docsvision

Руководство разработчика

Версия и дата сборки: dv6, 2025-11-28

Документация для разработчиков

Данный раздел документации предназначен для разработчиков Docsvision. Документация поделена на две основные части:

- "Руководство по разработке на платформе Docsvision" — работа с API системы и разработка компонентов для модулей Платформа, Базовые объекты, Управление процессами и других на языке C#.
- "Руководство по разработке web-решений" — разработка решений и компонентов на базе модуля Web-клиент на ReactJS.

Руководство по разработке платформы Docsvision

Первоочередная задача этого руководства — рассказать о возможностях использования и дать разработчику практические примеры работы с API Docsvision. При необходимости примеры сопровождаются фрагментами API Docsvision на языке C# для разработки в среде Visual Studio.

Для получения справочной информации по программному коду системы рекомендуется обращаться к описанию объектной модели Docsvision: "Библиотека классов".

Документация касается общего устройства Docsvision и приложений системы, а также даёт рекомендации по работе с прогрммным кодом системы. В соответствующих разделах приводятся примеры разработки скриптов и карточек, расширяющих стандартную функциональность системы и необходимые меры предосторожности при модификации стандартных компонентов.

При отсутствии необходимой функциональности в стандартной поставке системы разработчик может создать собственные карточки и решения, добавляющие или расширяющие стандартную бизнес-логику. Данное руководство описывает разработку скриптов в бизнес-процессах для реализации действий, недоступных по умолчанию.

Если функциональности стандартных утилит, конструкторов и справочников недостаточно,разработчик может интегрировать в Docsvision сторонние утилиты, которые будут получать данные и передавать их в систему.

Руководство по разработке web-решений

Раздел "Руководство по разработке web-решений" посвящён разработке на базе модуля Web-клиент и включает, но не ограничивается следующей информацией: создание клиентских скриптов, созданией расширений модуля и программы Конструктор Web-разметок, изменение и добавление новых элементов управления, дополнительных компонентов модуля, комментарии к исходному коду Web-клиента и прочую информацию, необходимую для разработки в web-среде.

Руководство по разработке web-решений требует от разработчика опыта в разработке компонентов на ReactJS и общего знания системы Docsvision.

Руководство по разработке Модуля интеграции с операторами ЭДО

Раздел Разработка компонентов модуля Модуля интеграции с операторами ЭДО описывает возможности по разработке компонентов для *Справочника настроек операторов ЮЗДО*. Сюда входят как компоненты для создания, чтения и изменения карточек Docsvision, так и компоненты коннекторов к операторам ЭДО, включая сервис сообщений, импор и работа с ЭП.

Вынесенная в компоненты функциональность модуля позволяет абстрагировать механизм отправки электронных документов от видов карточек и механизмов взаимодействия с операторами ЭДО.

Руководство по разработке Модуля интеграции с реестром МЧД

В разделе Описание взаимодействия с API модуля описывается API и обращение с программным кодом модуля. В разделе документации описывается API для выполнения действия с доверенностью, такие как регистрация, проверка статуса и отзыв, а также для обработки сообщений.

Руководство по разработке модуля Служба фоновых операций

Документация разработчика модуля Служба фоновых операций содержит описание объектной модели модуля.

Что нового

Далее перечислены изменения в документации:

Изменения в API

API для работы с МЧД

1. Добавлены новые классы:
    - `FindPowerOfAttorneyCardsResults – класс` — Содержит результаты поиска использования кодов полномочий в СКД.
    - `FindPowerOfAttorneyCardsResultsItem – класс` — Результат поиска использования кода полномочий в СКД.
    - `BaseCardExternalPowerOfAttorney` — Раздел списка подписей для хранения информации о машиночитаемых доверенностях контрагентов.
    - `PowerOfAttorney` — системная карточка доверенности.
    - `PowerOfAttorneyMainInfo` — представляет секцию "Основная информация" системной карточки доверенности.
    - `PowerOfAttorneyRepresentative` — представляет секцию "Представитель" системной карточки доверенности.
    - `PowerOfAttorneyRepresentativesPowers` — представляет секцию "Полномочия" системной карточки доверенности.
    - `PowerOfAttorneySubsidiaryPowersOfAttorney` — представляет секции "Системные карточки" дочерних доверенностей.
    - `Powers` — представляет полномочия для справочника полномочий МЧД.
    - `PowersCode` — представляет коды полномочий для справочника полномочий МЧД.
    - `PowersGroup` — представляет группы полномочий для справочника полномочий МЧД.
    - `PowersPowerOfAttorneyFormat` — представляет типы доверенности карточки справочник полномочий МЧД.
    - `IPowerOfAttorneyMachineReadableProvider` — определяет методы формирования и чтения машиночитаемой доверенности.
    - `IPowerOfAttorneyService` — предоставляет методы для работы с системной карточкой доверенности.

- `PowerOfAttorneyData` — Базовый класс сведений о доверенности.
- `PowerOfAttorneyDigest` — Представляет собой сведения о доверенности.
- `PowerOfAttorneyEMCHDData` — Сведения о доверенности в формате единой формы доверенности.

  ▼ *Внутренние классы:*

  - `AddressInfo` — Сведения об адресе (АдрТип).
  - `BranchOfRussianEntityInfo` — Сведения о филиале (обособленном подразделении) юридического лица (ФилПерПолн).
  - `ConfirmationOfAuthorityDocument` — Документ, подтверждающий полномочия лица, действующего без доверенности (ДокПдтвТип).
  - `DelegatedPowerOfAttorneyPrincipal` — Сведения о лице, передавшем полномочия (ПередПолн).
  - `FIO` — Фамилия, имя, отчество (при наличии) (ФИОТип).
  - `ForeignEntityInfo` — Сведения об иностранном юридическом лице (СвИнОргТип).
  - `HandwrittenSignature` — Рукописная подпись (ПодпРукопис)
  - `IndividualInfo` — Сведения о физическом лице (СведФЛТип)
  - `IndividualPrincipalInfo1` — Сведения о физическом лице (ФЛПерПолн).
  - `IndividualPrincipalInfo` — Сведения о физическом лице (доверителе) (ФЛДоверТип).
  - `IrrevocablePowerOfAttorneyInfo` — Сведения о безотзывной доверенности (Безотзыв).
  - `LegalEntityInfo` — Сведения о юридическом лице (или филиале (обособленном подразделении) юридического лица) (СвОргТип).
  - `MachineReadablePowerRestrictions` — Сведения об ограничении (ограничениях) к полномочию (ОгрСвПолн).
  - `MachineReadablePowersInfo` — Сведения о машиночитаемом полномочии (машиночитаемых полномочиях) (МашПолн).
  - `NotaryCertificateInfo` — Сведения о нотариальном удостоверении (СвНотУдТип).
  - `NotaryDeputyInfo` — ВРИО нотариуса (ВрионотТип).

- `NotaryInfo` — Сведения о нотариусе, совершившем нотариальное действие (СвНотДействТип).
- `PowerOfAttorneyDocument` — Состав и структура документа (Документ).
- `PowerOfAttorneyDocumentData` — Доверенность (Довер).
- `PowerOfAttorneyInfo` — Сведения о доверенности (СвДовТип).
- `PrincipalInfo` — Сведения о доверителе (Доверит).
- `PrincipalsInfo` — Сведения о доверителе (доверителях) (СвДоверит).
- `RepresentativesInfo` — Сведения о представителе (представителях) (СвУпПредТип).
- `RepresentativePowersInfo` — Сведения о полномочиях представителя (представителей) (СвПолнТип).
- `PowerOfAttorneyRevocationApplicantInfo – класс` — Информация о лице, отзывающем доверенность.
- `PowerOfAttorneyRevocationData – класс` — Данные отзыва доверенности.
- `PowerOfAttorneyRevocationInfo – класс` — Содержит информацию об отзыве доверенности.
- `RepresentativesInfo` — Сведения о представителе (представителях) (СвУпПредТип).
- Вспомогательные перечисления:
  - `AuthorityType` — Тип полномочия.
  - `CitizenshipType` — Признак наличия гражданства
  - `EntityType` — Тип лица, передавшего полномочия / тип представителя.
  - `Gender` — Пол.
  - `JointRepresentationType` — Признак совместных полномочий.
  - `NotarialActionParticipantStatus` — Статус участника нотариального действия.
  - `PowerOfAttorneyForm` — Форма доверенности.
  - `PowerOfAttorneyKind` — Вид доверенности.
  - `PowerOfAttorneyLossOfAuthorityType` — Признак утраты полномочий при передоверии.

        - PowerOfAttorneyOption — Признак доверенности.
        - PrincipalType — Тип доверителя.
        - RevocationCondition — Условие отзыва доверенности.
        - RevocationPossibleType — Признак безотзывной доверенности.
        - SoleExecutiveAuthorityType — Вид полномочий единоличного исполнительного органа.
    - PowerOfAttorneyFNSData — Базовый класс сведений о доверенности в формате ФНС.
    - PowerOfAttorneyFNSDOVBBData — Сведения о доверенности ФНС в формате DOVBB.

        ▼ *Внутренние классы:*

        - AddressInfo — Сведения об адресе (АдрТип)
        - BasicPowerOfAttorneyInfo — Сведения об Основной доверенности (СвОснДовер)
        - BasicPowerOfAttorneyPrincipalInfo — Сведения о доверителе Основной доверенности (СвДовер0)
        - BranchManagerInfo — Сведения о руководителе обособленного подразделения (СвРукОП)
        - ConfirmationOfAuthorityDocument — Реквизиты документа, подтверждающего полномочия (РеквДокПдтвТип)
        - DelegatedAuthorityPrincipalInfo — Сведения о лице, передавшем полномочия (СвЛицПередПолн)
        - ElectronicDocumentTransferMethod — Способ передачи электронного нотариального документа (СпПрдЭНотДок)
        - FIO — Фамилия, имя, отчество (при наличии) (ФИОТип)
        - ForeignEntityInfo — Сведения об иностранном юридическом лице (СвИнОргТип)
        - ForeignLegalEntityPrincipalInfo — Сведения о доверителе – иностранном юридическом лице (ИнОргДовер)
        - HandwrittenSignature — Рукописная подпись (ПодпРукопис)
        - IdentityCardOfIndividual — Сведения о документе, удостоверяющем

личность физического лица (УдЛичнФЛТип)

- `IndividualDelegatedAuthorityInfo` — Сведения о лице, передавшем полномочия – физическом лице (ФЛПрдПолн)
- `IndividualInfo0` — Сведения по физическому лицу (СвФЛ)
- `IndividualInfo1` — Сведения по физическому лицу (СвПоФЛ)
- `IndividualInfo2` — Сведения о физическом лице (СведФизЛТип)
- `IndividualInfo` — Сведения о физическом лице (СведФЛТип)
- `IndividualInfoBase` — Управляет получением сведений о физическом лице.
- `IndividualPrincipalInfo` — Сведения о доверителе – физическом лице (ФЛДоверТип)
- `IrrevocablePowerOfAttorneyInfo` — Сведения о безотзывной доверенности (БезотзывТип)
- `LegalEntityInfo` — Сведения об организации (СвОргТип)
- `LegalRepresentativeInfo` — Сведения о законном представителе физического лица (СвЗакПредТип)
- `NotaryCertificateInfo` — Сведения о нотариальном удостоверении (СвНотУдТип)
- `NotaryDeputyInfo` — ВРИО нотариуса (ВрионНот)
- `NotaryInfo` — Сведения о нотариусе, совершившем нотариальное действие (СвНотДейств)
- `NotaryPaymentInfo` — Сведения об оплате за совершение нотариального действия (ОплатНотДейст)
- `OrganizationInfo` — Сведения об организации (СвОрг)
- `PowerOfAttorneyDocument` — Состав и структура документа (Документ)
- `PowerOfAttorneyDocumentData` — Доверенность (Довер)
- `PowerOfAttorneyInfo` — Сведения доверенности (СвДовТип)
- `PrincipalInfo` — Сведения о доверителе (СвДоверит)
- `PrincipalWithoutPowerOfAttorneyInfo` — Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов)

- `RepresentativeInfo` — Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип)
- `RepresentativePowerInfo` — Сведения о полномочиях представителя (представителей) (СвПолнТип)
- `RetrustPowerOfAttorneyInfo` — Сведения доверенности, выданной в порядке передоверия (СвДовПер)
- `RetrustPowerOfAttorneyInfoData` — Передоверие (Передов)
- `RussianEntityInfo` — Сведения о российском юридическом лице (СвРосОргТип)
- `RussianLegalEntityPrincipalInfo` — Сведения о доверителе – российском юридическом лице (РосОргДовер)
- `SoleProprietorInfo0` — Сведения об индивидуальном предпринимателе (СведИПТип)
- `SoleProprietorInfo1` — Сведения об индивидуальном предпринимателе (СвИПТип)

◦ `PowerOfAttorneyFNSDOVELDat` — Сведения о доверенности ФНС в формате DOVEL.

▼ *Внутренние классы:*

- `AddressInfo` — Сведения об адресе (АдрТип)
- `FIO` — Фамилия, имя, отчество (при наличии) (ФИОТип)
- `IndividualInfo` — Сведения о физическом лице (СведФЛТип)
- `IndividualPrincipalInfo` — Сведения о доверителе — физическом лице (ФЛДоверТип)
- `OrganizationInfo` — Сведения об организации (СвОрг)
- `PowerOfAttorneyDocumentData` — Доверенность (Довер)
- `PowerOfAttorneyInfo` — Сведения доверенности (СвДовТип)
- `PrincipalInfo` — Сведения о доверителе (СвДоверит)
- `PrincipalWithoutPowerOfAttorneyInfo` — Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов)
- `RepresentativeInfo` — Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип)

- `PowerOfAttorneyMachineReadableInfo` — Содержит информацию о МЧД.
- `PowerOfAttorneyVerification` — Содержит результат проверки действительности доверенности
- `ImportESNSIResults` — Результат импорта полномочий из ЕСНСИ.

2. Добавлены новые перечисления::
   - `PowerOfAttorneyRegTransferStatuses` — статус передачи доверенности в распределённый реестр ФНС.
   - `PowerOfAttorneyRetrustType` — признак возможности оформления передоверия.
   - `PowerOfAttorneyRevocationApplicantType – перечисление` — Тип заявителя отзыва доверенности.
   - `PowerOfAttorneyRevocationType – перечисление` — Тип заявления на отзыв.
   - `PowerOfAttorneySignatureFormat` — формат подписи.
   - `PowerOfAttorneyStatus` — статус доверенности.
3. Добавлены новые интерфейсы:
   - `ICompressService` — сервис для работы с архивами.
     - Интерфейс предоставляет метод `CompressFolder – метод (string, string, CompressionFormat)`, сжимающий содержимое папки.
   - `IPowersService` — Сервис справочника полномочий.
4. Интерфейс `IBaseCardService` дополнен новыми методами, добавляющими данные МЧД в список подписей:
   - `AddExternalPowerOfAttorney(SignatureList, Guid, BaseCardExternalPowerOfAttorneyStatus)`
   - `AddExternalPowerOfAttorney(SignatureList, Guid, BaseCardExternalPowerOfAttorneyStatus, string)`
5. Класс `BaseCardSignature` дополнен новым свойством `ExternalPowerOfAttorney`, позволяющим установить и прочитать стороннюю МЧД.

Добавлен метод

Добавлено описание метода `RefreshObject(IObjectRef)` для обновления корневого объекта.

Уведомления о заданиях

Класс `TaskMessagesEventHandlerService` расширен методами, позволяющими определить собственную логику по рассылке писем с CC и BCC:

- `GetNotificationCc – метод (Task, StaffEmployee)` — позволяет вычислить адресатов в BCC.
- `GetNotificationBcc – метод (Task, StaffEmployee)` — позволяет вычислить адресатов в CC.

Класс `TaskGroupMessagesEventHandlerService` также расширен методами, позволяющими определить собственную логику по рассылке писем с CC и BCC:

- `GetNotificationBcc(TaskGroup, StaffEmployee)` — позволяет вычислить адресатов в BCC.
- `GetNotificationCc(TaskGroup, StaffEmployee)` — позволяет вычислить адресатов в CC.

Цитаты из файла

- Пространство имён `DocsVision.Platform.ObjectManager` дополнено классами и методами для работы с цитатами файлов:

  Классы:

  - `CitationFileItem` — список цитат, сгруппированных по файлам.
  - `FilesCitationInfo` — список цитат, сгруппированных по карточкам.
  - `SearchFilesCitationsRequest` — запрос цитируемых данных из файла.

  Методы:

  - `CardManager.GetSearchFilesCitationsRequest(SearchFilesCitationParentObjectsType, string)` — создаёт класс запроса, включающий параметры поиска и формирования цитат.
  - `CardManager.SearchFilesCitations(SearchFilesCitationsRequest)` — осуществляет непосредственно получение данных.

- Реализующий модель полнотекстового серверного поиска класс `FullTextSearch` дополнен следующими свойствами:

  - `FullTextSearch.Mode` — получает или задаёт режим поиска.
  - `FullTextSearch.QueryString` — получает или задаёт значение поисковой

строки.

- `FullTextSearch.SearchQuery` — получает поисковый запрос.
- `FullTextSearch.WithCitations` — определяет, будут ли использованы цитаты из файлов в результатах поиска.

Согласование

- Пространство имён `DocsVision.ApprovalDesigner.ObjectModel.Services` дополнено двумя классами:
  - `ApprovalStageService` — класс содержит методы для работы с этапом согласования.
  - `ApprovalStageEventHandlerService` — обрабатывает события этапа согласования.
- Интерфейс `IApprovalStageService` расширен новым методом `GetStageApprovers(ApprovalStage approvalStage, Document document)`, который возвращает согласующих этапа в результате вызова `CopyApproversFromDocument`.

Агрегация данных грида в Web-клиенте

- Класс `CardManager` расширен методами API для работы с агрегатами поиска:
  - `GetSearchAggregationItemsRequest (SearchAggregationItemsResultType, Guid, Guid, string, int?)` — возвращает результат агрегации.
  - `FindAggregationCardsInfo (SearchAggregationItemsRequest)` — получает список агрегатов и/или результаты фильтрации.
- В пространство имён `DocsVision.Platform.ObjectManager` добавлены новые классы:
  - `SearchAggregationItemsRequest` — представляет объект запроса
  - `SearchAggregationItemInfo` — представляет список объектов.
- В пространство имён `DocsVision.Platform.ObjectManager.SearchModel` добавлено перечисление:
  - `SearchAggregationItemsResultType` — результат поиска агрегатов.

Руководство по разработке

Базовая поставка платформы Docsvision содержит полный набор средств для разработки собственных решений, удовлетворяющих потребности организации в сфере электронного документооборота и его автоматизации.

К таким средствам можно отнести:

- Набор конструкторов — мощное и простое средство настройки существующего программного продукта.
- Возможность разработки новых шлюзов, функций и сценариев для использования их в бизнес-процессах.
- Возможность создавать новые типы объектов платформы (карточек) для полноценного представления объектов целевой сферы деятельности.
- Публичный API для разработки отдельных утилит, сценариев бизнес-процессов.

Основное назначение данного раздела документа — предоставить разработчику информацию о возможностях API Docsvision через практические примеры.

Помимо этого, документ содержит другие сведения, напрямую связанные с созданием решений:

- Создание собственных карточек и решений.

  Собственные типы карточки разрабатываются, например, если невозможно или сложно реализовать необходимую бизнес-логику, либо иную функциональность в т.ч. пользовательский интерфейс при помощи конструкторов Docsvision.

- Разработка сценариев в карточках.

  Сценарии (скрипты) в карточках библиотеки *Базовые объекты* позволяют расширить их функциональность штатным образом без разработки нового решения.

- Разработка сценариев в бизнес-процессах.

  Сценарии (скрипты) в бизнес-процессах являются одним из способов реализации специфических действий, не предусмотренных стандартными функциями бизнес-процессов.

- Разработка прикладных приложений (утилит).

Разработка внешних по отношению к Docsvision приложений, выполняющих какие-либо действия в системе, может быть востребована при интеграции Docsvision с другими системами (синхронизация справочных данных). Примерами таких утилит могут выступать приложения для импорта данных в справочники сотрудников и контрагентов, входящие в пакет разработчика.

В данном разделе приведена справочная информация об общем устройстве платформы и приложения *Workflow*, а также список ограничений на модификацию платформы, которые должны учитываться при разработке решений:

- Использование API Docsvision для взаимодействия с объектами системы.
- Разработка новых типов карточек.
- Основы разработки расширений и модулей для платформы и приложения *Workflow*.

Примеры могут сопровождаться или не сопровождаться описанием задействованных в примере сущностей API Docsvision. В любом случае для получения справочной информации об объектах API Docsvision рекомендуется обращаться к их описанию, приведенному в разделе Библиотека классов.

Все примеры, за исключением тех случаев, когда указано иное, разработаны на языке C# в среде разработки Visual Studio.

Для разработки внешних приложений потребуется установленный на компьютере разработки клиент Docsvision.

В этом разделе представлена следующая информация:

Общие сведения

Вместе с платформой разработчик получает два уровня API:

- Базовый API — универсальное средство программного доступа ко всем функциям платформы Docsvision. Может быть использован в проектах, основанных на .NET (сборка `DocsVision.Platform.ObjectManager.dll`) и COM (сборка `ObjectManager.dll`) технологиях.
- Высокоуровневый слой, объектная модель Docsvision (далее — "объектная модель") — набор базовых объектов и функциональных возможностей, с помощью которых могут быть реализованы специализированные модели сущностей выбранной предметной области.

  Использование готовых базовых типов позволяет снизить издержки на разработку собственного решения, обеспечит прозрачность архитектуры, простоту расширения и интеграции нескольких решений, включая *Базовые объекты* (являются частью платформы Docsvision). API данного уровня является удобным средством для работы с карточками. Сборки данного уровня API имеют только .NET-версию.

В зависимости от применимости функциональных возможностей выбранного уровня API, можно выделить четыре группы задач разработчика:

1. Задачи, при решении которых могут быть задействованы как базовый API, так и объектная модель.

   Это, например, с некоторыми ограничениями, доступ к данным карточек.

2. Задачи, которые эффективнее решать с использованием методов объектной

модели.

Одной из таких задач является запуск задания на исполнение.

3. Задачи, которые могут быть решены только методами базового API.

   Например: поиск данных по секции карточки.

4. Задачи, которые не могут быть решены с помощью API Docsvision.

Если при решении определённой задачи потребуется задействовать методы обоих уровней API нужно помнить, что каждый слой обладает собственным механизмом кэширования, что может повлиять на актуальность данных на одном слое по сравнению с другим.

В примерах использования API, рассмотренных в разделе Руководство по разработке, если это возможно, одна задача будет решена как методами базового API, так и методами объектной модели, что позволит оценить удобство и функциональные возможности разных уровней при решении конкретной задачи.

Далее приведена информация относительно некоторых ключевых объектов и терминов API Docsvision, которые необходимы для понимания последующих примеров разработки.

Пользовательская сессия и контекст объектов

В API Docsvision существуют две ключевые сущности, представляющие функциональные возможности своего слоя API: *пользовательская сессия* (или сессия пользователя) и *контекст объектов*.

Пользовательская сессия

Пользовательская сессия является главным объектом в базовом API и обеспечивает единую точку доступа к остальным функциям данного API.

С физической точки зрения, сессия — контекст сеанса связи с сервером для конкретного пользователя. Сессия ассоциируется с доменной учетной записью, от имени которой установлено соединение, и в дальнейшем все привилегии на доступ к данным из этой сессии проверяются относительно этой учетной записи.

В рамках сессии кэшируются все данные, к которым производилось обращение, для ускорения последующего доступа к ним. Размер кэша и

продолжительность пребывания в нём данных автоматически регулируются платформой, на основании внутренних алгоритмов. Разработчик может только явно потребовать удаления конкретных объектов из кэша (при помощи методов этих объектов).

Создание сессии является первоочередной задачей при разработке любых внешних (относительно платформы) приложений, работающих с Docsvision. При разработке внутрисистемных объектов, таких как скрипты карточек (в *Конструкторе скриптов* или *Конструкторе разметок*) или бизнес-процессы, готовая сессия передается при инициализации.

Контекст объектов

Контекст объектов — элемент объектной модели, который обеспечивает загрузку и сохранение данных карточек. *Контекст объектов* также предоставляет доступ к функциям, реализующим основную бизнес-логику для работы с бизнес-объектами. Функциональные возможности контекста объектов обеспечивают инициализацию объектной модели карточки при загрузке её данных из базы данных, а также последующее связывание данных.

Контекст объектов должен быть проинициализирован, чтобы работать с объектной моделью карточки. Исключением являются разработка скриптов карточек и бизнес-процессов — в этих случаях готовый *контекст объектов* будет доступен из базового класса (для скрипта карточки) и из параметров вызванного метода (для бизнес-процесса).

Для инициализации *контекста объектов* потребуется *пользовательская сессия*.

Менеджеры базового API

Менеджеры — сущности базового API, в которых собраны методы для работы с различными объектами платформы.

Например, есть менеджеры для работы с карточками, файлами, безопасностью и прочие:

- `AccessManager` — менеджер объектов безопасности, в котором представлены методы управления правами доступа на объекты системы, а также методы для работы с крипто-объектами.
- `CardManager` — менеджер карточек. Данный менеджер предоставляет полный набор средств для работы с экземплярами карточек: создание, получение и удаление.

- `ExtensionManager` — менеджер расширений. Данный менеджер предоставляет единственный метод `GetExtensionMethod`, возвращающий метод серверного расширения.
- `FileManager` — менеджер файлов, предоставляющий методы для работы с файлами: создание, получение и удаление, а также некоторые другие.
- `LicenseManager` — менеджер лицензий. Данный менеджер предоставляет два метода для учета числа подключений в лицензии дополнительного модуля.
- `LockManager` — менеджер блокировок. Предоставляет методы получения объектов блокировки с объекта системы. Для полнофункциональной работы могут потребоваться права администратора.
- `LogManager` — менеджер журнала. Данный менеджер содержит операции для работы с системным журналом.
- `ReportManager` — менеджер хранимых процедур. Предоставляет метод для получения данных хранимой процедуры.

Чтобы получить доступ к интересующему *менеджеру*, необходимо обратиться к соответствующему свойству сущности пользовательской сессии:

```
CardManager cardManager = userSession.CardManager;
```

Сервисы объектной модели

На уровне объектной модели Docsvision принято выносить бизнес-логику из объектной модели карточки в отдельные *сервисы*, доступ к которым можно получить непосредственно из *контекста объектов*. *Сервисы* выполняют задачи, аналогичные тем, что выполняют *менеджеры* базового API, но со специализацией на конкретных типах карточек — есть сервисы для работы с документами, с заданиями, со справочниками и т.п.

На программном уровне, *сервис* — это пара, состоящая из интерфейса, определяющего функциональные возможности *сервиса*, и класса, реализующего данный интерфейс.

В базовой поставке Docsvision предоставляются системные сервисы (для работы с блокировками, доступом и т.п.), а также сервисы для работы с карточками библиотеки карточек *Базовые объекты*. Полный список сервисов приведён далее.

Таблица 1. Полный список сервисов

Область действия	Сервис	Назначение
Общего назначения	IAccessCheckingService	Определяет функции получения списка ролей и доступных операций сотрудника в пределах заданной карточки. Предоставляет методы сброса кэша ролевой модели.
	IBaseCardService	Предоставляет методы установки и проверки ЭП, генерации дайджеста и управления бизнес-процессом.
	ILockService	Позволяет управлять состоянием блокировки объектов, получать информацию о текущем состоянии и владельце блокировки.
	ILogService	Определяет методы добавления и получения записей журнала карточки.
	IServerExtensionProxyService	Позволяет выполнять методы серверного расширения BackOffice.
	ISettingsCardService	Предоставляет методы доступа к системным настройкам.
	ICryptService	Сервис шифрования файлов карточек приложения *Базовые объекты*

Область действия	Сервис	Назначение
Карточки		

		с карточками типа *Карточка файла с*
Область действия	**Сервис**	**Назначение**
	`ISurveyService`	Предназначен для работы с карточками типа *Список опросов*
	`IUserProfileCardService`	Предназначен для работы с карточками типа *Карточка настроек пользователя*
Конструкторы	`IRoleModelService`	Конструктор ролей
	`IBaseUniversalService`	Карточка строки справочника
	`ILayoutService`	Конструктор разметок
	`IScriptingService`	Конструктор скриптов
	`IStateService`	Конструктор состояний
Справочники	`ICategoriesService`	Справочник категорий
	`IKindService`	Справочник видов карточек
	`ISettingsService`	Предоставляет методы для работы с настройками расширений справочника видов.
	`IStaffService`	Справочник сотрудников
	`IPartnersService`	Справочник контрагентов
	`IServersService`	Справочник серверов
	`ISignatureLabelService`	Справочник меток подписей
	`ILinkService`	Справочник ссылок

Чтобы получить один из *сервисов*, необходимо использовать метод `GetService` *контекста объектов*, уточнив тип (публичный интерфейс) запрашиваемого сервиса:

```
IDocumentService documentService = objectContext.GetService<IDocumentService>(); ① ②
```

① `objectContext` — сущность контекста объектов.

② `IDocumentService` — интерфейс, реализуемый *сервисом*.

Преобразователи данных

В объектной модели Docsvision карточки представляются в виде сущностей конкретной предметной области, к примеру: задания, документы, книги и т.п. Каждый тип таких сущностей обладает, как правило, собственной объектной моделью, которая определённым образом связана с данными карточки.

Чтобы определить механизм этого связывания, реализуется специальный класс — *преобразовать данных*,- в котором определяется связь между элементом карточки и свойством объектной модели, представляющим данный элемент.

При получении объекта из *контекста объектов*, данные карточки автоматически загружаются в объектную модель карточки, а при сохранении данных выполняется обратная операция. Для работы данного механизма соответствующий преобразователь данных должен быть загружен в *контекст объектов*.

В базовую поставку Docsvision входят *преобразователи данных*, которые необходимы для работы с объектной моделью карточек библиотеки *Базовые объекты*. Для работы с объектной моделью собственных типов карточек, разработчику предлагается реализовать собственный *преобразователь данных*.

Объекты хранения данных

Ключевым объектом Docsvision, соответствующим сущностям целевой системы, является *карточка*. С карточками работают как пользователи, так и разработчики. Если для пользователя карточка — это прежде всего графический интерфейс, на который определённым образом выведены данные, то для разработчика — это объектная модель, метаданные и сами данные.

Прежде чем перейти к дальнейшему описанию необходимо дать определения нескольким ключевым понятиям:

- *Тип карточки* — описание сущности целевой системы в Docsvision.

  Если проводить аналогию с программированием, то тип карточки — это класс.

В Docsvision представлено /dv6/schemas/dv6/[большое количество] типов карточек. Если существующие типы недостаточно точно отражают объект целевой системы, может быть разработан собственный с уникальной бизнес-логикой.

- *Экземпляр карточки* — экземпляр карточки определённого типа. Экземпляр карточки обладает уникальным идентификатором, по которому карточка может быть получена из базы данных. Если следовать аналогии, то экземпляр карточки — это объект.
- *Поле* — элемент структуры карточки, предназначенный для непосредственного хранения данных определённого типа. Также может ссылаться на другие элементы данной или другой карточки.
- *Секция* — элемент структуры карточки, объединяющий группу полей.

  С точки зрения метаданных, карточка — это набор *секций* с коллекциями *полей*. Каждая секция карточки обладает собственным набором полей, что обеспечивает необходимый уровень гибкости для реализации структур хранения данных.

На уровне базы данных секция — это таблица, а поле — это столбец данной таблицы.

Конкретный набор секций и полей, а также их характеристики определяет *схема карточки*. Каждый тип карточек обладает собственной схемой, что позволяет реализовать в системе объекты, приближенные по характеристикам к сущностям выбранной предметной области. К примеру, для хранения информации о книгах библиотеки, может быть реализована карточка типа `Книга` с полями: название, автор, дата издания и т.п. Для хранения информации о входящих документах компании можно реализовать карточку типа `Входящий документ` с полями: отправитель, дата отправки, специалист, который должен получить данный документ.

На уровне экземпляра карточки (конкретного документа, задания и т.п.) наборы данных хранятся не просто в полях секций карточки, а в *строках* секций — полях подчиненных по отношению к секциям сущностях. Строки обеспечивают возможность хранения в одной секции коллекции наборов данных, в т.ч. с определённой иерархией.

Это может быть востребовано, к примеру, если в карточке типа `Книга` должны

храниться сведения о нескольких авторах этой книги. Помимо табличного варианта организации строк секции, предусмотрена иерархическая структура, при которой строка секции может содержать подчиненные строки из этой же секции. Такая структура секции обычно используется в справочниках — карточки, представленные в единственном экземпляре (ограничение устанавливается схемой карточки), данные из которых используются в других карточках. Примером справочника может служить *Справочник сотрудников* из библиотеки карточек *Базовые объекты*, данные которого используются во многих других типах карточек данной библиотеки.

Структура секции определяются её типом:

- Плоская секция — может содержать только одну строку. Создание второй строки будет воспринято как ошибка.
- Коллекционная секция — может содержать набор строк. Секция данного типа по сути аналогична обычной таблице.
- Иерархическая секция — может содержать иерархию строк, в которой строка может иметь в подчинении строки этой же секции.

На уровне базы данных секция строка секции — это строка таблицы, а секция — сама таблица.

Далее, для обобщения изложенной выше информации, представлена структура экземпляра карточки некого типа. Подобную структуру можно составить на основе данных, предоставляемых программой "Docsvision Explorer" из комплекта "Resource Kit".

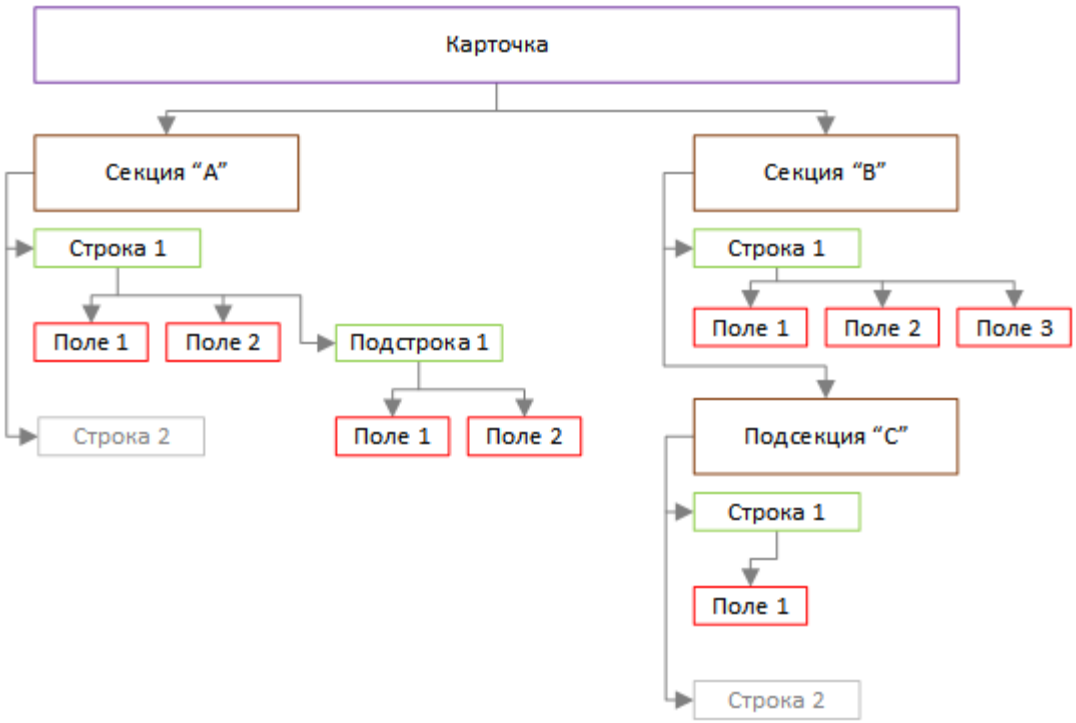


Рисунок 1. Структура экземпляра карточки

Приведенную карточку (в сочетании со схемой типа карточки) можно охарактеризовать следующим образом:

- Карточка содержит две секции: Секция "A" (иерархическая) и Секция "B" (плоская).
- Секция "A" содержит две строки: Строка 1 и Строка 2. У Строки 1 есть подчиненная строка — Подстрока 1. В секции определено два поля: Поле 1 и Поле 2.
- Секция "B" содержит единственную строку (Строка 1), что определено ограничениями плоской секции. В секции определены поля: Поле 1, Поле 2 и Поле 3.
- Секция "B" также имеет подчиненную секцию — Подсекция "C" (коллекционная), которая содержит две строки: Строка 1 и Строка 2. В секции определено единственное поле — Поле 1.

Для представления данных карточки в API Docsvision реализовано несколько классов, специфичных для выбранного уровня (слоя) API. Далее приведены такие типы для обоих уровней API Docsvision.

Объекты хранения данных на уровне базового API

Типы, приведенные далее, определены в сборке `DocsVision.Platform.ObjectManager.dll`.

- `CardData` — данные экземпляра карточки. Включает совокупность всех данных секций и атрибутов конкретного экземпляра карточки.
- `SectionData` — данные секции карточки, из которой может быть получена коллекция всех её строк.
- `RowDataCollection` — коллекция строк секции, из которой могут быть выбраны конкретные строки.

Если коллекция получена непосредственно из карточки, то она считается "живой", т.е. при создании/удалении строки изменения в коллекции будут видны сразу же. Если коллекция получена поисковым запросом, она является "моментальным снимком" данных, доступным только на чтение.

- `RowData` — строка секции. Выбрать строку из коллекции строк можно по уникальному идентификатору, по номеру в коллекции либо перебором.
- `SubSectionData` — подмножество строк (подсекция) секции, подчинённых определённой строке из родительской секции.
- `Field` — поле строки секции, а точнее его значение.

Обычный сценарий доступа к данным карточки с привлечением приведенных типов следующий:

1. Получить экземпляр карточки — `CardData`.
2. Получить нужную секцию из карточки: `CardData.Sections[Section_ID]`.
3. Получить коллекцию строк выбранной секции: `SectionData.Rows`.
4. Выбрать конкретную строку из секции: `SectionData.Rows[Row_ID]`.
5. Получить значение поля по псевдониму: `RowData[FieldAlias]`.

Объекты хранения данных на уровне объектной модели

Типы, приведенные далее, определены в сборках: `DocsVision.BackOffice.ObjectModel.dll` — содержит типы, описывающие сущности карточки и `DocsVision.Platform.ObjectModel.dll` — содержит общие базовые классы.

Приведенные далее сведения относятся только к карточкам, объектная модель которых унаследована от базового класса `BaseCard`. Если объектная модель создана на основе базовых классов из сборки `DocsVision.Platform.ObjectModel.dll`, механизм доступа к данным будет отличаться.

Объектная модель карточки предлагает гораздо меньшее (в сравнении с базовым API) число типов:

- BaseCard — базовый класс карточки, который содержит коллекции всех строк её секций.
- `BaseCardSectionRow` — строка секции, предоставляющая доступ к своим полям.

Как видно из списка, в объектной модели нет отдельных классов для секций и полей, что упрощает сценарий доступа к данным, который в общем случае будет следующий:

1. Получить экземпляр карточки — тип `BaseCard`, либо унаследованный от него.
2. Выбрать строки конкретной секции, воспользовавшись методом `BaseCard.GetSection`.
3. Выбрать строку из полученной коллекции.
4. Получить значение нужного поля по его псевдониму: `BaseCardSectionRow["FieldAlias"]`.

Если для типа карточки была реализована собственная объектная модель, то обращение к данным карточки будут выглядеть ещё проще:

1. Получить экземпляр карточки, к примеру, типа `SampleCard` — унаследован от типа `BaseCard`.
2. Получить коллекцию строк секции, из соответствующего публичного свойства класса: `SampleCard.SampleSection`.
3. Выбрать нужную строку секции: `SampleCard.SampleSection[0]`. Если секция является плоской и это учтено при реализации объектной модели, то данный шаг пропускается.
4. Получить значение поля: `SampleCard.SampleSection[0].SampleField`.

Помимо типов, приведенных выше и относящихся к доступу к данным карточки, в процессе разработки типов могут быть задействованы дополнительные типы, к примеру, относящиеся к справочникам. Если такие типы будут использованы далее, будет приведено их описание. Также описание большинства типов API Docsvision приведено в разделе Библиотека классов.

Подключение к серверу Docsvision

Первым шагом на пути разработки внешнего, по отношению к серверу и клиенту (Windows-клиенту) Docsvision, приложению является подключение к серверу

Docsvision. При этом должен быть получен экземпляр *пользовательской сессии* (или сессии пользователя) — главный объект базового API, обеспечивающий единую точку доступа к остальным функциям API данного уровня. При разработке скриптов карточки (в *Конструкторе скриптов*, либо в *Конструкторе разметок*) и бизнес-процессов создавать пользовательскую сессию не требуется — готовую сессию можно получить из класса скрипта карточки, а также из параметров вызываемых в бизнес-процессе методов.

С физической точки зрения, сессия — контекст сеанса связи с сервером для конкретного пользователя. Сессия ассоциируется с конкретной доменной учетной записью, от имени которой установлено соединение, и в дальнейшем все привилегии на доступ к данным из этой сессии будут проверяться именно для этой учетной записи.

Для выполнения операции с привилегиями другого пользователя необходимо явно указать его данные при создании сессии.

С точки зрения разработки, *пользовательская сессия* — экземпляр класса `UserSession`), который создаётся следующим образом:

```
UserSession CreateUserSession()
{ ①
 SessionManager sessionManager = SessionManager.CreateInstance();

 string connectionString =
"ConnectAddress=http://{SERVERNAME}/DocsVision/StorageServer/StorageServerService.asmx;Us
erName={USERNAME};Password={PASSWORD}"; ② ③

 sessionManager.Connect(connectionString); ④

 UserSession userSession = sessionManager.CreateSession(); ⑤

 return userSession;
}
```

① Создаем экземпляр менеджера сессий.

② `SERVERNAME` — имя сервера Docsvision.

③ `USERNAME` и `PASSWORD` — данные учетной записи для подключения к серверу Docsvision.

④ Передаем менеджеру сессий параметры подключения к серверу (описание см. по ссылке).

⑤ Создаем сессию пользователя.

Чтобы использовать приведенные в коде типы, потребуется:

1. Добавить в проект сборку Docsvision:

   `DocsVision.Platform.ObjectManager.dll`.

   Указанный DLL-файл входит в состав клиента и сервера Docsvision.

2. Подключить пространство имён:

   ```
   using DocsVision.Platform.ObjectManager;
   ```

   В приведенном выше примере строка подключения к серверу содержит логин и пароль пользователя, если эти сведения не указываются, подключение будет выполнено от имени пользователя, запустившего программу. Полный список параметров соединения с сервером Docsvision приведён в разделе Параметры соединения с сервером Docsvision.

Каждая пользовательская сессия обладает набором определённых параметров, значение которых устанавливается автоматическию Значение некоторых параметров можно изменить, но только на время работы в сессии. Значение определённого параметра пользовательской сессии можно получить из массива `UserSession.Properties`.

Описание доступных параметров приведено в разделе Список свойств пользовательской сессии.

Для работы с пользовательскими сессиями могут быть полезны два других метода менеджера сессий:

- `CloseSession(Guid)` — принудительно закрывает любую пользовательскую сессию (требуются права администратора системы).
- `GetOpenSessions` — возвращает список всех открытых сессий.

Инициализация контекста объектов

После подключения к серверу Docsvision, создания пользовательской сессии, может быть создан контекст объектов для работы с методами API уровня бизнес-логики.

Создавать контекст объектов не требуется, если:

- Пишется код для скрипта карточки, т.к. в этом случае готовый контекст объектов может быть получен непосредственно из класса скрипта.
- Предполагается использовать исключительно методы базового API.

Чтобы инициализировать контекст объектов, потребуется:

1. Получить экземпляр класса `ObjectContext` — сущность контекста объектов.
2. Используя платформенные сервисы, загрузить в контекст объектов преобразователи данных и необходимые сервисы.

Создание и инициализация *контекста объектов* является затратной операцией с точки зрения потребления аппаратных ресурсов. Рекомендуется использовать созданный и проинициализированный *контекст объектов*, а не создавать новый.

Далее приведён полный код инициализации контекста объектов для возможности работы с системными карточками и карточками библиотеки *Базовые объекты*.

▼ *Разверните, чтобы увидеть полностью:*

```
ObjectContext CreateObjectContext(UserSession userSession)
{
 var sessionContainer = new ServiceContainer();
 sessionContainer.AddService(typeof(UserSession), userSession);

 var objectContext = new ObjectContext(sessionContainer);

 var mapperFactoryRegistry = objectContext.GetService<IObjectMapperFactoryRegistry>();
 mapperFactoryRegistry.RegisterFactory(typeof(SystemCardsMapperFactory));
 mapperFactoryRegistry.RegisterFactory(typeof(BackOfficeMapperFactory));
 mapperFactoryRegistry.RegisterFactory(typeof(Docsvision.WorkerService.ObjectModel.Mapp
ing.WorkerServiceMapperFactory));


 var serviceFactoryRegistry = objectContext.GetService<IServiceFactoryRegistry>();
 serviceFactoryRegistry.RegisterFactory(typeof(BackOfficeServiceFactory));
 serviceFactoryRegistry.RegisterFactory(typeof(SystemCardsServiceFactory));
 serviceFactoryRegistry.RegisterFactory(typeof(Docsvision.WorkerService.ObjectModel.Ser
vices.WorkerServiceServiceFactory));


 objectContext.AddService<IPersistentStore>(DocsVisionObjectFactory.CreatePersistentSto
re(new SessionProvider(userSession), null));
```

```
 IMetadataProvider metadataProvider = DocsVisionObjectFactory.CreateMetadataProvider
(userSession);
 objectContext.AddService<IMetadataManager>(DocsVisionObjectFactory.CreateMetadataManag
er(metadataProvider, userSession));
 objectContext.AddService<IMetadataProvider>(metadataProvider);

 return objectContext;
}
```

Чтобы использовать приведенные в коде типы, потребуется:

1. Добавить в проект сборки Docsvision:
   - DocsVision.Platform.ObjectModel.dll
   - DocsVision.Platform.ObjectManager.dll
   - DocsVision.Platform.SystemCards.ObjectModel.dll
   - DocsVision.BackOffice.ObjectModel.dll
   - DocsVision.Platform.StorageServer.dll

   Указанные DLL-файлы входят в состав клиента и сервера Docsvision.
2. Подключить пространства имён:

```
using DocsVision.BackOffice.ObjectModel;
using DocsVision.BackOffice.ObjectModel.Mapping;
using DocsVision.BackOffice.ObjectModel.Services;
using DocsVision.Platform.Data.Metadata;
using DocsVision.Platform.ObjectManager;
using DocsVision.Platform.ObjectModel;
using DocsVision.Platform.ObjectModel.Mapping;
using DocsVision.Platform.ObjectModel.Persistence;
using DocsVision.Platform.SystemCards.ObjectModel.Mapping;
using DocsVision.Platform.SystemCards.ObjectModel.Services;
using System.ComponentModel.Design;
```

Работа с карточками

В данном разделе приведены наиболее востребованные сценарии использования API Docsvision при работе с карточками.

В этом разделе:

- Получение экземпляра карточки из базы данных

- Получение данных из секции карточки
- Изменение данных карточки
- Создание нового экземпляра карточки
- Удаление объектов
- Блокировка и снятие блокировки
- Архивирование и извлечение карточки из архива
- Поиск
- Экспорт, импорт и подготовка карточек к печати
- Подписание и шифрование
- Библиотека карточек "Базовые объекты"

Получение экземпляра карточки из базы данных

Платформа Docsvision предоставляет несколько способов получения карточки: её можно запросить по уникальному идентификатору в формате `Guid`, либо получить из результата работы поискового запроса. Здесь рассмотрен первый способ, для которого необходимо знать идентификатор получаемой карточки.

Идентификатор карточки может быть получен, как минимум, двумя способами:

- Значение можно посмотреть в свойствах карточки в Windows-клиенте, где идентификатор приведён в строке *Ссылка*.
- Найти карточку при помощи программы "Docsvision Explorer". Идентификатор указан в строке `Selected card ID`.

Базовый API

Получить данные карточки методами базового API можно следующим образом:

```
CardData card = userSession.CardManager.GetCardData(cardId) ① ② ③
```

① `userSession` — пользовательская сессия.

② `CardManager` — менеджер карточек, в котором собрана основная функциональность базового API для работы с карточками.

③ `cardId` — идентификатор запрашиваемой карточки. Обычно (как и `GetCardData`) методы принимают идентификаторы, приведенные к типу `System.Guid`:

```
Guid cardId = new Guid("00000000-0000-0000-0000-000000000000");
```

После получения данных карточки, можно:

- Получить данные из её секций.
- Установить или снять блокировку.
- Отправить карточку в архив или извлечь из архива.
- Экспортировать данные карточки в формате XML в файл.
- При необходимости изменить режим чтения данных.

Полный список доступных методов и свойств см. в описании типа `CardData`.

Объектная модель

Для получения данных карточки, а точнее её объектной модели, *контекст объектов* предоставляет метод `GetObject`, который может быть вызван следующим образом:

```
BaseCard card = objectContext.GetObject<BaseCard>(cardId); ① ② ③
```

① `objectContext` — проинициализированный *контекст объектов* (см. Подключение к серверу Docsvision).

② `BaseCard` — класс объектной модели карточки. Здесь использован базовый класс карточек библиотеки *Базовые объекты*. В любом случае указываемый тип должен быть наследником типа `ObjectBase` — базовый класс для всех сущностей данного уровня API.

Список классов объектных моделей карточек библиотеки *Базовые объекты* приведён в разделе Библиотека карточек "Базовые объекты".

③ `cardId` — идентификатор запрашиваемой карточки.

Метод `GetObject` достаточно универсален и может вернуть из *контекста объектов* любой объект, для которого разработана объектная модель и в контексте объектов зарегистрирован *преобразователь данных* (будет рассмотрен позднее).

После получения объектной модели карточки, возможно:

- Получить данные из её секций, если класс объектной модели карточки был унаследован от класса `BaseCard` или его потомков.

Если класс карточки наследуется напрямую от `ObjectBase`, функциональность получения данных секций потребуется реализовать самостоятельно.

- Получить объектную модель секций, если это предусмотрено объектной моделью карточки.

Объектная модель секции карточки, а точнее её строк, позволяется работать с секцией как с объектом, обладающим набором определённых свойств, представляющих поля карточки. Это отличает данный способ от ранее приведенного, идеология которого ближе к принципам работы с секциями в базовом API. В базовом API необходимо знать идентификаторы секций и точные названий полей.

Получение данных из секции карточки

После получения данных карточки, либо её объектной модели можно получить доступ к данным её секций.

Базовый API

На данном уровне API, секция карточки представлена типом `SectionData`. Получить данные секции карточки можно следующим образом:

```
SectionData section = card.Sections[sectionId]; ① ② ③
```

① `card` — данные карточки, из которой выбираются данные секции.

② `Sections` — коллекция всех секций карточки.

③ `sectionId` — идентификатор секции, данные которой должны быть получены.

Идентификаторы секций карточек базовой поставки Docsvision приведены в разделе /dv6/schemas/dv6/[Описание полей стандартной карточки].

После получения данных секции можно:

- Получить все её строки (для иерархической секции будут возвращены строки только первого уровня):

```
RowDataCollection rows = section.Rows;
```

- Получить все строки без учета иерархии, т.е. для иерархической секции будут возвращены все строки:

```
RowDataCollection rows = section.GetAllRows();
```

- Найти строку, пользуясь клиентской фильтрацией, либо строки, используя поиск по секции.
- Получить строку секции по идентификатору строки:

```
RowData row = section.GetRow(rowId);
```

- Добавить новую строку, либо удалить существующую, воспользовавшись методами `CreateRow` и `DeleteRow`, соответственно.
- Выбрать первую строку — востребовано для плоской секции, которая содержит единственную строку:

```
RowData row = section.FirstRow;
```

Полный список методов и свойств класса секции см. в описании типа `SectionData`.

С большой долей вероятности, целью получения данных секции (`SectionData`) является доступ к полям её строк. В качестве примера, рассмотрим полный код, который выводит названия (поле `Name`) всех подпапок первого из *карточки папок*:

```
CardData card = userSession.CardManager.GetCardData(new Guid("DA86FABF-4DD7-4A86-B6FF-
C58C24D12DE2")); ①

SectionData section = card.Sections[new Guid("FE27631D-EEEA-4E2E-A04C-D4351282FB55")]; ②

RowData row = section.GetRow(new Guid("FFFFFFFF-FFFF-0003-FFFF-000000000000")); ③

foreach (var folderRow in row.ChildRows) ④
{

 Console.WriteLine(folderRow["Name"]); ⑤
}
```

① Получаем данные карточки папок (см. /dv6/schemas/dv6/[Описание полей стандартной карточки]).

② Выбираем секцию "Папки".

③ Выбираем строку "Папки". Данная строка содержит подстроки, отражающие папки одноименного раздела в дереве папок Windows-клиента.

④ Перебираем строки первого уровня. Коллекцию строк без учета иерархии можно получить из свойства `AllChildRows`.

⑤ Получаем значение конкретного поля `Name` (Название) строки и выводим его (значение) в консоль.

Таким образом, чтобы получить данные из секции карточки, достаточно выбрать из неё конкретную строку (либо перебирать их все) и получить значение поля по его псевдониму.

Объектная модель

Способ получения данных из секции карточки на этом уровне API зависит от способа представления этой секции в объектной модели карточки:

1. Для предоставления данных отдельной секции объектная модель карточки содержит свойство, возвращающее сущность секции, а точнее коллекцию сущностей её строк. Таким образом данные секции могут быть получены из свойства класса объектной модели карточки. Такой способ реализации является основным для объектной модели карточек библиотеки *Базовые объекты*.
2. Объектная модель карточки не содержит отдельных свойств, возвращающих сущность секции (как в первом случае), но класс объектной карточки унаследован от класса `BaseCard`. В этом случае коллекция строк секции может быть получена с помощью метода `GetSection`.

Метод `GetSection` не предназначен для работы с иерархическими секциями — при получении секции данного типа, будут возвращены строки только первого уровня иерархии. Для работы с иерархическими секциями потребуется самостоятельно разработать объектную модель карточки (первый способ получения данных секции), либо использовать базовый API.

Чтобы сравнить приведенные способы получения данных секции карточки, рассмотрим сценарий, в котором нужно получить названия всех основных

файлов карточки типа *Документ*.

В классе объектной модели карточки типа *Документ* — `Document` — объявлено свойство `Document.Files`, возвращающее коллекцию файлов документа (типа `DocumentFile`), из коллекции файлов может быть получено название.

Код, предоставляющий названия файлов, может быть следующим:

```
Document document = objectContext.GetObject<Document>(cardId); ①

ObjectCollection<DocumentFile> files = document.Files; ②

foreach (DocumentFile file in files)
{

 if (file.FileType = DocumentFileType.Main)
  Console.WriteLine(file.FileName); ③
}
```

① Получаем карточку типа *Документ* по идентификатору карточки.

② Получаем коллекцию всех файлов документа.

③ Выбираем основные файлы и выводим их название — свойство `FileName`.

Для реализации сценария вторым способом потребуется самостоятельно получить название из *Карточки файла с версиями*:

```
BaseCard card = objectContext.GetObject<BaseCard>(cardId); ①


var files = (IEnumerable<BaseCardSectionRow>)card.GetSection(new Guid("A6FA8BAF-2EA4-
4071-AA3E-5C4E71646A90")); ②

foreach (var file in files)
{
 if ((int)file["FileType"] = 0) ③
 {
  var versionedFileCard = userSession.CardManager.GetCard(file.GetGuid("FileId")) as
VersionedFileCard; ④
  Console.WriteLine(versionedFileCard.CurrentVersion.Name); ⑤
 }
}
```

① Получаем карточку (используется из первого примера)

Тип возвращаемого объекта не уточняется намеренно, для демонстрации

возможностей базового класса.

② Получение всех строк коллекционной секции `Файлы` карточки типа *Документ* по идентификатору секции.

③ Проверяем тип файла: `0` — основной, `1` — дополнительный.

④ Получаем карточку файла с версиями.

⑤ Выводим название файла.

То есть, чтобы получить данные из секций, если объектная модель секции не реализована, требуется:

1. Получить строки этой секции с помощью метода `BaseCard.GetSection`:

```
var files = (IEnumerable<BaseCardSectionRow>)card.GetSection(new Guid("A6FA8BAF-2EA4-4071-AA3E-5C4E71646A90"));
```

2. Получить значение нужного поля из строки:

```
int type = (int)file["FileType"]; ①

Guid fileId = file.GetGuid("FileId"); ②
```

① Способ получения значения поля по псевдониму с приведением к нужному типу.

② Способ получения значение поля, если заранее известно, что тип поля — `Guid`.

Чтобы получить родительский объект, например документ типа `Document`, если есть объектная модель файла этого документа типа `DocumentFile`, используйте метод `ObjectHelper.GetParent`:

```
DocumentFile documentFile;
Document document = ObjectHelper.GetParent<Document>(docFile);
```

Изменение данных карточки

Под изменением данных карточки обычно подразумевается изменение значения полей её секций, т.к. именно поля выступают основным хранилищем её данных. Последовательность доступа к полям карточки в этом случае абсолютно

аналогична приведенной в Получение данных из секции карточки.

Базовый API

Изменение данных карточки в обычном режиме:

```
CardData card = userSession.CardManager.GetCardData(cardId); ①

RowData sectionRow = card.Sections[sectionId].FirstRow; ②

sectionRow["Field"] = "Новое значение поля"; ③
```

① Получаем данные карточки с идентификатором `cardId`.

② В качестве примера получим первую строку секции с идентификатором `sectionId`.

③ Изменяем содержимое поля с псевдонимом `Field`.

Если необходимо изменить значения нескольких полей, то предварительно рекомендуется активировать режим отложенных изменений:

```
CardData card = userSession.CardManager.GetCardData(cardId);

card.BeginUpdate(); ①
RowData sectionRow = card.Sections[sectionId].FirstRow;

sectionRow["FieldFirst"] = "Новое значение первого поля"; ②
sectionRow["FieldSecond"] = "Новое значение второго поля"; ③

card.EndUpdate(); ④
```

① Включаем режим отложенных изменений.

② Изменяем значение поля с псевдонимом `FieldFirst`.

③ Изменяем значение поля с псевдонимом `FieldSecond`.

④ Фиксируем изменения в базе данных.

Если в сценарии изменения данных предполагается изменение одного поля, то активировать режим отложенных изменений не требуется

Объектная модель

На высоком уровне API сценарий изменения данных зависит от способа представления секции в объектной модели карточки.

Если сущность секции представлена в объектной модели карточки в виде свойства, то код изменяющий, к примеру, название карточки типа *Документ* будет следующим:

```
Document card = objectContext.GetObject<Document>(cardId); ①

card.MainInfo.Name = "Новое название документа";②

objectContext.SaveObject(card); ③
```

① Получаем карточку типа *Документ*.

② Изменяем значение поля.

③ Сохраняем изменения.

Если необходимо, вместо сохранения изменений одного объекта, можно сохранить все изменения контекста объектов:

```
objectContext.AcceptChanges();
```

Если объектная модель карточки не предоставляет непосредственного доступа к секции, можно воспользоваться методом `GetSection` базового класса `BaseCard`:

```
BaseCard baseCard = objectContext.GetObject<BaseCard>(cardId); ①

BaseCardSectionRow row = (BaseCardSectionRow)baseCard.GetSection(new Guid("30EB9B87-822B-
4753-9A50-A1825DCA1B74"))[0]; ②

row["Name"] = "Новое название документа"; ③

objectContext.SaveObject(row); ④
```

① Получаем карточку.

② Получаем первую строку секции.

③ Переданный идентификатор соответствует идентификатору секция `Основная информация` карточки типа *Документ*.

④ Изменяем значение поля.

Создание нового экземпляра карточки

Базовый API

Менеджер карточек базового API предоставляет метод `CreateCardData` для создания новых экземпляров карточек определённого типа. В качестве примера, новый экземпляр *карточки файла с версиями* может быть создан следующим образом:

```
CardData card = userSession.CardManager.CreateCardData(new Guid("6E39AD2B-E930-4D20-AAFA-C2ECF812C2B3")); ① ②
```

① `userSession` — пользовательская сессия.

② `6E39AD2B-E930-4D20-AAFA-C2ECF812C2B3` — идентификатор карточки типа *Карточка файла с версиями*.

Идентификатор типа карточки можно посмотреть в разделе /dv6/schemas/dv6/[Описание полей стандартной карточки], либо в схеме карточки, например, используя программу "CardManager" из комплекта "Resource Kit".

Если необходимо создать карточку с определённым идентификатором, можно воспользоваться соответствующей перегрузкой метода:

```
CardData card = userSession.CardManager.CreateCardData(new Guid("6E39AD2B-E930-4D20-AAFA-C2ECF812C2B3"), newCardId); ①
```

① `newCardId` — идентификатор создаваемой карточки.

После создания экземпляра карточки, если необходимо, её данные могут быть изменены по аналогии с примером, приведенным в разделе Изменение данных карточки. Возможность изменения определённых данных карточки, а также тип этих данных определяются типом карточки, описание которых приведено в разделе /dv6/schemas/dv6/[Описание полей стандартной карточки].

Объектная модель

API данного уровня не предлагает отдельного метода для создания нового экземпляра карточки. Данная задача распределена между преобразователем данных соответствующего типа карточки и сервисом для работы с этим типом карточек.

Полное описание *сервисов*, предоставляемых в рамках базовой

> поставки Docsvision, было приведено в разделе Сервисы объектной модели, также можно обращаться к описанию интерфейсов сервисов в разделе Библиотека классов.

Далее приведён пример использования методов для создания карточек типа документ и задание:

```
IDocumentService documentService = objectContext.GetService<IDocumentService>(); ①
ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Document document = documentService.CreateDocument(); ③

Task taks = taskService.CreateTask(); ④

objectContext.AcceptChanges(); ⑤
```

① Получаем из контекста объектов два сервиса: для работы с документами и для работы с заданиями.

② Метод `GetService` возвращает сервис по его интерфейсу.

③ Создаем экземпляр карточки типа *Документ*.

④ Создаем экземпляр карточки типа *Задание*.

⑤ Сохраняем все изменения контекста объектов.

Как видно из примера, типовой сценарий создания экземпляра карточки включает три шага:

1. Получение сервиса для работы с карточками определённого типа.
2. Вызов метода, отвечающего за создание карточки.
3. Сохранение изменений методами `ObjectContext.AcceptChanges` или `ObjectContext.SaveObject`.

Перед сохранением данных, если необходимо, можно изменить данные карточки.

Удаление объектов

Платформа Docsvision позволяет удалять не только карточки, но и отдельные строки их секций. Для этого каждый уровень API предлагает собственные методы.

Удаление объекта методами базового API

Для удаления экземпляра карточки на уровне базового API предназначен метод `DeleteCard` *менеджера карточек*, принимающий идентификатор удаляемой карточки:

```
userSession.CardManager.DeleteCard(cardId); ① ②
```

① `userSession` — пользовательская сессия.

② `cardId` — идентификатор удаляемой карточки.

Приведенный выше код удаляет карточку без возможности восстановления, при этом другая перегрузка метода позволяет пометить карточку как удалённую (в Windows-клиенте отображается курсивом) без фактического удаления:

```
userSession.CardManager.DeleteCard(cardId, false); ①
```

① Для возможности восстановления второй параметр должен иметь значение `false`.

Чтобы восстановить карточку, помеченную к удалению, используйте метод `RestoreCard`:

```
userSession.CardManager.RestoreCard(cardId);
```

Чтобы удалить строку из секции карточки, необходимо использовать метод `DeleteRow` сущности секции. В качестве примера, удалим первую строку из секции карточки:

```
CardData card = userSession.CardManager.GetCardData(cardId); ①

SectionData section = card.Sections[sectionId]; ②

Guid firstRowId = section.FirstRow.Id; ③

section.DeleteRow(firstRowId); ④
```

① Получаем карточку, из которой удаляется строка.

② Получаем сущность секции.

`sectionId` — идентификатор секции, из которой удаляется строка. Чтобы

получить идентификатор секции стандартных карточек, обратитесь к разделу /dv6/schemas/dv6/[Описание полей стандартной карточки].

③ Получаем идентификатор первой строки.

④ Выполняем удаление.

Обычно идентификатор удаляемой строки заранее известен, поэтому фактически удаление можно привести к виду:

```
card.Sections[sectionId].DeleteRow(rowId);
```

Удаление объекта на уровне объектной модели

Данный уровень API предлагает использовать для удаления объектов единый метод контекста объектов — `DeleteObject`. В качестве примера его применения, удалим один из файлов документа (карточка типа *Документ*):

```
Document document = objectContext.GetObject<Document>(cardId); ①

DocumentFile file = document.Files.First(); ②

objectContext.DeleteObject(file); ③
objectContext.AcceptChanges();
```

① Получаем карточку типа *Документ*.

② Для простоты, выбираем для удаления первый файл по списку.

③ Удаляем файл и сохраняем изменения.

Данный код фактически выполняет удаление строки из секции `Файлы` карточки *Документ*, по аналогии с рассмотренным выше примером для базового API. Аналогичным образом может быть удалён и сам документ.

Метод `DeleteObject` удаляет объект без возможности восстановления. Если требуется пометить карточку как удалённую, используйте базовый API.

Контекст объектов предлагает функциональность по отмене изменений объекта до его сохранения (вызова метода `AcceptChanges` или `SaveObject`), основанную на использовании методов `RollbackChanges` и `RollBackObject` (см. Режим отложенных изменений).

Блокировка и снятие блокировки

Чтобы предупредить возникновение конфликтных ситуаций, вытекающих из одновременной работы нескольких пользователей с одним объектом, платформа Docsvision предлагает использовать механизм блокировок, который защищает отдельный объект (карточку или строку) от изменений из сессий других пользователей, оставляя его доступным на чтение.

Система предлагает два варианта блокировки:

- Временная, ограниченная сроком существования пользовательской сессии, из которой блокировка была установлена. Такая блокировка снимается автоматически после завершения пользовательской сессии, если объект не был освобожден пользователем самостоятельно.
- Постоянная, действующая во время существования пользовательской сессии, из которой блокировка установлена, а также после её завершения. Данная блокировка может быть снята только явно — тем, кто её установил, либо администратором системы.

Базовый API

Базовый API предоставляет методы для установки блокировки на карточку, строку и файл.

На карточку может быть установлена как временная, так и постоянная блокировка:

```
CardData cardData = userSession.CardManager.GetCardData(cardId); ①

card.PlaceLock(); ②

card.PlaceLock(true); ③
```

① Получаем карточку.

② Устанавливаем временную блокировку.

③ Устанавливаем постоянную блокировку.

Аналогичным образом устанавливается временная или постоянная блокировка на строку секции:

```
SectionData section = card.Sections[sectionId] ①

section.FirstRow.PlaceLock() ②
```

① Получаем секцию, содержащую блокируемую строку.

② Для примера, устанавливаем временную блокировку на первую строку секции.

Таким же образом работает блокировка файлов (не путать с карточками файла с версиями и файлами документов):

```
FileData fileData = userSession.FileManager.GetFile(fileId); ①

fileData.PlaceLock(true); ②
```

① Получаем файл.

② Устанавливаем постоянную блокировку.

Помимо методов блокировки, классы карточек (`CardData`), строк (`RowData`) и файлов (`FileData`) содержат методы снятия блокировки, определяемые интерфейсом `ILockable`:

- `RemoveLock` — снимает блокировку с объекта, установленную в собственной сессии.
- `ForceUnlock` — принудительно снимает блокировку, в т.ч. установленную другим пользователем — необходимы административные права.

Также интерфейс `ILockable` определяет несколько свойств:

- `LockStatus` — состояние блокировки.
- `LockOwner` — учетная запись сотрудника, выполнившего блокировку.

Устанавливать блокировку на объект рекомендуется во всех сценариях, связанных с изменением данных, непосредственно после получения объекта. В сочетании с проверкой состояния блокировки код может быть следующим:

```
if (cardData.LockStatus = LockStatus.Free) ①
{

 cardData.PlaceLock(); ②
}else
{
 MessageBox.Show("Карточка заблокирована пользователем " + cardData.LockOwner);
}
```

① Проверяем наличие блокировки у карточки.

`cardData` — карточка, которая должна быть заблокирована.

② Установка временной блокировки.

Помимо методов, которые предлагают классы карточек, строк и файлов, базовый API предоставляет специальный объект для работы с блокировками — *менеджер блокировок*.

Менеджер блокировок можно получить из сессии пользователя:

```
LockManager lockManager = userSession.LockManager;
```

Менеджер блокировок предлагает два метода:

- `GetLockableObject` — возвращает интерфейс `ILockable` для карточки с определённым идентификатором.
- `GetLockedObjects` — предоставляет заблокированные объекты определённого типа, с возможностью получения объектов, заблокированных только текущим пользователем.

Объектная модель

API данного уровня предоставляет две точки доступа к механизму блокировки:

- Методы *контекста объектов*:
  - `ObjectContext.LockObject` — установка блокировки, в т.ч. постоянной.
  - `ObjectContext.UnlockObject` — снятие собственной блокировки.

Заблокировать карточку, используя методы контекста объектов, можно следующим образом:

```
BaseCard card = objectContext.GetObject<BaseCard>(cardId); ①

objectContext.LockObject(card, true); ②
```

① Получаем карточку (или строку).

② Устанавливаем постоянную блокировку.

Таким же образом блокируется строка карточки:

```
BaseCardSectionRow row = objectContext.GetObject<BaseCardSectionRow>(rowId); ①
objectContext.LockObject(row, true); ②
```

① `rowId` — идентификатор блокируемой строки.

② Методы сервиса для работы с блокировками — ILockService:

- `LockObject` — устанавливает на объект временную блокировку.

- `UnlockObject` — снимает с объекта временную блокировку.

Далее приведён пример, в котором документ (карточка типа *Документ*) блокируется для изменения, после чего блокировка снимается.

```
ILockService lockService = objectContext.GetService<ILockService>(); ①

Document document = objectContext.GetObject<Document>(cardId); ②

if (!(lockService.IsObjectLockedByAnotherUser(document)) ③
{
 lockService.LockObject(document); ④

 document.MainInfo.Name = "Новое название документа"; ⑤
 objectContext.SaveObject(document);

 lockService.UnlockObject(document); ⑥
}
```

① Получение сервиса для работы с блокировками.

② Получение изменяемого документа.

③ Проверяем, что объект не заблокирован.

④ Блокируем документ для изменения.

⑤ Вносим изменение. В качестве примера, изменяется название документа. Блокировка и снятие блокировки со строк выполняется аналогично.

Архивирование и извлечение карточки из архива

Чтобы убрать из списка карточки, работа с которыми завершена, их можно поместить в архив.

За архивацию карточек на уровне базового API отвечает метод `Archive`, который предоставляется классом карточки — `CardData`:

```
CardData cardData = userSession.CardManager.GetCardData(cardId); ①

cardData.Archive(ArchiveOptions.None); ②
```

① Получаем данные карточки.

② Помещаем карточку в архив.

Второй метод класса `CardData` позволяет извлечь карточку из архива:

```
cardData.Dearchive(ArchiveOptions.None)
```

Параметр, передаваемый в методы, определяет сценарий помещения и извлечения карточки из архива. Данный параметр может принимать значения:

- `ArchiveOptions.None` — без дополнительных параметров.
- `ArchiveOptions.Delay` — архивирование будет выполнено с задержкой.
- `ArchiveOptions.IncludeLinkedCards` — будут также архивированы карточки, связанные с архивируемой.
- `ArchiveOptions.IncludeLinkedFiles` — будут также архивированы файлы, связанные с архивируемой карточкой.

Объектная модель не предлагает методов для работы с архивом. При необходимости воспользуйтесь методами базового API.

Поиск

Возможности поисковых механизмов, предлагаемых базовым API и объектной моделью, значительно отличаются. Базовый API предлагает больше функциональности в части формирования, а также использования поисковых запросов, а объектная модель — простой и функциональный механизм поиска.

Поиск объектов с использованием базового API

Возможности поиска объектов методами базового API можно разделить на две принципиально разные группы:

Серверный поиск

Наиболее мощный, выполняющийся на сервере метод. Клиент подготавливает XML-строку, содержащую поисковый запрос в специальном формате, и передает её серверу Docsvision, который должен вернуть результат в виде строк, помещаемых затем в кэш.

Поисковый запрос может быть передан в виде готовой XML-строки, либо в виде объекта, сформированного методами `ObjectManager`.

Клиентская фильтрация

Простая фильтрации строк XML-документа карточки с использованием XPath-

выражений, предлагаемая `ObjectManager`.

Таким образом необходимо самостоятельно решить, какой метод поиска будет более приемлемым (учитывая производительность и простоту реализации) в конкретной ситуации.

Серверный поиск

Поисковый запрос для выполнения серверного поиска описывается строкой в формате XML. Он строится по специальной схеме (`Search.xsd`), которая описывает все возможные комбинации условий поиска.

Конкретный XML поискового запроса может быть получен, например при помощи визуального редактора (диалог *Расширенный поиск* в Windows-клиенте), где пользователь может оперировать семантически значимыми понятиями, такими как *тип карточки — секция — поле — значение*. Но результатом работы этого визуального редактора все равно в конечном счете станет XML-строка запроса, которая и будет передана на сервер для выполнения. Этот XML-текст может быть скопирован из диалога и использован в коде для выполнения аналогичных запросов.

Также текст поискового запроса может быть задан программно — например при помощи набора стандартных объектов для работы с XML (XML Parser). Однако, для упрощения программного создания и редактирования запросов, в объектной модели ObjectManager предусмотрен ряд специализированных объектов.

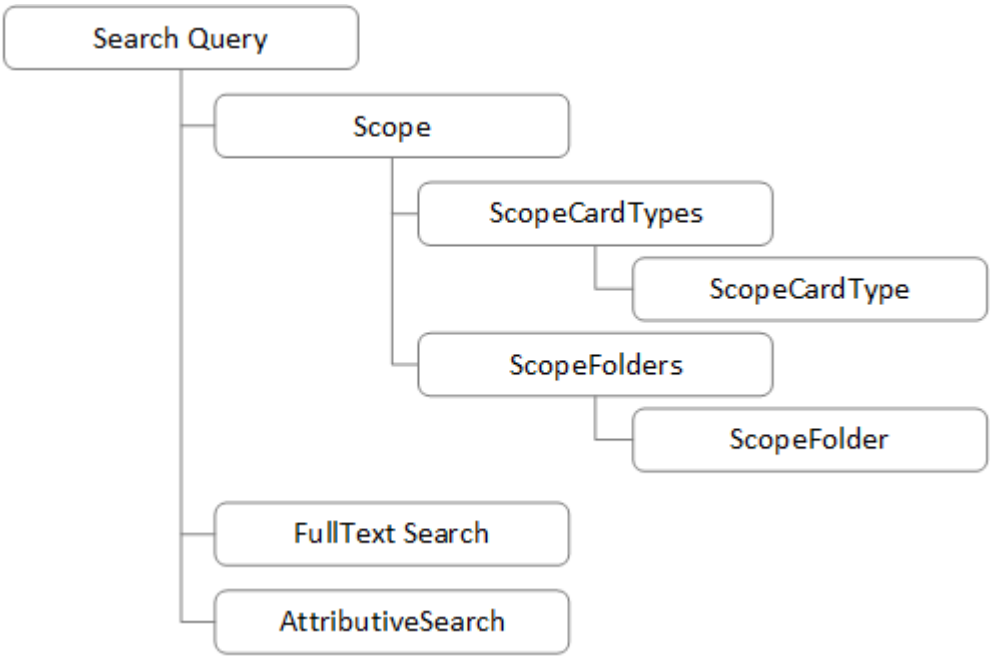


Рисунок 2. Структура запроса

Все объекты для работы с поиском расположены в выделенном пространстве имён `DocsVision.Platform.ObjectManager.SearchModel`, поэтому для их использования нужно подключить данное пространство имён к проекту.

Базовым объектом для построения поискового запроса является объект `SearchQuery`. Для создания этого объекта, предназначен специальный метод `CreateSearchQuery` пользовательской сессии (`UserSession`).

Сам запрос состоит из двух частей — атрибутивного поиска (`AttributiveSearch`) и полнотекстового запроса (`FullTextSearch`). Эти части независимы друг от друга и могут фигурировать в запросе как по отдельности, так и одновременно. Возможно объединение результатов этих частей по И/ИЛИ, для этого предназначен флаг **`CombineResults`**.

Количество карточек в результатах поиска можно ограничить свойством `Limit`.

Пространство карточек, на котором исполняется запрос, ограничивается объектом `Scope`. Это ограничение накладывается и на атрибутивную, и на полнотекстовую части запроса и может включать в себя ограничение по конкретным типам карточек (`CardTypes`) и по папкам, в которых расположены ярлыки карточек (`Folders`).

```
SearchQuery query = session.CreateSearchQuery();


query.Scope.CardTypes.AddNew(new Guid("C1FED883-08DE-420F-8FB4-C16CEFFC1630"));
query.Scope.CardTypes.AddNew(new Guid("FA0C389E-1095-4BC1-BEDC-793463742571")); ①

query.Scope.Folders.AddNew(new Guid("C0BCEB41-A19B-4813-A355-82EB6FD4F5F0"
)).IncludeSubfolders = true; ②
```

① Установка ограничения по типам карточек.

② Установка ограничения по папкам.

Отменить наложенные ограничения может свойство `AllCards` — при его установке поиск производится по всем карточкам, независимо от указанных ограничений.

Свойство `IncludeLinkedCard` позволяет включить в результаты поиска карточки, связанные с уже найденными по ссылкам на определённую глубину вложенности (глубина задаётся в качестве значения свойства). Дополнительно можно ограничить тип ссылок (`LinkTypes`) и их направление (`LinkDirections`).

Сходное по действию свойство `IncludeTopicCards` позволяет включить в результаты поиска карточки, принадлежащие к одной теме обработки с найденными.

Признак `IncludeArchived` сигнализирует о необходимости включать в область поиска также и все архивные карточки. Аналогичный признак `IncludeDeleted` позволяет искать также и в удалённых карточках.

Чтобы получить сформированную строку запроса в формате XML надо воспользоваться методом `GetXml()`. В этом методе параметр *ForSearch*=**true** означает, что строка формируется с целью отправки на сервер, и не должна содержать неактивные настройки (например, такие, как поисковые слова).

Значение *ForSearch*=**false** означает, что строка получается для сохранения где-либо и поэтому в неё включаются все настройки поиска. Второй параметр метода — `Parameters` — содержит массив значений, которые будут подставлены на место параметров, заданных в поиске.

Существует также обратный метод — `ParseXml`(**string***queryXml*). С его помощью можно проинициализировать объект `SearchQuery` из сохранённой строки XML-запроса.

Полнотекстовая часть поискового запроса (`FullTextSearch`) очень проста и позволяет задать только саму строку для поиска (`QueryString`) и режим поиска (`Mode`), чтобы ограничить поиск только файлами или всеми карточками. Поиск возможен только по карточкам, для которых в схеме разрешён поиск. В результате поиска по файлам будут возвращены карточки-владельцы найденных файлов, а не сами файлы.

Пример кода построения полнотекстового запроса:

```
SearchQuery query = session.CreateSearchQuery();

query.FullTextSearch.QueryString = "Договор на обслуживание";
query.FullTextSearch.Mode = FullTextSearchMode.Files; ①
```

① Условия полнотекстового поиска

Атрибутивная часть описывает условия на значения полей карточек, которые требуется найти. Она состоит из запросов к карточкам (`CardTypeQuery`). Если требуется найти все карточки, следует выставить в **true** параметр `AllCards`. Каждый запрос к типу карточки состоит из набора запросов к секциям карточки

(SectionQuery).

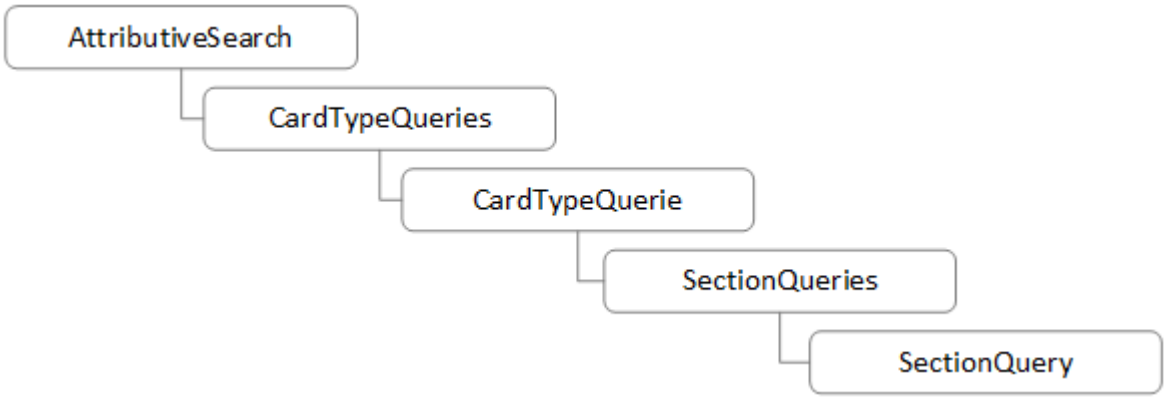


Рисунок 3. Структура атрибутивного поиска

Каждый запрос к секции состоит из набора условий (Condition), накладываемых на конкретные поля секции. Условия организуются в иерархические группы (ConditionGroup). Группа условий — это аналог скобок в выражении. Условия внутри ConditionGroup объединяются операцией (И/ИЛИ), задаваемой свойством Operation. Группы ConditionGroup могут быть вложенными.

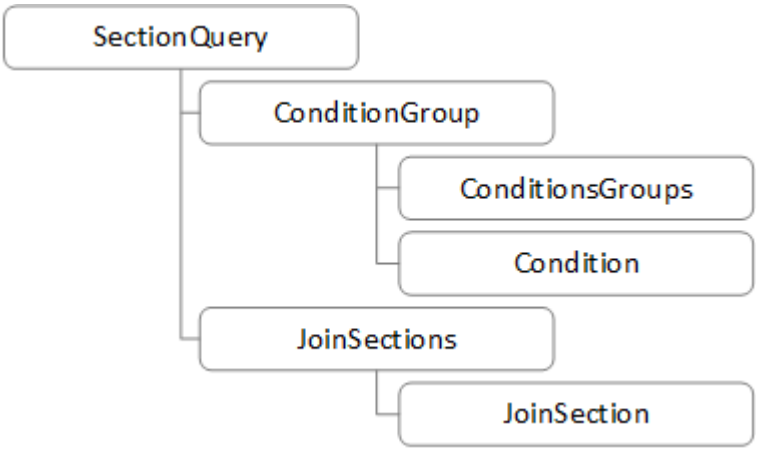


Рисунок 4. Структура поиска по секции

Сформированный поисковый запрос можно выполнить, вызвав метод CardManager.FindCards (в параметре метода передается XML-строка сформированного запроса). Метод возвращает CardDataCollection — коллекцию найденных карточек, соответствующих условиям запроса. Если же в результате выполнения запроса не было найдено ни одной карточки, коллекция будет пуста.

В качестве примера рассмотрим построение запроса, который вернёт все карточки типа "Документ", с названием Sample. Перед тем как строить запрос, необходимо выяснить идентификаторы типа карточки, который собираемся искать. Также необходимо узнать идентификаторы секций, в полях которых

производится поиск. Целесообразно оформить эти идентификаторы в виде констант:

```
const string ID_CARDTYPE = "{B9F7BFD7-7429-455E-A3F1-94FFB569C794}";
const string ID_SECTION = "{30EB9B87-822B-4753-9A50-A1825DCA1B74}";

SearchQuery searchQuery = userSession.CreateSearchQuery();

CardTypeQuery typeQuery = searchQuery.AttributiveSearch.CardTypeQueries.AddNew(new Guid
(ID_CARDTYPE)); ①

SectionQuery sectionQuery = typeQuery.SectionQueries.AddNew(new Guid(ID_SECTION)); ②

sectionQuery.ConditionGroup.Conditions.AddNew("Name", FieldType.Unistring,
ConditionOperation.Equals, "Sample"); ③

string query = searchQuery.GetXml(); ④

CardDataCollection coll = userSession.CardManager.FindCards(query); ⑤
```

① Поиск по типу карточки.

② Поиск по секции.

③ Поиск по значению поля.

④ Получение текста запроса.

⑤ Выполнение запроса.

В результате будет выполнен следующий запрос (см. содержимое переменной `query`):

```
<Search Version="4300" CombineResults="OR">
 <AttributiveSearch>
  <CardTypeQuery CardTypeID="{B9F7BFD7-7429-455E-A3F1-94FFB569C794}">
   <SectionQuery Version="4300" SectionTypeID="{30EB9B87-822B-4753-9A50-A1825DCA1B74}">
    <ConditionGroup Alias="alias0" Operation="OR">
     <Condition Alias="alias1">
      <Field FieldType="unistring">Name</Field>
      <Op>EQ</Op>
      <Value>'Sample'</Value>
     </Condition>
    </ConditionGroup>
    <Options Limit="-1" />
   </SectionQuery>
  </CardTypeQuery>
```

```
 </AttributiveSearch>
 <Scope />
 <FulltextSearch Mode="CardsAndFiles">
  <QueryString />
 </FulltextSearch>
</Search>
```

При построении условий к секциям можно также использовать присоединенные секции (JoinSections). Такая необходимость может возникнуть, если поле, по которому нужно искать, физически содержится в секции другой карточки. В этом случае к запросу по секции (SectionQuery) нужно присоединить секцию связанной карточки (JoinSections.AddNew). Для присоединённой секции (JoinSection) необходимо указать псевдоним (Alias), поле секции, по которому идёт соединение (SectionField), а также идентификатор (ID) или псевдоним (JoinWith) присоединяемой секции или системной таблицы (TableName) и имя её поля (WithField). А в самом условии (Condition) необходимо, помимо имени поля, задать также псевдоним секции (SectionAlias), которому это поле принадлежит.

Пример: основная секция карточки, по которой строится запрос, содержит поле Регистратор — представляющее собой ссылку (REF_ID) на запись о сотруднике (из справочника сотрудников) и нужно найти все карточки, у которых E-Mail регистратора содержит подстроку "mail.ru". Такие детали, как E-Mail сотрудника, хранятся только в соответствующей секции справочника сотрудников — поэтому её необходимо присоединить при построении такого запроса:

```
const string REFSTAFF_EMPLOYEES = "{DBC8AE9D-C1D2-4D5E-978B-339D22B32482}"; ①

JoinSection join = sectionQuery.JoinSections.AddNew("RegisteredBy_Info"); ②
join.Id = new Guid (REFSTAFF_EMPLOYEES);
join.SectionField = "RowID";
join.WithField = "RegisteredBy";

Condition condition = sectionQuery.ConditionGroup.Conditions.AddNew("Email", FieldType
.Unistring, ConditionOperation.EndsWith, "mail.ru"); ③
condition.SectionAlias = "RegisteredBy_Info";
```

① Идентификатор секции справочника сотрудников.

② Присоединение секции справочника сотрудников.

③ Добавление условия на поле присоединенной секции.

Таким образом, при построении запросов можно использовать следующий алгоритм:

1. Создать новый объект запроса (`SectionQuery`).
2. При необходимости установить ограничения области поиска (`Scope`) на типы карточек (`CardTypes`) и папки (`Folders`).
3. Если запрос полнотекстовый, задать строку поиска (`FullTextSearch.QueryString`).
4. Если запрос содержит атрибутивные условия:
    i. Определить тип карточки, которая должна являться результатом поиска (`CardTypeQuery`).
    ii. Определить, по какой секции карточки строится запрос (`SectionQuery`). Если поле, по которому производится поиск, физически находится в секции другой (связанной) карточки, то присоединить её (`JoinSections`).
    iii. Сформировать как минимум одну группу условий (`ConditionGroup`) и определить операцию для условий (`Operation`).
    iv. Добавить в группу условия (`Condition`), для каждого из которых определить следующее:

        - Имя поля, по которому производится сравнение (`FieldAlias`) и его тип (`FieldType`), а если поле принадлежит связанной секции — то имя этой секции (`SectionAlias`).
        - Условие (`Operation`).
        - Значение для сравнения (`Value`).

        Если условие использует агрегацию:

        - Задать имя функции (`AggregateFunctionName`).
        - При необходимости задать дополнительные условия на агрегируемые значения (AggregateConditionGroup).
5. Выполнить запрос (`CardManager.FindCards`).

Помимо поиска экземпляров карточек, объектная модель может использоваться и для поиска строк в конкретной секции карточки. Для этого предназначен метод `SectionData.FindRows`, который возвращает коллекцию строк (`RowDataCollection`), соответствующих условиям запроса, или пустую коллекцию, если ни одна строка не подходит.

Текст запроса для поиска строк соответствует аналогичному запросу для поиска карточек, но в сокращённом варианте. Он начинается сразу с уровня объекта

SectionQuery, который также является создаваемым при помощи соответствующего метода объекта UserSession:

```
SectionQuery secQuery = session.CreateSectionQuery();
```

Для поиска строк секций могут использоваться только атрибутивные, а не полнотекстовые, условия.

При построении поисковых запросов внутри секции можно пользоваться всеми возможностями поисковой модели — присоединенными секциями, агрегациями, и т.д. Для этого используются те же самые объекты.

В приведенном далее примере выполняется поиск всех строк в справочнике сотрудников с фамилией Иванов:

```
const string REFSTAFF_CARDTYPE = "6710B92A-E148-4363-8A6F-1AA0EB18936C";
const string REFSTAFF_EMPLOYEES = "DBC8AE9D-C1D2-4D5E-978B-339D22B32482";

CardData staffData = session.CardManager.GetDictionaryData(new Guid(REFSTAFF_CARDTYPE));
①

SectionData section = staffData.Sections[new Guid(REFSTAFF_EMPLOYEES)]; ②

SectionQuery query = session.CreateSectionQuery(); ③

query.ConditionGroup.Conditions.AddNew("LastName", FieldType.Unistring,
ConditionOperation.Equals, "Иванов"); ④

RowDataCollection results = section.FindRows(query.GetXml()); ⑤
```

① Получение данных справочника.

② Получение секции сотрудников.

③ Создание поискового запроса по секции.

④ Условие по полю фамилия.

⑤ Выполнение запроса.

Клиентская фильтрация

Клиентская фильтрация позволяет осуществить выборку строк с использованием XPath-выражений. Её имеет смысл применять в тех случаях, когда данные уже загружены на клиента и выполнять серверный поиск нет необходимости.

Фильтрация поддерживается для данных секций (`SectionData` и `SubSectionData`), а также для произвольных коллекций строк (`RowDataCollection`).

Выражения для поиска строятся стандартным для XPath способом (описание синтаксиса XPath-запросов можно найти в оригинальном стандарте, в MSDN или в специализированных изданиях).

В качестве элементов XPath-запроса могут выступать поля секции, по которой осуществляется поиск (как собственные поля секции, так и связанные поля секций других карточек).

Полный перечень этих полей для стандартных типов карточек можно посмотреть:

- При помощи утилиты "CardManager", входящей в Resource Kit, открыв схему соответствующей карточки (схемы всех базовых типов карточек входят в состав пакета разработчика);
- В разделе /dv6/schemas/dv6/[Описание полей стандартных карточек].

Другим простым способом, который может помочь в построении XPath-запроса, является получение непосредственно данных секции в формате XML на примере любой карточки интересующего типа (при помощи команды *Экспорт и печать* в Windows-клиенте). Используя эти данные как образец, можно строить запросы непосредственно на базе этого XML-документа.

Клиентская фильтрация возвращает только один результат (строку). Это всегда первая по порядку строка, удовлетворяющая условиям поиска (если таким условиям удовлетворяет несколько строк).

Пример поиска сотрудника в справочнике сотрудников по фамилии:

```
const string ID_CARDTYPE = "{6710B92A-E148-4363-8A6F-1AA0EB18936C}";
const string ID_SECTION  = "{DBC8AE9D-C1D2-4D5E-978B-339D22B32482}";

string name = "Иванов"; ①

CardData card = session.CardManager.GetDictionaryData(new Guid(ID_CARDTYPE));
SectionData section = card.Sections[new Guid(ID_SECTION)];
RowData row = section.FindRow("@SurName=\"" + strName + "\""); ②
```

① Фамилия сотрудника.

② Клиентская фильтрация.

Для поиска строк в коллекции (RowDataCollection) параметры и способ использования метода идентичны соответствующему методу для секций. Единственное отличие заключается в том, что метод Filter может возвращать более чем один результат (RowDataCollection).

Сохранение поискового запроса

Поисковый запрос, созданный вручную или с использованием объектной модели, может быть сохранен в системе для дальнейшего использования (например, в виртуальных папках).

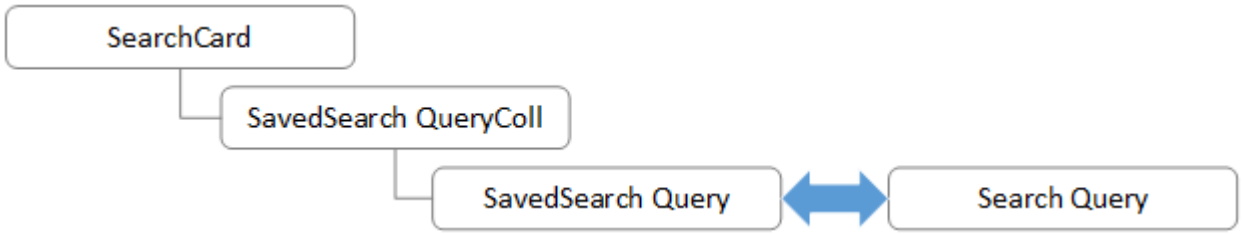


Рисунок 5. сохранённый запрос

Для хранения сформированных запросов в системе предусмотрена специальная карточка — карточка сохранённых поисковых запросов (SavedSearchCard), расположенная в системной библиотеке карточек (DocsVision.Platform.ObjectManager.SystemCards). Чтобы избежать необходимости оперировать с данными этой карточки напрямую, в объектной модели реализован соответствующий объект — SearchCard. Данная карточка является справочником, и поэтому может быть получена по идентификатору типа:

```
const string SEARCH_CARD_TYPE = "{05E4BE46-6304-42A7-A780-FD07F7541AF0}";
SearchCard searchCard = session.CardManager.GetDictionary(new Guid(SEARCH_CARD_TYPE)) as
SearchCard;
```

Пример создания нового поискового запроса и сохранения его в карточке:

```
SearchQuery query = session.CreateSearchQuery(); ①

const string SEARCH_CARD_TYPE = "{05E4BE46-6304-42A7-A780-FD07F7541AF0}";
SearchCard searchCard = (SearchCard)session.CardManager.GetDictionary(new Guid
(SEARCH_CARD_TYPE)); ②

searchCard.Queries.AddNew("Мой новый запрос").Import(oQuery); ③
```

① Формирование поискового запроса.

② Получение карточки сохранённых запросов.

③ Сохранение запроса в карточку.

Возможности поиска в контексте объектов

Контекст объектов предоставляет достаточно простой и функциональный механизм поисковых запросов — метод `FindObject` контекста объектов, принимающий тип искомого объекта.

```
BaseCard foundCard = objectContext.FindObject<BaseCard>(query); ① ② ③
```

① `objectContext` — контекст объектов.

② `BaseCard` — тип возвращаемого объекта. В качестве типа возвращаемого объекта может быть передан тип объектной модели карточки или строки.

③ `query` — поисковый запрос типа `QueryObject`.

Чтобы найти в *Справочнике сотрудников* сотрудника с определённой учетной записью, воспользуйтесь примером, приведенным далее, в котором параметры поискового запроса передаются непосредственно в конструктор класса `QueryObject`.

```
QueryObject query = new QueryObject(StaffEmployee.AccountNameProperty.Name,
"MYDOMAIN\\KukoffSA"); ①

StaffEmployee employee = objectContext.FindObject<StaffEmployee>(query); ②
```

① Формируем поисковый запрос по полю `AccountName`.

`StaffEmployee.AccountNameProperty.Name` — возвращает название поля, по которому выполняется поиск.

`"MYDOMAIN\\KukoffSA"` — искомое значение.

② Получаем сотрудника с искомой учетной записью.
`StaffEmployee` — класс (объектной модели) искомого объекта.

Также возможны более сложные поисковые запросы, которые включают несколько критериев поиска. К примеру, далее приведён код запроса, выполняющего поиск контрагента по названию и типу (организация или подразделение).

```
QueryObject query = new QueryObject();
```

```
query.AddCriterion(PartnersCompany.NameProperty.Name, "ООО Берег"); ①

query.AddCriterion(PartnersCompany.TypeProperty.Name, (int)PartnersCompanyType
.Organization, CompareOperation.Equals); ②

query.Operation = QueryOperation.And; ③

PartnersCompany partnersCompany = objectContext.FindObject<PartnersCompany>(query); ④
```

① Добавляем критерий поиска по названию контрагента.

② Ограничиваем поиск только среди организаций.

③ Определяем логическую операцию И, применяемую к результатам поиска.

④ Получаем контрагента, соответствующего критериям.

Приведенные выше примеры используют метод `FindObject`, который предназначен для возврата единственного результата. Если в результате поиска будет найдено более одного объекта, будет возвращена ошибка. Данный метод следует использовать только в случае, когда точно известно, что будет возвращен максимум один объект. Если ожидается возвращение нескольких объектов, используйте метод `FindObjects`, который возвращает коллекцию объектов. Метод вызывается аналогично первому.

Экспорт, импорт и подготовка карточек к печати

Функции экспорта и импорта доступны только в базовом API.

Экспорт

Платформа позволяет экспортировать любые данные (карточки, секции, наборы строк и отдельные строки) в формат XML, а также импортировать их обратно. С этой функциональностью неразрывно связана также и задача печати данных — так как проще всего реализовать её через специальный формат экспорта.

Для получения XML-представления данных карточки можно использовать метод SaveXml объектов `CardData`, `SectionData`, `SubSectionData`, и `RowData`, а также коллекций этих объектов.

```
FileStream file = new FileStream("..\\MyCard.xml", FileMode.Create);
card.SaveXml(file, ExportFlags.LinkedCards & ExportFlags.LinkedRows);
```

Для управления логикой экспорта можно создать собственный инспектор на

основе базового класса `DocsVision.Platform.ObjectManager.ExportCardInspector`.

Импорт

Позволяет загрузить в систему данные на основании строки (или файла) в формате XML, ранее выгруженного из этой же (или другой) базы данных Docsvision.

Корректность импорта не гарантируется, если версия базы данных, из которой данные были экспортированы, отличается от версии базы, куда они импортируются.

Для импорта предназначен метод `ImportCards` объекта `CardManager`. Импортируемый XML должен соответствовать стандартной схеме карточек Docsvision. Проверка на соответствие этой схеме обязательно выполняется перед началом собственно загрузки данных, и в случае несоответствия менеджер объектов вернет ошибку. Таким образом, передавать на вход метода произвольный XML-документ бессмысленно.

Для управления логикой работы импорта предназначен специальный тип объектов — `DocsVision.Platform.ObjectManager.ImportCardInspector`. Этот объект полностью реализует процесс импорта, осуществляет контроль целостности данных и выполняет разрешение конфликтов.

Карточки, содержащие отсылки к версии компонентов, не рекомендуется напрямую импортировать в версию 6.1 и выше, если они выгружены из версии 5.5.

Необходимость наличия сложного механизма разрешения конфликтов при импорте обусловлена многообразием возможных конфигураций системы. Например, даже при импорте одной отдельно взятой карточки может возникнуть ситуация, когда такая карточка с таким идентификатором уже существует в базе. Это сразу порождает несколько возможных вариантов дальнейшего поведения импортера:

- Заменить полностью существующую в базе карточку на новую.
- Объединить данные существующей карточки с данными импортируемой.
- Создать новый экземпляр карточки (с другим идентификатором) и наполнить его импортируемыми данными.

- Прервать процесс импорта.
- Другие варианты.

Однако даже объект-инспектор не всегда может гарантировать корректность импорта, так как некоторые ограничения активируются только при отправке уже импортированных данных на сервер.

К числу таких ограничений относятся:

- Права доступа к объектам.
- Ограничения ссылочной целостности (foreign key constraints).
- Ограничения уникальности (unique constraints).

Если импортируемые данные нарушают одно их этих ограничений, то ошибка возникнет уже на сервере при попытке сохранить импортированные данные в БД. В этом случае импорт будет отменен полностью в рамках одной транзакции.

По умолчанию в *менеджере объектов* реализовано 2 типа объектов-инспекторов, обладающих разным поведением:

- `ImportCardInspectorCopier` — создаёт копию карточки (с другими идентификаторами всех строк), используя данные из XML в качестве шаблона.
- `ImportCardInspectorReplacer` — заменяет содержимое импортируемой карточки. Существующие строки будут заменены, отсутствующие строки буду добавлены в существующий экземпляр.

Объект-инспектор нужно создавать явно, передав в его конструктор ссылку на сессию:

```
FileStream file = new FileStream("myCard.xml", FileMode.Open);
ImportCardInspectorReplacer inspector = new ImportCardInspectorReplacer(session);
CardDataCollection cards = session.CardManager.ImportCards(file, inspector);
```

Если существующих типов инспекторов недостаточно для реализации необходимой логики импорта, можно разработать собственный объект-инспектор. Для этого нужно создать новый класс, унаследовав его от базового `DocsVision.Platform.ObjectManager.ImportCardInspector`, и переопределить методы этого класса, отвечающие за импорт отдельных структур.

Печать

Печать данных карточки может быть выполнена как стандартными средствами системы, так и любым другим способом на усмотрение разработчика. При использовании собственных методов печати целесообразно получить данные карточки в формате XML, а затем преобразовать их в формат, удобный для печати (либо экспортировать во внешнюю систему, осуществляющую печать).

Использование стандартных средств предполагает применение к данным карточки XSLT- или InfoPath-преобразования, описание которого задаётся на этапе проектирования схемы данных карточки (см. Раздел "Transformations").

Получить преобразования для конкретного типа карточки можно из мета-данных:

```
CardType type = session.CardManager.CardTypes[new Guid("идентфикатор_типа")];
CardTransformationCollection trans = type.Transformations;
```

После применения XSLT-преобразования данные карточки принимают формат XML или HTML-документа, который впоследствии может быть распечатан. Для применения XSLT-преобразования можно воспользоваться стандартными методами XML-парсера:

```
CardData card = session.CardManager.GetCardData(cardId); ①
MemoryStream stream = new MemoryStream();
card.SaveXml(stream);
stream.Position = 0;
XmlDocument cardXML = new XmlDocument();
cardXML.Load(stream);

CardType type = session.CardManager.CardTypes[new Guid("идентфикатор_типа")]; ②
CardTransformation trans = type.Transformations[0];

XmlDocument transformXML = new XmlDocument();
transformXML.LoadXml(trans.Data.ToString());
XslCompiledTransform XSLTransform = new XslCompiledTransform();
XSLTransform.Load((IXPathNavigable)transformXML); ③

System.Text.StringBuilder builder = new System.Text.StringBuilder();
XmlWriter html = XmlWriter.Create(builder); ④
XSLTransform.Transform((IXPathNavigable)cardXML, html);
```

① Получаем данные карточки получение XML карточки.

② Получаем шаблон печати.

③ Формируем трансформацию.

④ Применяем преобразование.

Преобразованный документ может быть передан для печати во внешнее приложение (например, Microsoft Internet Explorer или Microsoft Word), либо в элемент управления WebBrowser, если создаётся .NET-приложение.

Подписание и шифрование

В платформе Docsvision реализована функциональность по защите и контролю целостности информации, за счет подписания и шифрования объектов. На обоих уровнях API эти методы защиты представлены в той или иной мере.

Возможность работы с зашифрованными данными карточек (секций, строк) в платформе не поддерживается.

Базовый API

На уровне базового API реализация механизма подписания и шифрования обеспечивает возможность создания, чтения и удаления специального типа объектов — крипто-объектов. Данные объекты могут быть двух типов, перечисленных в `DocsVision.Platform.ObjectManager.CryptObjectType`:

- `CryptObjectType.Signature` — подпись.
- `CryptObjectType.Key` — публичный ключ шифрования.

Крипто-объекты не являются зашифрованными или подписанными версиями самих объектов, а лишь хранят соответствующие ключи.

Крипто-объекты сохраняются в специальной системной таблице (**dvsys_crypto**) и привязаны напрямую конкретному подписанному или зашифрованному объекту системы.

Платформа оперирует уже готовыми крипто-объектами, а выполнение самих операций подписи и шифрования при этом должно выполняться другими средствами, например при помощи библиотеки `System.Security.Cryptography` для .NET Framework.

Для работы с крипто-объектами в объектной модели предусмотрен ряд методов, предоставляемых объектом `AccessManager`:

- `StoreCryptObject` — сохранение новой подписи или ключа шифрования.
- `GetCryptObjects` — получает список крипто-объектов для заданного файла.
- `GetCryptObject` — возвращает данные подписи или зашифрованного объекта по его собственному идентификатору.
- `GetCryptObjectInfo` — получает дополнительную информацию о крипто-объекте с указанным идентификатором.
- `DeleteCryptObject` — удаляет крипто-объект по его собственному идентификатору.
- `DeleteCryptObjects` — удаляет все крипто-объекты заданного типа для конкретного файла.
- `GetKey` — получает ключ шифрования заданного объекта.
- `StoreKey` — сохраняет ключ шифрования.
- `StoreSignature` — сохраняет ключ ЭП.

Для файлов, подпись и шифрование которых реализованы в самой платформе, время жизни связанных крипто-объектов контролируется автоматически. То есть, например, при удалении файла все наложенные на него подписи также будут автоматически удалены. Но при создании крипто-объектов собственного типа (не относящихся к файлам) разработчик должен сам контролировать время их жизни и заботиться об удалении.

Рассмотрим пример получения всех подписей.

```
FileData file = session.FileManager.GetFile(new Guid("идентификатор_файла"));
InfoRowCollection coll = session.AccessManager.GetCryptObjects(file.Id, CryptObjectType
.Signature);

foreach (InfoRow cryptrow in coll)
{
   MessageBox.Show("Файл подписан " + cryptrow["CreationDate"].ToString() + "
пользователем " + cryptrow["AccountName"].ToString());
}
```

При выполнении операций подписи или шифрования можно использовать текущие системные настройки криптографии, заданные в справочнике настроек (к их числу относятся крипто-

провайдер, алгоритм, длина ключа и т.д.).

Объектная модель

На данном уровне API подписание и шифрование реализовано для карточек типа *Документ* и *Задание*. Ниже представлена сводная таблица методов для работы с данной функциональностью.

Метод	Назначение
`IDocumentService.AddSignature(Document, X509Certificate2, Boolean, ICollection<CardFieldSetting>)`	Данный метод используется для инициализации и добавления подписи в список подписей карточки *Документ*, и устанавливает ЭП на поля карточки, а также основные и дополнительные файлы.
`IDocumentService.AddSignatures(Document, SignedCms)`	Данный метод используется для добавления коллекции подписей в список подписей карточки *Документ*. Позволяет использовать при подписании криптографическое сообщение вместо сертификата. Непосредственно выполняет установку ЭП на поля карточки, основные и дополнительные файлы.
`IDocumentService.CheckActualSignature(Document, ICollection<CardFieldSetting>, BaseCardSignature)`	Данный метод позволяет определить актуальность подписи по отношению к данным карточки. Например, путем сравнения версии основного файла с версией, приведенной в подписи.

Метод	Назначение
`IDocumentService.DecodeSignedDocument(String, Document, SignedCms)`	Метод DecodeSignedDocument позволяет получить из файла `.p7s` сообщение CMS/PKCS 7, используемое в других методах (например, `IDocumentService.AddSignatures`). `
`IDocumentService.VerifySignature(Document, BaseCardSignature, X509Certificate2, Boolean, Boolean, ICollection<CardFieldSetting>)` `	Данный метод используется для проверки ЭП, установленной на полях и файлах карточки *Документ*.
`ICryptService`	Сервис `ICryptService` предоставляет методы шифрования и дешифровки файлов.
`ITaskService.AddSignature(Task, X509Certificate2, StatesOperation, Guid, IEnumerable<CardFieldSetting>)`	Данный метод используется при подписании карточки *Задание*.

Метод	Назначение
`ITaskService.AddSignature(Task, X509Certificate2, StatesOperation, Guid, IEnumerable<CardFieldSetting>, IEnumerable<Document>)`	Данный метод используется при подписании карточки *Задание* и основного документа.
`ITaskService.VerifySignature(Task, BaseCardSignature, X509Certificate2, Boolean, IEnumerable<CardFieldSetting>)`	Данный метод используется для проверки ЭП, установленной на полях и файлах карточки *Задание*.

Библиотека карточек "Базовые объекты"

Объектная модель платформы предоставляемая в рамках базовой поставки Docsvision, включает коллекцию *сервисов* и сущностей, которые реализуют объектные модели и *преобразователи данных* для работы с карточками библиотеки *Базовые объекты*. В данном разделе собрана основная информация, необходимая для работы с карточками данной библиотеки карточек.

В этом разделе:

- Бизнес-календарь
- Документ
- Группа заданий

- Задание
- Конструктор ролей
- Конструктор состояний
- Конструктор справочников
- Список ссылок на карточки заданий
- Справочник видов карточек
- Справочник контрагентов
- Справочник сотрудников

Бизнес-календарь

Назначение	Предназначена для хранения информации о расписании работы сотрудников (количество рабочих дней, рабочих часов и др.) на определённый период времени.
Класс карточки	`BackOffice-ObjectModel-Calendar:Calendar_CL.adoc[DocsVision.BackOffice.ObjectModel.Calendar]`

Для работы с карточкой предназначен сервис `ICalendarService`. Сервис предоставляет следующие операции:

- Добавление рабочих дат в бизнес-календарь.
- Добавление рабочего время.
- Получение сроков рабочего времени для различных условий.

В примере создаётся бизнес-календарь в котором все пятницы 2014 года объявляются сокращенными (рабочее время с 10 до 15 часов). По созданному календарю будет работать текущий пользователь. Дополнительных проверок не проводим.

```
ICalendarService calendarService = objectContext.GetService<ICalendarService>();
IStaffService staffService = objectContext.GetService<IStaffService>(); ①

CardData cardData = userSession.CardManager.CreateCardData(DocsVision.BackOffice.CardLib
.CardDefs.CardCalendar.ID); ②
```

```
DocsVision.BackOffice.ObjectModel.Calendar calendar = objectContext.GetObject<DocsVision
.BackOffice.ObjectModel.Calendar>(cardData.Id); ③
calendar.MainInfo.Name = "bestWorks";
calendar.Description = "Легкий день";
int year = 2014;

CalendarYear calendarYear = calendarService.AddCalendarYear(year); ④

calendarYear.Days = new ObjectCollection<CalendarDay>();
DateTime startDate = new DateTime(year, 1, 1); ⑤

for (DateTime i = startDate; i <= new DateTime(year, 12, 31); i = i.AddDays(1))
{
 if (i.DayOfWeek = DayOfWeek.Friday)
 { ⑥
  CalendarWorkTime calendarWorkTime = calendarService.AddCalendarWorkTime(new System
.DateTime(2000, 1, 1, 10, 0, 0), new System.DateTime(2000, 1, 1, 15, 0, 0)); ⑦

  CalendarDay calendarDay = calendarService.AddCalendarDay(i.DayOfYear);
  calendarDay.WorkTime = new ObjectCollection<CalendarWorkTime>();
  calendarDay.WorkTime.Add(calendarWorkTime); ⑧

  calendarYear.Days.Add(calendarDay); ⑨
 }
}

calendar.Years.Add(calendarYear); ⑩

staffService.GetCurrentEmployee().CalendarID = cardData.Id; ⑪

objectContext.AcceptChanges(); ⑫
```

① Инициализация контекста объектов и получение сервисов.

② Инициализация карточки типа Бизнес-календарь (использует `CardManager`, старый API).

③ Доступ к карточке через объектную модель.

④ Создание объекта типа "Год".

⑤ Инициализация коллекции дней в году.

⑥ Заполнение дней.

⑦ Создание объекта содержащего укороченное рабочее время.

⑧ Создание объекта типа "День" и добавление в него рабочего времени.

⑨ Добавление дня в год.

⑩ Сохранение года в календаре.

⑪ Присвоение календаря текущему пользователю.

⑫ Сохранение всех изменений.

Документ

Назначение	Карточки данного типа предназначены для работы с файлами: хранения файлов в различных форматах, предварительный просмотр, учет версий, экспорта в различные форматы, наложение электронных подписей в формате `.p7s`.
Класс карточки	`DocsVision.BackOffice.ObjectModel.Document`

Для работы с карточкой предназначен сервис `IDocumentService`, с помощью которого можно выполнять различные операции с документом, начиная с создания документа, и заканчивая работой с файлами и подписями документа.

Доступные операции:

- Создание документа.
- Синхронизация свойств с файлом.
- Управление подписями документа.
- Работа с файлами документа.
- Экспорт документа.

Далее перечислены базовые сценарии работы с объектной моделью карточки "Документ", для которых действительно следующее:

- Контекст объектов может быть получен в соответствии с примером, приведенным в разделе Инициализация контекста объектов.
- Сервисы получаем стандартным способом:

```
IDocumentService documentService = objectContext.GetService<IDocumentService>(); ①
ILinkService linkService = objectContext.GetService<ILinkService>(); ②
IReferenceListService referenceListService = objectContext.GetService
<IReferenceListService>(); ③
IStaffService staffService = objectContext.GetService<IStaffService>(); ④
```

```
IStateService stateService = objectContext.GetService<IStateService>(); ⑤
```

① Сервис для работы с документами.

② Сервис для работы со "Справочником ссылок".

③ Сервис для работы со списками ссылок.

④ Сервис для работы со "Справочником сотрудников".

⑤ Сервис для работы с "Конструктором состояний".

- Воспользуйтесь поиском, чтобы получить информацию по классам и переменным, используемым в примерах.

Создание нового Документа

```
static void CreateDocument()
{

 KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("8E40F327-9517-
4A43-998D-BF2BD619588D")); ①


 Document document = documentService.CreateDocument(null, kind);
 document.Description = "Дайджест карточки Документ"; ②
 document.MainInfo.Name = "Название документа"; ③
 document.MainInfo.Author = document.MainInfo.Registrar = staffService.
GetCurrentEmployee(); ④
 document.MainInfo["Content"] = "Содержимое документа"; ⑤
 objectContext.SaveObject(document);
}
```

① Получение вида карточки Исходящий.

② Создание документа.

③ Отображается в сетке Windows-клиента.

④ Отображается при открытии карточки.

⑤ Регистратор/подготовил, а также автор.

Добавление основного файла в Документ

```
static void AddFile()
{
 string filePath = @"z:\Sample.docx"; ①
```

```
 Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-000000000000")); ②

 documentService.AttachMainFile(document, filePath);
 objectContext.SaveObject<Document>(document); ③
}
```

① Прикрепляемый документ.

② Получение документа, к которому добавляется файл.

③ Добавление основного документа.

Добавление в "Документ" ссылки на другую карточку

```
static void AddLink()
{
 Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-000000000000"));
 BaseCard addedCard = objectContext.GetObject<BaseCard>(new Guid("00000000-0000-0000-0000-000000000001")); ①

 LinksLinkType linkType = linkService.FindLink("В ответ на"); ②

 referenceListService.CreateReference(document.MainInfo.ReferenceList, linkType, addedCard, "Ссылка на документ", false); ③
}
```

① Получение документа, в который добавляется ссылка на карточку.

② Получение типа ссылки.

③ Добавление ссылки в список ссылок документа document.

Смена состояния Документа

```
static void ChangeDocumentState()
{
 Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-000000000000")); ①

 StatesState endState = stateService.GetStates(document.SystemInfo.CardKind).First(t => t.DefaultName.Equals("Is signed")); ②

 stateService.ChangeState(document, endState); ③
}
```

① Получение документа.

② Получение конечного состояния "Документа". В данном случае — Подписан.

③ Смена состояния.

Подписание "Документа" и основных файлов

```
static void SignDocument()
{

 Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ①


 ICollection<CardFieldSetting> fields = documentService.GetKindSettings(document
.SystemInfo.CardKind).DocumentSignature.Fields; ②


 documentService.AddSignature(document, GetCertificate(), false, fields); ③
 objectContext.SaveObject(document);
}
```

① Получение подписываемого документа.

② Получение списка полей, которые должны быть подписаны. Настройки получаем из "Справочника видов карточек".

③ Подписание документа. Будут подписаны только основные файлы и отдельные поля карточки. Дополнительные файлы подписаны не будут.

Группа заданий

Назначение	Предназначена для создания детализированных распоряжений по документу, разбитых на отдельные подзадачи. Каждая подзадача предназначена для определённого исполнителя или группы исполнителей, снабжена описанием работ, сроками их исполнения и другими параметрами, а также предусматривает необходимый контроль за ходом его реализации.
Класс карточки	DocsVision.BackOffice.ObjectModel.TaskGroup

Для работы с карточкой предназначен сервис `ITaskGroupService`, с помощью которого можно работать с группами заданий. Позволяет создавать группы заданий, отправлять на исполнение и отзывать.

Доступные операции:

- Создание группы заданий.
- Добавление в задание дочерней группы заданий.
- Добавление исполнителей, в том числе с индивидуальными настройками.
- Отправка группы заданий на исполнение.
- Отзыв группы заданий.
- Проверка заданий на наличие ошибок.

Далее перечислены базовые сценарии работы с объектной моделью карточки "Группа заданий", для которых действительно следующее:

- Контекст объектов может быть получен в соответствии с примером, приведенным в разделе Инициализация контекста объектов.
- Сервисы получаем стандартным способом:

```
IStaffService staffService = objectContext.GetService<IStaffService>(); ①
IStateService stateService = objectContext.GetService<IStateService>(); ②
ITaskGroupService taskGroupService = objectContext.GetService<ITaskGroupService>(); ③
ITaskListService taskListService = objectContext.GetService<ITaskListService>(); ④
```

① Сервис для работы со "Справочником сотрудников".

② Сервис для работы с "Конструктором состояний".

③ Сервис для работы с "Группой заданий".

④ Сервис для работы со списком заданий.

- Воспользуйтесь поиском, чтобы получить информацию по классам и переменным, используемым в примерах.

Создание "Группы заданий"

```
static void TaskGroupCreate()
{
 KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("6D76D0A7-5434-
40F2-912E-6370D33C3151")); ①
```

```
 TaskGroup taskGroup = taskGroupService.CreateTaskGroup(kind);
 taskGroup.Description = "Новая группа заданий";
 taskGroup.MainInfo.Name = "Данная группы заданий создана из кода";
 taskGroup.MainInfo.Author = staffService.GetCurrentEmployee();
 taskGroup.MainInfo.StartDate = DateTime.Now;
 taskGroup.MainInfo.EndDate = taskGroup.MainInfo.StartDate.Value.AddHours(5D);
 taskGroup.MainInfo.Content = "Необходимо выполнить данные задания.\nСрок 5 часов."; ②

 taskGroupService.AddSelectedPerformer(taskGroup, staffService.GetCurrentEmployee()); ③

 objectContext.SaveObject(taskGroup);
}
```

① Получение вида для группы заданий.

② Создание группы заданий.

③ Добавление текущего пользователя в качестве исполнителя.

Добавление "Группы заданий" в документ

```
static void AddTaskGroupToDocument()
{
 Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ①

 TaskGroup taskGroup = objectContext.GetObject<TaskGroup>(new Guid("00000000-0000-0000-
0000-000000000001")); ②

 taskListService.AddTaskGroup(document.MainInfo.Tasks, taskGroup, document); ③
 objectContext.SaveObject(document);
}
```

① Получение документа, в который добавляется группа заданий.

② Получение добавляемой группы заданий.

③ Добавление группы заданий в документ.

Отправка Группы заданий на исполнение

```
static void SendToPerformance()
{
 TaskGroup taskGroup = objectContext.GetObject<TaskGroup>(new Guid("00000000-0000-0000-
0000-000000000000")); ①

 Guid kindId = taskGroupService.GetKindSettings(taskGroup.SystemInfo.CardKind
).DefaultTaskCardKind;
```

```
 KindsCardKind kindTask = objectContext.GetObject<KindsCardKind>(kindId); ②

 taskGroupService.SendToPerformance(taskGroup, kindTask); ③

 StatesState statesState = stateService.FindStateByBuiltIn(TaskGroup.PerformanceState,
taskGroup); ④

 stateService.ChangeState(taskGroup, statesState); ⑤
}
```

① Получение группы заданий, отправляемой на исполнение.

② Получение вида создаваемых заданий из параметров вида группы заданий.

③ Отправка на исполнение.

④ Получение состояния `На исполнение`.

⑤ Смена состояния карточки.

Задание

Назначение	Предназначена для доставки на рабочие места пользователей информации о необходимости выполнения того или иного действия в рамках налаженного документооборота, а также доставки файлов, карточек и ссылок на внешние по отношению к системе Docsvision объекты.
Класс карточки	`DocsVision.BackOffice.ObjectModel.Task`

Для работы с карточкой предназначен сервис `ITaskService`, с помощью которого можно работать с заданиями. Позволяет создавать задания, настраивать и управлять ими.

Доступные операции:

- Создание задания.
- Управление состоянием задания.
- Управление комментариями к заданию.
- Управление исполнителями и контроллерами задания.

- Управление делегированием задания.
- Управление подписями задания.
- Подготовка вложений задания.
- Управление настройками задания.
- Управление дочерними и родительскими заданиями.

Далее перечислены базовые сценарии работы с объектной моделью карточки "Задание", для которых действительно следующее:

- Контекст объектов может быть получен в соответствии с примером, приведенным в разделе Инициализация контекста объектов.
- Сервисы получаем стандартным способом:

```
IDocumentService documentService = objectContext.GetService<IDocumentService>(); ①
ILinkService linkService = objectContext.GetService<ILinkService>(); ②
IReferenceListService referenceListService = objectContext.GetService
<IReferenceListService>(); ③
IStaffService staffService = objectContext.GetService<IStaffService>(); ④
IStateService stateService = objectContext.GetService<IStateService>(); ⑤
ITaskService taskService = objectContext.GetService<ITaskService>(); ⑥
```

  ① Сервис для работы с документами.

  ② Сервис для работы со "Справочником ссылок".

  ③ Сервис для работы со списками ссылок.

  ④ Сервис для работы со "Справочником сотрудников".

  ⑤ Сервис для работы с "Конструктором состояний".

- Для получения информации по классам и переменным, используемым в примерах, воспользуйтесь поиском.

Создание нового "Задания"

```
static void CreateTask()
{
 KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("AB801854-70AF-
4B6C-AB48-1B59B5D11AA9")); ①

 StaffEmployee employee = objectContext.GetObject<StaffEmployee>(new Guid("00000000-0000-
0000-0000-000000000000")); ②
```

```
 Task task = taskService.CreateTask(kind); ③
 task.MainInfo.Name = "Название задания";
 task.Description = "Дайджест созданного задания";
 task.MainInfo.Author = staffService.GetCurrentEmployee(); ④
 taskService.AddSelectedPerformer(task.MainInfo, employee); ⑤
 task.MainInfo.Priority = TaskPriority.Low; ⑥
 task.MainInfo.EndDate = DateTime.Now.AddDays(1); ⑦
 task.MainInfo.Content = "Данное задание является примером и не требует исполнения";
 objectContext.SaveObject(task); ⑧
}
```

① Получение вида для создаваемого задания. AB801854-70AF-4B6C-AB48-1B59B5D11AA9 — На исполнение.

② Получение сотрудника, назначаемого исполнителем задания.

③ Создание задания и заполнение базовых данных.

④ Автор задания.

⑤ Добавление выбранного исполнителя.

⑥ Установка низкого приоритета.

⑦ Срок выполнения — "завтра".

⑧ Сохранение созданного задания.

Добавление файла в "Задание"

```
static void AddFile()
{
 string file = @"z:\Sample.docx"; ①

 Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ②

 ReferenceList referenceList = referenceListService.CreateReferenceList();
 task.MainInfo.ReferenceList = referenceList; ③

 LinksLinkType linkType = linkService.FindLink("К3_ДополнениеФайлы"); ④

 KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("F9A8D158-9884-
4765-859D-31C4EFCA149D")); ⑤
 Document fileCard = documentService.CreateDocument(file, kind); ⑥
 fileCard.MainInfo.Name = "Файл для задания";
 objectContext.SaveObject(fileCard); ⑦

 referenceListService.CreateReference(referenceList, linkType, fileCard, string.Empty,
```

```
true); ⑧
 objectContext.AcceptChanges();
}
```

① Загружаемый файл.

② Получение задания, в которое производится добавление файла.

③ Создание нового списка ссылок.

④ Получение типа ссылки по имени.

⑤ Создание карточки Файл для последующей загрузки в задание.

⑥ Получение вида карточки.

⑦ Создание документа.

⑧ Требуется сохранение.

Отправка Задания на исполнение

```
static void StartTask()
{
 Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ①

 taskService.StartTask(task); ②

 StatesStateMachineBranch branch = stateService.FindBranchByBuiltIn(Task
.InitializationToStartedByStart, task.SystemInfo.State); ③

 stateService.ChangeState(task, branch); ④
}
```

① Получение задания.

② Запуск задания на исполнение.

③ Получение перехода Подготовка — Не начато.

④ Изменение состояния.

Делегирование задания

```
static void DelegateTask()
{
 Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ①
```

```
 StaffEmployee newPerformer = staffService.Get(new Guid("00000000-0000-0000-0000-
000000000001")); ②

 taskService.Delegate(task, new StaffEmployee[] { newPerformer }, null, false, false,
"Необходимо выполнить задание"); ③

 stateService.ChangeState(task, stateService.FindStateByBuiltIn(Task.DelegatedState,
task)); ④
}
```

① Получение задания.

② Получение сотрудника, которому осуществляется делегирование задания.

③ Делегирование задания сотруднику "newPerformer".

④ Установка для задания состояния Делегировано.

Завершение "Задания"

```
static void CompleteTask()
{
 Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ①

 BuiltInState state = taskService.CompleteTask(task, "Задание было завершено успешно");
②

 StatesState cancelState = stateService.FindStateByBuiltIn(state, task); ③

 stateService.ChangeState(task, cancelState); ④
}
```

① Получение задания.

② Завершение задания с комментарием.

③ Поиск в "Конструкторе состояний" состояния, соответствующего "встроенного" состоянию Завершено.

④ Изменение состояния.

Метод CompleteTask выполняет проверку и установку параметров, необходимых для корректного завершения задания, но фактическое изменение состояние производит метод ChangeState.

При работе с методами сервиса ITaskService нужно учитывать, что при выполнении статусных методов (отозвать, принять, отклонить

> и т.д.) статус самого задание не изменяется. Для непосредственной смены статуса используется метод `IStateService.ChangeState(BaseCard, StatesState)`.

Конструктор ролей

Назначение	Определяет механизм распределения прав пользователей при работе с рядом приложений Docsvision на основе данных самой карточки.
Класс карточки	`DocsVision.BackOffice.ObjectModel.RoleModel`

Для работы с карточкой предназначен сервис `IRoleModelService`, с помощью которого можно получить доступ к сохранённым правилам и свойствам ролевой модели, а также создавать и управлять объектами данного уровня безопасности.

Доступные операции:

- Создание условий и групп условий для ролей.
- Создание пользовательских параметров и операций.
- Связывание настроек в сопоставление операция-состояние.
- Создание и получение настроек ролевой модели для вида карточки.
- Получение ролей для вида карточки.

В примере создаётся роль, которая может быть использована для ограничения времени доступа к документам в выходные дни.

```
IRoleModelService roleModelService = objectContext.GetService<IRoleModelService>(); ①

KindsCardKind kindsCardKind = objectContext.GetObject<KindsCardKind>(new Guid("8ACE1220-
A452-455D-8EEB-9EDF9DC6E327")); ②

RoleModelCardKindRoleSetting roleModelCardKindRoleSetting = roleModelService
.GetOrCreatePreferredCardKindSetting(kindsCardKind); ③
RoleModelRole roleModelRole = roleModelService.CreateRole(roleModelCardKindRoleSetting);
roleModelRole.Name = "Работа по выходным";

roleModelRole.ConditionGroup.Operation = RoleModelRoleConditionGroupOperation.Or; ④

RoleModelRoleCondition condition = roleModelService.AddCondition(roleModelRole
.ConditionGroup); ⑤
```

```
condition.Parameter = RoleModelRoleConditionParameter.Today;
condition.Operation = RoleModelRoleConditionOperation.Equals;
condition.Value = System.DayOfWeek.Saturday;

condition = roleModelService.AddCondition(roleModelRole.ConditionGroup); ⑥
condition.Parameter = RoleModelRoleConditionParameter.Today;
condition.Operation = RoleModelRoleConditionOperation.Equals;
condition.Value = System.DayOfWeek.Sunday;

objectContext.AcceptChanges(); ⑦

roleModelService.MakeRoleCommon(roleModelRole); ⑧
```

① Инициализация контекста объектов и получение сервиса.

② Получение вида карточки "Документ".

③ Создание настроек для вида карточки.

④ Логическая операция между условиями — `ИЛИ`.

⑤ Создание первого условия: `сегодня = суббота`.

⑥ Создание второго условия: `сегодня = воскресенье`.

⑦ Сохранение изменений.

Роль не создаёт реального ограничения, для настройки используется матрица доступа (недоступная для карточки "Документ", доступная для "ДокументУД").

Конструктор состояний

Назначение	Позволяет описать жизненный цикл карточки.
Класс карточки	`DocsVision.BackOffice.ObjectModel.StatesDictionary`

Для работы с карточкой предназначен сервис `IStateService`, с помощью которого можно работать с автоматом состояний. Он предоставляет возможность управления операциями, состояниями, ветвями перехода между состояниями и диаграммой состояний. Помимо этого, с помощью данного сервиса можно узнать, какие операции доступны для заданной карточки и в какое состояние её можно перевести.

Доступные операции:

- Создание ветки перехода из одного состояния в другое.
- Создание операции для смены состояния.
- Создание состояния.
- Получение состояний доступности операций.
- Создание разметки диаграммы состояний.
- Поиск ветки перехода.
- Поиск операций.
- Поиск состояний.
- Получение объекта настройки прав для вида карточки.
- Получение списка операций для вида карточки.
- Получение списка ветвей перехода.
- Получение разметки диаграммы состояний.
- Получение списка состояний.
- Удаление объекта настройки прав для вида карточки.
- Локализация названия операции.
- Локализация названия состояния.
- Определение возможности удаления операции.

В примере выведем в консоль названия всех состояний для вида документа *Внутренний*

```
IStateService stateService = objectContext.GetService<IStateService>(); ①

KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("7db9044e-91b7-447d-8caa-5e5a4967b8d4")); ②

IList<StatesState> states = stateService.GetStates(kind);
foreach (var item in states)
{
 Console.WriteLine(item.DefaultName); ③
}
```

① Инициализация контекста объектов и получение сервиса для работы со Конструктором состояний.

② Получение вида документа *Внутренний*.

③ Получение допустимых состояний для данного вида и вывод на консоль.

См. также пример из раздела "Задание"

Конструктор справочников

Назначение	Карточка предназначена для организации древовидной структуры простых текстовых данных.
Класс карточки	`DocsVision.BackOffice.ObjectModel.BaseUniversal`

Для работы с карточкой предназначен сервис `IBaseUniversalService`, с помощью которого можно управлять узлами и строками справочника.

Доступные операции:

- Создание узлов и строк.
- Копирование, перемещение и удаление узлов и строк справочника.
- Поиск узлов и строк.
- Создание и получение отображаемых полей.

Ниже приведён пример создания справочника техники магазина. Создаётся новый узел "Мониторы", в который помещаются узлы с марками и строки с моделями.

```
IBaseUniversalService baseUniversalService = objectContext.GetService
<IBaseUniversalService>(); ①

BaseUniversal baseUniversal = objectContext.GetObject<BaseUniversal>(DocsVision
.BackOffice.CardLib.CardDefs.RefBaseUniversal.ID); ②

BaseUniversalItemType techType = baseUniversalService.AddNewItemType(null); ③
techType.Name = "Мониторы";
techType.Description = "Мониторы всех фирм";

BaseUniversalItemType techMark = baseUniversalService.AddNewItemType(techType);
techMark.Name = "LG";
techMark.Description = "Производитель LG"; ④

BaseUniversalItem item = baseUniversalService.AddNewItem(techMark);
item.Name = "34UM95";
item.Description = "Модель 34UM95. Разрешение экрана 3440x1440. Матрица IPS."; ⑤
```

```
item = baseUniversalService.AddNewItem(techMark);
item.Name = "29UM65";
item.Description = "Модель 29UM65. Разрешение экрана 2560x1080. Матрица IPS.";

objectContext.AcceptChanges(); ⑥
```

① Инициализация контекста объектов и получение сервиса.

② Получение объектной модели Конструктора справочников.

③ Создания одного из узлов первого уровня. `null` — позволяет создать элемент без родителей, т.е. корневой.

④ Создание узла с названиями производителей.

⑤ Создание строк с моделями мониторов.

⑥ Сохранение изменений.

Пример можно использовать как часть проекта синхронизации внутреннего справочника фирмы и поставщика техники. Справочник можно подключить к карточке, которая будет играть роль бланка для выписывания техники со склада.

Список ссылок на карточки заданий

Назначение	Создаёт список заданий для последующего добавление в другую карточку (например, "Документ").
Класс карточки	`DocsVision.BackOffice.ObjectModel.TaskList`

Для работы с карточкой предназначен сервис `ITaskListService`, позволяющий создавать и получать карточки списков заданий, которые прикладываются к карточкам документа.

Доступные операции:

- Создать карточку списка заданий.
- Добавить задание в список.
- Удалить задание.
- Получить дерево заданий.
- Получить описание узла дерева заданий.

В примере создаётся список заданий, в который добавляется отдельное задание.

Список закрепляется за документом.

```
Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-000000000000")); ①

IStaffService staffService = objectContext.GetService<IStaffService>();
ITaskService taskService = objectContext.GetService<ITaskService>();
ITaskListService taskListService = objectContext.GetService<ITaskListService>(); ②

TaskList taskList = taskListService.CreateTaskList();
document.MainInfo.Tasks = taskList; ③

KindsCardKind taskKind = objectContext.GetObject<KindsCardKind>(new Guid("AB801854-70AF-4B6C-AB48-1B59B5D11AA9")); ④

Task task = taskService.CreateTask(taskKind);
task.MainInfo.Name = "Задача для Иванова";
task.MainInfo.Author = staffService.GetCurrentEmployee();
taskService.AddSelectedPerformer(task.MainInfo, staffService.GetCurrentEmployee()); ⑤

taskListService.AddTask(taskList, task, document); ⑥

objectContext.AcceptChanges(); ⑦
```

① Инициализация контекста объектов и получение документа с ID `00000000-0000-0000-0000-000000000000` для которого будет добавлено задание.

② Получение сервисов для работы с сотрудниками, заданиями и списками заданий.

③ Создание списка заданий и присвоение его документу.

④ Получение вида задания. В данном случае `На исполнение`.

⑤ Инициализация задания. Автором и исполнителем назначаем текущего пользователя.

⑥ Добавление списка заданий к документу.

Справочник видов карточек

Назначение	Предназначен для управления иерархией видов для существующих типов карточек, добавление и удаление видов, а также настройка определённых параметров видов.

Класс карточки	DocsVision.BackOffice.ObjectModel.Kinds

Для работы с карточкой предназначен сервис IKindService, с помощью которого можно программно создавать и удалять виды карточек, управлять настройками, кроме того, возможно добавление и удаление расширений.

Доступные операции:

- Добавление и удаление вида карточки в справочник.
- Добавление типа карточки.
- Управление настройками способов создания карточек видов.
- Добавление и удаление расширений.
- Управление настройками расширений.
- Управление настройками бизнес-процессов в карточке.

В приведенном примере создаётся новый вид для карточки "Документ"

```
IKindService kindService = objectContext.GetService<IKindService>(); ①

KindsCardType kindsCardType = kindService.GetCardType(DocsVision.BackOffice.CardLib
.CardDefs.CardDocument.ID); ②

KindsCardKind kindsCardKind = kindService.AddCardKind(kindsCardType);
kindsCardKind.Name = "Тестовый вид карточки"; ③

kindsCardKind.UseOwnSettings = true;
kindsCardKind.UseOwnExtendedSettings = true;
kindsCardKind.UseOwnLayouts = true; ④

kindService.AddProcessSettings(kindsCardKind, new Guid("12345678-0000-0000-0000-
000000000000"), "Отправить на согласование"); ⑤

objectContext.AcceptChanges(); ⑥
```

① Инициализация контекста объектов и получение сервиса.

② Получение типа карточки "Документ".

③ Создание новый вид в полученном типе, с названием "Тестовый вид карточки".

④ Вид будет использовать свои настройки.

⑤ Добавление БП в настройки вида.

⑥ Сохранение изменений.

Карточка "Справочник видов карточек" блокируется для записи при открытии справочника в Windows-клиенте.

Справочник контрагентов

Назначение	Предназначен для хранения информации о структуре и сотрудниках внешних организаций, с которыми взаимодействует компания.
Класс карточки	`DocsVision.BackOffice.ObjectModel.Partners`

Для работы с карточкой предназначен сервис `IPartnersService`, с помощью которого можно программно создавать и удалять контрагентов, сотрудников и группы компаний.

Доступные операции:

- Управление подразделениями.
- Управление сотрудниками.
- Управление группами.
- Осуществление поиска по различным параметрам.

В приведенном примере создаётся новый вид для карточки "Документ"

```
IPartnersService partnersService = objectContext.GetService<IPartnersService>(); ①

PartnersOrgType orgType = partnersService.AddNewOrgType();
orgType.Name = "ЗАО"; ②

PartnersPosition positionDir = partnersService.AddNewPosition();
positionDir.Name = "Директор";
PartnersPosition positionMan = partnersService.AddNewPosition();
positionMan.Name = "Начальник отдела продаж"; ③

PartnersCompany partnersCompany = partnersService.AddNewUnit(null);
partnersCompany.Name = "ЗАО \"Луч\"";
partnersCompany.FullName = "Закрытое акционерное общество \"ЛУЧ\""; ④
partnersCompany.Type = PartnersCompanyType.Organization;
partnersCompany.OrgType = orgType;
partnersCompany.IsVendor = true; ⑤
```

```
PartnersEmployee partnersEmployee = partnersService.AddNewEmployee(partnersCompany);
partnersEmployee.LastName = "Иванов";
partnersEmployee.FirstName = "Александр";
partnersEmployee.MiddleName = "Сергеевич"; ⑥
partnersEmployee.Position = positionDir; ⑦

partnersCompany.Manager = partnersEmployee; ⑧

PartnersCompany partnersDepartment = partnersService.AddNewUnit(partnersCompany);
partnersDepartment.Name = "Отдел продаж";
partnersDepartment.Type = PartnersCompanyType.Department; ⑨
PartnersEmployee partnersEmployee2 = partnersService.AddNewEmployee(partnersDepartment);
partnersEmployee2.LastName = "Константинов";
partnersEmployee2.FirstName = "Сергей";
partnersEmployee2.MiddleName = "Иванович";
partnersEmployee2.Position = positionMan;
partnersDepartment.Manager = partnersEmployee2;

objectContext.AcceptChanges(); ⑩
```

① Инициализация контекста объектов и получение сервиса

② Добавление типа юр. лица

③ Добавление должностей

④ Создание организации

⑤ Ставим признак, что организации

⑥ Добавление сотрудника

⑦ Должность "Директор"

⑧ Назначаем сотрудника руководителем

Справочник сотрудников

Назначение	Предназначен для хранения сведений о структуре организации: входящих в её состав подразделениях и отдельных сотрудниках.
Класс карточки	`DocsVision.BackOffice.ObjectModel.Staff`

Для работы с карточкой предназначен сервис `IStaffService`, с помощью которого можно программно создавать и удалять объекты справочника сотрудников, управлять ими, кроме того, возможна синхронизация с Active Directory.

Доступные операции:

- Добавление объектов в справочник.
- Удаление объектов из справочника.
- Поиск объектов в справочнике.
- Проверить уникальность данных.
- Получение объектов справочника.
- Синхронизация с ActiveDirectory.
- Управление данными сотрудника и подразделения.
- Управление структурой справочника.
- Управление правами.

В простом примере, заведем нового сотрудника в существующем подразделении.

```
IStaffService staffService = objectContext.GetService<IStaffService>(); ①

StaffUnit unit = objectContext.GetObject<StaffUnit>(new Guid("00000000-0000-0000-0000-
000000000000")); ②

StaffEmployee employee = staffService.AddNewEmployee(unit); ③

employee.AccountName = "myDomain/myName";
employee.BirthDate = DateTime.Now;
employee.LastName = "Иванов";
employee.FirstName = "Иван"; ④

objectContext.AcceptChanges(); ⑤
```

① Инициализация контекста объектов и получение сервиса для работы со справочником сотрудников.

② Получение подразделения в которое поступает сотрудник.

③ Создание нового сотрудника.

④ Заполнение данных.

⑤ Сохранение контекста.

Работа с папками и ярлыками

Для создания организованной визуальной структуры в платформе Docsvision предусмотрены специализированные объекты — Папки, которые реализуются

при помощи специальной системной карточки — *Карточки папок*. *Карточка папок* хранит иерархическое дерево папок, каждая из которых имеет ряд собственных свойств (название, иконка и т.д.), а также ряд дочерних элементов — подпапок и ярлыков.

Карточка папок

Разработчик может взаимодействовать с данными карточки папок непосредственно через объект `CardData` по тем же принципам, как с любой другой карточкой системы. Однако, для упрощения этой задачи в объектной модели предусмотрен ряд специализированных объектов, лучше отражающих семантику: `Folder`, `Shortcut`. Ключевым объектом этой модели является сама карточка папок — объект `FolderCard`.

Все объекты расположены в специальном пространстве имён для работы с системными карточками — `DocsVision.Platform.ObjectManager.SystemCards`.

Как и все системные карточки, этот объект может быть получен через стандартный интерфейс методами `CardManager.GetCard` или `CardManager.GetDictionary`.

Так как карточка папок является справочником, то её идентификатор всегда известен и неизменен — у справочников он всегда соответствует идентификатору типа карточки.

```
const string FOLDER_CARD_TYPE = "DA86FABF-4DD7-4A86-B6FF-C58C24D12DE2"; ①
FolderCard folderCard = (FolderCard)userSession.CardManager.GetDictionary(new Guid
(FOLDER_CARD_TYPE));
```

① Получение карточки папки.

Папка

Папка содержит некоторое количество собственных свойств (название, иконка и т.д.), а также ряд дочерних элементов, к числу которых относятся:

- Подпапки — дочерние папки.
- Ярлыки, лежащие в папке.
- Ограничения на типы карточек, которые могут быть созданы в папке.

- Ограничения на представления, которые могут быть применены.
- Ограничения на шаблоны экспорта.

У папки есть три режима работы:

- Обычный — отображается содержимое в виде списка карточек.
- URL — отображается страница.
- Карточка — в правой панели отображается интерфейс карточки.

Из объектной модели доступен переключатель режима работы папки `Folder.CurrentStyle` и идентификатор карточки папки `Folder.PropCardID`. Также можно выполнить соответствующую настройку из Windows-клиента.

При создании карточки папки надо учитывать, что, когда пользователь открывает папку, она отображается не модально и скрывается, как только пользователь переходит на другую папку. У карточки нет возможности отменить закрытие (ей не приходит вызов `CardControl.CardClosing`), поэтому в её интерфейсе не должно быть средств редактирования чего-либо, что может остаться несохранённым к моменту перехода на другую папку. Лучше всего использовать интерфейс карточки для отображения чего-либо, а всё редактирование вынести на отдельные формы, по закрытии которых будет производиться сохранение данных.

В примере проверяется наличие у папки подпапки с названием "завершённые". Если папка с заданным именем отсутствует, она будет создана.

```
Folder folder = folderCard.GetFolder(new Guid("00000000-0000-0000-0000-000000000000"));
string folderName = "завершённые"; ①

if (!folder.HasSubfolders || folder.Folders.Count(m=>m.Name.Equals(folderName)) = 0)
{
 folder.Folders.AddNew(folderName); ②
}
```

① Получение папки с идентификатором `00000000-0000-0000-0000-000000000000`.

② Если папка не имеет подпапок, либо не содержит подпапки с именем "завершённые", такая папка будет создана.

Ярлык

Физически ярлык — это просто строка карточки папок, которая хранит в себе ссылку на карточку.

Как и ссылка, ярлыки бывают двух видов — *слабые* и *сильные*. Слабый ярлык держит слабую ссылку на карточку и отображается с overlay-иконкой, аналогичной ссылке в Windows. При удалении такого ярлыка удаляется только сам ярлык.

Сильный ярлык держит сильную ссылку на карточку, и отображается обычной иконкой карточки. Для пользователя сильный ярлык полностью соответствует нахождению карточки в этой папке, поэтому правомерно говорить о том, что "карточка лежит в папке". При удалении ярлыка удаляется сама карточка, если на неё нет других сильных ссылок. Если программно нужно удалить карточку, на которую могут быть ярлыки, то надо воспользоваться методом `FolderCard.DeleteCard`, который удаляет карточку с учетом существующих ярлыков.

Для каждой карточки можно создать только один сильный ярлык. В случае повторного вызова метода создания сильного ярлыка для той же карточки — метод вернет ошибку.

По аналогии с письмами Outlook ярлык на карточку обладает признаком прочитанности. Непрочитанные ярлыки отображаются в Windows-клиенте жирным шрифтом. Когда пользователь открывает карточку, он устанавливает этот признак в "прочитано". Этот признак можно выставить программно, если необходимо сигнализировать пользователю о том, что ему необходимо обратить внимание на новый ярлык.

Кроме этого, ярлык может содержать также дополнительные атрибуты, например, режим открытия карточки `Shortcut.ModeId`. Таким образом, можно создать на одну карточку несколько ярлыков с разными режимами открытия карточки.

В объектной модели для работы с ярлыками предназначен объект `Shortcut`.

В примере, для заданной карточки удаляются все ярлыки, кроме "сильного":

```
ShortcutCollection shortcutCollection = folderCard.GetShortcuts(new Guid("00000000-0000-
0000-0000-000000000000")); ①

foreach (DocsVision.Platform.ObjectManager.SystemCards.Shortcut shortcut in
shortcutCollection)
{
 if (!shortcut.IsHardLink) folderCard.DeleteShortcut(shortcut.Id, true); ②
```

```
}
```

① Получение всех ярлыков на карточку `00000000-0000-0000-0000-000000000000`.

② Если ярлык не "сильный", то удалить.

Чтобы создать в папке ярлык на существующую карточку, используйте метод CreateShortcut *карточки папок*:

```
FolderCard folderCard = (FolderCard)userSession.CardManager.GetDictionary(new Guid
("DA86FABF-4DD7-4A86-B6FF-C58C24D12DE2")); ①

folderCard.CreateShortcut(new Guid("FB9A01EC-3412-42B2-8555-8016674D4DE9"), new Guid
("A65C892C-339F-E511-9417-90E6BA57B9F8"), false); ②
```

① Получение карточки папок

② Создание ярлыка.
Первый параметр — идентификатор папки, в которой создаётся ярлык.

Второй параметр — идентификатор карточки, на которую создаётся ярлык.

Третий параметр — признак создания "сильного ярлыка".

Работа с файлами

Платформа Docsvision позволяет загружать в базу данных и извлекать из базы данных пользовательские файлы произвольного формата. Для работы с файлами в системе существует специальная системная карточка — *карточка файла*, а также набор интерфейсов менеджера объектов для манипуляций с файлами.

Для работы с файлами можно использовать несколько видов объектов:

- Системный объект — **файл** — реализует простейшую функциональность для хранения файлов в базе данных Docsvision.
- Системный объект — **версия файла** — реализует дополнительные атрибуты для нумерованной версии файла.
- Системная **карточка файла** — поддерживают функциональность по хранению нескольких версий файла, комментариев к ним и дополнительных механизмов работы с файлом.
- Карточка файла из решения *Делопроизводство* (не рассматривается).

- Карточка — список файлов из решения *Делопроизводство* (не рассматривается).

Системный объект — файл

Для работы с простыми файлами в системе предусмотрен объект `FileData`. Это самый низкоуровневый объект для работы с двоичными данными файла — поэтому у него нет таких понятий как "версия" или "свойства".

Работа со стандартными файлами осуществляется при помощи менеджера файлов, который может быть получен из интерфейса менеджера объектов `UserSession.FileManager`. Менеджер файлов позволяет выполнять простейшие операции с файлами — создавать, удалять, копировать и осуществлять поиск файлов в системе

В примере, создадим **файл** с названием `NewFile` и загрузим в него файл из файловой системы `NewFile.doc`, после чего присвоим переменной `newFileId` значение идентификатора созданного файла

```
FileData file = userSession.FileManager.CreateFile("NewFile"); ①

file.Upload("..\\NewFile.doc"); ②

Guid newFileId = file.Id; ③
```

① Создание нового файла.

② Загрузка файла из файловой системы.

③ Получение идентификатора для дальнейшего использования.

Внешнее хранение файлов

В системе Docsvision файл имеет две составляющие: информационную и бинарную. В информационной части хранится информация о файле (название, размер и т. п.), в бинарной — бинарное содержимое файла.

Информационная часть файла храниться в БД Docsvision, бинарная может храниться в основной БД или в другом поддерживаемом хранилище.

Подробная информация о внешнем хранении файлов приведена в документации по администрированию модуля *Платформа*, раздел "Внешние хранилища", и документации "Общее описание системы",

раздел /dv6/system/dv6/binary-storage/[Хранение бинарных данных файлов].

При работе с данными файла следует учитывать следующую особенность: между информационной и бинарной частью файла устанавливается связь один ко многим — несколько информационных частей могут ссылаться на одни бинарные данные. Например, данная ситуация возникает при копировании файла с помощью метода `FileData.Copy`. После изменения содержимого файла (с использованием стандартного API) оригинал и копия станут ссылаться на собственный (уникальный) экземпляр бинарных данных.

Офлайн режим файла

Для файлов поддерживается офлайн режим, при котором файл перемещается в хранилище, поддерживающее режим офлайн.

Файл в режиме офлайн не участвуют в обработке информации (прежде всего — построение полнотекстового каталога), а также недоступны для получения и изменения до возвращения в режим онлайн. Если попытаться читать или записывать данные файла, выгруженного из базы (и если выключена автоматическая загрузка его обратно в базу), то клиенту будет возвращена ошибка. Такая же ошибка будет возвращена, если попытаться выгрузить уже выгруженный файл. Если попытаться загрузить файл, уже находящийся в базе, клиенту также будет возвращена ошибка.

Изменение оперативного статуса файлов выполняется с помощью методов `FileData.TakeOffline` и `FileData.BringOnline`.

Для выполнения операций вытеснения/восстановления файлов из внешнего хранилища требуется наличие привилегии оператора архива (членство в группе **DocsVision Power Users**).

Текущее состояние файла определяется элементом из перечисления `OfflineState` — перечисление.

Системный объект — версия файла

Версия файла (объект `FileVersion`) расширяет функциональность стандартного файла и наследует его интерфейс. Свойства и методы версии файла повторяют аналогичные свойства файла и добавляют к ним собственные.

Объект версии файла не может быть создан разработчиком напрямую и

используется только в системной карточке файла с версиями `VersionedFileCard`.

Системная карточка файла

Системная карточка файла используется для обеспечения удобства работы с файлами и предоставляет дополнительные возможности, такие как:

- Поддержка неограниченного количества версий каждого файла.
- Возможность задания комментариев к версиям файла.
- Обеспечение безопасности при работе с файлом.
- Поддержка механизма единоличного владения файлом (`CheckIn`/`CheckOut`).

Для работы с системной карточкой файла в менеджере объектов предусмотрен объект `VersionedFileCard`.

Поскольку карточка файла является обычной карточкой Docsvision, то для её создания необходимо прибегнуть к помощи объекта — менеджера карточек (`CardManager`).

Пример создания и первичной инициализации карточки файла:

```
VersionedFileCard fileCard = (VersionedFileCard)userSession.CardManager.CreateCard(new
Guid("6E39AD2B-E930-4D20-AAFA-C2ECF812C2B3")); ①

fileCard.Initialize("..\\NewFile.doc", Guid.Empty, false, false); ②

FileVersion fileVersion = fileCard.CurrentVersion; ③

Console.WriteLine(fileVersion.VersionString); ④
```

① Создание карточки файла с версиями, `6E39AD2B-E930-4D20-AAFA-C2ECF812C2B3` — идентификатор такого типа карточек.

② Первичная инициализация карточки файла.

③ Получение первой версии файла.

④ Вывод номера версии.

Таким образом, для работы с файлами:

- При использовании системы в качестве хранилища файлов произвольного формата для простоты следует использовать системный объект-файл.
- При необходимости ведения сложной иерархии версий файла и реализации механизмов `CheckIn`/`CheckOut` следует использовать системную карточку файла.

Работа с правами доступа

Система позволяет назначать права на отдельные объекты, для чего используется система безопасности Windows, основанная на пользователях и группах (доменных или локальных) и описателях прав (дескрипторах).

Право дается или запрещается пользователю при помощи объекта ACE (`access control entry`), который задаёт маску прав. Права могут быть даны (`allowed`) или запрещены (`denied`) конкретному пользователю или группе (`security principal`).

ACE объединяются в список ACL (`access control list`). DACL (`discretionary access control list`) хранится в описателе прав (`security descriptor`), в котором, кроме списка DACL, есть ещё информация о владельце объекта (`owner`), а также настройки аудита в виде списка SACL (`system access control list`). Кроме того, начиная с версии 4.3, поддерживается использование в дескрипторе метки уровня доступа (`mandatory access control, MAC`).

В текущей версии платформы аудит не поддерживается.

Права, которые можно настраивать на объектах, делятся на общие (`general`) и специфические (`specific`). Общие отображаются в главном диалоге настройки безопасности. Специфические появляются в списке в окне *Advanced*. Общие права используются для упрощения администрирования. Специфические права дают администратору возможность управлять поведением объектов на более тонком уровне.

Общие права (`AccessGeneral`) состоят из набора частных прав:

- Чтение (`R`, `RP`).
- Изменение (`W`, `CC`, `DC`).
- Удаление (`D`).
- Полный доступ (все права).

Специфические права (`AccessSpecific`):

- Чтение данных объекта (`R`).
- Изменение данных объекта (`W`).
- Создание дочерних объектов (`CC`).
- Удаление дочерних объектов (`DC`).

- Удаление объекта (`D`).
- Чтение разрешений (`RP`).
- Изменение разрешений (`SP`).
- Смена владельца (`TO`).

Система позволяет назначать права на все низкоуровневые объекты:

- Типы карточек (`CardType`).
- Карточки (`CardData`).
- Секции (`SectionData`).
- Строки (`RowData`).
- Файлы (`FileData`).
- Хранимые процедуры (`Report`).
- Пользовательские объекты.

Для этого в каждом из этих объектов предусмотрено два специальных метода: `GetAccessControl` — возвращающий описатель прав и `SetAccessControl`, устанавливающий описатель прав.

Наследование прав

В системе реализовано наследование прав от родительских к дочерним объектам следующим образом:

- Карточка наследует права от других карточек, которые держат на данную карточку сильные ссылки (а физически от строки, в которой содержится ссылка). При установке первой сильной ссылки на карточку её права меняются, чтобы отразить включившееся наследование. При изменении прав на родительскую карточку права на подчинённую карточку меняются соответственно.
- Секция карточки наследует от экземпляра карточки, которому он принадлежит, или от родительской строки если это подчинённая секция.
- Строка наследует права от своей секции.
- Файл наследует от карточки владельца, если она задана.

Частным, но важным случаем является наследование прав от папки на карточку. Папка — это строка в карточке папок — от неё права наследуют все ярлыки в папке. По сильным ярлыкам права переходят на карточки. Таким образом, права

на карточки, на которые существуют сильные ярлыки, наследуются от карточки папок.

В Windows-клиенте права на карточку папок могут быть отредактированы на узле *Папки*. Данный частный случай относится только к Docsvision 4.5. В более поздних версиях Docsvision ярлык является слабой ссылкой — дискреционные права, установленные для папки, не будут унаследованы карточками, ярлыки на которые размещены в папке.

Наследованием прав можно управлять при помощи специальных флагов, определяющих порядок наследования:

- `ContainerInheritAce` — права распространяются на дочерние контейнеры;
- `ObjectInheritAce` — права распространяются на дочерние объекты;
- `InheritOnlyAce` — права распространяются только на потомков, но не на сам объект.

Таким образом, возможные сочетания этих флагов определяют многообразие типов наследования прав:

Кто наследует	**Сочетание флагов**
Этот объект, дочерние контейнеры и объекты	`ContainerInheritAce` `ObjectInheritAce`
Этот объект и дочерние контейнеры	`ContainerInheritAce`
Этот объект и дочерние объекты (не контейнеры)	`ObjectInheritAce`

Только дочерние контейнеры и объекты	`ContainerInheritAce`
	`ObjectInheritAce`
	`InheritOnlyAce`

Только дочерние контейнеры	`ContainerInheritAce` `InheritOnlyAce`
Только дочерние объекты (не контейнеры)	`ObjectInheritAce` `InheritOnlyAce`
Только этот объект	

Все стандартные объекты системы (карточки, секции, строки) являются **контейнерами** и именно в таком качестве рассматриваются при наследовании прав.

Программное назначение прав

Объекты связанные с реализацией модели безопасности расположены в специальном пространстве имён — `DocsVision.Platform.Security.AccessControl`, которое реализовано в отдельной сборке — `DocsVision.Platform.dll`. Поэтому для работы с безопасностью, эту сборку следует предварительно подключить к своему проекту, а также определить используемое пространство имён:

```
using DocsVision.Platform.Security.AccessControl;
```

Классы для работы с правами карточек, файлов, и хранимых процедур унаследованы от базового класса `DVObjectSecurity`, который содержит основные механизмы для работы с правами. Кроме этого, каждый из них содержит ряд специфических свойств, характерных именно для данного объекта (схемы прав). Этот класс по сути дела описывает дескриптор (`SD`), и включает в себя методы для работы с разрешениями (`ACE`), аудитами, и другими свойствами дескриптора.

Обращаем внимание, что численные значения некоторых элементов перечислений, описывающих права доступа были изменены относительно Docsvision 4.x. И в случае, если в системе имеется унаследованное от Docsvision 4.x решение, в котором работа с флагами прав доступа осуществлялась через использование целочисленных констант, а не через использование вышеописанных перечислений, то рекомендуется проверить соответствие использованных целочисленных констант актуальным значениям перечислений.

Общий алгоритм работы с правами в рамках данной модели можно сформулировать следующим образом:

Чтобы добавить новое разрешение на карточку, файл или процедуру необходимо:

1. Получить объект для работы с разрешениями `CardData.GetAccessControl`
2. Создать новое разрешение `CardDataAccessRule` с необходимыми атрибутами.
3. Добавить разрешение к описателю прав `AddAccessRule` или `SetAccessRule`.

4. Сохранить изменённый описатель прав `SetAccessControl`.

   В примере, для существующей карточки с идентификатором `00000000-0000-0000-0000-000000000000` добавляются права на чтение для пользователя `IvanovII`.

```
CardData card = userSession.CardManager.GetCardData(new System.Guid("00000000-0000-0000-0000-000000000000")); ①

CardDataSecurity cardDataSecurity = card.GetAccessControl(); ②

CardDataAccessRule cardDataAccessRule = new CardDataAccessRule("DOMAIN\\IvanovII", CardDataRights.Read, AccessControlType.Allow); ③

cardDataSecurity.SetAccessRule(cardDataAccessRule); ④

card.SetAccessControl(cardDataSecurity); ⑤
```

   ① Получение с сервера данных карточки с известным идентификатором.

   ② Получение описателя прав карточки.

   ③ Создание нового разрешения — совокупного права чтения.

   ④ Добавление нового разрешения к описателю прав.

   ⑤ Сохранение изменённого описателя прав.

Чтобы удалить разрешения на карточку, файл или процедуру, необходимо:

5. Получить объект для работы с разрешениями `CardData.GetAccessControl`.
6. Удалить разрешения для субъекта:
   a. Методом `PurgeAccessRules`, чтобы удалить **все** разрешения для субъекта
   b. Методом `RemoveAccessRuleSpecific`, чтобы удалить конкретное разрешение для субъекта.
   c. Методом `SetAccessRule`, чтобы удалить все разрешения для субъекта и добавить вместо них новое.
7. Сохранить изменённый описатель прав `SetAccessControl`.

   Ниже приведён пример, в котором у пользователя `IvanovII` отбираются права на карточку с идентификатором `00000000-0000-0000-0000-000000000000`.

```
CardData card = userSession.CardManager.GetCardData(new System.Guid("00000000-0000-0000-0000-000000000000"));
```

```
CardDataSecurity cardDataSecurity = card.GetAccessControl();

cardDataSecurity.PurgeAccessRules(new NTAccount("DOMAIN\\IvanovII")); ①

card.SetAccessControl(cardDataSecurity); ②
```

① Удаление всех прав пользователя.

② Сохранение изменённого описателя прав.

Значения битовых флагов стандартных прав приведены в таблице: "Битовые флаги стандартных прав доступа".

Использование отчётов

Отчёт — механизм, предоставляющий возможность получения прямого доступ к БД Docsvision с целью получения сложных выборок данных, повышения производительности работы Решения, получения данных из внешней БД и т.п.

Отчёт работает с данными, минуя модели безопасности Docsvision, поэтому задача обеспечения защиты данных, в данном случае, возлагается на разработчика.

Создание отчётов требует от разработчика понимания внутреннего устройства БД Docsvision.

Создание отчёта

Создание нового отчёта включает два этапа:

1. Разработка SQL-процедуры, реализующей требуемый механизм взаимодействие с данными в БД.
2. Настройка отчёта в схеме библиотеки карточек.

Отчёт представляет собой SQL процедуру вида:

```
BEGIN
 SELECT tblMy.Info1, tblMy.Info2
 FROM [dvtable_{SOME_DV_TABLE_GUID_HERE}] tblMy
 WHERE tblMy.Param1 = @Param1 and tblMy.Param2 = @Param2
 ORDER BY tblMy.Info2
END
```

В процедуре можно использовать параметры (с символом `@`), значения которых

будут переданы при вызове отчёта. Параметры объявлять не требуется.

Процедура может возвращать или не возвращать значения. Возвращаемые значения могут представлять собой единственное значение или таблица.

Процедура должна размещаться в блоке `BEGIN .. END`.

Заголовок хранимой процедуры и описания параметров будут сгененированы автоматически.

Процедура должна быть сохранена в файл `.sql` в подпапку (например, "Sql") в одном каталоге со схемой библиотеки карточек.

```
CardLib
|  MyCardLib.xml
|  MyCard.xml
|--Sql
   |  MyProcedure1.sql
   |  MyProcedure1.sql
```

Регистрация отчёта в схеме библиотеки карточек выполняется с помощью программы "CardManager". Инструкция по работе с программой приведена в Руководстве по работе с комплектом утилит Resource Kit, см. раздел "/dv6/resource-kit/6.1/cardmanager/util/[CardManager]".

В качестве примера далее будет выполнена регистрация отчёта `GetLogMessages`, возвращающего записи из журнала Docsvision за определённый период:

```
BEGIN
    SELECT appLog.EmployeeID, appLog.Date, appLog.Data
    FROM [dbo].[dvsys_log_application] appLog WITH(NOLOCK)
    INNER JOIN [dbo].dvsys_users users WITH(NOLOCK)
    ON (users.UserID = appLog.UserID)
    WHERE appLog.Date BETWEEN @DateFrom AND @DateTo
END
```

Чтобы настроить новый отчёт:

1. Откройте в программе "CardManager" свойства собственной библиотеки карточек и перейдите на вкладку *Reports*.

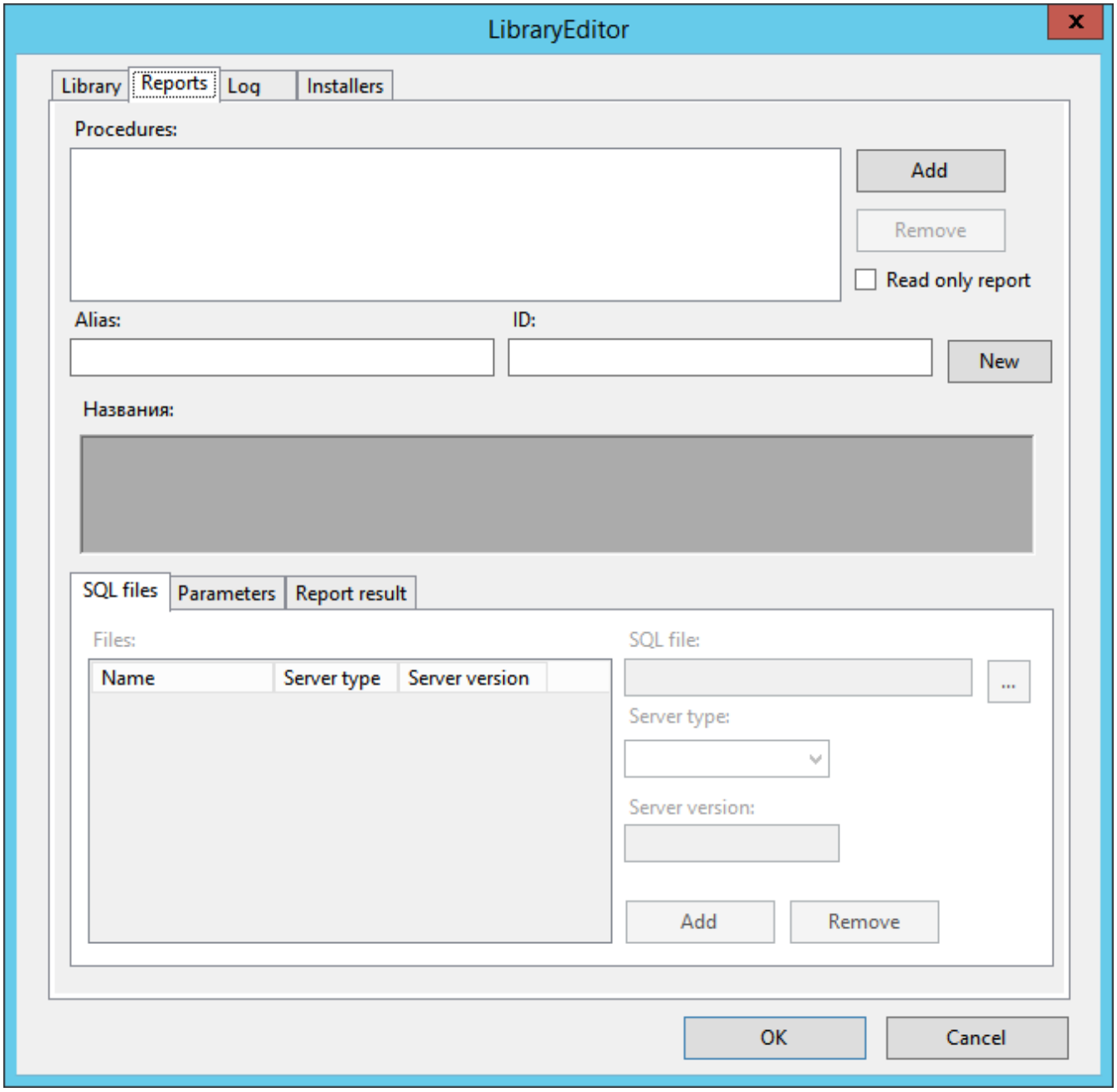


Рисунок 6. Вкладка настройки отчётов библиотеки карточек в программе "CardManager"

2. Добавьте новый отчёт в схему и укажите основные свойства отчёта.

   a. Нажмите кнопку **Add** в блоке *Procedures*. В список *Procedures* будет добавлена новая запись.

   b. В поле *Alias* введите псевдоним отчёта.

   c. В таблице *Названия* введите локализованные названия отчёта.

   В поле ID будет указан автоматически сгенерированный идентификатор отчёта, по которому отчёт может быть вызван с использованием API.

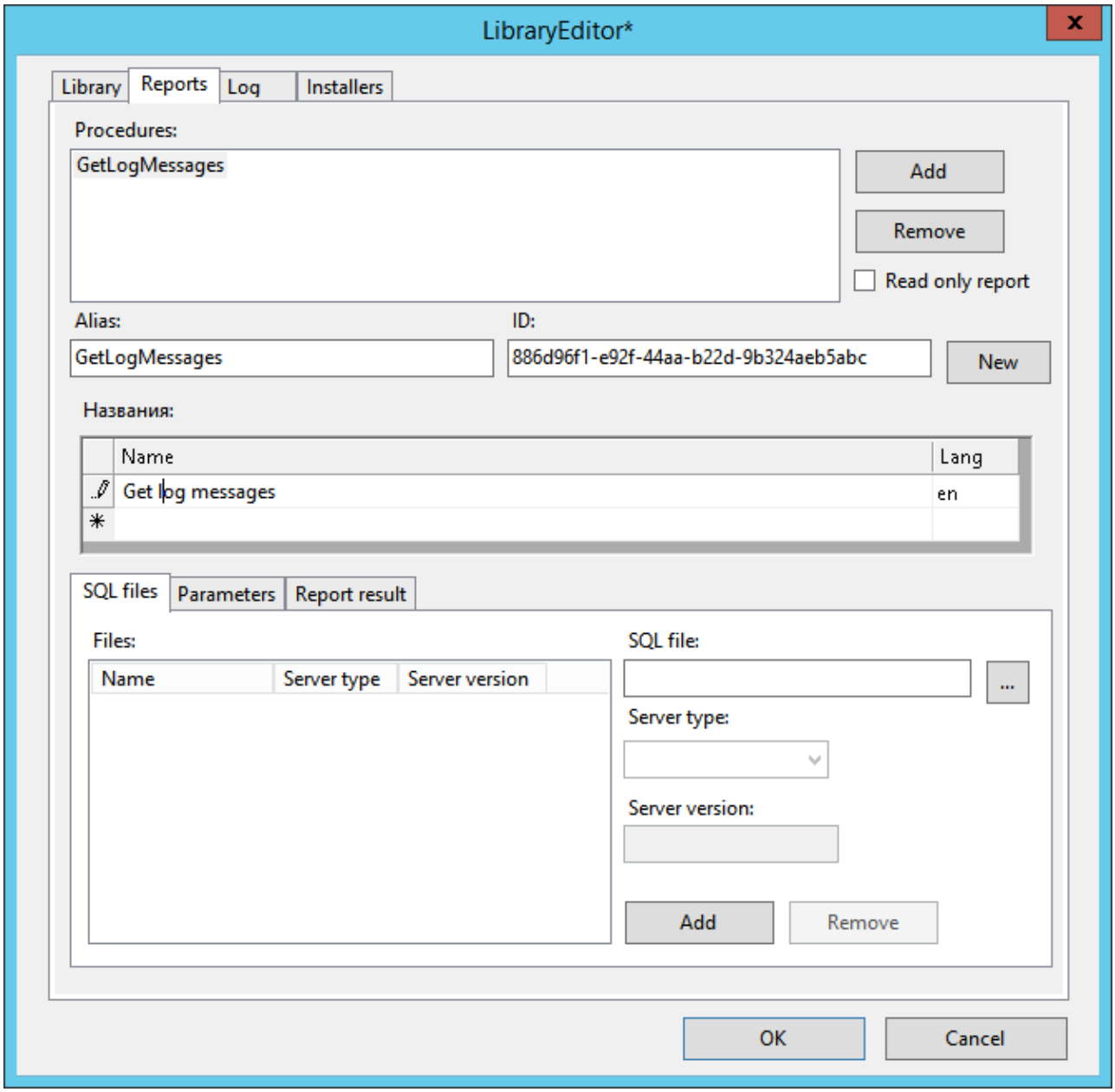


Рисунок 7. Пример настройки основных свойств отчёта

3. Добавьте в отчёт файлы с SQL процедурами на вкладке *SQL files*.
   a. Нажмите кнопку **Add** на данной вкладке. В список *Files* будет добавлена новая запись.
   b. В поле *SQL file* выберите разработанный ранее SQL-файл с процедурой отчёта. Приведите путь к файлу к относительному виду.
   c. В списке *Server type* выберите тип СУБД, под управлением которой работает БД Docsvision, для которой создаётся отчёт: ***Microsoft SQL*** или ***PostgreSQL***.

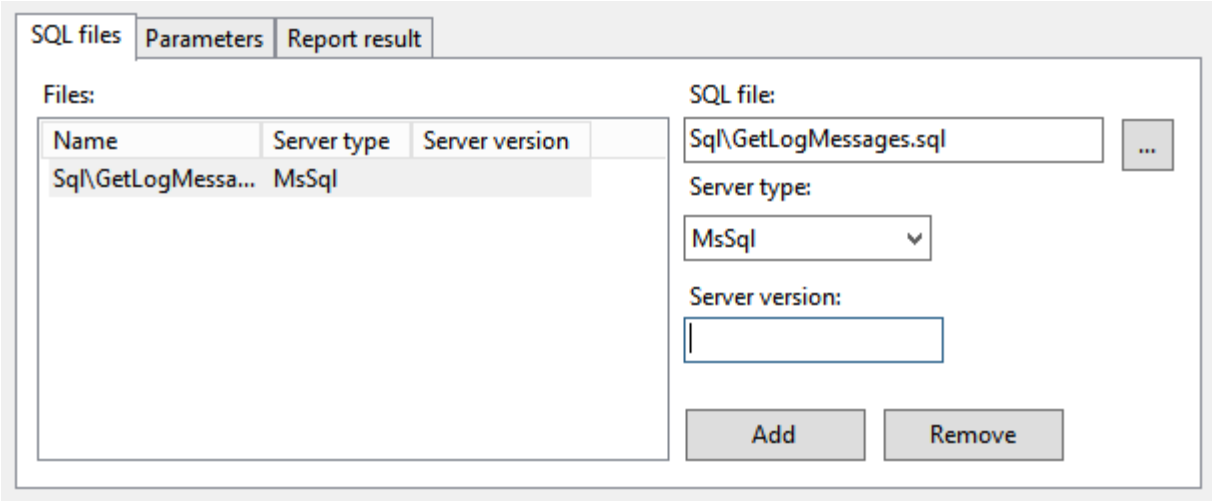


Рисунок 8. Пример настройки файлов отчёта

4. Настройте параметры отчёта на вкладке *Parameters*.
   a. Нажмите кнопку **Add** на данной странице. В список *Parameters* будет добавлена новая запись.
   b. В поле *Parameter* name введите название параметра, которое ожидается в соответствии с разработанной SQL-процедурой.
   c. В поле *Parameter type* укажите тип параметра, соответствующий типу параметра, используемого в SQL-процедуре.
   d. Аналогичным образом добавьте все параметры, ожидаемые разработанной SQL-процедурой.

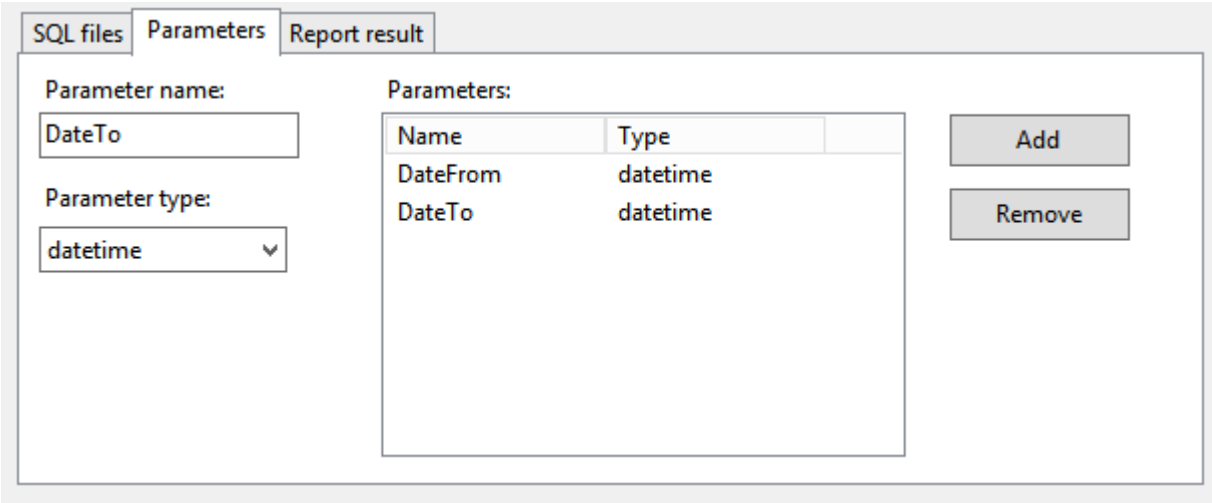


Рисунок 9. Пример настройки параметров отчёта

5. Настройте возвращаемые данные отчёта на вкладке *Report result*.
   a. Выберите формат возвращаемого значения:
      - ***Scalar result*** — если возвращается единственное значение.

- ***TableResult*** — если возвращается коллекция значений.

b. Если выбран формат ***Scalar result***, укажите:

   - *Data type* — тип возвращаемых данных.
   - *Precision*, *Scale*, *Size* — точность, масштаб и длина (см. подробнее на сайте Microsoft.).
   - *Nullable* — если может быть пустым.

c. Если выбран формат ***TableResult***:

   i. Нажмите кнопку **Add**. В список *Columns* будет добавлена новая запись.

   ii. В поле *Name* введите название возвращаемого процедурой поля.

   iii. Укажите (см. описание выше): *Data type*, *Precision*, *Scale*, *Size* и *Nullable*.

   iv. Аналогичным образом добавьте **все** возвращаемые процедурой поля. Данное условие критически важно при работе с PostgreSQL.

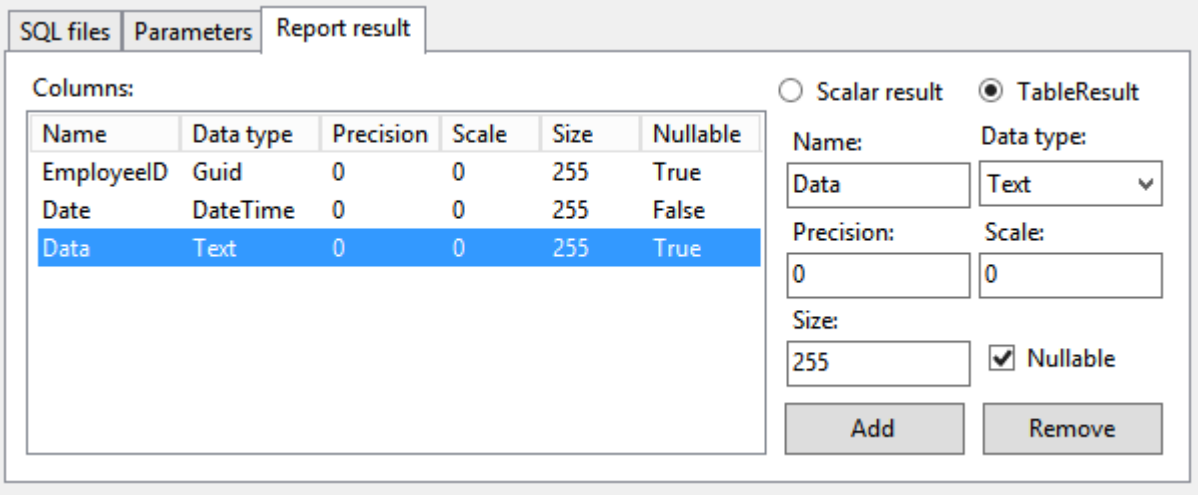


Рисунок 10. Пример настройки результатов отчёта

6. Сохраните настройки библиотеки карточек и загрузите её в Docsvision стандартным образом.
7. Перезапустите сервер Docsvision.

Вызов отчёта

Для работы с отчётами API Docsvision предоставляет менеджер отчётов типа `ReportManager`.

Все отчёты содержатся в поле *Reports* класса `ReportManager`.

Получить требуемый отчёт можно по его идентификатору, который был присвоен в схеме библиотеки карточек:

```
var report = userSession.ReportManager.Reports[Guid.Parse("886d96f1-e92f-44aa-b22d-9b324aeb5abc")];
```

Значения параметров отчёта устанавливаются в поле `Parameters`*:*

```
report.Parameters["DateFrom"].Value = DateTime.Parse("01.01.2012");
report.Parameters["DateTo"].Value = DateTime.Parse("01.01.2019");
```

Выполнение отчёта запускается методом `report.GetData()`, который возвращает набор строк с результатами выполнения.

Следующий код демонстрирует пример вызова разработанного отчёта с отображением записей журнала Docsvision в консоли.

```
var report = userSession.ReportManager.Reports[Guid.Parse("886d96f1-e92f-44aa-b22d-9b324aeb5abc")]; ①

report.Parameters["DateFrom"].Value = DateTime.Parse("01.01.2012");
report.Parameters["DateTo"].Value = DateTime.Parse("01.01.2019"); ②

InfoRowCollection results = report.GetData(); ③

foreach (InfoRow result in results)
{
  Console.WriteLine("{0} | {1} | {2}", result["EmployeeID"].ToString(),result["Date"
].ToString(), result["Data"].ToString()); ④
}
```

① Получение отчёта с идентификатором `886d96f1-e92f-44aa-b22d-9b324aeb5abc`.

② Заполнения параметров отчёта — начальной и конечной даты выборки.

③ Выполнение отчёта.

④ Отображение результатов выполнения отчёта в консоли.

Способ отображения данных отчёта в пользовательском интерфейсе реализуется разработчиком самостоятельно.

Работа с нумераторами

Для упрощения автоматической нумерации документов в системе

предусмотрены специальные объекты — *нумераторы*.

Нумераторы выполняют следующие функции:

- Выделение диапазонов номеров для документов.
- Ассоциация (закрепление) диапазонов номеров за пользователями системы.
- Автоматическое выделение номеров новым документам.
- Резервирование номеров.
- Контроль и освобождение занятых и зарезервированных номеров.

В системе может одновременно существовать несколько нумераторов.

Каждый нумератор отвечает следующим принципам:

- Определяет непрерывное пространство номеров (например, 1…1000).
- Пространство номеров может содержать несколько диапазонов, закрепленных за конкретными пользователями системы (например, 1..100, 500..800).
- Может содержать несколько именованных зон, номера в которых могут совпадать (например, "2004 год", "2005 год").
- Каждый номер может обладать собственным статусом:
  - Свободен.
  - Занят.
  - Зарезервирован.

Для реализации работы с нумераторами в состав платформы входит специальная системная карточка нумератора, а в менеджере объектов определён набор объектов для выполнения операций с карточкой нумератора.

Карточка нумератора

Системная карточка нумератора `NumeratorCard` содержит свойства и методы для работы с конкретным экземпляром нумератора. Для создания нового нумератора требуется создать экземпляр этой карточки.

Зона нумератора

Зона нумератора представляет собой именованную совокупность номеров. Внутри одного нумератора может существовать несколько параллельных зон, номера в которых совпадают. Вся работа, связанная с получением или освобождением номеров, производится именно в конкретной зоне нумератора.

Для работы с зонами в менеджере объектов предусмотрен объект `NumeratorZone`.

Диапазон нумератора

Диапазон нумератора представляет собой подмножество номеров, закрепленных за определённым пользователем системы. Для работы с диапазонами в менеджере объектов предусмотрен объект `NumeratorRange`.

Таким образом, для работы с нумератором следует выполнить следующие действия:

- Создать экземпляр карточки нумератора.
- Определить границы пространства номеров.
- Создать в нумераторе как минимум одну нумерованную зону.
- При необходимости добавить один или несколько диапазонов номеров.

Пример работы с нумератором:

```
const string NUMERATOR_CARD_TYPE = "959FF5E2-7E47-4F6F-9CF6-E1E477CD01CF"; ①

NumeratorCard numerator = (NumeratorCard)session.CardManager.CreateCard(new Guid
(NUMERATOR_CARD_TYPE)); ②

numerator.ChangeStartNumber(1, true);
numerator.ChangeEndNumber(1000, true);
numerator.Name = "MyNumerator"; ③

NumeratorZone zone = numerator.Zones.AddNew(DateTime.Now.Year.ToString()); ④

Guid outId;
int number = zone.GetNumber(Guid.Empty, true, out outId); ⑤
```

① Константа, идентификатор типа карточки нумератора.

② Создание новой карточки нумератора.

③ Первичная инициализация нумератора.

④ Создание зоны.

⑤ Получение номера.

Работа с представлениями

Механизм представлений является частью функциональности платформы, предназначенной для построения отчётов. Этот механизм предоставляет возможность получать произвольные выборки данных, описывая их структуру

при помощи визуального редактора, и отображать полученные результаты в пользовательском интерфейсе Windows-клиента, с возможностью их форматирования (подсветки, группировки и так далее).

Можно выделить две основные задачи, которые разработчик на платформе может решать с использованием представлений:

- Программное создание новых описаний представлений, которые затем будут доступны пользователям в интерфейсе Windows-клиента.
- Получение данных конкретного представления, на основании уже имеющегося описания, и их последующая обработка.

В ряде случаев использование представлений эффективно также для повышения производительности работы решения. Например, если нужно получить большую выборку данных (карточек) со сложной структурой взаимосвязей, вместо использования нескольких поисковых запросов будет более эффективно получить данные одним вызовом представления.

При работе с представлениями, имеет смысл оперировать такими понятиями, как описание представления (это схема, которая определяет принципы выборки данных с конкретной структурой) и данные представления (это результат применения запроса, сгенерированного на основании описания, к конкретной базе данных в конкретный момент времени). На основании одного описания, можно произвольное количество раз получать данные, которые будут иметь одинаковую структуру (число и состав столбцов), но разное содержание (число и состав строк).

В свою очередь, каждое описание представления разделено на две логические части:

- Описание (схема) данных — эта часть определяет, какие именно данные и каким способом необходимо выбрать из базы данных.

  Это описание в формате XML физически хранится в специальной системной карточке сохранённых представлений (`SavedViewCard`). При каждом изменении этой схемы происходит автоматическая генерация хранимых процедур в БД, которые предназначены для получения собственно данных (эти процедуры имеют имена `dvview_get_data_ID-представления)`.

- Описание форматирования — эта часть определяет, как именно полученные данные нужно показать в пользовательском интерфейсе (имена и порядок колонок, группировки, сортировки и т.д.).

Это описание является частью другой системной карточки — Windows-клиента (`NavigatorCard`) и хранится в специальной её секции `ViewSettings`.

Такой принцип разграничения сущности данных от их визуализации потенциально позволяет реализовать и другие формы отображения для одних и тех же данных. К примеру, штатным способом данные представления можно просматривать в Windows-клиенте (и поэтому он хранит собственное описание форматирования данных), однако потенциально возможно реализовать другой клиент (например, для мобильных устройств), который будет отображать те же данные в другом формате (с учетом ограничений мобильных устройств).

В данном разделе

- Описание (схема) данных представления.
- Описание (схема) форматирования.
- Получение данных представления.

Описание (схема) данных представления

Описание данных представления хранится в специальной системной карточке и представляет собой XML-документ (так же как схемы карточек и библиотек). Однако, работать с этим описанием, оперируя непосредственно узлами и атрибутами XML-документа, было бы слишком сложно и неудобно. Поэтому в объектной модели системы предусмотрен ряд специальных объектов, которые позволяют работать с описанием представления уже на уровне семантических объектов (полей карточек, операций с ними).

Базовым объектом в этой модели является объект `View`. Этот объект является создаваемым (например, его можно создать при добавлении в систему нового описания представления). Также он может быть проинициализирован на основании существующего описания представления, сохранённого в карточке `ViewCard`.

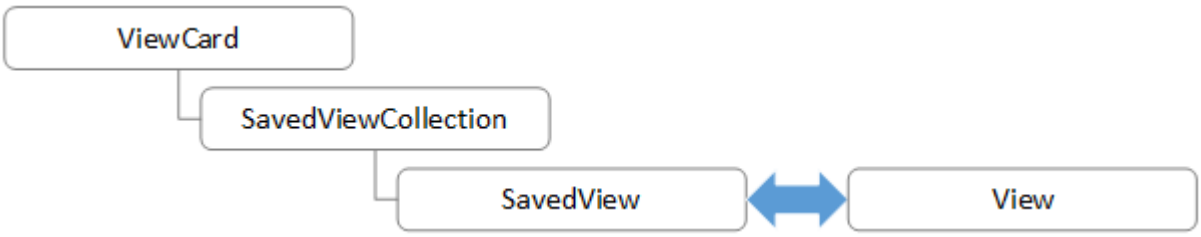


Рисунок 11. Доступ к представлению из карточки представления

Для загрузки существующего описания представления используется вспомогательный объект `SavedView` (сохранённое описание представления),

который можно получить из карточки сохранённых представлений (ViewCard).

Карточка сохранённых представлений является справочником, поэтому её экземпляр всегда можно получить:

```
const string VIEW_CARD_ID = "17F8F0B3-7E93-45E9-B250-EED4E93F3FA3";
ViewCard vcard = (ViewCard)session.CardManager.GetDictionary(new Guid(VIEW_CARD_ID));
```

Из коллекции сохранённых описаний объекта свойство Views, необходимое выбрать нужное представление и вызвать метод SavedView.Export, чтобы на выходе получить низкоуровневый объект View для непосредственного редактирования описания представления.

Отдельно стоит рассмотреть использование виртуальных полей. Виртуальные поля представляют собой наборы правил, согласно которым можно извлечь данные из физических таблиц базы данных. Для этого необходимо использовать вычисления, агрегации, присоединении и т.д. Определения виртуальных полей привязаны к конкретному типу секции, от которой начинается процесс вычислений.

Виртуальные поля могут быть определены разработчиком в процессе создания схемы карточки (такие поля всегда будут доступны для использования во всех представлениях). Доступ к определениям этих полей можно получить через объектную модель метаданных CardType.VirtualFields. Элементы этой коллекции доступны только для чтения, так как они жестко определены в схеме карточки.

Также виртуальные поля можно создать (выделить) в процессе построения конкретного представления, и сохранить как отдельные объекты для дальнейшего использования в других представлениях по аналогичной секции. Такие виртуальные поля создаются на основе готовых элементов представлений ViewElement при помощи метода ExtractVirtualField. Они сохраняются в той же системной карточке ViewCard, что и описания представлений, и доступны для последующего повторного использования через её объектную модель. Для работы с такими виртуальными полями используется другой тип объектов — SavedVirtualField.

Пример добавления предварительно сохранённого виртуального поля к описанию представления:

```
const string SECTION_TYPE_ID = "{8C77892A-21CC-4972-AD71-A9919BCA8187}";
```

```
SavedVirtualFieldCollection savedColl = viewcard.GetVirtualFields(new Guid
(SECTION_TYPE_ID)); ①

SavedVirtualField  savedField = savedColl["Performer"]; ②

ViewElement element;
element.MergeVirtualField(savedField.Export(), "Performers", "Performers"); ③
```

① Получение сохранённых определений виртуальных полей для конкретного типа секции.

② Получение поля с псевдонимом `Performer`.

③ Добавление поля к описанию представления.

Типичный алгоритм создания нового описания представления может включать в себя следующие шаги:

1. Определить количество и состав колонок будущего представления `View.Columns`, задать порядок сортировки данных в колонках `View.Sorting`.
2. Сформировать как минимум один элемент представления `ViewElement`, определяющий источники данных для колонок:

   a. Задать основную секцию карточки, из которой будут извлекаться данные `SectionId`.

   b. Физические поля секции, которые выводятся в представление в непосредственном виде, перечислить в коллекции `SectionFields`.

   c. Если для данной секции ранее были определены виртуальные поля (в схеме карточки или в конкретной базе данных), то загрузить их описание при помощи метода `MergeVirtualField`.

   Чтобы вывести данные из полей другой секции или таблицы:

   d. Присоедините эту секцию (или таблицу) к основной секции в коллекции `JoinDefinitions`.

   e. Выбрать поля присоединенной секции (или таблицы), добавив их в коллекцию `SectionFields` вместе с псевдонимом присоединенной секции.

3. Если представление должно содержать данные, не присутствующие в физических полях секций, то такие данные нужно оформить в виде вычисляемых полей `ComputedFields`.

   Для каждого вычисляемого поля `ComputedFileld` необходимо:

   a. Определить набор физических источников данных (`ComputationParts`),

каждый их которых может быть значением физического поля секции или результатом выполнения арифметической функции.

b. Определить агрегацию (`Aggregation`), выполняемую над физическими источниками данных.

c. Определить операцию (`Operation`), выполняемую над физическими источниками данных.

d. Задать ожидаемый тип результата вычислений (`Type`).

e. При необходимости можно сопоставить каждому из возможных результатов вычислений конкретное значение, которое будет итоговым результатом (`ResultSwitch`).

Пример кода: Создание нового представления.

При построении представления можно использовать псевдонимы трёх предопределённых секций:

- `Main` — базовая секция представления.
- `Inst` — строка из системной таблицы **dvsys_instances**, соответствующая текущей карточке.
- `Shortcuts` — строка из секции ярлыков карточки папок, соответствующая текущей карточке.

Описание (схема) форматирования

Важной частью представления, помимо получения данных, является их форматирование (а также, возможно, дополнительная обработка) при отображении в конечном клиентском приложении.

Windows-клиент, который является базовым средством для работы с представлениями, предлагает следующие возможности обработки и форматирования данных:

- Фильтрация данных (при помощи дополнительных фильтров).
- Агрегация (вычисление статистических величин).
- Группировка данных в таблице.
- Сортировка данных в таблице.
- Указание шрифта для таблицы.

- Разметка линий сетки в таблице.
- Указание ширины строк в таблице.
- Задание имён колонок.

Для работы с настройками представлений Windows-клиента в объектной модели предусмотрен объект `ViewSettings`, доступный из карточки Windows-клиента `NavigatorCard`.

Каждому конкретному представлению соответствуют индивидуальные настройки отображения, поэтому общее число объектов `ViewSettings` соответствует количеству настроенных в системе представлений.

Пример кода работы с настройками представления:

```
SavedView view = viewcard.Views[0]; ①

const string NAVIGATOR_CARD_TYPE = "{A7F9784B-96A4-4B3E-B820-2E714A2A1463}";
NavigatorCard nav = (NavigatorCard)session.CardManager.GetDictionary(new Guid
(NAVIGATOR_CARD_TYPE)); ②

ViewSettings settings = nav.ViewSettings[view.Id]; ③

foreach(ViewColumnSettings col in settings.Columns)
{
 MessageBox.Show(col.Caption); ④
}
```

① Получение первого попавшегося представления.

② Получение карточки Windows-клиента.

③ Получение настроек представления.

④ Перебор всех колонок и вывод их названий.

Получение данных представления

Другой важной задачей при работе с представлениями является получение конкретной выборки данных на основании предварительно подготовленного (программно или через пользовательский интерфейс) описания представления.

Чтобы получить данные представления, необходимо воспользоваться методом `CardManager.GetViewData`.

Пример получения данных системного представления Дайджест, примененного к конкретной папке:

```
var viewReadRequest = session.CardManager.GetViewRequestFromSearch(string);
viewReadRequest.CursorType = CursorType.Keyset;
viewReadRequest.ViewId = viewId;

/*что-либо ещё*/

InfoRowCollection infoRows = session.CardManager.GetViewData(viewReadRequest);
```

Пример получения данных представления по XML-описанию на основании результатов сформированного поискового запроса:

```
SearchQuery query = session.CreateSearchQuery(); ①

ViewSource source = ViewSource.FromSearch(query.GetXml(true, null));
InfoRowCollection viewData = session.CardManager.GetViewData(source, view.GetXml()); ②
```

① Формирование поискового запроса.

② Получение данных представления.

Метод возвращает `InfoRowCollection` — набор строк, соответствующий результатам запроса. Возможность его сортировки в объектной модели не предусмотрена, то есть сортировать его можно только внешними средствами.

Работа со ссылками

Ссылка — это специальный тип поля, которое ссылается на другую карточку (тип `RefCardID`) или на строку другой секции (`RefID`).

Ссылка на карточку может быть одного из четырёх типов: простая, слабая, сильная и автоматическая. Тип ссылки задаётся в описании карточки и, следовательно, устанавливается один раз для всех экземпляров карточки такого типа. Управлять поведением ссылки во время исполнения нельзя.

Тип ссылки влияет на поведение сервера в следующих случаях:

- При изменении значения ссылочного поля.
- При удалении карточки, на которую существуют ссылки.

Рассмотрим поведение различных типов ссылок подробнее.

Простая ссылка

Простая ссылка позволяет просто сохранить в поле идентификатор связанной карточки и определить ссылочные поля. Сервер не выполняет никаких дополнительных действий при записи или удалении значения в этом поле, а также при удалении связанной карточки. В этом смысле, простая ссылка аналогична полю типа `uniqueid(System.Guid)`. Контроль ссылочной целостности в этом случае полностью остается за разработчиком.

Слабая ссылка

Слабая ссылка имеет отличие от простой — при удалении связанной карточки соответствующее поле ссылки будет автоматически обнулено (установлено в значение NULL). Это позволит разработчику не заботиться о проблеме "зависших" ссылок на несуществующие карточки.

Дополнительно появляется возможность узнать, какие карточки ссылаются на данную и на какие карточки ссылается она сама. Для этого предназначены методы `GetLinksToCard` и `GetLinksFromCard` класса `CardManager`.

Сильная ссылка

Сильная ссылка выражает отношения владения — если карточка держит сильную ссылку на другую карточку, то можно сказать, что она ею владеет или является родительской по отношению к ней.

Характеристики сильных ссылок:

- Сильных ссылок на карточку может быть несколько, то есть несколько карточек могут "владеть" одной карточкой.
- При создании сильной ссылки на карточку на неё распространяются (наследуются) права из карточки-владельца.
- Если на карточку владельца назначаются права, то при распространении прав вниз по иерархии они распространяются и на подчинённую карточку.

  Таким образом, подчинённая карточка имеет набор прав (и, соответственно, видимость) в зависимости от того, на какую из карточек владельцев были последними назначены права. Такой принцип используется потому, что в Windows нет множественного наследования, и система сама поступает так при реализации аналога сильных ссылок на файлы.

- Если на подчинённую карточку есть сильные ссылки, то её напрямую удалить нельзя. При удалении карточки-владельца сервер отмечает удаление сильной

ссылки. Если это последняя сильная ссылка на карточку, то она сама тоже удаляется. Таким образом, подчинённая карточка гарантированно живет столько, сколько самый последний из её владельцев.

Если возникает необходимость очистить значение сильной ссылки, но при этом сохранить карточку, то надо воспользоваться методом `ClearLink`. Сильные ссылки нужны для реализации владения, контроля области видимости и срока жизни карточек.

Автоматическая ссылка

Автоматическая ссылка по своему поведению подобна слабой, но обладает следующим дополнительным свойством — при удалении связанной карточки из исходной карточки будет удалена вся строка, содержащая ссылку.

Например, в стандартных карточках документов "Делопроизводства" есть секция *Файлы и ссылки*, содержащая ссылки на другие карточки. Все ссылки там имеют тип *Авто*. Таким образом, при удалении карточки из системы все существующие ссылки на неё из карточек документов автоматически удаляются.

Ссылки на строки

Ссылка на строку (`RefID`), в отличие от ссылки на карточки, не имеет типа и поэтому не обладает возможностями контроля целостности. Единственным способом управлять поведением связанного объекта для такого рода полей является признак **Удалять связанную строку при изменении значения** (Delete linked row on value change), который должен быть установлен при создании схемы карточки в программе "CardManager", входящей в Resource Kit. Если этот признак установлен, то любое изменение значения заполненного ссылочного поля (установка нового значения или сброс в null) приведёт к тому, что связанная строка (другой карточки) будет удалена.

Это поведение имеет смысл, если связанная строка расположена в коллекционной или иерархической секции.

Типичным сценарием применения этого поведения является, например, работа с номерами в карточках "Делопроизводства". Если карточка занимает какой-то номер (ей соответствует строка в секции занятых номеров карточки нумератора), то при удалении карточки занятый ею номер автоматически освобождается (строка в карточке нумератора удаляется).

Управление режимом загрузки данных и кэширование

Режим загрузки данных устанавливается для всех секций карточки на этапе создания её схемы, но может быть переопределен свойствами: `CardData.FetchMode` (действует на всю карточку) и `SectionData.FetchMode` (действует на конкретную секцию).

В приведенном примере Схема карточки определяет механизм загрузки её данных из базы данных, Схема карточки определяет режим загрузки её данных из базы данных.

Для сценариев, связанных с получением данных карточки, актуально такое понятие, как режим загрузки данных, который определяет порцию информации, загружаемой с сервера Docsvision, за одно обращение. Режим загрузки может иметь одно из следующих значений (см. описание перечисления `FetchMode`):

- `Card` — будут загружены данные всех секций карточки (актуально только для карточки в целом).
- `Section` — при обращении к какой-либо строке секции будут загружены все её строки, в т.ч. данные её подсекций.
- `SubSection` — будут загружены данные одной подсекции, к которой идет обращение.
- `Level` — будут загружены данные только запрашиваемого уровня иерархической секции.

Для повышения производительности данные читаются не по одной строке, а блоками — уровнями или подсекциями. Сделано это из соображений о том, что задержки на передачу данных содержат в себе константу — стоимость сетевого вызова. Таким образом, иногда выгоднее получить больше данных за один раз, чем несколько раз обращаться к серверу за маленькими порциями.

С другой стороны, могут существовать карточки, которые содержат сотни тысяч строк (например, справочники). Попытка загрузить все данные такой карточки за один вызов приведёт к чрезмерной задержке, поэтому в такой ситуации более оптимальным было бы получать данные меньшими порциями по мере их востребованности.

Режим загрузки данных определяется для всей карточки в целом на этапе создания схемы данных в программе "CardManager", входящей в Resource Kit и

распространяется на все её секции. Однако, в режиме исполнения (runtime) разработчик может переопределить это поведение (для карточки или для конкретных секций). Для этого служит свойство `FetchMode` объектов `CardData` и `SectionData`.

Режим построчной загрузки (`FetchMode.Row`) в данный момент не реализован. Однако получить данные для отдельных строк все же можно. Для этого у объекта `SectionData` есть набор методов, работающих с одиночной строкой. Их использование позволяет не загружать на клиента ненужную информацию, что, конечно же, предпочтительнее с точки зрения производительности.

В зависимости от того к каким данным карточки требуется доступ, можно выделить несколько объектов реализующих его:

- `CardData` — точка входа к основным данным карточки (контейнеру).
- `SectionData` — доступ к секциям карточки.
- `SubSectionData` — доступ к подсекциям карточки.

Кэширование данных

Говоря про загрузку данных, нельзя не упомянуть про кэширование. В `ObjectManager` имеется кэш на чтение и, таким образом, повторное чтение строк вернёт уже загруженные данные. При этом надо понимать, что строки в кэше могли быть изменены на сервере, а это может привести к тому, что на клиенте окажутся неактуальные данные.

Для этого в `ObjectManager` предусмотрен механизм обновления строк:

- Явный — у всех объектов есть метод `Refresh` (возможно, с параметрами), с помощью которого можно обновить данные вручную.
- Неявный — при получении карточки (`CardData`) производится проверка наличия данных карточки в кэше. Если данные есть, но они потеряли актуальность, например на сервере произошло изменение строки, принадлежащей карточке, вызывается процедура обновления.

Неявный характер обновления данных во втором случае нужно всегда иметь в виду и избегать частого повторного получения данных карточки. В этом случае всегда производится серверный вызов — обновляется информация о статусе карточки (если такая карточка уже получалась ранее).

Существует также возможность очистить весь кэш одним вызовом — для этого

предназначен метод `PurgeCache` экземпляра `CardManager`.

Режим отложенных изменений

Отложенные изменения в базовом API

При изменении данных в обычном режиме работы автоматически вызывается серверный метод, отвечающий за запись изменений в базу данных. Такой вызов будет выполнен при каждом изменении значения поля карточки, что может привести к значительным накладным расходам при большом количестве изменений.

Для оптимизации подобных сценариев, связанных с изменением большого количества значений, предусмотрен т.н. *режим отложенных изменений*, при котором все изменения накапливаются на клиентской стороне и передаются на сервер одним вызовом.

Режим отложенных изменений устанавливается вызовом у объекта, реализующего интерфейс `IUpdatable`, метода `BeginUpdate`:

```
CardData card = userSession.CardManager.GetCardData(new Guid("00000000-0000-0000-0000-
000000000000")); ①

card.BeginUpdate(); ②
```

① Получение данных карточки.

② Включение режима отложенных изменений для полученной карточки.

Когда в карточку были внесены изменения, их нужно зафиксировать в базе данных, вызвав метод `EndUpdate`:

```
card.EndUpdate();
```

Также существует возможность отменить несохранённые изменения, вызвав метод `CancelUpdate`:

```
card.CancelUpdate();
```

Режим отложенных изменений поддерживает не только карточка, но и другие объекты, реализующие интерфейс `IUpdatable`.

Общий список объектов, поддерживающих режим отложенных изменений, следующий:

- `CardData`.
- `SectionData`.
- `SubSectionData`.
- `RowData`.
- `RowDataCollection`.
- `FileData`.
- `Folder`.
- `Shortcut`.

У данного режима есть несколько важных особенностей:

- Права доступа и ограничения для изменяемых объектов проверяются не сразу, а в момент обработки пакета изменений на сервере.
- Множество изменений одних данных не объединяются (не заменяют друг друга), а накапливаются. Например, если значение одного поля было изменено 10 раз, то в итоговой пакет будет записано 10 транзакций, а после получения пакета сервером будут выполнены все 10 операций.

Также следует помнить, что в пределах секции режим отложенных изменений устанавливается в т.ч. и на все подчиненные объекты — подсекции, строки и поля. Если строки находятся в режиме отложенных изменений, то после перевода в этот режим родительской секции их изменения будут учитываться в контексте родительского объекта. В обратную сторону наследование не распространяется.

Если режим отложенных изменений включён для родительского объекта, то управление режимом из подчиненного объекта не допускается, т.е. нельзя, к примеру, отменить режим для секции, если он установлен в карточке.

Как правило, режим отложенных изменений устанавливается на всю карточку, так как по умолчанию вся карточка блокируется пользователем для изменения. Но возможны случаи, когда это не так, например в случае работы со справочниками.

Практический пример доступен по ссылке Пример использования режима отложенных изменений

Отложенные изменения в объектной модели

API уровня бизнес-логики по умолчанию работает в режиме, эквивалентном режиму отложенных изменений, т.е. фактически изменения накапливаются на клиентской стороне и отправляются на сервер одним вызовом. Причем в отличие от режима отложенных изменений базового API, в объектной модели данный режим действует сразу для всех объектов, с которыми выполняются изменения.

Для работы с отложенными изменения на данном уровне API предусмотрено несколько методов:

- `ObjectContext.AcceptChanges` — сохраняет все изменения в контексте объектов:

```
BaseCard card = objectContext.GetObject<BaseCard>(new Guid("00000000-0000-0000-0000-000000000000")); ①

card.Description = "Новое описание"; ②

BaseCard otherCard = objectContext.GetObject<BaseCard>(new Guid("00000000-0000-0000-0000-000000000001")); ③

otherCard.Description = "Новое описание для другой карточки"; ④

objectContext.AcceptChanges(); ⑤
```

① Получение карточки.

② Изменение данных.

③ Получение другой карточки.

④ Изменение данных другой карточки.

⑤ Сохранение всех изменений контекста объектов.

- `ObjectContext.SaveObject` — сохраняет изменения одного конкретного объекта:

```
objectContext.SaveObject(card); ①
```

① Сохранение изменений в изменённом объекте `card`.

- `ObjectContext.RollbackChanges` — отменяет все несохранённые изменения контекста объектов:

```
objectContext.RollbackChanges();
```

- `ObjectContext.RollBackObject` — отменяет несохранённые изменения одного объекта:

```
objectContext.RollBackObject(card);
```

Контекст объектов также предоставляет методы для сохранения и отмены изменения нескольких карточек. Полный список доступных методов см. в описании типа `ObjectContext`.

Журналирование

Функциональность журналирования обеспечивает единый механизм для фиксации всех операций, производимых в системе, и возможность их последующего анализа.

Каждая запись в журнале может относиться к одному из следующих типов:

- Информация.
- Предупреждение.
- Аудит.
- Ошибка.

Журнал Docsvision разбит на три разных части:

- Система — записи в этом журнале фиксирует сервер Docsvision. Сюда могут попадать как ошибки, так и аудит низкоуровневых операций, выполняемых сервером Docsvision.
- Безопасность — в этом журнале фиксируются результаты аудит операций над объектами, для которых он явно настроен.
- Приложения — в этот журнал попадают операции, фиксируемые конкретными приложениями (Делопроизводство, Управление процессами, и т.д.). Сюда же попадают все сообщения от решений сторонних разработчиков.

В системном журнале (Система) автоматически фиксируются следующие низкоуровневые операции сервера:

- Операции с карточками:
  - Создание карточки.
  - Пометка карточки как удалённой.
  - Пометка карточки как прочитанной.

- Удаление карточки.
- Копирование карточки.
- Изменение прав карточки.
- Добавление карточки к теме обработки.
- Архивирование карточки.
- Разархивирование карточки.
- Открытие карточки (если оно производилось из Windows-клиента).
- Экспорт карточки (если он производился из Windows-клиента).

- Операции со строками:
  - Добавление строки.
  - Обновление строки.
  - Удаление строки.
  - Перемещение строки.
  - Копирование строки.
  - Изменение прав доступа к строке.
- Безопасность:
  - Изменение прав доступа к строке.
  - Изменение прав доступа к карточке.
  - Изменение прав доступа к файлу.
  - Изменение прав доступа к элементу.
  - Изменение прав доступа к отчёту.
  - Изменение прав доступа к типам карточек.
- Операции с файлами:
  - Создание файла.
  - Открытие файла.
  - Запись в файл.
  - Закрытие файла.
  - Удаление файла.
  - Изменение прав доступа к файлу.

    - Копирование файла.
    - Замена файла.
    - Выгрузка файла из базы данных.
    - Загрузка файла в базу данных.
    - Архивирование файла.
    - Разархивирование файла.
- Операции с нумераторами:
    - Изменение левой границы нумератора.
    - Изменение правой границы нумератора.
    - Выделение номера.
    - Выделение диапазона номеров.
    - Освобождение номера.
    - Освобождение диапазона номеров.
- Операции с ярлыками:
    - Создание ярлыка.
    - Удаление ярлыка.
    - Перемещение ярлыка.
    - Копирование ярлыка.

Разработчику на платформе доступны следующие возможности при работе с журналом (все они требуют наличия административных привилегий для выполнения):

- Получение определённого набора записей из журнала по фильтру.
- Поиск конкретной записи в журнале.
- Удаление записей из журнала.
- Изменение стратегии автоматической очистки журнала.
- Экспорт и импорт записей журнала в формат XML.

Кроме этого, программно доступна возможность фиксировать в журнале собственные операции (для этого административные привилегии не требуются). Эти записи могут быть любого типа (отладочные и информационные сообщения, предупреждения). и их характер зависит от семантики конкретной ситуации.

Разработчики карточек могут расширять набор журналируемых операций, для этого в редакторе библиотеки `CardManager` на вкладке *Log* необходимо определить новые операции, указать их идентификатор, псевдоним и имя:

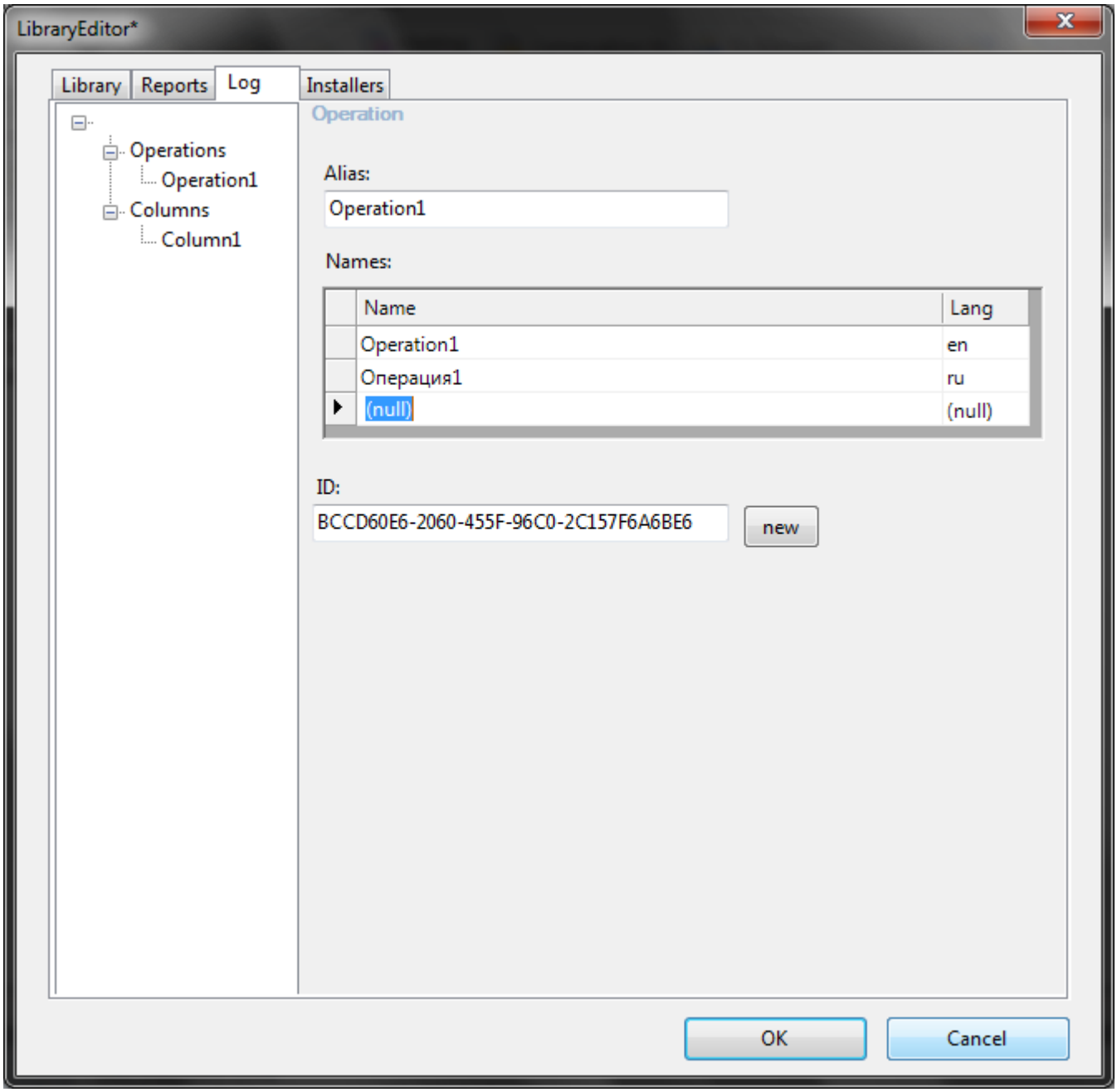


Рисунок 12. Определение операций журнала

Дополнительные атрибуты операций (`Columns`) определяются там же. Необходимо указать их псевдоним, тип данных и имя. Колонку можно ассоциировать с какой-либо операций или указать, что она обязательна для всех операций данной библиотеки.

При записи Custom-сообщения в журнал приложение должно сформировать xml-документ по схеме `EventData.xsd` и сохранить там все дополнительные атрибуты.

Для работы с журналом предназначен специальный объект `LogManager`, доступ к которому можно получить из объекта сессии (`UserSession`).

Первоочередной задачей является установка стратегии журнала, указав параметры хранения записей. Сделать это можно вызвав метод `LogManager.SetLogStrategy` с соответствующими параметрами.

Записывать собственные сообщения в журнал можно вызывая метод `LogMessage` или `LogMessageEx`.

Поиск сообщений осуществляется методом `FindMessages`, а более полную информацию по конкретному сообщению можно получить вызывая `GetMessage` или `GetMessageDetails`.

Разработка решений на платформе Docsvision

В данном разделе приведены практические методики использования API Docsvision при разработке готовых решений на базе Docsvision. Данные сценарии демонстрируют возможности Docsvision для создания законченных программных продуктов, удовлетворяющих потребности конкретной предметной области.

В этом разделе:

- Разработка карточек и библиотек карточек
- Разработка расширений Docsvision
- Разработка элемента управления для Конструктора разметок
- Распространение решения
- Отладка и тестирование

Разработка карточек и библиотек карточек

Модификация карточек и решений, входящих в стандартную поставку системы Docsvision ("Базовые объекты" и "Управление процессами"), не допускается ни при каких условиях. Их исходный код не подлежит распространению.

Система может быть расширена путем добавления новых пользовательских модулей — *карточек*. Система изначально содержит ряд карточек, необходимых для её работы. Такие карточки называются *системными объектами*.

В логическом смысле карточка представляет собой описание пользовательского бизнес-объекта, содержащее в себе все необходимые данные и возможности для выполнения операций над ними. Физически карточка — это совокупность программных объектов (компонент), объектов в базе данных системы (таблиц и хранимых процедур), а также мета-описание структур данных в XML формате.

Каждая карточка относится к одному из типов. Тип карточки — логическое понятие, введённое с целью отличать разнородные карточки друг от друга. Каждый тип имеет уникальное имя и идентификатор, например, идентификатор типа карточки "Документ" имеет значение `B9F7BFD7-7429-455E-A3F1-94FFB569C794` (идентификаторы стандартных карточек приведены в разделе "/dv6/schemas/dv6/[Описание полей стандартной карточки]"). При необходимости не только различать отдельные карточки, но и группировать сходные карточки, типы карточек объединяются в библиотеки по их схожести для пользователя. Принадлежность типа карточки к какой-либо библиотеке определяется автором карточки.

Следует различать тип карточки и экземпляр карточки. Регистрация новой карточки в системе добавляет один новый тип, для которого впоследствии может быть создано неограниченное количество экземпляров карточек. Все созданные экземпляры будут принадлежать одному типу, то есть использовать для работы одни и те же программные компоненты и структуры данных.

Все экземпляры карточек обладают уникальными идентификаторами. Идентификаторы используются для получения данных от сервера, для ссылок на другие карточки и на данные карточек. Идентификатор — это обязательный атрибут объектов системы, поддерживаемый самой платформой (то есть разработчик карточки не должен явно резервировать место для хранения идентификаторов).

Программные компоненты карточки реализуют ряд предопределённых интерфейсов, предназначенных для их интеграции в среду Docsvision. Карточка может также реализовывать собственные интерфейсы (например, для связи с другими карточками). Как правило, модуль карточки также реализует пользовательский интерфейс, позволяющий пользователю просматривать и редактировать содержимое карточки.

Каждый тип карточки также может декларировать ряд действий, применимых для данного типа. Под действиями в данном случае понимаются определённые команды, "понятные" карточке, которые при инициализации пользователем, приведут к выполнению программными компонентами карточки некоторых

операций.

Иерархия

Помимо карточек в системе определяется иерархия для их организации. Каждый узел иерархии — это контейнер-папка, содержащая ссылки на карточки. Папка также может содержать в себе подпапки. С точки зрения пользователя — это иерархия "папок, содержащих карточки и другие папки", что схоже с представлением размещения файлов в файловой системе. Администратор может задавать категории или типы карточек, которые могут храниться в конкретной папке. Вся информация о папках и их иерархии хранится в области данных одной из системных карточек — карточке папок.

Атрибуты

Папки ссылаются на карточки посредством их идентификаторов, а также ряда дополнительных атрибутов. Совокупность всех сведений об одной карточке в папке представляет собой *ярлык*. Для каждого экземпляра карточки может существовать неограниченное количество ярлыков, размещенных в разных папках.

Подробнее папки и ярлыки рассмотрены в разделе Работа с папками и ярлыками

Представление

Содержимое папки может быть представлено пользователю несколькими способами, которые определяет администратор системы или сам пользователь. Отображение содержимого папки одним из способов называется *представлением*. Представление строится из отдельных элементов представления каждой карточки. Элемент представления — это набор данных карточки, который может отображаться в папке одновременно.

При помощи представлений и связанного с ними интерфейса пользователь может просматривать список карточек, осуществлять поиск, выбирать отдельные карточки и исполнять над ними определённые действия, в том числе, "открывать" карточки — запускать их пользовательский интерфейс. Карточка может поддерживать несколько вариантов пользовательского интерфейса или способов взаимодействия с пользователем — режимов. Информация о возможных режимах хранится в ярлыке карточки. В разных режимах над карточками могут выполняться различные операции, определённые разработчиком карточки.

Пользователь системы может создавать новые карточки известных системе типов или удалять существующие карточки. Для добавления новых типов

карточек необходимо пройти несколько этапов и начать нужно с Разработки схемы данных.

Обратите внимание на то, что для работы с новой библиотекой карточек через Консоль настройки Docsvision, например, для обновления базы данных, необходимо реализовать модуль расширения Консоли настройки, передав в нем схему новой библиотеки.

В данном разделе

- Разработка схемы данных карточки
- Создание объектной модели карточки
- Разработка компонента карточки
- Разработка компонента библиотеки карточек

Разработка схемы данных карточки

Архитектура Docsvision предусматривает строго определённый способ взаимодействия ядра системы с пользовательскими карточками. Чтобы карточку можно было использовать в системе, структуры данных карточки и её программные компоненты должны соответствовать определённым стандартам.

Хранилище для данных карточки создаётся в базе данных Docsvision автоматически. При этом в качестве исходных данных используется XML-файл с описанием структуры данных — схема карточки, которая должна быть подготовлена заранее. Файл содержит сведения о количестве данных в карточке, их типах и связях между ними. Фактически этот файл используется и при разработке программных компонент карточки, так как многие функции системы используют в качестве параметров вызова идентификаторы (GUID), которые закреплены за определёнными секциями данных в файле описания структуры.

Схема данных карточки описывает разделы (секции), которые в свою очередь состоят из строк. Под секциями понимаются группы данных, объединенные по общим признакам (полям или атрибутам). Строки секций могут ссылаться на другие секции (подчинённые секции).

Для удобства организации данных выделяются три вида секций:

- *Плоская секция* — набор единичных значений атрибутов (полей).

- *Коллекционная секция* — таблица значений, которая может содержать несколько строк с однотипными полями.
- *Иерархическая секция* — таблица, строки которой ссылаются друг на друга, образуя древовидную (иерархическую) структуру. Поскольку каждая строка-узел может ссылаться на несколько дочерних строк, иерархическую секцию можно представить себе как иерархию секций одного типа.

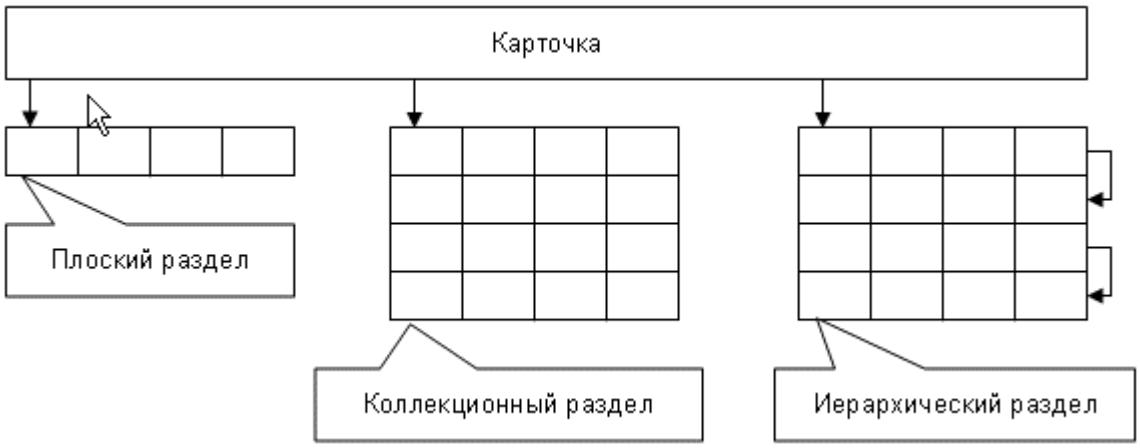


Рисунок 13. Виды секций

Каждая секция имеет собственный уникальный идентификатор.

Схема также может определять виртуальные поля, которые используются при построении представлений (отчётов). Каждое виртуальное поле описывает логическую единицу данных карточки, значимую для пользователя (например, "Исполнители" или "Статус связанного процесса").

Кроме этого, схема карточки описывает возможные действия карточки с указанием их уникальных идентификаторов, режимы работы карточки — возможные варианты отображения её пользовательского интерфейса, преобразования данных карточки — варианты трансформации данных в другой формат при помощи XSLT и InfoPath-преобразований.

Разработка схемы карточки состоит из четырёх основных этапов:

- Подготовка базы данных.
- Подготовка библиотеки карточек.
- Создание и редактирование схемы данных.
- Загрузка готовой схемы в базу данных.

Для выполнения всех этих операций используется программа "CardManager", включённой в Resource Kit.

Подключение базы данных Docsvision

В данном разделе приведена последовательность действий по подключению БД, в которую будут загружаться схемы карточек Docsvision, к программе CardManager.

Выберите пункт меню *Register database* из контекстного меню *Databases*, и укажите данные для подключения базы данных: название сервера СУБД, название БД Docsvision, имя пользователя и пароль для подключения.

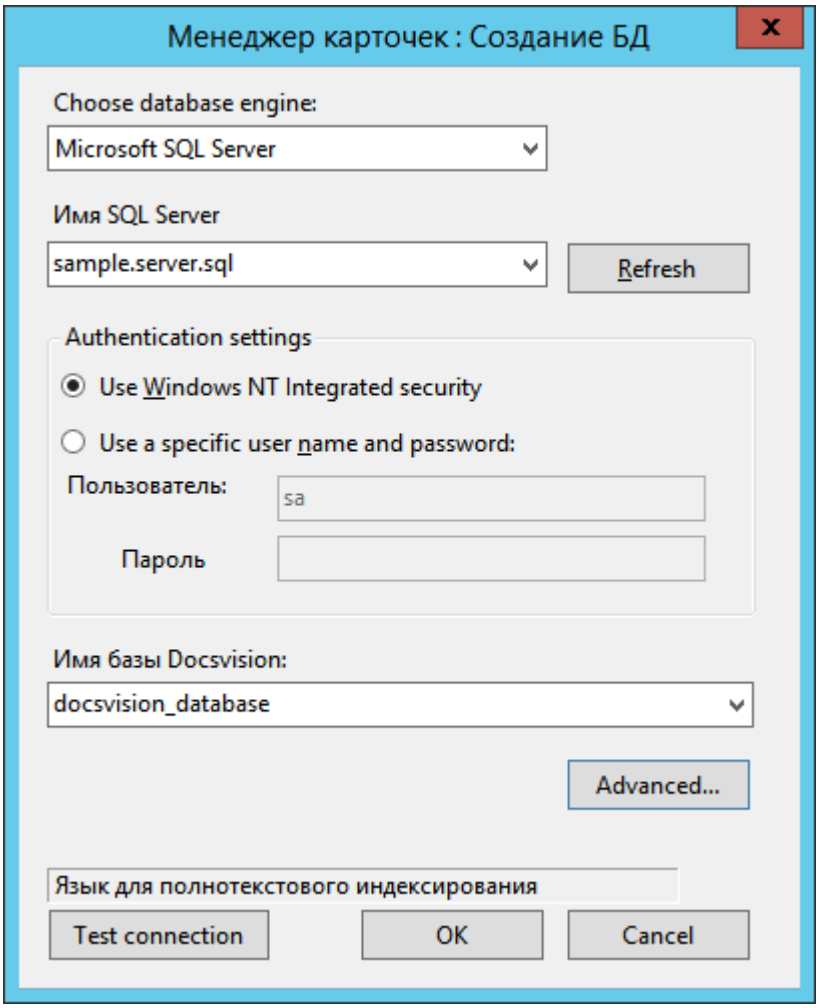


Рисунок 14. Подключение к базе данных

Отключить подключенную БД можно с помощью команды *Unregister database*.

Подготовка библиотеки карточек

Библиотека карточек объединяет несколько описаний карточек в одну логическую единицу — решение. Например, системные карточки объединяются в одну библиотеку, а "Базовые объекты" — в другую. В большинстве случаев одно разрабатываемое решение на платформе Docsvision соответствует одной

библиотеке.

При последующем распространении библиотеки карточек устанавливаются на клиентские компьютеры целиком.

Библиотека отображается в качестве родительской ветки для входящих в неё карточек в дереве карточек в *Windows-клиенте* (ветка *Карточки*).

Каждая библиотека карточек имеет описание (схему), собственный программный компонент, иконку и установочный пакет (подробнее о компонентах библиотек — в разделе Разработка компонента библиотеки карточек).

Схему карточки можно добавить в уже существующую библиотеку, либо в созданную новую.

Для создания библиотеки карточек вызовите команду *New library* из контекстного меню узла *Card libraries* дерева объектов.

При этом нужно указать имя файла, для сохранения описания библиотеки карточек, а также её свойства:

- *Alias* — псевдоним библиотеки, с помощью которого к ней можно обращаться в коде.
- *Названия* — локализованное имя библиотеки, отображаемое пользователю для выбранного языка.
- *ID* — уникальный идентификатор библиотеки, определённый на этапе разработки (кнопка **Новый** позволяет сгенерировать новый идентификатор).
- *Version* — номер версии библиотеки (его необходимо изменять вручную при необходимости обновить библиотеку на всех клиентах).
- *ProgID* — программный идентификатор компонента библиотеки.
- *Activation string* — идентификатор класса или программный идентификатор CLSID компонента библиотеки.
- *MSI Product Code* — код установочного компонента библиотеки для автоматической установки.
- *MSI Package Name* — путь к установочному компоненту библиотеки.
- *Icon file* — иконка библиотеки (отображается в Windows-клиенте).

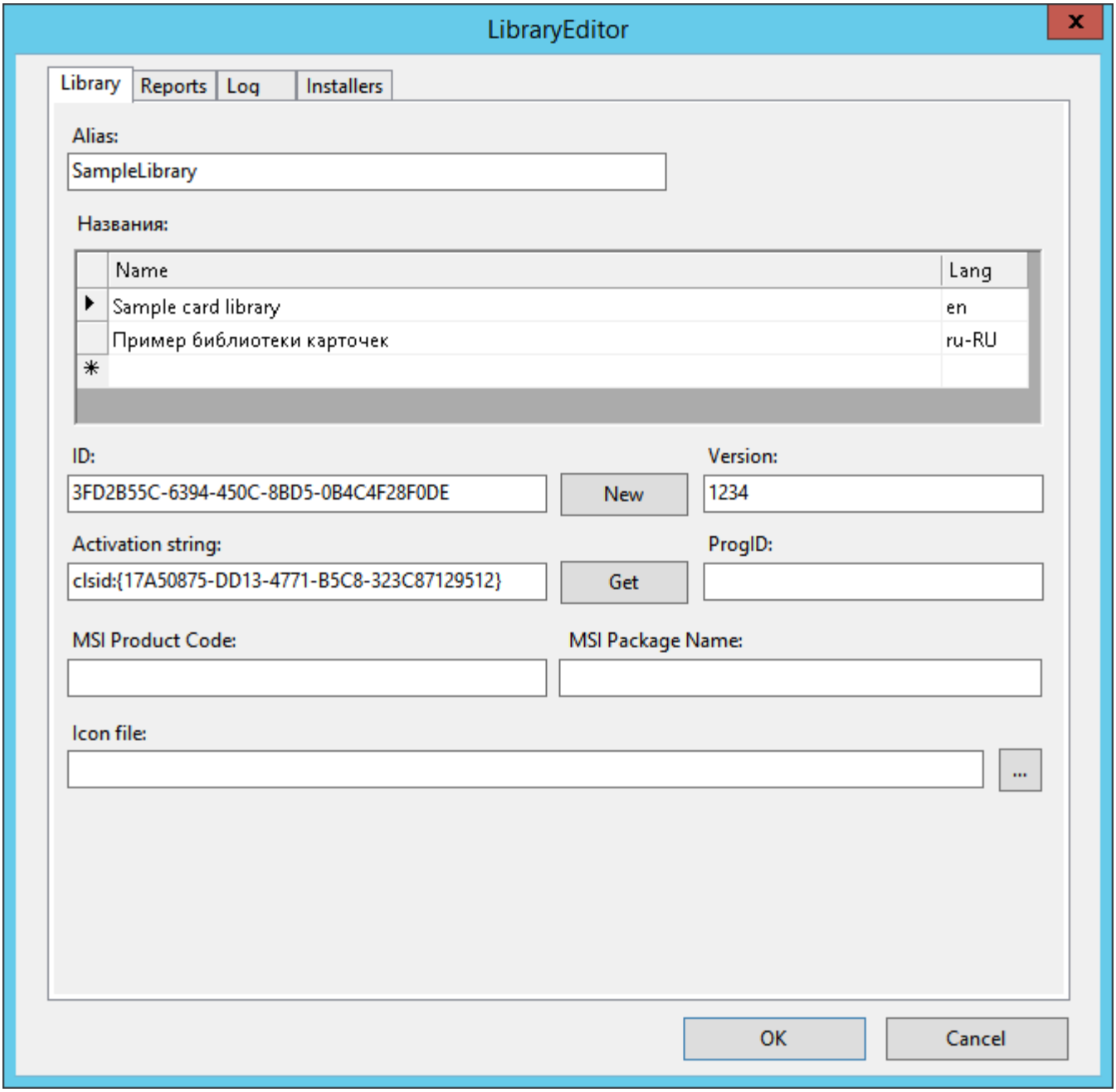


Рисунок 15. Описание библиотеки

Работа с вкладкой *Reports* описана в разделе "Использование отчётов", с вкладкой *Installers* — в разделе "Распространение решения", а с вкладкой *Log* — в разделе "Журналирование".

После создания и описания всех свойств библиотеки её название появится в левой панели окна *CardManager*. После чего можно приступать к описанию схем метаданных карточек, входящих в библиотеку.

Создание и редактирование схемы данных

Чтобы добавить новую схему данных (карточку) в библиотеку, нужно выбрать

необходимую библиотеку в левой панели и выполнить команду *New card*. Если схема данных была подготовлена ранее, её можно включить в библиотеку при помощи команды *Open card*. После этого надо указать имя XML-файла, содержащего описание схемы данных карточки.

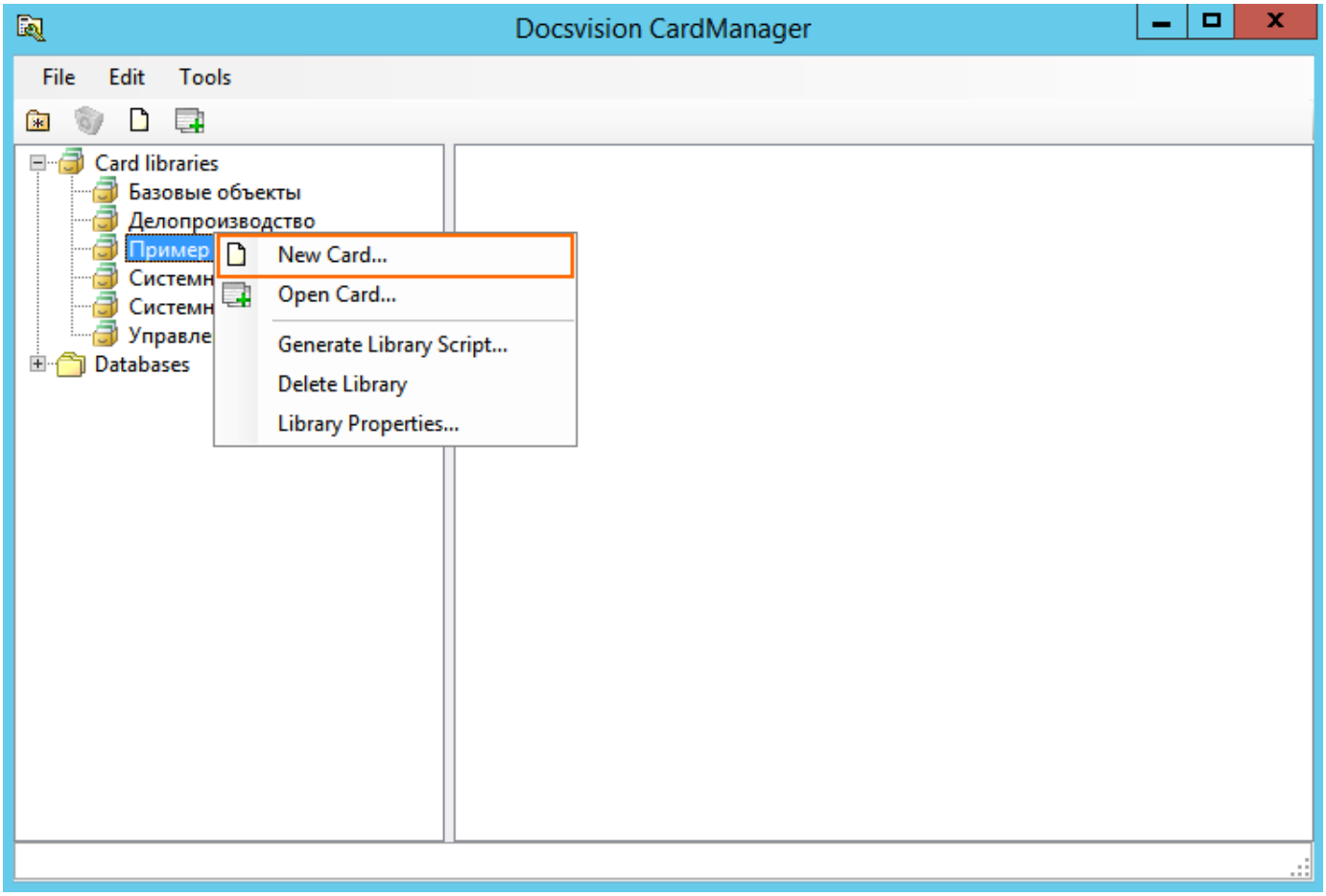


Рисунок 16. Добавление карточки в библиотеку

После создания карточки можно приступать к редактированию схемы данных (для редактирования существующей карточки нужно выбрать пункт меню **Edit › Card Propeties**).

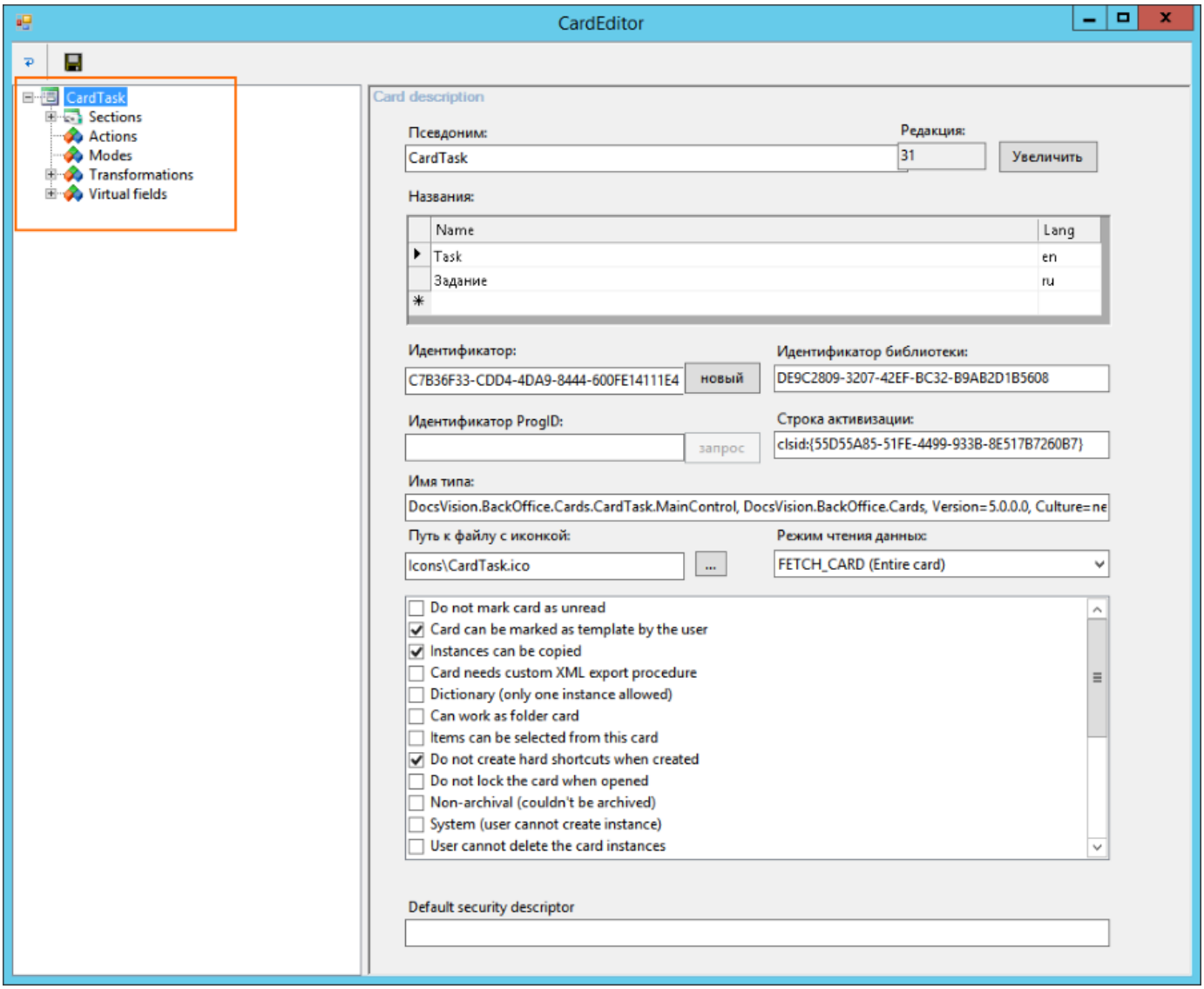


Рисунок 17. Редактирование метаданных карточки

Редактирование схемы данных представляет собой описание атрибутов карточки в нескольких различных разделах:

- *Card description* — общая информация. В данном разделе описываются общие свойства карточки: её имя, описание, идентификатор, иконка, системные атрибуты.
- *Sections* — описание секций, входящих в карточку, их полей, свойств и связей между ними.
- *Virtual fields* — логические элементы данных карточки, доступные для вывода в представления.
- *Actions* — описывает возможные действия карточки.
- *Modes* — возможные режимы работы карточки (например, варианты отображения её интерфейса).

- *Transformations* — варианты трансформации данных в другой формат при помощи XSLT и InfoPath-преобразований.

Заполнение метаданных карточки рассмотрено на примере схемы карточки *Документ*.

Описание карточки, Общая информация карточки

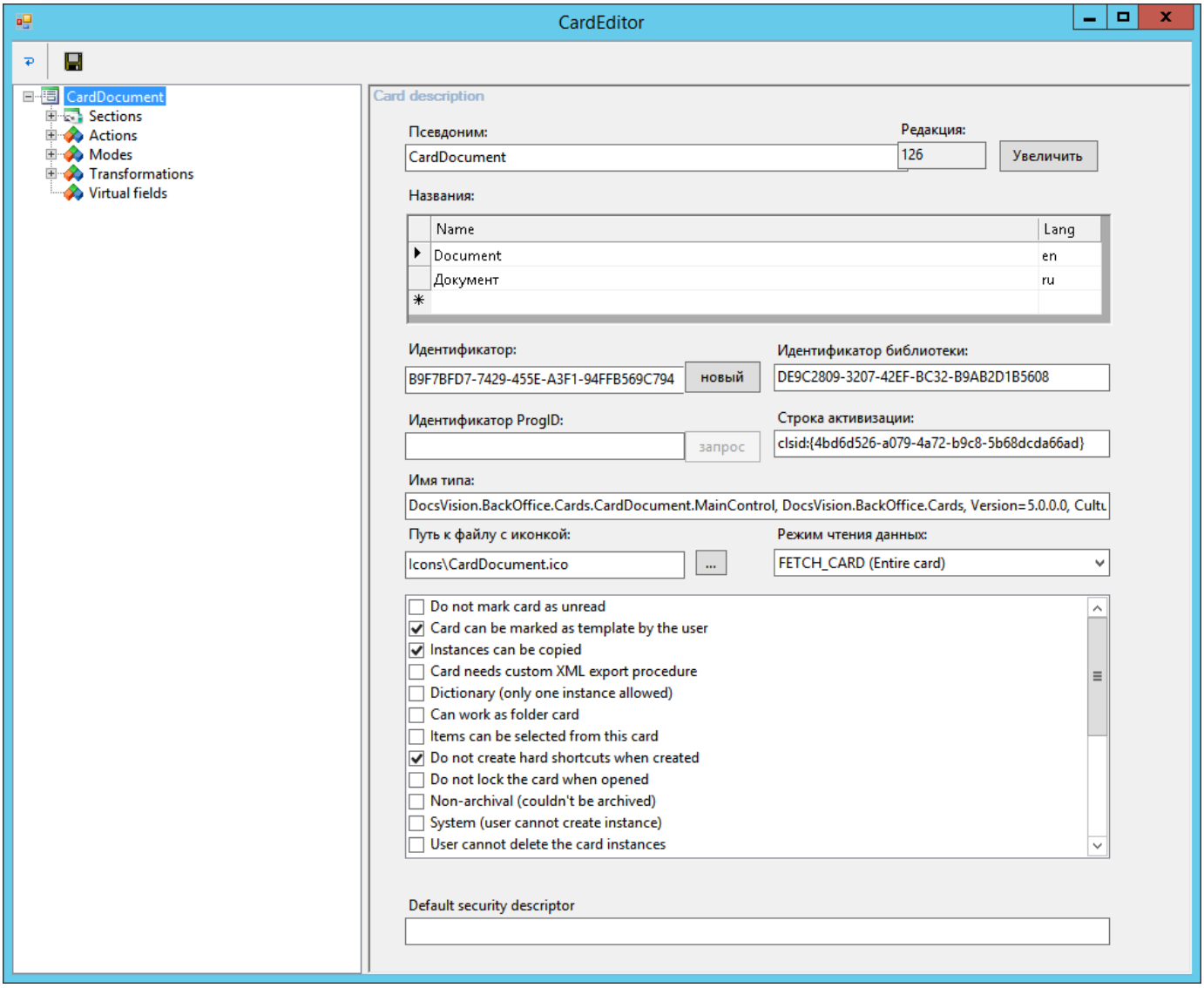


Рисунок 18. Общая информация карточки

Общее описание карточки предполагает заполнение следующих полей:

- *Псевдоним* — псевдоним карточки, с помощью которого к ней можно обращаться в коде. Псевдоним должен быть уникальным в рамках библиотеки.
- *Редакция* — номер версии карточки.

- *Названия* — локализованные названия карточки, отображаемые пользователю.
- *Идентификатор* — уникальный идентификатор данного типа карточек. Этот идентификатор генерируется автоматически при создании новой карточки. Идентификатор должен быть уникален в рамках всей базы данных карточек.
- *Идентификатор библиотеки* — идентификатор библиотеки, которой принадлежит карточка. Заполняется автоматически.
- *Идентификатор ProgID* — программный идентификатор COM-компонента, реализующего функциональность карточки. Для .Net карточек не требуется.
- *Строка активизации* — строка активизации компонента карточки. Может содержать программный идентификатор (`ProgID`), идентификатор класса (`CLSID`) и данные о лицензировании. Для .Net карточек не требуется.
- *Имя типа* — полное имя класса компонента карточки. Для .Net карточек следует указывать данный параметр вместо `ProgID` и строки активации.
- *Путь к файлу с иконкой* — иконка данного типа карточек, отображаемая для пользователя. Рекомендуется использовать относительные пути при указании иконки.
- *Режим чтения данных* — позволяет выбрать механизм чтения данных карточки. Возможны следующие варианты режима чтения:
  - *FETCH_CARD* (карточка целиком) — при обращении к карточке все её данные будут переданы клиенту.
  - *FETCH_SECTION* (секция) — при обращении к данным карточки будет прочитана целиком соответствующая секция.
  - *FETCH_SUBSECTION* (подсекция) — чтение только данных подсекций.
  - *FETCH_LEVEL* (уровень) — чтение только данных одного уровня дерева (для иерархических секций).
  - *FETCH_ROW* — чтение только данных одной строки.
- Дополнительные атрибуты, определяющие особенности поведения карточки в системе:
  - *Do not mark card as unread* — по умолчанию при создании экземпляра карточки ярлык на неё помечается как "непрочитанный" (выделяется жирным шрифтом в окне Windows-клиента). Данный атрибут позволяет отключить эту возможность.

- *Card can be marked as template by user* — данный атрибут позволяет использовать механизм шаблонов для данного типа карточек.
- *Instances can be copied* — разрешает или запрещает копирование данных карточки.
- *Card needs custom XML export procedure* — карточки этого типа используют нестандартную процедуру выгрузки в XML (стандартные пункты меню будут недоступны).
- *Dictionary (only one instance allowed)* — разрешается существование в системе только одного экземпляра карточки. Используется для справочников.
- *Can work as folder card* — для карточек, реализующих функциональность папок (такую функциональность реализует системная карточка папок, которую разработчик может подменить собственной).
- *Items can be selected from this card* — карточку можно использовать для выбора её элементов (запускать в режиме выбора).
- *Do not create hard shortcuts* — атрибут регулирует механизм создания ярлыков для экземпляров карточек. По умолчанию при создании экземпляра карточки для неё создаётся "сильный" ярлык, при удалении которого удаляется и сама карточка. Данный атрибут позволяет отключить эту возможность.
- *Do not lock the card when opened* — разрешает или запрещает автоматическую блокировку данных карточки при её открытии из Windows-клиента. Если блокировка установлена, то одновременная работа пользователей с одним экземпляром карточки запрещается на системном уровне. Если блокировка отключена, то разработчик карточки должен сам контролировать ситуации одновременного доступа к данным.
- *Non-archival* — разрешает или запрещает перемещение данных карточки в архивные таблицы или архивную базу данных (при использовании дополнительного модуля архивации).
- *System (user cannot create instance)* — пользователь не может создать экземпляр карточки. Используется для карточек, не имеющих пользовательского интерфейса, и утилит. Экземпляры таких карточек создаются только программно.
- *User cannot delete card instances* — карточки данного типа не могут удаляться пользователями (соответствующие пункты меню и панели

инструментов будут недоступны).

- *Card cannot be replicated* — данная опция регулирует возможность репликации всех экземпляров данной карточки.
- *Non-searchable* — карточка не участвует в поиске и не индексируется.
- *Hidden* — экземпляры данного типа карточек не отображаются пользователю. Обычно используется совместно с атрибутом "Системная".
- *Replicate only card templates* — в процессе репликации будут участвовать только те карточки данного типа, которые помечены в качестве шаблонов.
- *Allow to open as linked card* — атрибут для связанных карточек. Разрешает или запрещает открытие пользовательского интерфейса карточки из других карточек.
- *Card is securable* — регулирует механизм выдачи прав на данные карточки. При наличии данного признака, права могут быть назначены на все структурные элементы карточки (секции, строки секций). При отсутствии данного признака используется так называемый упрощённый режим — права выдаются на карточку целиком.

Для повышения производительности крайне НЕ рекомендуется устанавливать этот признак для карточек без особой необходимости!

- *Provides UI Extension* — атрибут для карточек, которые интегрируются с интерфейсом Windows-клиента для реализации прикладных функций.
- *High changes rate* — атрибут следует устанавливать у тех карточек, изменение данных в которых происходит с экстремально большой частотой (например, он установлен у системной карточки папок). При установленном флаге `timestamp` данной карточки не обновляется при изменении, а вычисляется по набору timestamp'ов секций данной карточки.
- *Use server extension when checking access to the card* — атрибут по умолчанию включён для карточек документов и заданий. Если флаг снят, ролевая модель при проверке прав на карточки не используется.
- *Client-side data caching on disk is restricted* — атрибут позволяет запретить кэширование клиентских данных на диск. По умолчанию эта функция отключена.

- *Default security descriptor* — дескриптор безопасности, который определяет набор, назначаемых прав по умолчанию при создании экземпляров карточки.

В этом дескрипторе можно использовать только константные идентификаторы объектов (well-known security identifiers), т.к. создаваться экземпляры карточки будут в другом домене чем тот, где она разрабатывается.

Раздел "Sections", Описание секции

Заполнение данных о секциях предполагает создание структуры (иерархии) секций данных карточки, указание их свойств и полей. Для добавления новой секции нужно воспользоваться командой *New section* в секции *Sections*. После этого необходимо указать свойства новой секции и список входящих в неё полей.

Для корректной работы карточки требуется наличие у неё, как минимум, одной секции.

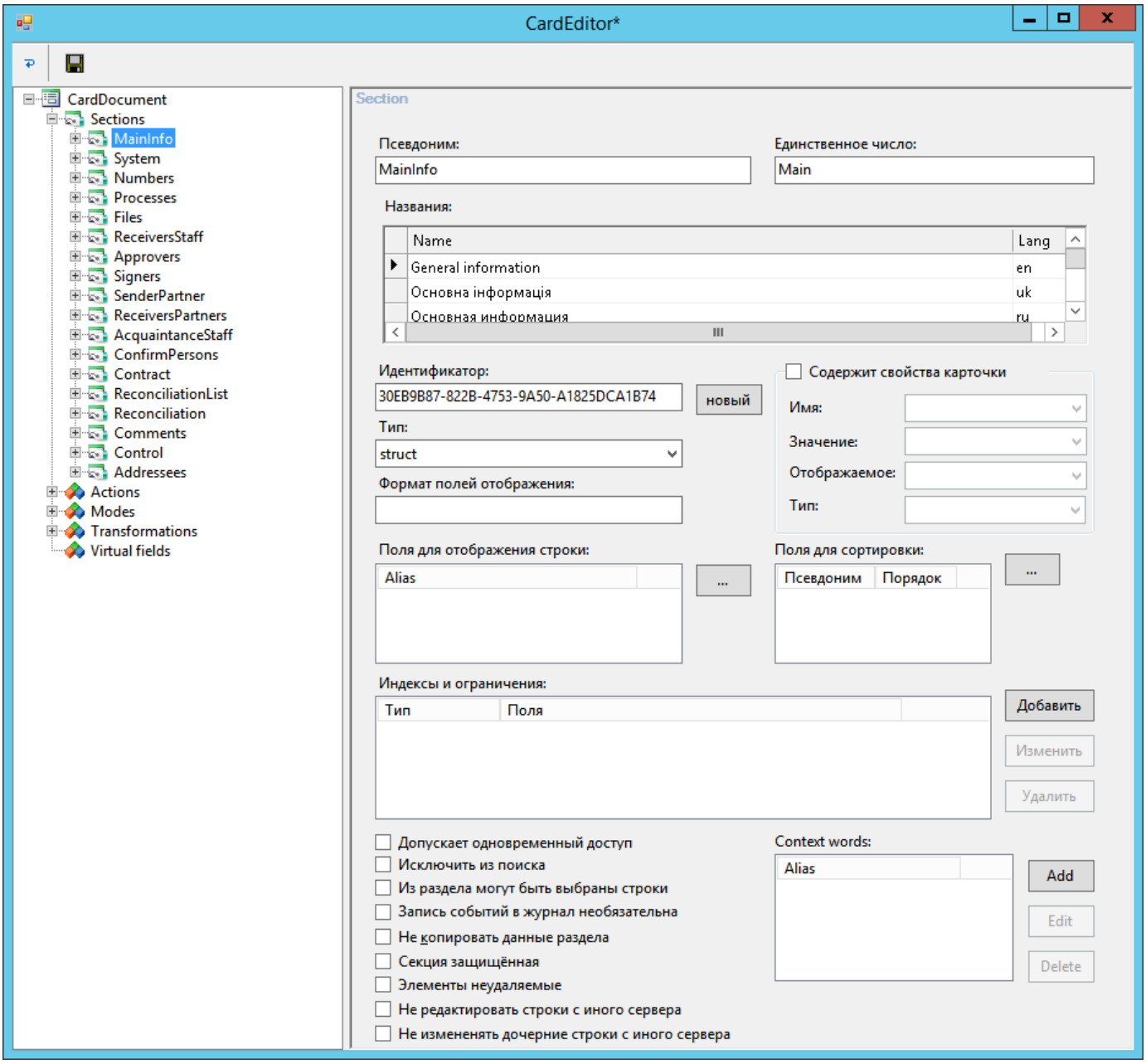


Рисунок 19. Описание секции карточки

Описание секции содержит следующие свойства:

- *Псевдоним* — псевдоним секции, с помощью которого к ней можно обращаться в коде.

Псевдоним секции должен отличаться от псевдонима карточки. Псевдоним должен быть уникальным в рамках карточки.

- *Единственное число* — строка, указывающая наименование соответствующего класса при использовании утилит авто-генерации кода по описанию карточки.

- *Названия* — локализованные названия секции, отображаемые для пользователя.
- *Идентификатор* — уникальный идентификатор данной секции. Этот идентификатор генерируется автоматически при добавлении новой секции и может быть сгенерирован заново при нажатии кнопки **New**.
- *Тип* — тип секции:
  - *struct* — плоская.
  - *table* — коллекционная.
  - *tree* — иерархическая.
- *Формат полей отображения* — формат описания (дайджеста) строки секции. Дайджесты строк — это текстовые комментарии, уникальным образом характеризующие конкретную строку внутри секции. В основном, они используются в элементах управления, предназначенных для выбора строк секции.

В текущей версии формат описания строки не применяется и зарезервирован для будущего использования.

- *Поля для отображения строк* — поля, из которых составляются описания (дайджесты) строк секции. Значения этих полей приводятся к строчному выражению, объединяются в порядке их описания и отделяются друг от друга пробелом.

Например, секция содержит данные о сотрудниках организации и имеет несколько полей, среди которых есть *Name* (Имя), *SurName* (фамилия) и *MiddleName* (Отчество). В этом случае, если выбрать эти три поля в качестве полей для отображения строки, то во всех элементах управления выбранная строка данной секции будет отображаться как *Фамилия Имя Отчество*.

- *Содержит свойства карточки* — атрибут для секций свойств. Это специальный вид секций, содержащих ряд предопределённых полей, которые могут выступать в качестве пользовательских атрибутов карточки (свойств). Платформа позволяет обрабатывать свойства карточки специальным образом — например, выводить их в представления или синхронизировать их значения со свойствами файлов Office. При отметке какой-либо секции как *содержащей свойства*, требуется дополнительно указать ключевые поля этой

секции:

- *Имя* — поле, содержащее название свойства (строка).
- *Значение* — поле, содержащее значение свойства (чаще всего — variant).
- *Отображаемое* — поле, содержащее отображаемое значение свойства (строка).
- *Тип* — поле, определяющее тип свойства (перечисляемое).

- *Поля для сортировки* — описание полей секции, по которым производится сортировка строк при обращении к данным секции. Можно указать прямой или обратный порядок сортировки для конкретных полей. Число полей для сортировки не ограничено. Сортировка производится в порядке следования этих полей в описании.
- *Индексы и ограничения* — ограничения на поля секции. Каждое ограничение устанавливает одно требование к одному полю или набору полей секции. Можно указать следующие ограничения:
  - *Unique globally* — значение поля (или набора полей) уникально в рамках сервера Docsvision.
  - *Unique within card* — значение поля (или набора полей) уникально в рамках экземпляра карточки.
  - *Unique within section* — значение поля (или набора полей) уникально в рамках секции.
  - *Unique within tree* — значение поля (или набора полей) уникально в рамках всех уровней иерархической секции.
  - *Clustered index* — по полю строится кластерный индекс.
  - *Non- clustered index* — по полю строится некластерный индекс.
  - *Unique clustered index* — по полю строится уникальный кластерный индекс.
  - *Unique non — clustered index* — по полю строится уникальный некластерный индекс.

Необходимо внимательно отнестись к созданию индексов, так как их не оптимальное создание (или попросту отсутствие) способно значительно понизить производительность всей системы.

- Дополнительные настройки секции:
  - *Допускает одновременный доступ* — допускается одновременная работа нескольких пользователей с данными секции (чаще всего этот признак устанавливается для справочников и требует отказа от блокировки карточки).
  - *Исключить из поиска* — поля секции не будут отображаться в диалоге поиска.
  - *Из раздела могут быть выбраны строки* — секцию можно использовать для выбора строк (чаще всего этот признак устанавливается для секций справочников).
  - *Запись событий в журнал необязательна* — сообщение об изменении строки секции не будет записываться в системный журнал.

Для повышения производительности крайне рекомендуется устанавливать этот режим для секций, с которыми производится интенсивная работа (например, для секций с данными журналирования).

  - *Не копировать данные раздела* — при копировании карточки строки данной секции не будут копироваться.
  - *Секция защищенная* — признак наличия собственных дескрипторов у строк секции (используется в случае полной безопасности на карточке в целом).
  - *Элементы неудаляемые* — признак, показывающий, что строки из данной секции не могут быть удалены.
  - *Не редактировать строки с иного сервера* — устанавливает запрет на изменение строк секции, если строки созданы на другом сервере. Например, подразделение, созданное на одном сервере, и реплицированное на другие сервера, будет доступно "только для чтения" на других серверах. Зарезервирован для модуля "Межфилиального обмена".
  - *Не изменять дочерние строки с иного сервера* — устанавливает запрет на изменение строк секции, если секция создана на другом сервере. Например, сотрудники подразделения, созданного на одном другом сервере, будут доступны только для чтения на других серверах. Зарезервирован для модуля "Межфилиального обмена".

Раздел "Sections", Добавление полей и дочерних секций

Создание секции также включает в себя добавление полей и дочерних секций. Для этого следует выбрать нужную секцию и воспользоваться командой *New field* или *New section*. Дочерние секции содержат те же свойства, что и секции первого уровня.

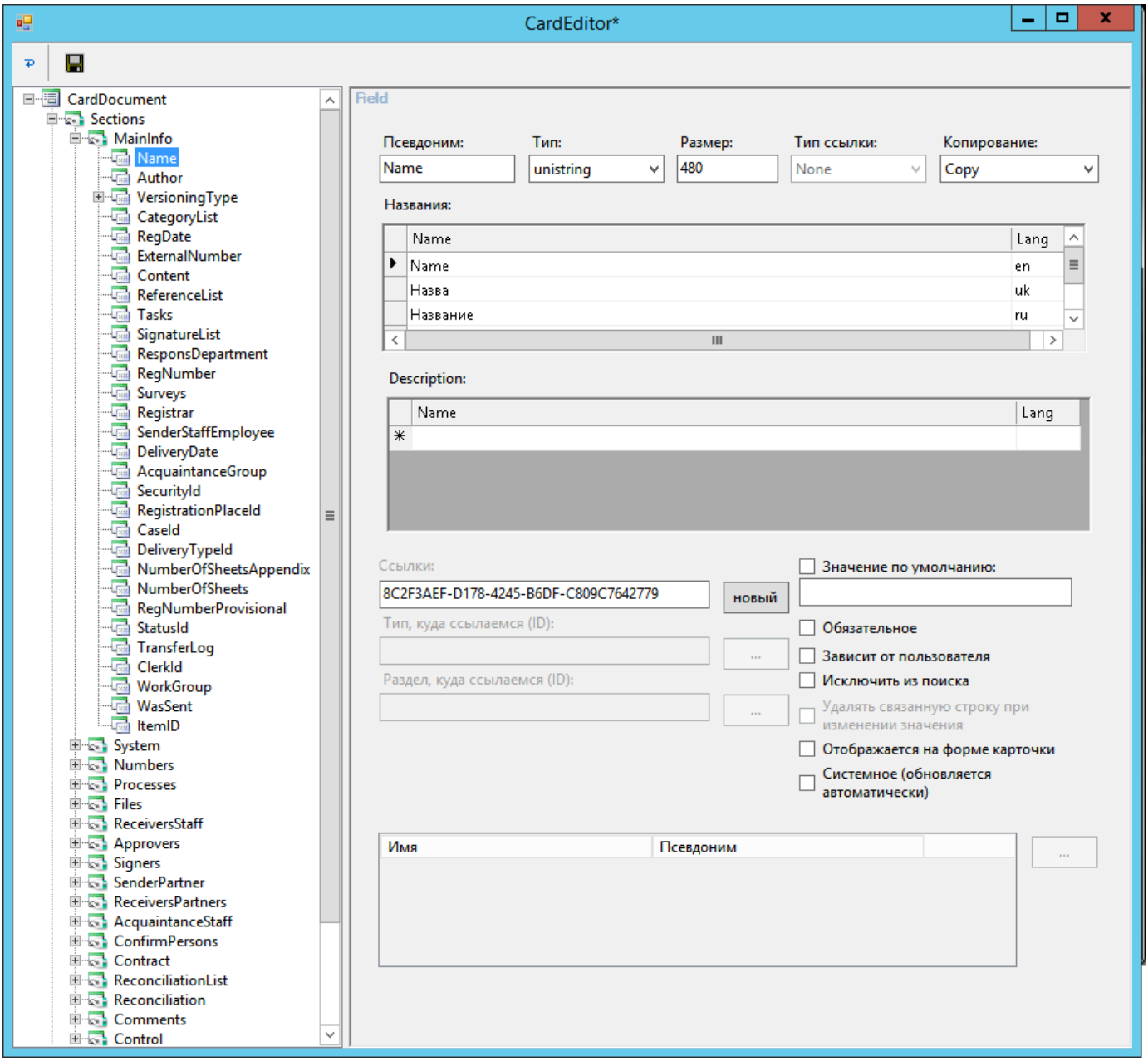


Рисунок 20. Редактирование поля секции]

Описание поля содержит следующие свойства:

- *Псевдоним* — псевдоним поля, с помощью которого к нему можно обращаться в коде. Псевдоним должен быть уникальным в рамках секции.
- *Тип* — тип (формат) данных в поле. Возможные типы полей:

- *int* — целое.
- *bool* — булево значение (Да/Нет).
- *datetime* — дата и время.
- *date* — дата.
- *time* — время.
- *enum* — перечисление (предопределённый набор уникальных значений).
- *bitmask* — битовая маска.
- *uniqueid* — уникальный идентификатор (`GUID`).
- *userid* — идентификатор пользователя.
- *string* — строка. Ограничение по размеру: не более `8000` символов.
- *unistring* — строка в формате `Unicode`. Ограничение по размеру: не более `4000` символов.
- *fileid* — идентификатор файла Docsvision.
- *float* — число с плавающей точкой.
- *refid* — ссылка на строку другой секции.
- *refcardid* — ссылка на карточку.
- *text* — текст. Ограничение по размеру: 2^{31-1} байт (2 Гб).
- *unitext* — текст в формате Unicode. Ограничение по размеру: 2^{31-1} байт (2 Гб).
- *binary* — двоичные данные.
- *image* — изображение.
- *sdid* — идентификатор безопасности.
- *decimal* — десятичное число.
- *variant* — значение переменного типа.

- *Размер* — ограничение длины поля (используется в основном для полей строчного типа).
- *Тип ссылки* — указание типа ссылки для ссылочных полей (`refid`, `refcardid`).

  Тип ссылки характеризует поведение объектов, связанных ссылкой.

  Возможны следующие типы ссылок:

- *Weak* — связанный объект разрешено удалять. При удалении объекта, содержащего ссылку, связанный объект останется.
- *Hard* — связанный объект запрещено удалять, пока на него существует ссылка. При удалении объекта, содержащего ссылку, связанный объект также будет автоматически удалён.
- *None* — при удалении связанного объекта значение ссылочного поля будет очищено.
- *Auto* — при удалении связанного объекта будет удалена вся строка секции, содержащей ссылку.

- *Копирование* — режим копирования данных поля. Возможны следующие режимы:
  - *Null* — значение поля при копировании обновляется.
  - *Copy* — значение поля копируется.
  - *CreateNew* — создание нового значения (используются для ссылок).
- *Названия* — локализованные названия поля, отображаемые у пользователя.
- *Description* — словесное описание поля, будет отображаться в виде подсказки к полю в диалогах поиска и представлений.
- *Идентификатор* — уникальный идентификатор поля.
- *Тип, куда ссылаемся (ID)* — уникальный идентификатор типа карточки, с полями или секциями которой связано данное поле. Тип карточки можно выбрать в диалоге типов.
- *Раздел, куда ссылаемся (ID)* — идентификатор секции связанной карточки, с полями которого осуществляется связь.
- *Значение по умолчанию* — значение поля, которое будет автоматически задаваться при создании новых экземпляров карточки.
- *Обязательное* — значение данного поля не может быть пустым. Система не позволит сохранить карточку без заданного значения этого поля.
- *Зависит от пользователя* — значение поля сопоставляется с конкретным пользователем системы. Таким образом, для каждого пользователя может храниться собственное уникальное значение этого поля.
- *Исключить из поиска* — поле не участвует в поиске и не отображается в диалоге поиска.

- *Удалять связанную строку при изменении значения* — признак удаления связанных строк в случае изменения значения этого поля (используется для ссылочных полей).
- *Отображается на форме карточки* — поле может быть сопоставлено с элементом пользовательского интерфейса. Этот признак используется в справочнике типов, если данная карточка обеспечивает возможность "тонкой" настройки.
- *Системное* — признак того, что поле является системным. Учитывается при некоторых операциях с данными.
- Также здесь содержится список полей связанной карточки, значения которых ассоциируются со строкой этой секции. Используются для ссылочных полей. Данная возможность позволяет при обращении к данным секции вернуть не только значения их собственных полей, но и ряда полей связанных карточек или секций. Это может быть удобным, например, чтобы снизить число обращений к серверу при чтении данных карточки или при её экспорте и печати.

Раздел "Actions"

В разделе *Actions* можно указать действия (команды), "понятные" данной карточке. В качестве команд могут выступать, например, такие действия как *Открыть в другом режиме*, *Переслать*, *Списать в архив* и т. д.

Разработчик сам определяет перечень этих команд, их названия и семантику. Фактически, все описанные команды появляются в меню действий над карточкой в окне Windows-клиента. При выборе пользователем одной из команд будет вызван клиентский компонент карточки с указанием идентификатора активированной команды и требуемые операции нужно будет запрограммировать самостоятельно в коде этого компонента.

Для добавления действий следует перейти в раздел *Actions* и выбрать пункт меню *New action*.

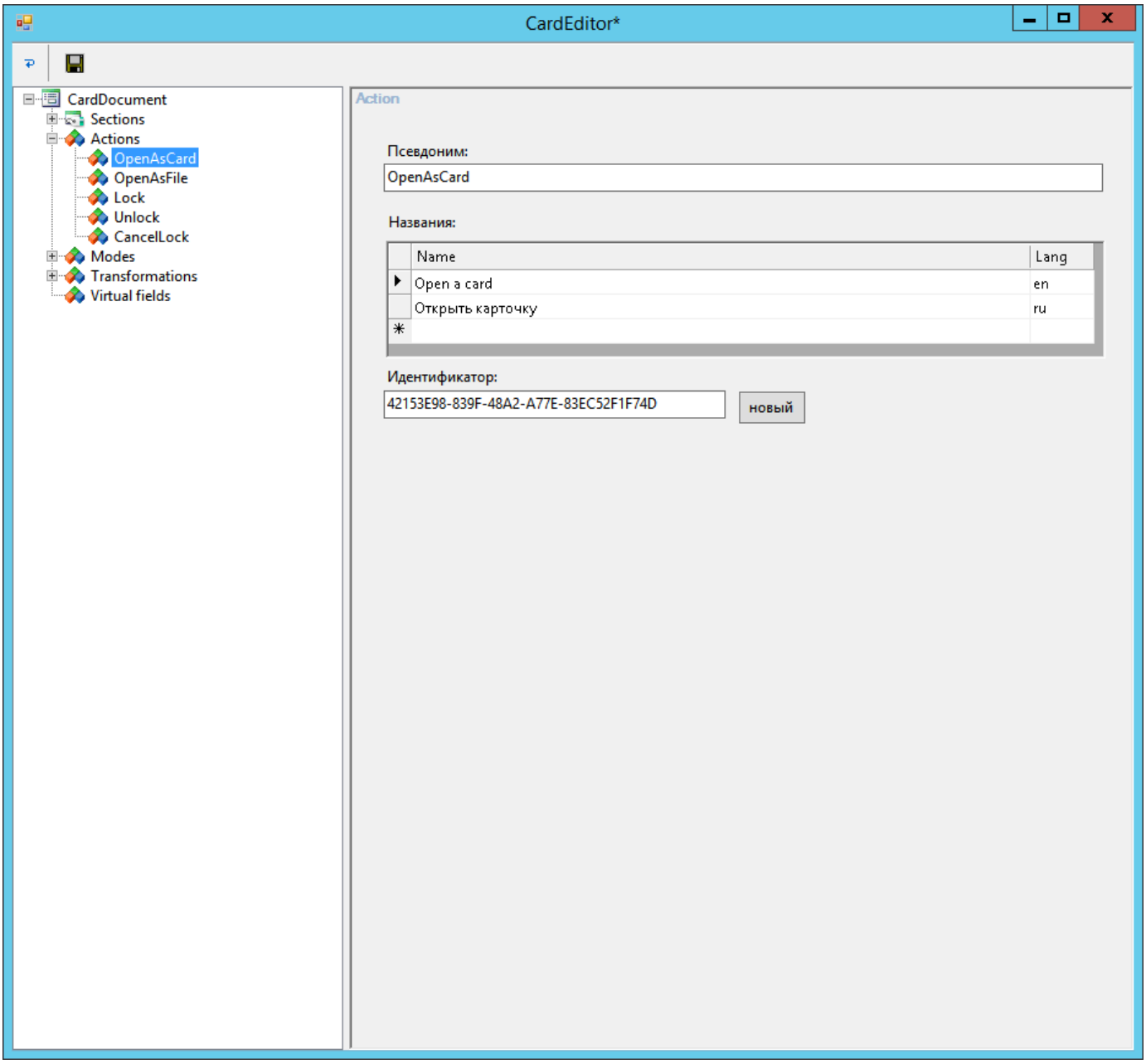


Рисунок 21. Описание действий над карточкой

Свойства действия (метода) включают в себя:

- *Псевдоним* — псевдоним действия, с помощью которого к нему можно обращаться в коде. Псевдоним должен быть уникальным в рамках карточки.
- *Названия* — локализованные названия команды, отображаемые у пользователя в меню действий над карточкой.
- *Идентификатор* — уникальный идентификатор метода.

Раздел "Modes"

В разделе *Modes* можно определить возможные режимы работы карточки. Режим работы карточки проще всего воспринимать, как вариант реализации

пользовательского интерфейса карточки. Можно, например, представить себе финансовый документ — бюджет, который может использоваться в различных вариантах:

- Подготовка.
- Принятие.
- Модификация поправками.

У карточки такого документа будет три соответствующих режима работы. Для каждого режима разработчик карточки сам определяет вид пользовательского интерфейса и семантику работы. Выбрать конкретный режим открытия экземпляра карточки можно либо при помощи контекстного меню Windows-клиента, либо режим может быть жёстко определён в ярлыке карточки.

Для добавления новых режимов работы следует перейти в раздел *Modes* и воспользоваться командой *New mode*. Далее нужно определить свойства режима.

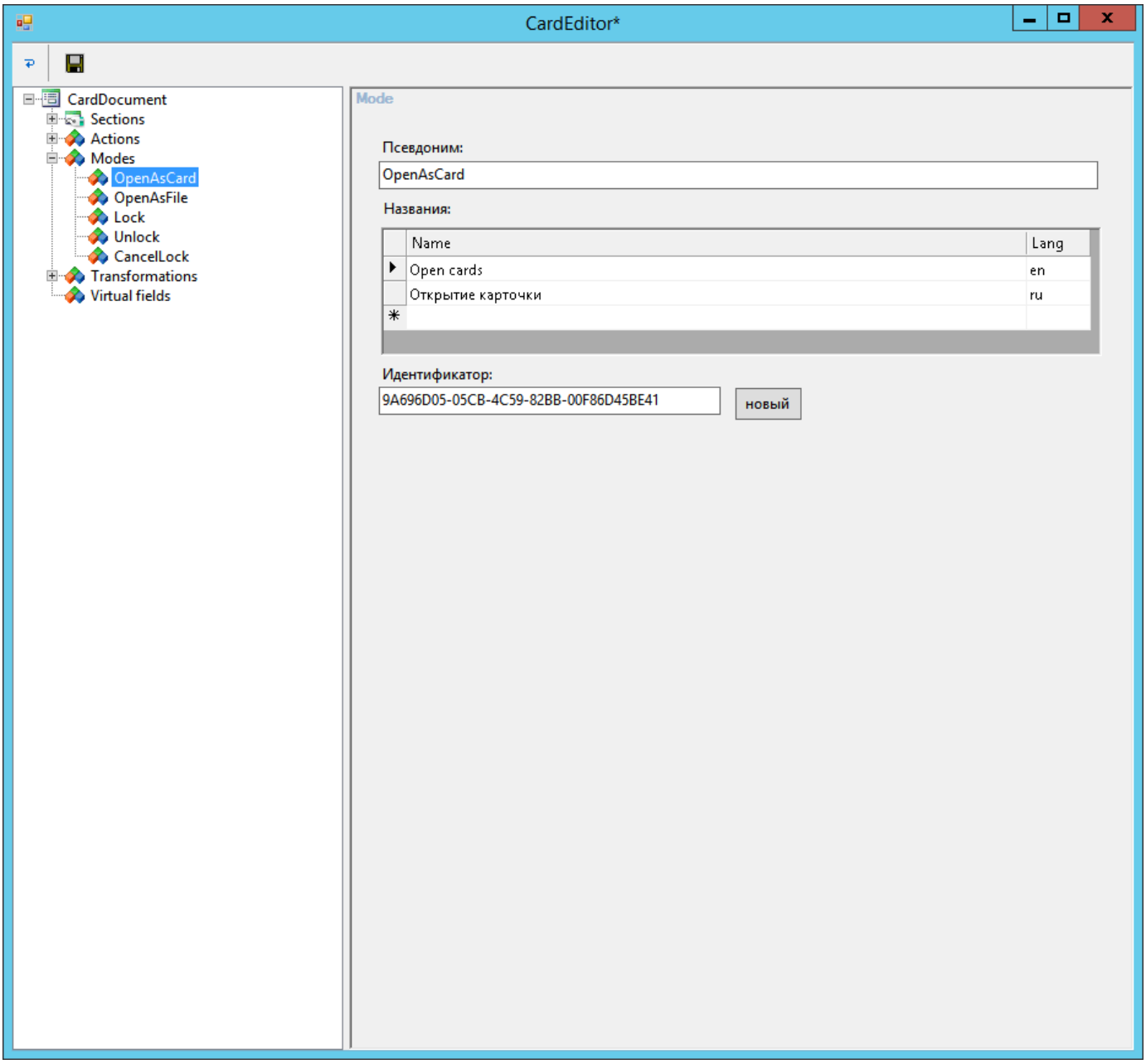


Рисунок 22. Свойства режима работы карточки

Свойства режима следующие:

- *Псевдоним* — псевдоним режима, с помощью которого к нему можно обращаться в коде. Псевдоним должен быть уникальным в рамках карточки.
- *Названия* — локализованные названия режима, отображаемые у пользователя.
- *Идентификатор* — уникальный идентификатор режима.

Раздел "Transformations"

В секции "Transformations" можно определить возможные преобразования данных карточки. Преобразования применяются для трансформации данных

карточки в другой формат, описанный в преобразовании. Например, преобразования могут применяться при печати карточек, экспорте данных в другие системы или при публикации карточек на web-страницах.

Системой поддерживаются два типа преобразований: XSLT-преобразования (они используются для получения HTML-представления данных карточки при выводе на печать) и преобразования Microsoft InfoPath.

Для определения преобразований следует перейти в раздел *Transformations* и выбрать пункт меню *New transformation*. Затем нужно определить свойства преобразования.

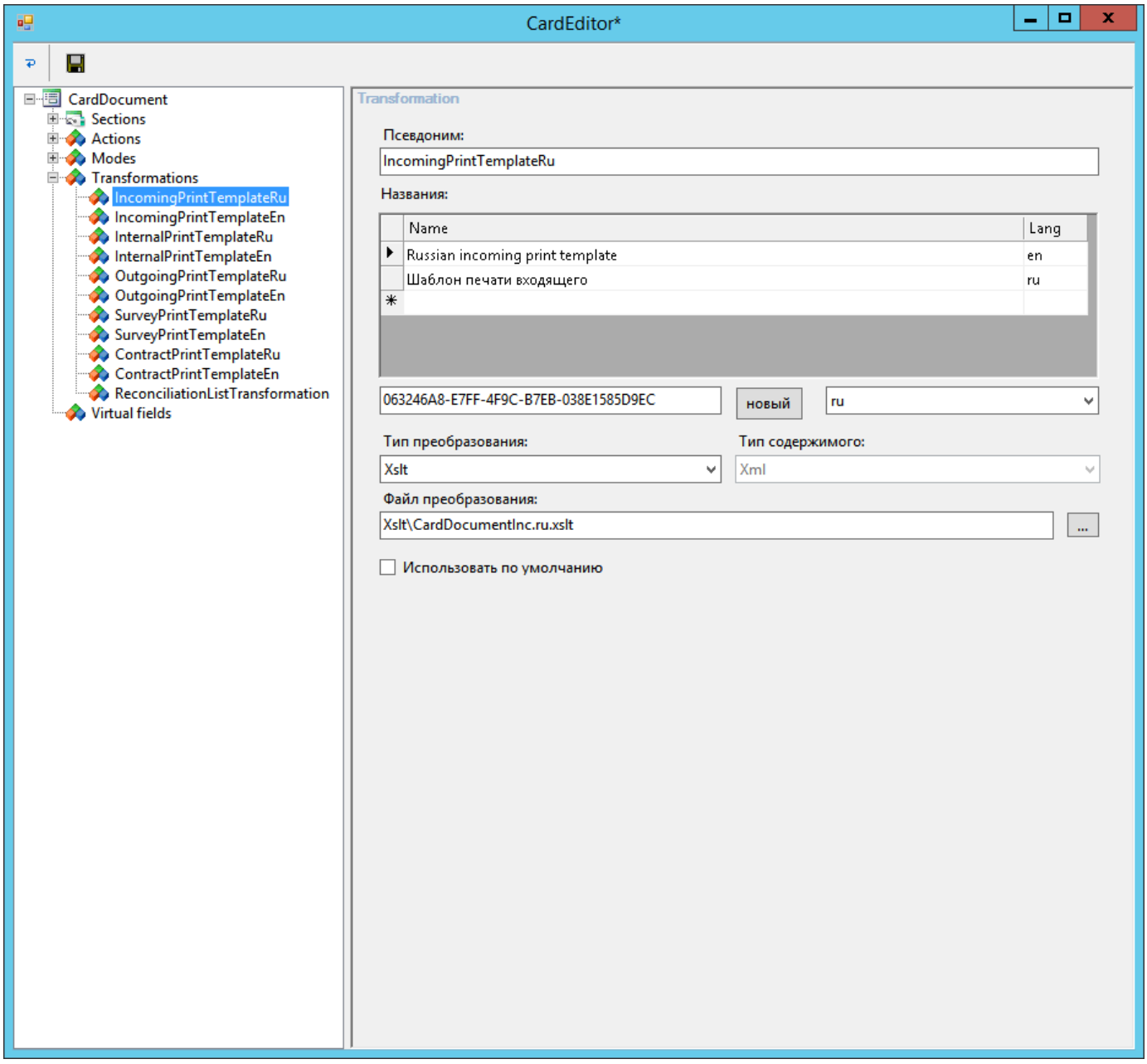


Рисунок 23. Свойства преобразования

Для преобразования предусмотрены следующие свойства:

- *Пседоним* — псевдоним преобразования, с помощью которого к нему можно обращаться в коде. Псевдоним должен быть уникальным в рамках карточки.
- *Названия* — локализованные названия преобразования, отображаемые у пользователя.
- Уникальный идентификатор преобразования.
- Язык локализации преобразования (каждое преобразование относится только к конкретному языку; нельзя использовать одно и то же преобразование сразу для всех языков).
- *Тип преобразования*:
  - *Xslt* — XSLT.
  - *InfoPath* — Microsoft InfoPath.
  - *Custom* — пользовательский тип.
- *Тип содержимого* — тип данных преобразования (выбирается только для пользовательских преобразований). Может иметь одно из следующих значений:
  - *Text* — текст.
  - *Xml* — XML.
  - *Binary* — двоичные данные.
- *Файл преобразования* — путь к файлу с данными преобразования (например, XSLT).
- *Использовать по умолчанию* — признак основного преобразования.

Преобразования, указанные в схеме карточки, будут присутствовать в её типе по умолчанию. Впоследствии в конкретной базе данных этот набор преобразований может быть дополнен силами пользователей.

После завершения редактирования схемы данных карточки её необходимо сохранить.

Раздел "Virtual fields"

После описания секций и полей карточки можно переходить к описанию *виртуальных полей*. Виртуальные поля представляют собой правила для формирования данных, которые физически не хранятся в самой карточке, но могут быть интересны при построении представлений (отчётов) с участием

карточек такого типа.

В качестве источника данных для виртуального поля могут выступать:

- Физическое поле самой карточки.
- Физическое поле карточки другого типа, которая по каким-то признакам связана с данной.
- Системные данные о карточке, например дата последнего изменения.
- Результат вычислений, произведённых над любым из вышеперечисленных полей.

Виртуальные поля, определённые в схеме карточки, будут всегда доступны в рамках соответствующих секций в диалоге настройки представлений. Помимо этого, пользователи смогут самостоятельно доопределить дополнительные виртуальные поля уже в процессе работы с системой.

Для создания нового виртуального поля нужно перейти в раздел *Virtual fields* и выбрать соответствующий пункт контекстного меню.

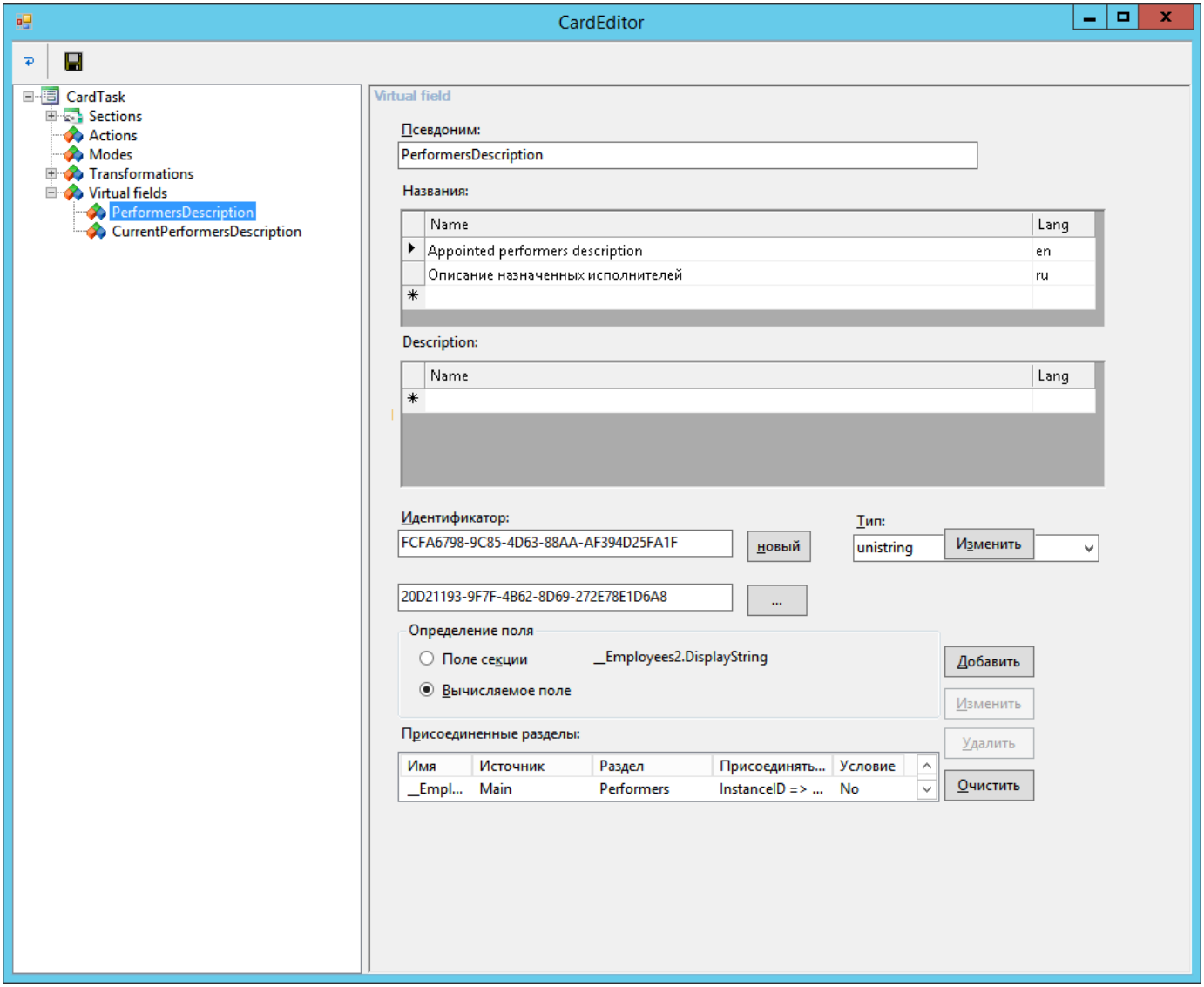


Рисунок 24. Свойства виртуального поля

Основные свойства виртуального поля:

- *Псевдоним* — псевдоним виртуального поля, с помощью которого к нему можно обращаться в коде. Псевдоним должен быть уникальным в рамках карточки.
- *Названия* — локализованные названия виртуального поля, отображаемые у пользователя (в диалоге настройки представлений).
- *Description* — словесное описание виртуального поля, будет отображаться в виде подсказки к полю в диалогах поиска и представлений.
- *Идентификатор* — уникальный идентификатор виртуального поля.
- *Тип* — результирующий тип данных виртуального поля.

  Доступны только простые типы:

- *int* — целое.
- *bool* — булевское значение.
- *datetime* — дата и время.
- *uniqueid* — уникальный идентификатор (`GUID`).
- *string* — строка.
- *unistring* — строка в формате Unicode.
- *float* — число с плавающей точкой.
- *unitext* — текст в формате Unicode.
- *decimal* — десятичное число.
- *binary* — двоичные данные.

- Идентификатор секции, на основе данных которой строится виртуальное поле. Из этой секции будут выбираться физические поля для вычислений, и к ней будут последовательно присоединяться другие секции или таблицы.
- *Определение поля* — источник данных для виртуального поля.

  Доступные источники:

  - *Поле секции* — физическое поле основной или присоединённой секции.
  - *Вычисляемое поле* — правила вычислений над физическим полем основной или присоединённой секции.
- *Присоединенные разделы* — позволяет определить связи основной секции виртуального поля с другими секциями или таблицами. Секции связываются между собой по определённому полю (обычно — уникальному идентификатору).

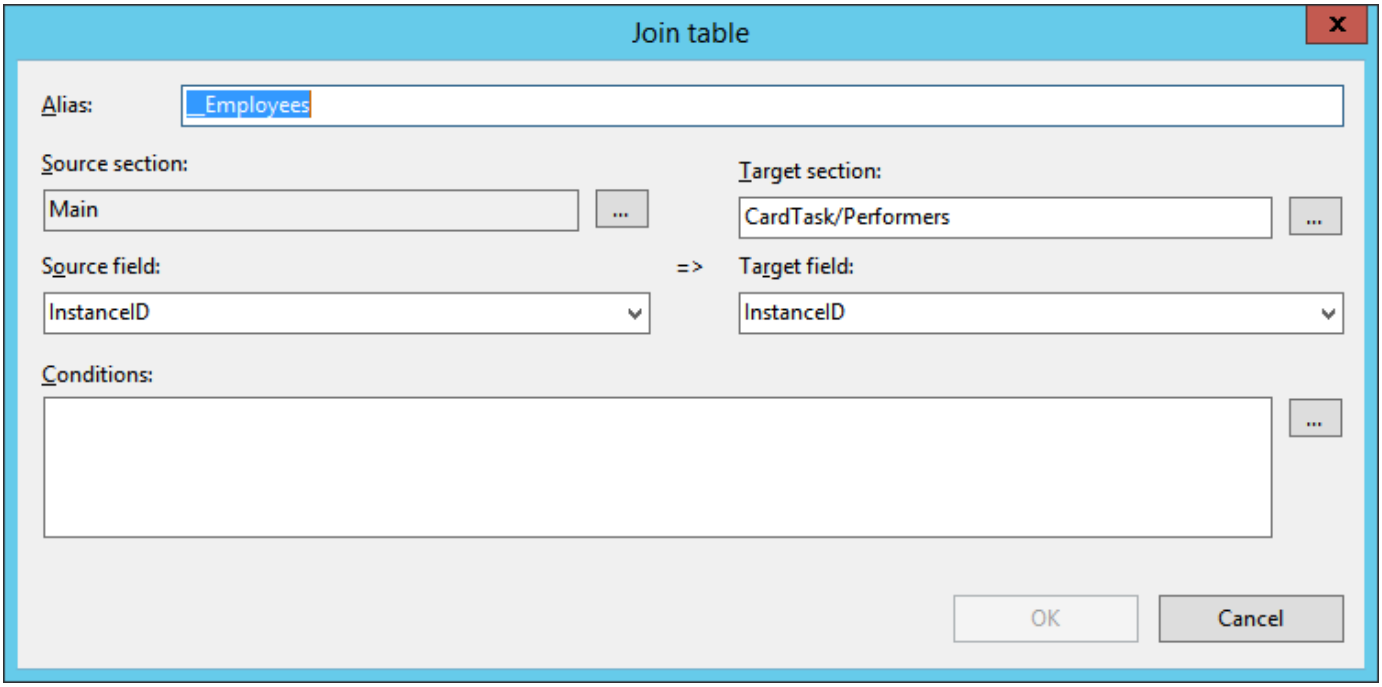


Рисунок 25. Присоединение секции

При присоединении секции, необходимо указать следующие значения:

- *Alias* — уникальный (в рамках определения виртуального поля) псевдоним присоединяемой секции.
- *Source section* — секция, к которой производится присоединение (основная секция виртуального поля или ранее присоединенная).
- *Source field* — поле оригинальной секции, по которому производится присоединение (`JOIN`).
- *Target section* — секция другой карточки или системная таблица.
- *Target field* — поле присоединяемой секции, по которому производится `JOIN`.
- *Conditions* — условия фильтрации записей присоединённой секции.

После присоединения всех необходимых секций можно задать правила построения вычисляемых полей. Для чего кликните по кнопке **Изменить** в блоке настроек поля.

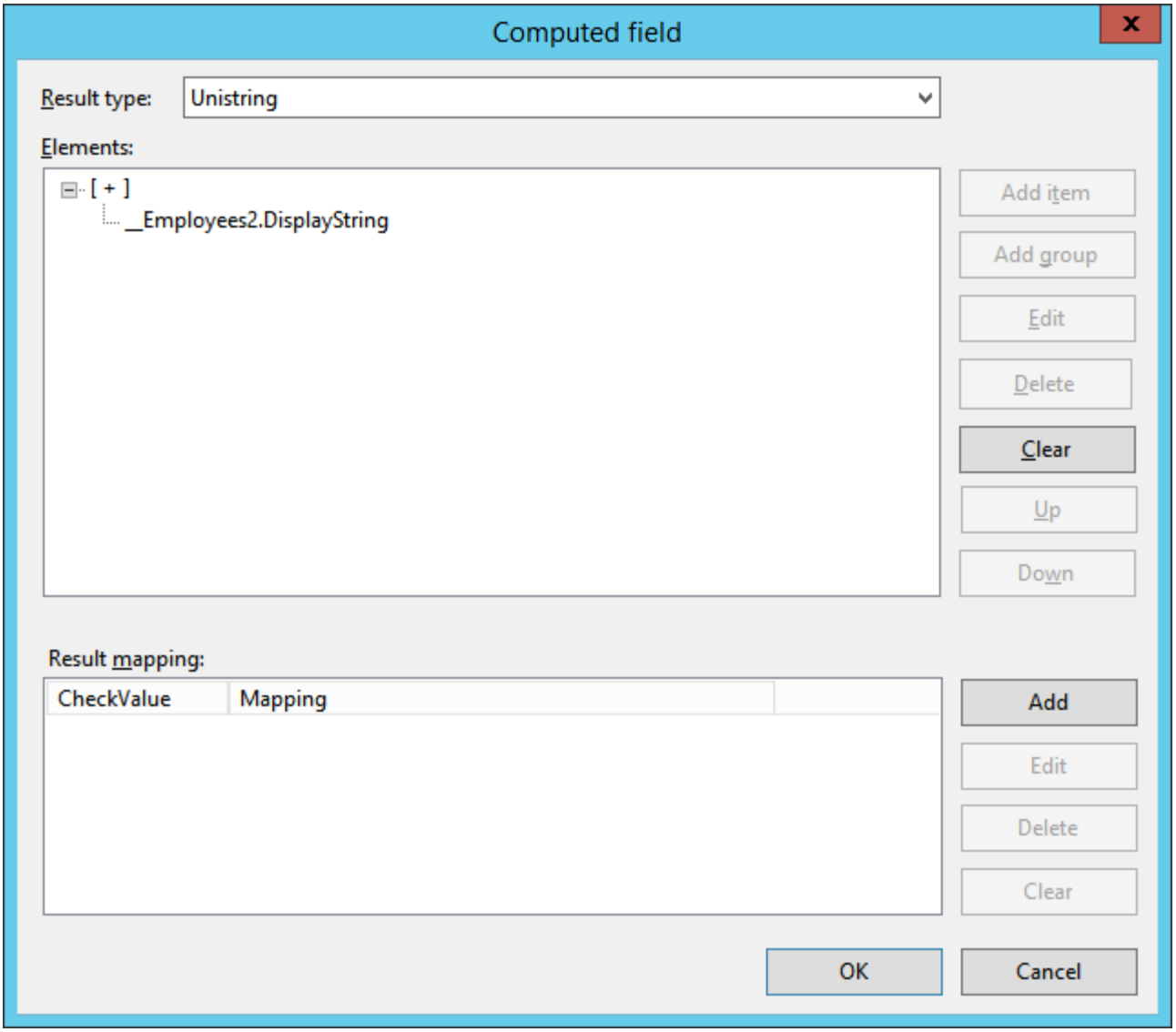


Рисунок 26. Свойства вычисляемого поля

Описание вычисляемого поля по сути представляет собой совокупность атомарных операций (элементов), выполняющихся в определённом порядке, который задаётся через иерархические группы. Внутри каждой группы элементы могут объединяться арифметической операцией (`+`, `-`, `*`, `/`), логической операцией (`AND`, `OR`) или агрегацией.

Вычисления одного уровня исполняются строго последовательно, поэтому важно контролировать порядок их выполнения. Для этого можно перемещать элементы внутри группы кнопками **Up** и **Down**.

Каждый элемент вычислений, в свою очередь, может быть одним из следующих видов:

- *Simple* — значение физического поля секции, предопределённое значение или результат применения к нему простейшей функции.

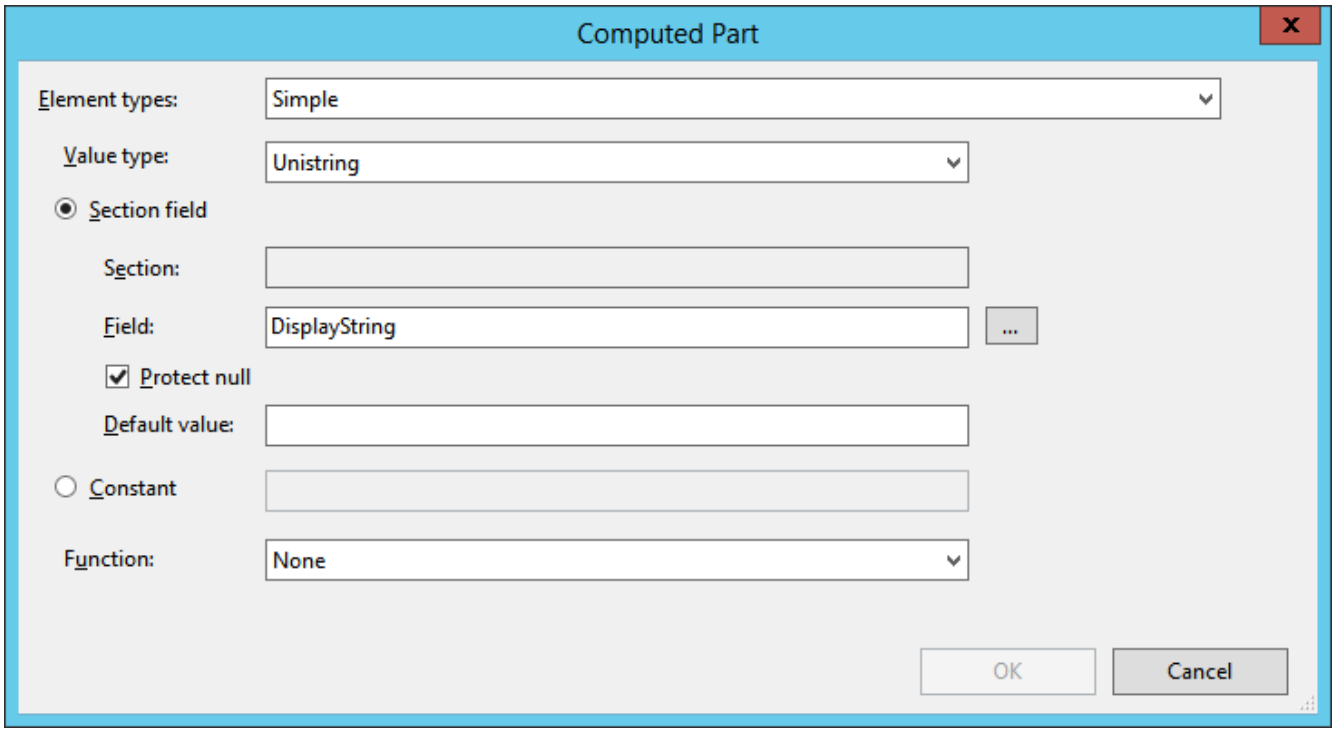


Рисунок 27. Элемент вычисления вида "Simple"

- *Switch* — результат селекции, сопоставляющей конкретным данным физического поля предопределённые значения.

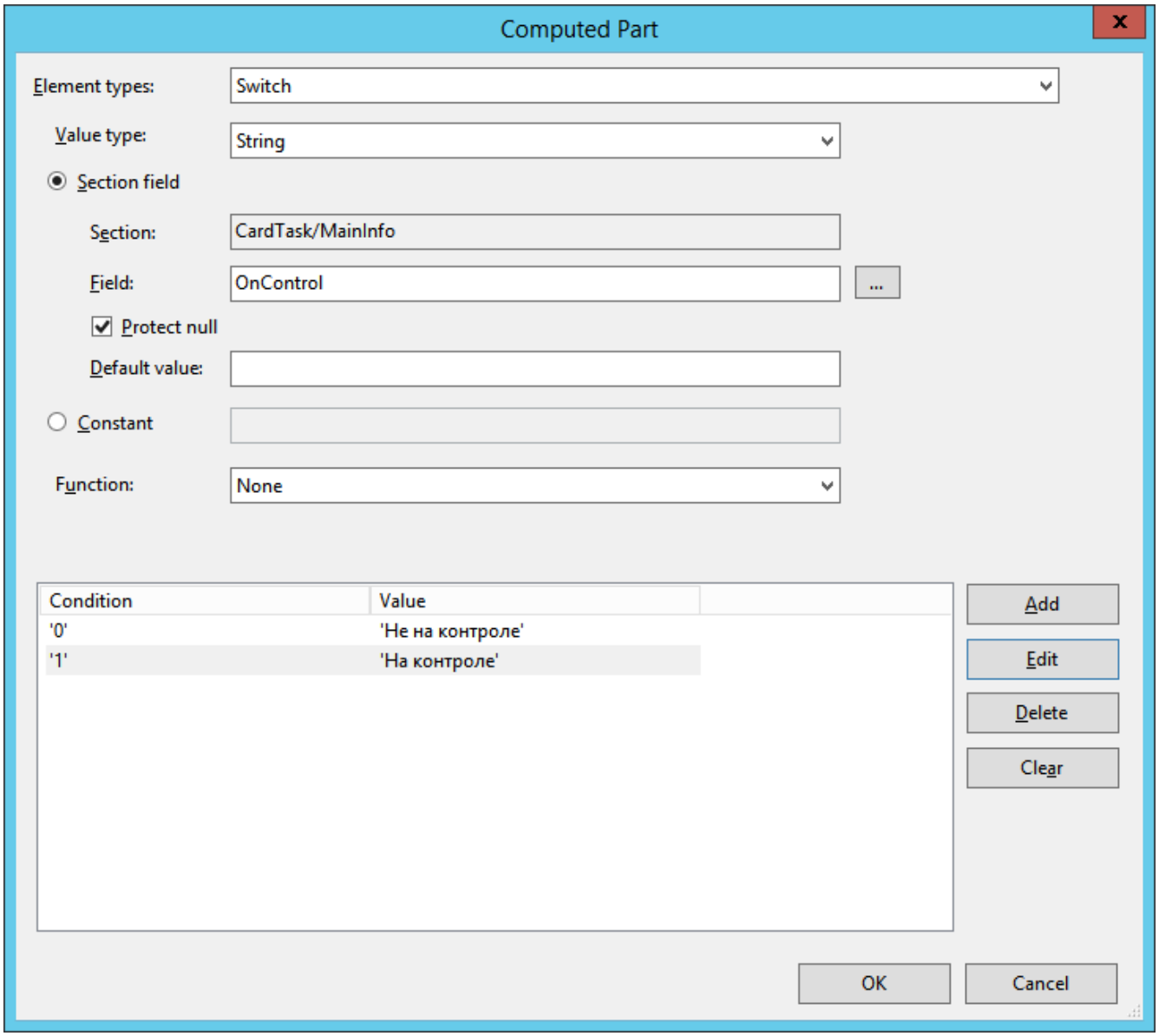


Рисунок 28. Элемент вычисления вида "Switch"

- *Conditions* — условный оператор, формирующий значение элемента в зависимости от значений других элементов.

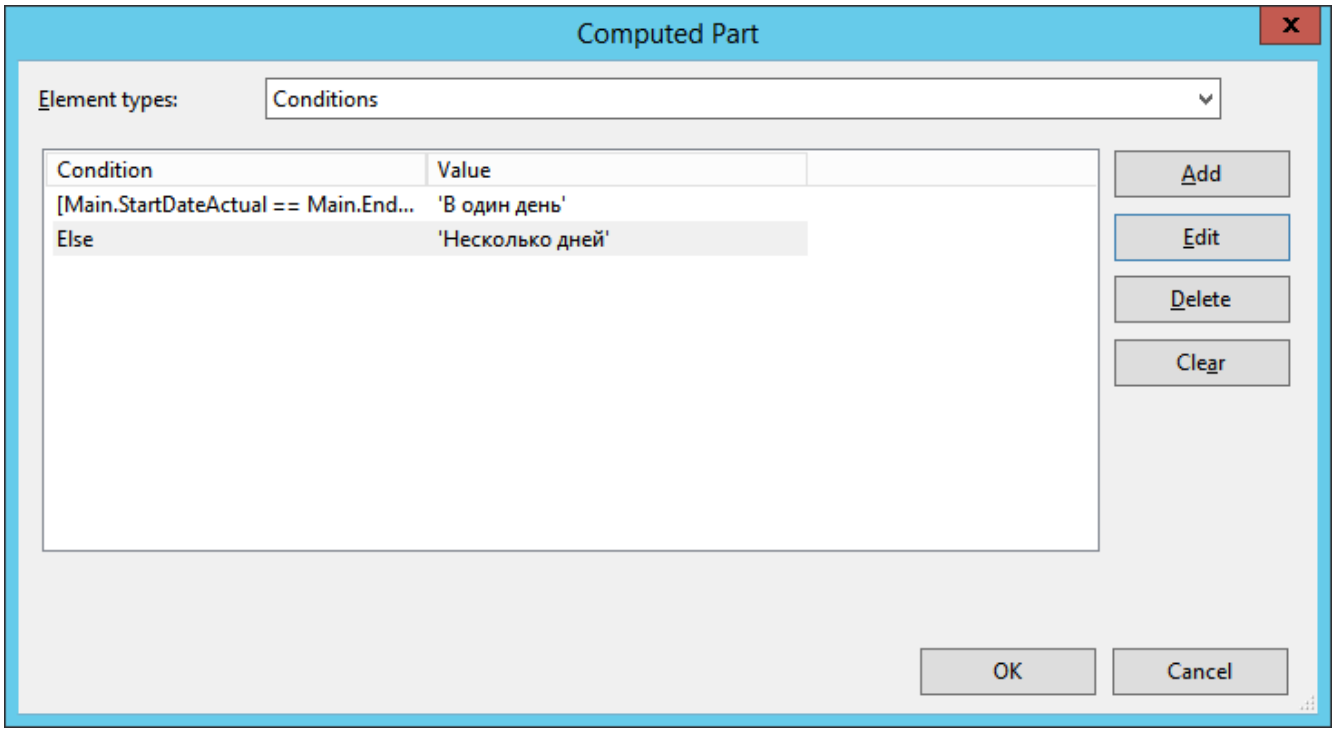


Рисунок 29. Элемент вычисления вида "Conditions"

Важно контролировать типы конкретных элементов вычислений и их групп, чтобы избежать ошибок в вычислениях и получить правильный тип результирующего значения.

Загрузка схем карточек в базу данных

Готовую схему данных карточки (или библиотеки карточек) необходимо загрузить в базу данных Docsvision, чтобы создать в ней необходимую инфраструктуру таблиц, индексов и хранимых процедур.

CardManager предоставляет возможность погрузить в базу данных описание библиотеки карточек с помощью команды *Load library to database*. При этом необходимо выбрать библиотеку для загрузки (из списка библиотек, подключенных к `CardManager`).

Для корректной загрузки собственных библиотек карточек в базу данных, в `CardManager` должны быть подключены описания всех стандартных библиотек, входящих в комплект поставки системы (`SystemCardLib`, `ManagedCardLib`. `TakeOfficeCardLib`, `WorkflowCardLib`). Эти библиотеки включены в состав пакета разработчика и находятся в папке `CardDefs`.

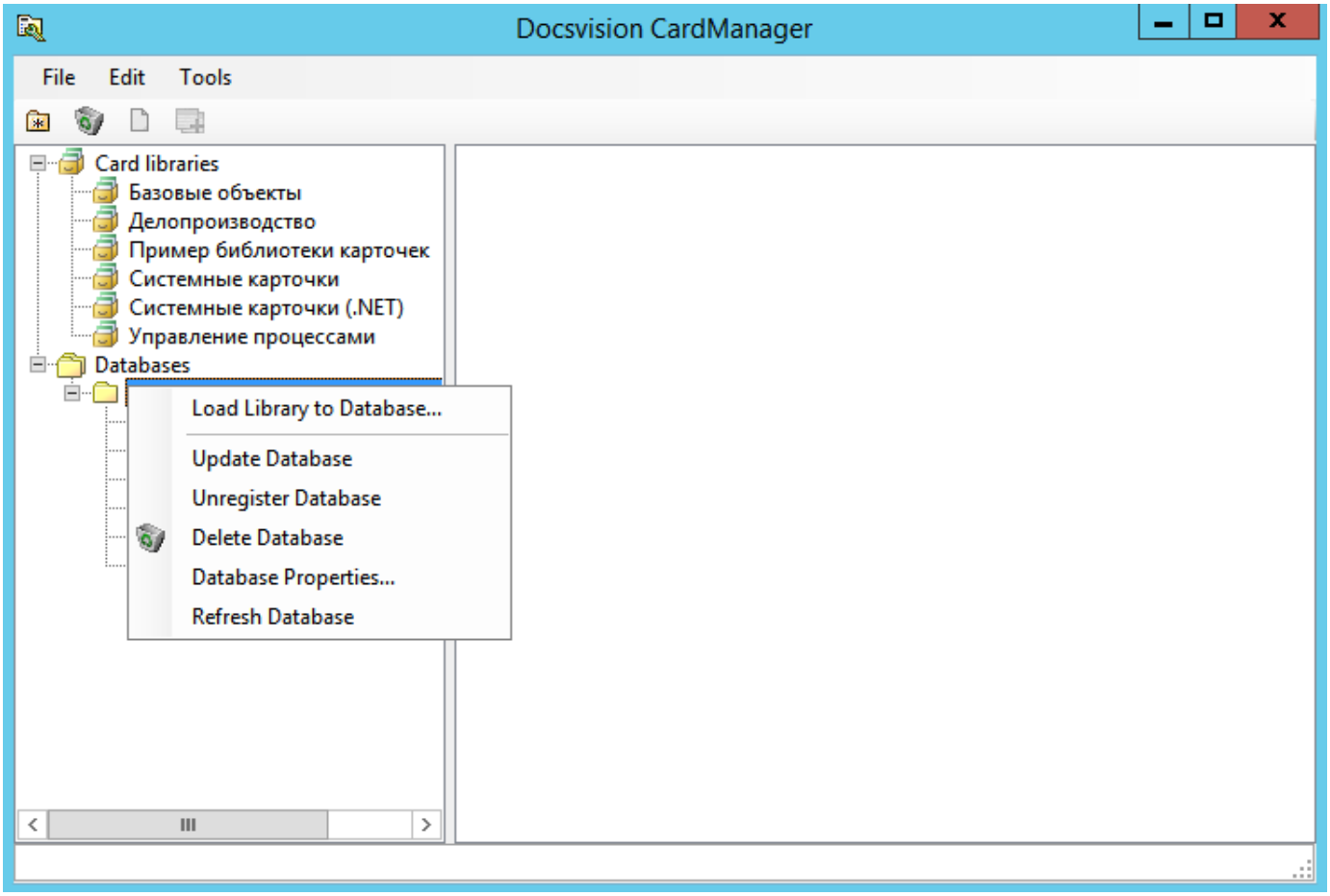


Рисунок 30. Выгрузка схемы карточки в базу данных Docsvision

Если описание карточки уже присутствует в указанной базе данных, существующие структуры данных будут автоматически обновлены согласно внесённым в схему изменениям.

При необходимости можно удалить созданные структуры данных и экземпляры карточек из базы данных. Для этого нужно воспользоваться командами *Delete card* или *Delete library*. Не рекомендуется удалять из базы данных описания системных карточек.

При изменении схемы карточки, чтобы получить возможность использовать модифицированную схему на клиенте, нужно обязательно выполнить следующие действия:

- Повторно скомпилировать *компонент библиотеки карточек*. Комплект библиотеки карточек кэширует схемы всех карточек библиотеки, что позволяет ускорить доступ к ним. Без повторной компиляции будет возвращаться старое описание.
- Перезапустить сервер Docsvision (служба **dvappserver**) — запущенный сервер также кэширует схемы всех карточек. Без

перезапуска будет возвращаться старое описание.

Создание объектной модели карточки

Разработка объектной модели карточки не является обязательным шагом, но в случае её реализации, взаимодействие с карточкой на программном уровне заметно упростится. Кроме этого объектная модель может быть дополнена алгоритмами проверки данных и необходимыми бизнес-функциями.

В этом разделе:

- Разработка объектной модели строки секции карточки
- Разработка объектной модели карточки
- Разработка преобразователя данных

Строки секции карточки объектной модели

В отличие от описания схемы метаданных карточки (или справочника), в данном случае, удобнее разработать сперва объектные модели строк секций, и лишь после этого приступать к самой карточке или справочнику.

Т.к. секции могут быть нескольких типов, далее приведён пример разработки классов для строк каждого из них.

В качестве базового класса строки секции выступает класс `ObjectBase`.

Плоская секция

Для примера опишем секцию, в которой содержится одно текстовое поле — `Name`, а также одно ссылочное `Author`, которое ссылается на сотрудника в *справочнике сотрудников*.

В таком случае объектная модель секции строки будет выглядеть так:

```
public class SampleCardStructSection : ObjectBase
{
 public static readonly ObjectProperty AuthorProperty; ①
 public static readonly ObjectProperty NameProperty; ②

 public StaffEmployee Author ③
 {
  get { return (StaffEmployee)base.GetValue(SampleCardStructSection.AuthorProperty); }
  set { base.SetValue(SampleCardStructSection.AuthorProperty, value); }
 }
 public string Name
 {
```

```
    get { return (string)base.GetValue(SampleCardStructSection.NameProperty); }
    set { base.SetValue(SampleCardStructSection.NameProperty, value); }
  }

  static SampleCardStructSection() ④
  {
    SampleCardStructSection.AuthorProperty = ObjectProperty.Register("Author", typeof
(StaffEmployee), typeof(SampleCardStructSection)); ⑤
    SampleCardStructSection.NameProperty = ObjectProperty.Register("Name", typeof(string),
typeof(SampleCardStructSection));
  }

  internal SampleCardStructSection() ⑥
  {
  }

  internal SampleCardStructSection(ObjectInitializationData data) : base(data) ⑦
  {
  }
}
```

① Вначале идет объявление всех доступных полей. Всегда типа `ObjectProperty`.

② Рекомендуется использовать название поля из схемы метаданных с добавлением суффикса `Property`.

③ Публичные свойства для обращения получения и записи значения соответствующего поля.

Тип свойства соответствует типу поля в схеме метаданных.

Для ссылочных полей тип соответствует типу сущности, на которую идет ссылка. В данном случае — это объект сотрудник.

④ Обязательный статический конструктор, в котором должна быть выполнена регистрация всех объявленных ранее свойств.

⑤ В метод `Register` передаем название поля (как в схеме метаданных), его тип (как определён выше) и тип самой секции (текущий класс).

⑥ Конструктор для новой карточки.

При желании здесь можно добавить собственную бизнес-логику, а также проверку данных перед записью.

Коллекционная секция

Реализация объектной модели для строки секции данного типа ничем не

отличается от рассмотренной ранее. Есть отличие на этапе описания секции в классе карточки (будет рассмотрено далее).

Иерархическая секция

Для такой секции свойственно наличие подчиненной секции, причем этой подчиненной секцией является сама секция. В этом основное отличие в реализации объектной модели.

```
public class SampleCardTreeSection : ObjectBase
{
 public static ObjectProperty DescendantsProperty; ①

 public ObjectCollection<SampleCardTreeSection> Descendants ②
 {
  get { return (ObjectCollection<SampleCardTreeSection>)base.GetValue
(SampleCardTreeSection.DescendantsProperty); }
 }

 static SampleCardTreeSection()
 {
  SampleCardTreeSection.DescendantsProperty = ObjectProperty.Register("Descendants",
typeof(ObjectCollection<SampleDescendantsSection>),  typeof(SampleCardTreeSection)); ③
 }

 internal SampleCardTreeSection()
 {
 }
 internal SampleCardTreeSection(ObjectInitializationData data) : base(data)
 {
 }
}
```

① Объявление подчиненной секции.

② Свойство, через которое будут доступны строки подчиненной секции, т.е. строки самой этой секции.

③ Связывание секции с соответствующим свойством.
`SampleDescendantsSection` — некая секция, которая в схеме метаданных карточки подчинена секции `SampleCardTreeSection`.

В объектной модели библиотеки карточек *Базовые объекты* базовым классом для строк выступает класс `BaseCardSectionRow`, который предоставляет некоторые дополнительные методы. Примерами методов могут быть: получение дочерней секции по идентификатору, получение значения поля с приведением к типу `Guid`.

Реализация объектной модели строек секций при использовании данного базового класса не изменится.

Разработка объектной модели карточки

В объектной модели карточки должны быть перечислены все (или необходимые) её секции.

Базовым классом карточки выступает всё тот же класс `ObjectBase`.

В качестве примера реализуем объектную модель карточки, содержащую секции, рассмотренные ранее. Регистрация секций в карточке, аналогична регистрации полей в секции:

```
public class SampleCard : ObjectBase
{
①
 public static readonly ObjectProperty StructSectionProperty; ②
 public static readonly ObjectProperty CollectionSectionProperty; ③
 public static readonly ObjectProperty TreeSectionProperty; ④

 public SampleCardStructSection StructSection ⑤
 {
  get
  {
   if (((ObjectCollection<SampleCardStructSection>)base.GetValue(SampleCard
.StructSectionProperty)).Count = 0) ⑥
   {
    ((ObjectCollection<SampleCardStructSection>)base.GetValue(SampleCard
.StructSectionProperty)).Add(new SampleCardStructSection());
   }
   return ((ObjectCollection<SampleCardStructSection>)base.GetValue(SampleCard
.StructSectionProperty)).First<SampleCardStructSection>(); ⑦
  }
 }

 public ObjectCollection<SampleCardCollectionSection> CollectionSection ⑧
 {
  get { return (ObjectCollection<SampleCardCollectionSection>)base.GetValue(SampleCard
.CollectionSectionProperty); }
 }

 public ObjectCollection<SampleCardTreeSection> TreeSection
 {
  get { return (ObjectCollection<SampleCardTreeSection>)base.GetValue(SampleCard
.TreeSectionProperty); }
```

```
    set { base.SetValue(SampleCard.TreeSectionProperty, value); } ⑨
  }

  static SampleCard()
  {
    SampleCard.StructSectionProperty = ObjectProperty.Register("StructSection", typeof
(ObjectCollection<SampleCardCollectionSection>), typeof(SampleCard));
    SampleCard.CollectionSectionProperty = ObjectProperty.Register("CollectionSection",
typeof(ObjectCollection<SampleCardCollectionSection>), typeof(SampleCard));
    SampleCard.TreeSectionProperty = ObjectProperty.Register("TreeSection", typeof
(ObjectCollection<SampleCardTreeSection>), typeof(SampleCard));
  }

  protected internal SampleCard() ⑩
  {
  }
  protected internal SampleCard(ObjectInitializationData data): base(data)
  {
  }
}
```

① Объявляются секции карточки.

② Плоская секция.

③ Коллекционная секция.

④ Иерархическая секция.

⑤ Предоставляется доступ к секциям, а точнее к их строчкам.

⑥ Если секция пустая (не содержит строчек), в ней создаётся строчка. Необязательно.

⑦ Т.к. в секции может быть несколько строчек, для плоской секции возвращается только первая строчка.

⑧ Коллекционная секция возвращает весь список своих строчек.

⑨ Добавляется возможность записывать значение секции напрямую. Необязательно.

⑩ Конструкторы для новой карточки и для существующей карточки.

Реализация объектной модели справочника ничем не отличается от реализации для карточки.

В библиотеке карточек *Базовые объекты* базовым классом карточки является класс `BaseCard`, дополняющий функциональность класса `ObjectBase`. Базовый

класс также можно использовать в качестве базового для объектной модели свой карточки. Для справочников в данном случае базовым классом выступает `BaseDictionaryCard`.

Разработка преобразователя данных

Неотъемлемой частью разработки объектной модели карточки является разработка *преобразователя данных* — объект, в котором объясняется каким образом элементы объектной модели карточки (или секций) связаны с данными этой карточки в базе данных.

Контекст объектов задействует *преобразователь данных* на этапе создания экземпляра карточки (к примеру, методом `GetObject`), а также при сохранении новых или изменённых данных.

Преобразователь данных разрабатывается для каждого созданного класса строки секции, а также для самой карточки.

С программной точки зрения, преобразователь данных должен вернуть объект типа `ObjectMap` (карта преобразователя), который содержит списки, связывающие элементы объектной модели с их эквивалентами в схеме метаданных карточки.

Далее рассматривается пример разработки преобразователей для строк секций и карточки, рассмотренных ранее.

Преобразователь данных для строки секции

Базовым классом преобразователя для строки (не зависит от типа секции) является класс `CardRowMapper`. Для класса `CardRowMapper` указывается тип, соответствующий классу объектной модели строки для которой он создаётся.

Для строки плоской секции, разработанной ранее, преобразователь будет выглядеть следующим образом:

```
internal sealed class SampleCardStructSectionMapper : CardRowMapper
<SampleCardStructSection>
{
 private static ObjectMap map;

 static SampleCardStructSectionMapper() ①
 {
  SampleCardStructSectionMapper.InitializeObjectMap(); ②
 }

 public SampleCardStructSectionMapper(ObjectContext context) : base(context) ③
```

```
 {
 }

 protected override ObjectMap GetObjectMap() ④
 {
  return SampleCardStructSectionMapper.map;
 }

 protected override SampleCardStructSection CreateObject(ObjectInitializationData data)
⑤
 {
  return new SampleCardStructSection(data);
 }

 private static void InitializeObjectMap() ⑥
 {
  SampleCardStructSectionMapper.map = new ObjectMap();

  SampleCardStructSectionMapper.map.ObjectTypeId = new Guid("78006F34-55DF-497F-BD62-
E0A33C8EEABF"); ⑦

  SampleCardStructSectionMapper.map.Field(SampleCardStructSection.NameProperty, "Name");
⑧

  SampleCardStructSectionMapper.map.Reference(SampleCardStructSection.AuthorProperty,
"Author"); ⑨
 }
}
```

① Обязательный статический конструктор.

② Создаем карту преобразователя.

③ Обязательный конструктор для передачи текущего контекста объектов.

④ Обязательный метод, предоставляющей карту.

⑤ Создание экземпляра объекта секции с заранее подготовленными данными.

⑥ Создание карты преобразователя.

⑦ Указываем идентификатор секции в схеме карточки.

⑧ Связываем обычное поле.

Первый аргумент объявление поля в классе секции, а второй аргумент — название поля в схеме карточки.

⑨ Связываем ссылочное поле.

Преобразовать для строки коллекционной секции не отличается от преобразователя для строки плоской секции.

Отличие в реализации преобразователя для строки иерархической секции в наличии в карте объявления подчиненных секций:

```
public class SampleCardTreeSectionMapper : CardRowMapper<SampleCardTreeSection>
{
 private static ObjectMap map;
 static SampleCardTreeSectionMapper()
 {
  SampleCardTreeSectionMapper.InitializeObjectMap();
 }
 public SampleCardTreeSectionMapper(ObjectContext context) : base(context)
 {
 }
 protected override ObjectMap GetObjectMap()
 {
  return SampleCardTreeSectionMapper.map;
 }
 protected override SampleCardTreeSection CreateObject(ObjectInitializationData data)
 {
  return new SampleCardTreeSection(data);
 }

 private static void InitializeObjectMap()
 {
  SampleCardTreeSectionMapper.map = new ObjectMap();
  SampleCardTreeSectionMapper.map.ObjectTypeId = new Guid("08ACDC17-EAB7-4E66-8C6D-
428F09490F3A");

  SampleCardTreeSectionMapper.map.Collection(SampleCardTreeSection.DescendantsProperty,
new Guid("80DB0D9B-D1E8-4A4D-8001-E71AC07B10CE")); ①
 }
}
```

① В карту добавляется подчиненная секция. Первым аргументом передается объявление секции в родительской секции, а вторым — её идентификатор из схемы метаданных.

Преобразователь данных для карточки

Преобразователь данных для карточки оформляется аналогичным образом, за исключением базового класса (должен быть класс `BaseCardMapper<T>`) и того, что в карте присутствуют исключительно секции.

```
public class SampleCardMapper : CardMapper<SampleCard>
{
 private static ObjectMap map;
 static SampleCardMapper()
 {
  SampleCardMapper.InitializeObjectMap();
 }
 public SampleCardMapper(ObjectContext context) : base(context)
 {
 }
 protected override ObjectMap GetObjectMap()
 {
  return SampleCardMapper.map;
 }
 protected override SampleCard CreateObject(ObjectInitializationData data)
 {
  return new SampleCard(data);
 }
 private static void InitializeObjectMap()
 {
  SampleCardMapper.map = new ObjectMap();

  SampleCardMapper.map.ObjectTypeId = new Guid("E1BF5846-FE0C-424B-9B71-B58D1A526BCF");
①

  SampleCardMapper.map.Collection(SampleCard.StructSectionProperty, new Guid("A3F0B456-
587E-4769-B019-467AD4EB9BF8")); ②
  SampleCardMapper.map.Collection(SampleCard.CollectionSectionProperty, new Guid
("F6E9E009-7447-435D-8DB8-C3E4187E2D61"));
  SampleCardMapper.map.Collection(SampleCard.TreeSectionProperty, new Guid("EA968311-
3702-4568-9663-2CE9CBE09CC6"));
 }
}
```

① Идентификатор типа карточки.

② Регистрация секций в карте преобразователя.

Преобразователь данных для справочника

Класс преобразователя данных для справочника реализуется по аналогии с преобразователя данных карточки. Единственное отличие — базовый класс здесь `DictionaryMapper<T>`.

```
public class SampleDictionaryMapper : DictionaryMapper<SampleDictionary>
{
```

```
 private static ObjectMap map;
 static SampleDictionaryMapper()
 {
  SampleDictionaryMapper.InitializeObjectMap();
 }
 public SampleDictionaryMapper(ObjectContext context) : base(context)
 {
 }
 protected override ObjectMap GetObjectMap()
 {
  return SampleDictionaryMapper.map;
 }
 protected override SampleDictionary CreateObject(ObjectInitializationData data)
 {
  return new SampleDictionary(data);
 }
 private static void InitializeObjectMap()
 {
  SampleDictionaryMapper.map = new ObjectMap();

  SampleDictionaryMapper.map.ObjectTypeId = new Guid("4E32BC51-5F50-4F68-9D0F-
3BFBB6523B14"); ①
  SampleDictionaryMapper.map.Collection(SampleDictionary.TreeSectionProperty, new Guid
("0CD96A19-59EF-44EC-AF47-977A56BDDB32"));
 }
}
```

① Идентификатор типа справочника.

Регистрация преобразователей данных

Когда классы преобразователей данных били разработаны, требуется создать ещё один класс — *фабрика преобразователей данных*. В созданном классе нужно связать тип преобразователя к классу карточки или секции, за который он отвечает. Базовым классом для *фабрики преобразователей данных* выступает класс `ObjectMapperFactory`.

```
public sealed class SampleCardsMapperFactory : ObjectMapperFactory
{
 public SampleCardsMapperFactory(ObjectContext context) : base(context)
 {
  base.RegisterObjectMapper(typeof(SampleCardStructSection), typeof
(SampleCardStructSectionMapper)); ①
  base.RegisterObjectMapper(typeof(SampleCardCollectionSection), typeof
(SampleCardCollectionSectionMapper));
  base.RegisterObjectMapper(typeof(SampleCardTreeSection), typeof
```

```
(SampleCardTreeSectionMapper));
  base.RegisterObjectMapper(typeof(SampleCard), typeof(SampleCardMapper));
 }
}
```

① В метод передается тип объектной модели строки секции или карточки, а также тип соответствующего преобразователя.

Чтобы контекст объектов мог работать с типами карточек и секций, для которых разработаны *фабрики преобразователей данных*, эти фабрики должны быть добавлены в контекст (см. на примере добавления фабрики `BackOfficeMapperFactory` в примере).

Разработка компонента карточки

После завершения проектирования схемы данных карточки и загрузки её в базу данных Docsvision необходимо разработать соответствующий программный компонент карточки.

Карточка может и не иметь пользовательского интерфейса, например если она используется только в качестве хранилища данных для других карточек. В этом случае данный этап может быть пропущен.

Программный компонент карточки представляет собой элемент управления ActiveX, предоставляющий пользовательский интерфейс для работы с данными карточки. Элемент управления вызывается Windows-клиентом при активации карточки ("открытии") или выполнении команд карточки.

Можно выделить несколько типов программных компонент карточек:

- **Карточки документов** — программные компоненты, реализующие только интерфейс для ввода и редактирования данных.

  Типичными примерами карточек документов являются карточки библиотеки "Базовые объекты": *Документ*, *Задание* и другие.

- **Карточки справочников** — программные компоненты, предназначенные для редактирования данных справочников.

  Помимо средств для ввода и редактирования данных, предоставляют специальный интерфейс для выбора элементов справочника.

- **Карточки расширения** — программные компоненты, реализующие расширения функциональности Windows-клиента.

  Например, при помощи карточек расширения можно реализовать специфические механизмы поиска, экспорта и импорта данных или интеграции с внешними системами. Такой компонент делается на основе класса `NavExtension`, унаследованного от класса `CardControl`. Компонент получает все события `CardControl` и в некоторых случаях может быть использован в том числе и для отображения пользовательского интерфейса карточки.

- **Специализированные карточки** — системные карточки, служебные компоненты или утилиты, предназначенные для использования в составе других карточек или реализации специфической функциональности.

  К числу таких карточек можно отнести системную карточку файла, карточку *Список файлов* из библиотеки "Базовые объекты" и другие.

Разработка программного компонента состоит из нескольких этапов:

- Реализация стандартных интерфейсов — программирование кода карточки в соответствии со стандартами платформы Docsvision для обеспечения её корректной работы в системе.
- Реализация пользовательского интерфейса — включает в себя проектирование и реализацию вариантов взаимодействия кода карточки с пользователем, дизайн и расположение элементов управления на форме карточки.
- Реализация дополнительной функциональности — дополнительные функции карточки, связанные с её назначением.
- Отладка и тестирование — проверка корректной работы разработанного компонента в составе системы.

В качестве инструмента для создания карточки может быть использована любая среда программирования, позволяющая реализовывать компоненты стандарта ActiveX и осуществлять взаимодействие с COM-компонентами.

Примерный перечень сред и языков, в которых может быть разработана карточка Docsvision:

- Microsoft Visual Studio 6.0 (Visual Basic, Visual C++);
- Microsoft Visual Studio 2003, 2005, 2008, 2010 (VB.NET, C#);

- Borland Delphi и другие.

В данном Руководстве примеры кода приведены для Microsoft .NET (C#).

Реализация стандартных интерфейсов

Сначала необходимо создать новый проект — компонент, который будет реализовывать визуальную часть карточки. Расширения Windows-клиента также являются специальными компонентами карточек. Такой компонент делается на основе класса `NavExtension`, унаследованного от класса `CardControl`. Компонент получает все события `CardControl` и в некоторых случаях может быть использован в том числе и для отображения пользовательского интерфейса карточки.

Проект должен иметь тип "Windows Forms Control Library":

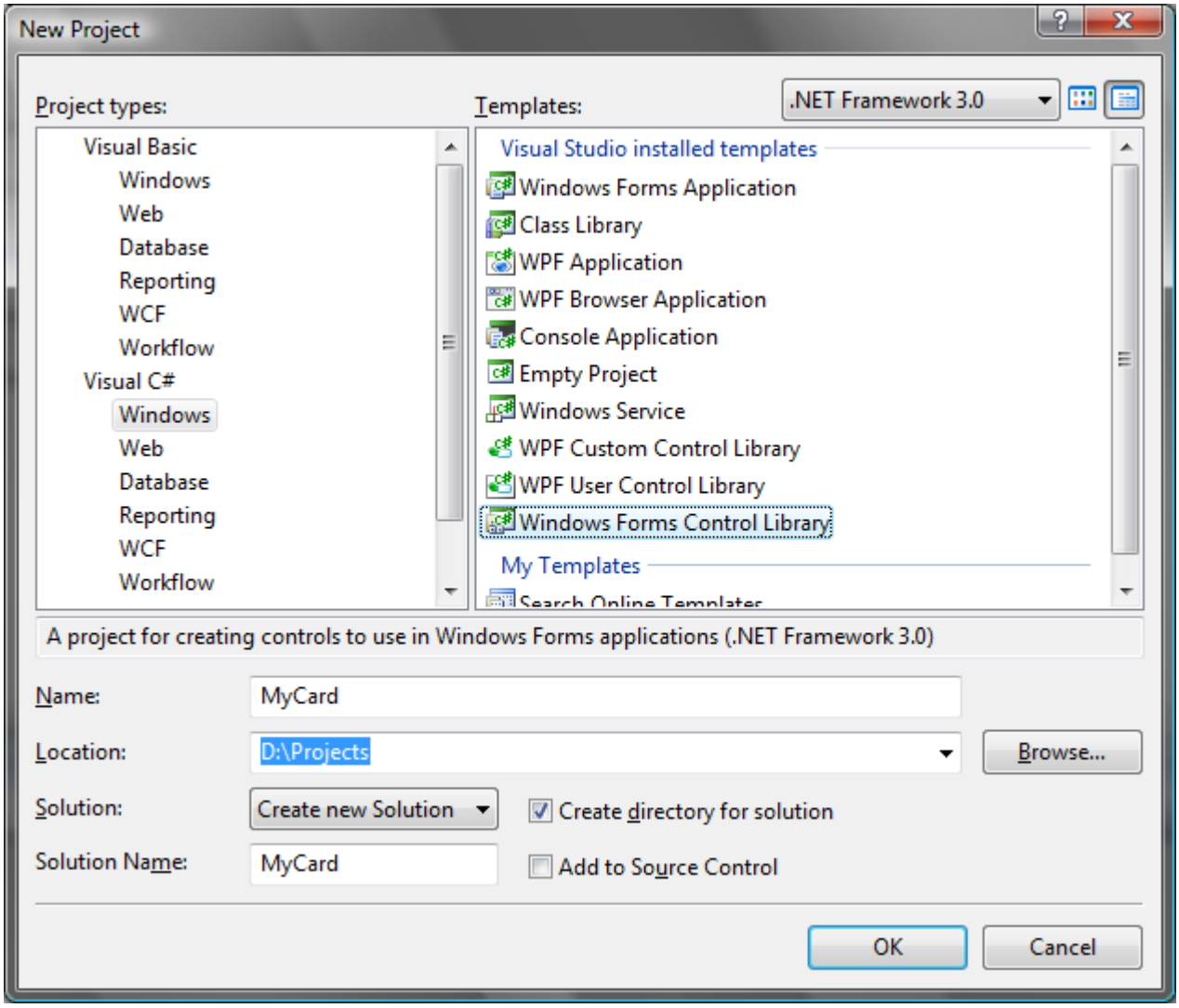


Рисунок 31. Создание проекта "Windows Forms Control Library"

К проекту требуется подключить следующие библиотеки:

- `DocsVision.Platform.WinForms.dll` — содержит интерфейс для взаимодействия с контейнером карточек (средой, в которой запускаются и работают компоненты карточек), а также элементы управления Docsvision для карточек в среде WinForms.

Вместо `DocsVision.Platform.WinForms.dll`, можно использовать библиотеку `DocsVision.Platform.WPF.dll` — она позволяет разрабатывать карточки по технологии Windows Presentation Foundation (WPF). При этом основные принципы разработки и интерфейсы остаются точно такими же, как и в случае WinForms.

- `DocsVision.Platform.ObjectManager.dll` — библиотека менеджера объектов. В этой библиотеке определены основные интерфейсы для взаимодействия с сервером и другими объектами системы (API).

Эти сборки входят в состав как серверной, так и клиентской программ инсталляции. В случае установки только клиентской программы, их можно взять из папки пользователя, для которого проведена установка: `C:\Users\USERNAME\AppData\Local\DocsVision\Client`.

При выполнении установки сервера сборки можно взять из папки: `\DocsVision\Platform\`

Следует подключить пространства имён этих сборок в область видимости текущего проекта при помощи директивы `using`:

```
using DocsVision.Platform.WinForms;
using DocsVision.Platform.ObjectManager;
```

Чтобы карточка могла быть использована совместно с другими компонентами платформы, её программный компонент должен реализовывать набор предопределённых интерфейсов, которые уже реализованы в специальном классе DocsVision.Platform.WinForms.CardControl. Все что остается сделать — это унаследовать основной класс своей карточки от базового класса `CardControl` вместо стандартного `UserControl`:

```
public sealed partial class TestCard : CardControl
{
```

```
}
```

После этого в коде карточки автоматически появится доступ к членам базового класса, среди которых самые важные:

- `CardData` — объект данных текущей карточки.
- `CardHost` — объект контейнера карточек.
- `Session` — текущая сессия пользователя (точка входа к остальным объектам API).

Дополнительные свойства, доступные из базового класса:

- `FolderID` — идентификатор папки, из которой открыта карточка.
- `ShortcutID` — идентификатор ярлыка, из которого открыта карточка.
- `ModeID` — идентификатор текущего режима карточки.

Готовый код компонента карточки может выглядеть следующим образом (такая карточка будет создаваться и активироваться, но не несет в себе никакой полезной функциональности):

```
using System;
using System.Windows.Forms;
using DocsVision.Platform.ObjectManager;
using DocsVision.Platform.WinForms;

namespace DocsVision.Test
{
 public sealed partial class TestCard : CardControl
 {
  public TestCard()
  {
   InitializeComponent();
  }
 }
}
```

Чтобы карточка поддерживала механизм настройки разметок через справочник "Конструктор разметок", а также обладала дополнительной встроенной функциональностью (работа с бизнес-процессами, переходы по автомату состояний) следует унаследовать класс не от `CardControl`, а от `BaseCardControl`, входящей в сборку `DocsVision.BackOffice.WinForms.dll`. Данная сборка входит в

клиентский инсталляционный пакет Docsvision и является одним из компонентов библиотеки карточек "Базовые объекты".

`BaseCardControl` в свою очередь унаследован от `CardControl`, поэтому все вышеуказанные свойства также будут доступны. Кроме того, в такой карточке автоматически можно использовать элементы объектной модели:

- `BaseObject` — базовый объект карточки, предоставляющий доступ к её основным свойствам.
- `ObjectContext` — контекст объектов, позволяющий работать с другими объектами системы.
- **Набор сервисов** для использования функциональности существующих карточек (каждый сервис также доступен через `ObjectContext`)

Шаблон класса карточки, при использовании базового класса `BaseCardControl` и Конструктора разметок, требует изменения:

```
[ComVisible(true)]
[Guid("E41A2D89-300A-4649-AAA7-39634E72FD91")]
[ClassInterface(ClassInterfaceType.None)]
[Customizable(true)] ①
[CardFrameWindowType(typeof(CardFrameForm))]
public partial class SampleCardClass : DocsVision.BackOffice.WinForms.BaseCardControl
{
 public SampleCardClass() ②
 {
  if (IsInDesignMode())
  {
   InitializeComponent();
  }
 }

 protected override void OnCardInitialized(EventArgs e) ③
 {
  base.OnCardInitialized(e);
  InitializeComponent();
 }
}
```

① Разрешает настраивать карточку, в т.ч. в "Конструкторе разметок".

② Для отображения элементов управления, определённых в коде, в Конструкторе разметок, необходимо изменить конструктор класса.

③ Для работы с разметками карточки, инициализация должна быть

осуществлена не в конструкторе, а в методе `OnCardInitialized`.

Если в настройке карточки нет необходимости, то можно упростить класс карточки:

```
[ComVisible(true)]
[Guid("E41A2D89-300A-4649-AAA7-39634E72FD91")]
[ClassInterface(ClassInterfaceType.None)]
[CardFrameWindowType(typeof(CardFrameForm))]
public partial class SampleCardClass : DocsVision.BackOffice.WinForms.BaseCardControl
{
 protected override void OnCardInitialized(EventArgs e)
 {
  base.OnCardInitialized(e);
  InitializeComponent();
 }
}
```

Обработка событий

Жизненный цикл компонента карточки предусматривает ряд ключевых событий, которые могут быть обработаны в коде компонента. Обработчики для этих событий уже существуют в базовом классе `CardControl`, и содержат код, необходимый для нормального функционирования компонента карточки.

Однако, при желании разработчик может переопределить одно или несколько из этих событий, чтобы реализовать в них собственную логику.

При переопределении событий, нужно не забыть вызвать обработчик базового класса!

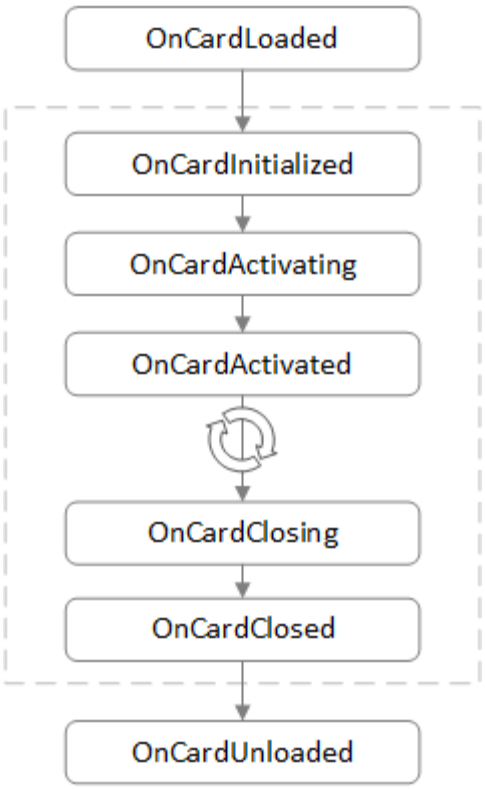


Рисунок 32. События жизненного цикла карточки

Внешний контур схемы охватывает полный жизненный цикл компонента карточки, от момента его создания до момента выгрузки из памяти. Внутренний контур содержит последовательность событий, возникающий при активации компонента карточки для каждого независимого экземпляра данных.

Windows-клиент кэширует компонент карточки в памяти после первого обращения к нему, и впоследствии активирует этот же самый компонент при открытии всех карточек такого типа.

Поэтому важно не забывать всегда инициализировать и очищать элементы управления, чтобы в них не осталось данных со времени предыдущей активации.

При использовании базовой карточки `BackOffice` реализована собственная обработка закрытия карточки — стандартные обработчики событий переопределять не требуется. Вместо этого следует переопределять более специализированные методы сохранения данных:

- `bool CheckDataChanges()` — в реализации данного метода необходимо вернуть значение `true` в случае, если пользователь внес какие-то изменения в карточку в процессе работы с ней, иначе — `false`.

Результат метода влияет на то, будет ли предложено пользователю сохранить карточку при её закрытии. Изменения базового объекта карточки будут проверены автоматически.

- `CommitDataChanges()` — в данном методе необходимо произвести сохранение всех изменений в карточке, совершенных пользователем на текущий момент.

  Если работа ведется только через `ObjectContext`, перегружать данный метод нет необходимости — изменения будут сохранены автоматически.

- `RollbackDataChanges()` — в данном методе следует произвести отмену совершенных пользователем изменений в карточке.

  Как и в случае с `CommitDataChanges`, если работа ведется только с объектами, полученными через `ObjectContext`, данный метод перегружать не надо — контекст автоматически будет возвращен к первоначальному состоянию.

Фоновый поток исполнения бизнес-логики при открытии карточки

Базовый класс `BaseCardControl` содержит методы и события, которые позволяют вынести часть логики, исполняемой при открытии карточки, в отдельный поток. Данная возможность может быть востребована, к примеру, если необходимо дополнить процесс открытия карточки собственным алгоритмом, исполнение которого может занять продолжительное время.

Чтобы вынести исполнения дополнительной логики в отдельный поток достаточно переопределить два метода базового класса:

- `OnCardActivatedAsync` — здесь размещается бизнес-логика карточки, которая должна быть выполнена в отдельном потоке.
- `OnCardActivatedAsyncCompleted` — здесь может размещаться дополнительная функциональность, связанная с обновлением интерфейса.

При реализации аналогичной логики в скрипте карточки (с использованием *конструктора разметок*), соответствующая логику привязывается к событиям `CardActivatedAsync` и `CardActivatedAsyncCompleted`.

Повторное использование компонента карточки

Компонент карточки может сообщить Windows-клиенту о поддержке им механизма повторного использования (кэширования), при активации которого

компонент карточки не выгружается из памяти при закрытии карточки, а кэшируется в специальном хранилище и используется повторно, при обращении к карточкам данного типа.

Чтобы задействовать данный режим, необходимо:

- Для карточек, унаследованных от типа `CardControl`, нужно реализовать в классе компонента карточки интерфейс `IReusableCardComponent`:

```
public sealed partial class TestCard : CardControl, IReusableCardComponent
{


 public bool CanBeReused
 {
  get { return true; } ①
 }

 public ReusageMode Mode
 {
  get { return ReusageMode.OwnLogic; } ②
 }
}
```

① Определяем необходимость повторного использования компонента: `true` — используется, `false` — не используется.

② Указываем режим повторного использования.

- Для карточек, унаследованных от типа `BaseCardControl`, нужно переопределить методы `GetCanBeReused` и `GetReusageMode`:

```
public sealed partial class TestCard : BaseCardControl
{


 protected override bool GetCanBeReused() ①
 {
  return true;
 }

 protected override ReusageMode GetReusageMode() ②
 {
  return ReusageMode.OwnLogic;
 }
```

```
}
```

① Определятся необходимость повторного использования компонента.

② Определяется собственный режим повторного использования.

В режиме `ReusageMode.OwnLogic` необходимость повторного использования определяется по значению переменной `CanBeReused`. В режиме `ReusageMode.NoReusage` компонент карточки кэшироваться не будет. В режиме `ReusageMode.CardHostLogic` необходимость кэширования компонента определяется внутренней логикой контейнера карточек.

Реализация пользовательского интерфейса карточки

После завершения реализации стандартных интерфейсов, следующим шагом в разработке компонента карточки является расположение на контрольном элементе карточки различных элементов, предназначенных для управления поведением карточки, ввода, редактирования и удаления данных, а также прочих необходимых функций.

В качестве элементов управления могут быть использованы стандартные элементы управления Visual Studio, дополнительные контрольные элементы сторонних производителей или специализированные элементы управления Docsvision.

Элементы управления Docsvision реализованы в библиотеке `DocsVision.Platform.WinForms.dll`. Для удобства, можно добавить элементы управления на панель инструментов Visual Studio (Toolbox), с выбором указанной библиотеки.

Для карточек, разрабатываемых по технологии WPF, отдельных элементов управления Docsvision не предусмотрено, поэтому при разработке карточки могут использоваться только стандартные элементы управления.

Элементы управления из библиотеки `DocsVision.Platform.WinForms.dll` можно условно разделить на три группы:

- **Источники данных** — эти элементы управления сами не отображаются на форме и не взаимодействуют с пользователем, а служат только в качестве источников данных для других элементов управления.

  К этой группе относятся:

  - `SessionSource` — соединение с сервером Docsvision.

- `CardDataSource` — набор данных карточек.
- `RowDataSource` — набор строк секции карточки.
- `InfoRowDataSource` — данные представления.
- `ReportDataSource` — данные хранимой процедуры.

- **Интерактивные элементы управления** — элементы управления, которые размещаются на форме и с которыми напрямую взаимодействует пользователь.

  К этой группе относятся:

  - `BoundChooseBox` — универсальный элемент управления для выбора значений ссылочных полей.
  - `CardChooseBox` — специализированный элемент управления для выбора карточек.
  - `FolderChooseBox`– специализированный элемент управления для выбора папок.
  - `RowChooseBox` — элемент управления для выбора любых объектов.
  - `BoundTreeView` — элемент управления для работы с иерархической секцией.
  - `WizardControl` — вспомогательный элемент управления для создания Мастеров.
  - `NavigationToolStrip` — навигационная панель инструментов.

- **Диалоги** — вспомогательные диалоги, которые могут быть вызваны из текущей формы, и открываются в новом окне:

  - `SelectDirectoryEntryDialog` — диалог выбора пользователя из Active Directory.
  - `SelectOrganisationalUnitDialog` — диалог выбора подразделения из Active Directory.
  - `SelectDomainDialog` — диалог выбора домена из Active Directory.
  - `SelectIconDialog` — диалог выбора иконки из файла на диске.

Описание данных элементов управление приведено в разделе Устаревшие элементы управления Docsvision.

Кроме стандартных элементов управления Docsvision, описанных выше, могут быть использованы элементы управления из сборки `DocsVision.BackOffice.WinForms.dll`, которые возможно добавить в Toolbox Visual

Studio.

Из сборки доступны следующие элементы управления:

- Элементы выбора данных Docsvision:
  - `CardChooseBox` — элемент управления для выбора карточки, аналогичен `CardChooseBox` из `DocsVision.Platform.WinForms`.
  - `RowChooseBox` — элемент управления для выбора строки справочника, аналогичен `RowChooseBox` из `DocsVision.Platform.WinForms`.
  - `RowMultiChooseBox` — элемент для выбора нескольких значений.
  - `CommunicativeChooseBox` — элемент для выбора сотрудника с возможностью интеграции с Microsoft Office Communicator/Lync.

    В текущей версии системы интеграция с Microsoft Lync и Skype For Business не поддерживается.

  - `CommunicativeMultiChooseBox` — элемент для выбора нескольких сотрудников.
  - `UniversalDirectoryChooseBox` — элемент для выбора значения из Конструктора справочников.
- Сложные элементы управления для работы с базовыми объектами:
  - `CategoryListView` — элемент выбора категорий карточки из дерева категорий.
  - `HistoryGrid` — таблица журнала карточки.
  - `ReferenceListView` — элемент для работы с файлами и ссылками.
  - `TaskTreeView` — элемент для работы со связанными задачами.
- Дополнительные элементы **выбора данных**:
  - `FileSystemFileBox` — элемент выбора файла из файловой системы.
  - `FolderChooseBox` — элемент выбора папки из файловой системы.
- **Вспомогательные**:
  - `CommunicatorControl` — элемент управления, отображающий статус пользователя в Microsoft Office Communicator/Lync, позволяющий осуществить сеанс связи с пользователем (звонок, отправка мгновенного сообщения, отправка электронной почты). Используется в `CommunicativeChooseBox` и `CommunicativeMultiChooseBox`.

    В текущей версии системы интеграция с Microsoft Lync и

Skype For Business не поддерживается.

- `Preview` — элемент просмотра содержимого файла. Используются средства предпросмотра, установленные в операционной системе (аналогично Microsoft Outlook).

Разработка компонента библиотеки карточек

Так же, как и для карточки, реализация библиотеки предусматривает два основных этапа: создание описания (схемы) библиотеки в формате XML и реализация программного компонента.

Основная задача компонента библиотеки карточек — это кэширование XML-описаний (схем) и иконок всех карточек, входящих в библиотеку. Это избавляет клиентов от необходимости выполнять запрос к серверу при каждом обращении к метаданным карточек и, таким образом, существенно повышает производительность их работы.

Помимо этого, по наличию или отсутствию регистрации данного компонента на клиентской машине определяется сам факт установки решения в целом.

Если решение зарегистрировано в базе данных, но при запуске Windows-клиент не может создать компонент библиотеки карточек, им будет предпринята попытка заново установить инсталляционный пакет решения на данной машине. В случае неудачи будет выдано сообщение о том, что библиотека карточек некорректно установлена.

По этой причине компонент библиотеки карточек является обязательной частью законченного решения. Его создание может быть пропущено или отложено в процессе разработки карточек решения (в пределах компьютера разработчика), но к нему обязательно необходимо будет вернуться при распространении созданного решения.

Компонент библиотеки карточек является реализацией интерфейса `ICardLibraryInfo`, который определяет методы получения информации о библиотеке. Данный интерфейс реализован в классе `DocsVision.Platform.WinForms.CardLibrary`, который можно использовать в качестве базового в собственных реализациях компонентов библиотек карточек. При этом нужно переопределить методы: `GetIcon`, `GetCardIcon`, `GetLibraryDefinition` и `GetCardDefinition`.

Номер версии в сборке библиотеки (`assemblyVersion.FileVersion`),

должен совпадать с номером версии, который указан в XML-описании библиотеки. В противном случае будет появляться предупреждение о несовпадении версий библиотеки на сервере и на клиенте.

В код проекта также необходимо включить ресурсный файл, который будет содержать иконки и XML-описания карточек, входящих в библиотеку, а также XML-описания самой библиотеки карточек. Такой файл можно создать при помощи встроенного редактора Visual Studio. А чтобы избежать дублирования, можно просто вставить в ресурсный файл **ссылки** на файлы из проекта, например следующим образом:

```
<data name="Icon_TestCard" type="System.Resources.ResXFileRef, System.Windows.Forms">
  <value>..\..\CardDefs\Icons\TestCard.ico;System.Drawing.Icon, System.Drawing, Version
=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a</value>
</data>

<data name="Definition_TestCard" type="System.Resources.ResXFileRef,
System.Windows.Forms">
  <value>..\..\CardDefs\TestCard.xml;System.String, mscorlib, Version=2.0.0.0, Culture
=neutral, PublicKeyToken=b77a5c561934e089;utf-8</value>
</data>
```

У ресурсного файла обязательно необходимо установить атрибут `build action` — `embedded resource`

Если при помощи нового инсталляционного пакета или пакета исправлений производится обновление сборки, содержащей компонент библиотеки карточек, может потребоваться перезагрузка приложения.

Если этот момент является критичным, то при разработке собственного решения рекомендуется реализовывать компонент библиотеки карточек как COM-класс, реализующий интерфейс `ICardLibraryInfo` и `ICardLibraryInfo2`.

Пример базового кода компонента библиотеки:

```
using System;
using System.Drawing;
using System.Runtime.InteropServices;
using DocsVision.Platform.WinForms; ①
```

```
namespace DocsVision.Test
{
    [ComVisible(true)]
    [Guid("9FD12D2E-F17F-4CCC-8598-38A52513E970")]
    [ClassInterface(ClassInterfaceType.None)]
    public class TestCardLib : CardLibrary
    {
        public TestCardLib() { }

        public override Icon GetIcon() ②
        {
            return Resources.TestCardLibIcon;
        }

        public override Icon GetCardIcon(Guid cardTypeID) ③
        {
            if (cardTypeID = TestCard.ID)
                return Resources.TestCardIcon;
        }

        public override string GetCardDefinition(Guid cardTypeID) ④
        {
            if (cardTypeID = TestCard.ID)
                return Resources.TestCardDefinition;
        }

        protected override string GetLibraryDefinition() ⑤
     {
            return Resources.TestLibDefinition;
        }
    }
}
```

① Особенность подключения пространства имён класса `CardLibrary`: если реализуется WinForms компонент, то CardLibrary определён в пространстве имён `DocsVision.Platform.WinForms`, если WPF — в пространстве имён `DocsVision.Platform.Wpf`.

② Определение иконки для библиотеки.

③ Определение иконки для карточек, в данном случае только для типа `TestCard`.

④ Возвращает XML-описание карточки, в данном случае только для типа `TestCard`.

⑤ Возвращает XML-описание библиотеки карточек.

Разработка расширений Docsvision

Полноценный процесс персонализации решения не может обойтись без создания дополнений в клиентской или серверной части продукта. Организации может потребоваться внести в интерфейс Windows-клиента изменения, не реализуемые в рамках объектной модели. Например, возможность формирования файла экспорта определённого формата непосредственно из Windows-клиента.

Также может потребоваться выполнять определённые методы в контексте сервера и с его привилегиями. Разработка расширений с использованием объектной модели Docsvision не сложнее процесса разработки карточки, но создание плагинов обладает своими преимуществами.

Методика разработки расширений для Windows-клиента во многом похожа на разработку карточек и описана в разделе "Расширение Windows-клиента".

Типы объектной модели, требующие реализации в процессе разработки серверных расширений, перечислены в разделе "Серверные расширения".

Раздел "Модуль расширения Консоли настройки" освещает создание специфического компонента для программы Консоль настройки Docsvision.

В данном разделе

- "Серверные расширения"
- "Модуль расширения Консоли настройки"

Разработка расширений для Linux

При разработке и регистрации расширений в Docsvision версии 6.1 необходимо учитывать:

- Проекты существующих расширений необходимо перевести на .NET 6, новые расширения должны разрабатываться в .NET 6.
- Расположение разработанных библиотек.
- Сборки расширений должны регистрироваться в конфигурационном файле модуля в формате `.json`.

  1. Существующие проекты расширений необходимо перевести с .NET 4 на .NET 6. Для упрощения задачи можно воспользоваться рекомендациями Microsoft по переносу кода в .NET Framework из .NET.
  2. Рекомендуется получать библиотеки из продукта и подключать их к проекту. При компиляции следует обращать внимание на ошибку CA1416.

Ошибка должна расцениваться как критическая, т.к. данными ошибками обозначены функции, которые не будут работать в Linux, соответственно, их необходимо изменить для поддержки кросс-платформенности.

3. Сборки могут размещаться в любом месте, главное условие — в системе необходимо указать расположение сборки. Например, расширения Web-клиента и Управление процессами загружаются из каталога `/usr/lib/docsvision/common/`.

   Доступны следующие варианты указания пути к сборке

   - Добавить еще один путь можно при помощи переменной окружения `DV_Docsvision__Shared__Paths__DigDes=/path/to/your/custom/extensions/`.
   - Прописать путь в конфигурационном файле `appsettings.json`:

```
{
  "DocsVision": {
    "Platform": {
      "Server": {
        "Extensions": {
          "WebClientServerExtension":
"DocsVision.WebClient.ServerExtension.ServerExtension,
DocsVision.WebClient.ServerExtension, Version=6.0.0.0, Culture=neutral,
PublicKeyToken=7148afe997f90519"
        }
      }
    },
    "Shared": {
      "Paths": {
        "DigDes": "/path/to/your/custom/extensions/"
      }
    }
  }
}
```

   - Указать в отдельном файле в формате `.config`.

   Править основной `appsettings.json` не обязательно, `.json` с регистрацией расширения можно добавить в каталог `/usr/lib/docsvision/platform/config/`. Новый файл можно создать по аналогии с уже находящимися в этом каталоге.

Расширение Windows-клиента

Платформа Docsvision позволяет авторам решений разрабатывать собственные расширения интерфейса Windows-клиента — аналог плагинов, дополняющих стандартную функциональность новыми возможностями, доступными пользователям. Расширения могут быть нескольких типов (описаны ниже) или совмещать в себе реализацию нескольких типов, поэтому конечная реализация расширения будет зависеть от его типа.

Технически расширение Windows-клиента представляет собой обычную карточку системы Docsvision, которая должна разрабатывается по стандартному сценарию.

В схеме карточки расширения должен быть установлен флаг `Реализует расширения интерфейса хоста / Provides UI Extension`. Также рекомендуется установить флаг `Является справочником / Dictionary (only one instance allowed)`, чтобы исключить возможность создания нескольких экземпляров карточки-расширения, что может привести к конфликтам при их вызове.

Т.к. карточка исполняет лишь функции проводника расширения, то, скорее всего, добавлять в неё рабочие секции и поля не предполагается. Если это так, то добавьте в свою схему фиктивные секцию и поле, чтобы схема карточки считалась корректной.

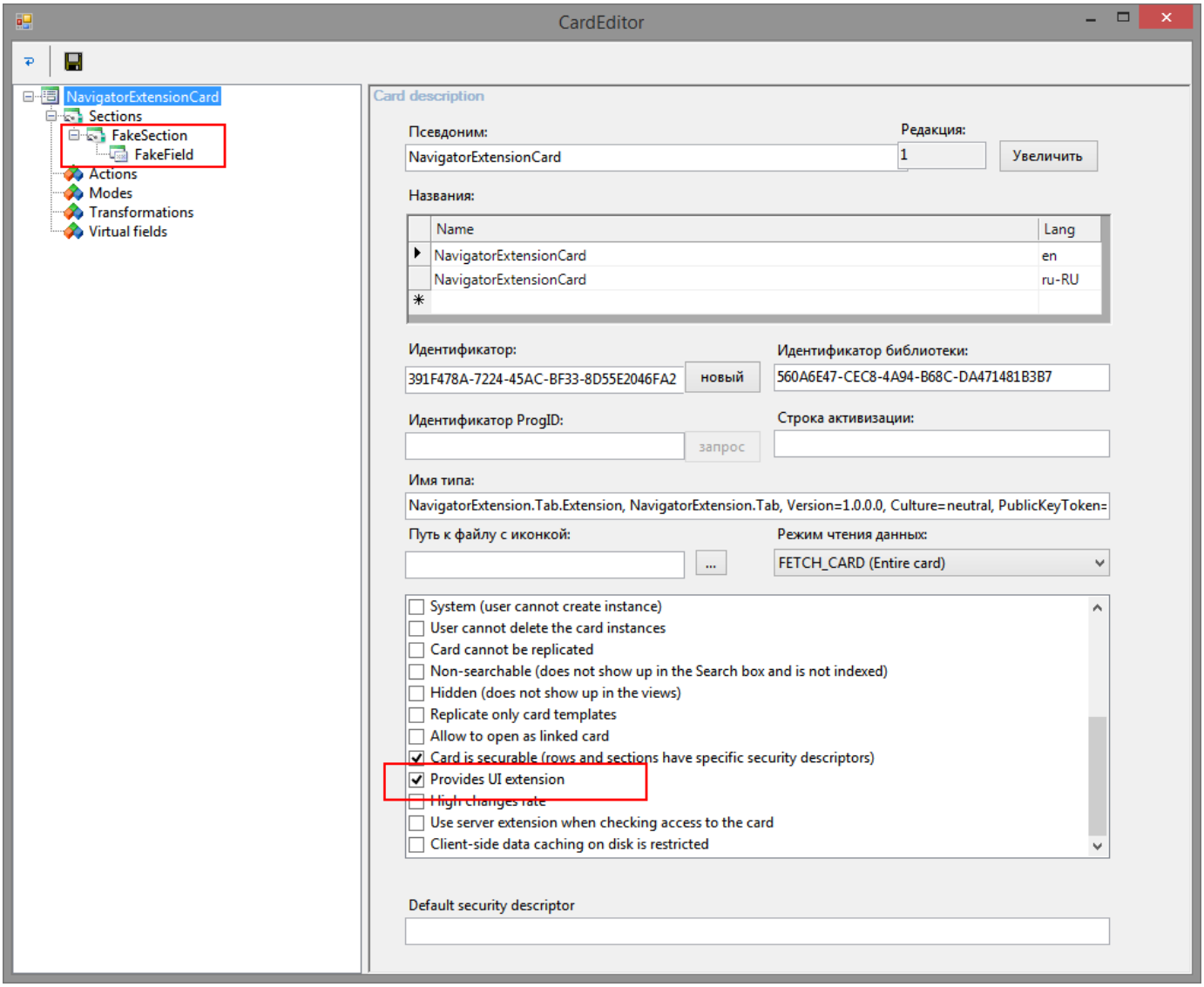


Рисунок 33. Фиктивная секция

Компонент карточки-расширения необходимо наследовать от класса `NavExtension`, в котором помимо стандартных интерфейсов компонента карточки реализованы специальные интерфейсы расширения Windows-клиента, также являющиеся специальными компонентами карточек:

```
public sealed partial class SampleExtension : NavExtension
```

Обязательным для расширения любого типа является переопределение свойства `SupportedTypes`, в котором необходимо указать тип или типы функций, реализующих расширение:

```
protected override NavExtensionTypes SupportedTypes
{
 get
```

```
 {
  return NavExtensionTypes.PropertyPages | NavExtensionTypes.Picker;
 }
}
```

Дальнейшая реализация зависит от типа функциональности, реализуемой расширением.

Расширение виртуальных папок

Тип `NavExtensionTypes.VirtualFolder`.

Позволяет программно изменить поведение виртуальных папок. Например, "на лету" сформировать новый текст поискового запроса для виртуальной папки или отобразить собственный диалог ввода параметров.

Расширение виртуальных папок не требует реализации никаких специальных интерфейсов. Если карточка расширения назначена в качестве фильтра для виртуальной папки (это нужно сделать явно, выбрав её в поле "Тип фильтра" в свойствах папки), то при активации такой папки Windows-клиент вызывает карточку расширения методом `CardHost.SelectFromCard`.

Соответственно, карточка расширения может вернуть результат выбора (настроенный запрос) через `FinishSelection`.

```
protected override void OnCardClosed(EventArgs e)
{
    if (this.DialogResult = DialogResult.OK)
    {
        this.CardFrame.FinishSelection(query.GetXML(true, null), false);
    }
    base.OnCardClosed(e);
}
```

Расширение экспорта

Тип `NavExtensionTypes.Report`.

Этот тип расширения появляется в списке доступных вариантов экспорта данных представления (по умолчанию в Windows-клиенте есть только возможность выгрузки данных представления в Excel). Когда пользователь выбирает соответствующую команду экспорта, управление получит код расширения, которое и будет реализовывать собственно логику экспорта данных

(например, во внешнюю систему или базу данных).

Расширение экспорта требует реализации в коде карточки специального метода, который будет вызван в момент выбора пользователем соответствующей команды экспорта: `CreateReport`

Командное расширение

Тип `NavExtensionTypes.Command`.

Расширение позволяет добавить собственные команды в панель инструментов (тулбар) Windows-клиента или в контекстное меню карточек и папок.

Командное расширение предполагает реализацию следующих дополнительных методов:

- `CreateCommands`.
- `QueryCommandStatus`.
- `InvokeCommand`.

Логика работы Windows-клиента с командами расширения может быть отражена в следующем алгоритме:

1. При первичной загрузке Windows-клиент опрашивает все доступные карточки расширения на предмет наличия команд. Если расширение поддерживает команды, то возвращает их массив (`CreateCommands`).
2. В процессе работы при выделении пользователем конкретных объектов (карточек, папок), но перед тем как отобразить контекстное меню Windows-клиент опрашивает расширение на предмет статуса каждой команды (`QueryCommandStatus`).
3. Наконец, при выборе пользователем конкретной команды, управление переходит в код карточки расширения (`InvokeCommand`).

Пример кода доступен по ссылке.

Обработчик событий

Тип `NavExtensionTypes.Event`.

Расширение позволяет добавить собственную обработку стандартных событий Windows-клиента: например, таких, как запуск или завершение работы. Данный тип расширений может быть полезен в качестве вспомогательного для других типов. Например, если расширение экспорта выгружает данные в какую-то внешнюю систему, то обработчик событий может при запуске Windows-клиента

пытаться заранее установить соединение с этой системой.

В расширение событий должно быть реализовано два специализированных метода:

- `OnStartup`.
- `OnShutdown`.

Пикер

Тип `NavExtensionTypes.Picker`.

Такой тип расширения позволяет разработчику реализовать собственный интерфейс для выбора учетных записей пользователей (это может быть полезно в случае разработки собственного, отличного от стандартного, справочника сотрудников). Расширение такого типа будет вызвано, например, при назначении прав на объекты системы, чтобы пользователь мог оперировать не доменными учетными записями, а более полными сведениями из справочника.

Расширение-пикер предполагает реализацию следующих методов:

- `LookupAccounts`.
- `LookupSids`.
- `LookupNames`.
- `PickAccounts`.

Контроль папки

Тип `NavExtensionTypes.Control`.

Это расширение позволяет динамически контролировать поведение папок, например, запретить отображать подпапки или не подсвечивать количество непрочитанных карточек.

Расширение контроля папки требует реализации единственного метода: `QueryFolderControl`.

Пример кода, в котором для простых папок не показываются дочерние элементы:

```
protected override NavFolderControlFlags QueryFolderControl(NavFolderControlType
folderType, Guid folderId)
{
  if (folderType = NavFolderControlType.Folder)
  {
    return NavFolderControlFlags.DoNotAskChildren;
  }
```

```
    else
  {
    return base.QueryFolderControl(folderType, folderId);
  }
}
```

Страницы свойств

Тип NavExtensionTypes.PropertyPages.

Расширение такого типа может добавлять собственные страницы свойств к свойствам папок и карточек. На этих дополнительных страницах можно расположить произвольную информацию и элементы управления.

Для реализации дополнительных страниц свойств карточка расширения должна реализовывать специальный метод CreatePropertyPages. Метод должен создавать коллекцию страниц свойств типа (NavPropertyPage).

```
protected override IEnumerable<NavPropertyPage> CreatePropertyPages()
{
    Trace.WriteLine("TestExtension.InitializePropertyPages()");
    return new NavPropertyPage[] {
        new NavPropertyPage() {
            PageType = NavPropertyPageTypes.All,
            Name = "TestPropertyPage",
            Clsid = typeof(TestPropertyPage).GUID,
        },
    };
}
```

В свою очередь, элемент управления, реализующий собственно страницу свойств, должен быть унаследован от специального класса — NavPropertyPageControl. И так же как и компоненты карточек, он должен иметь идентификатор COM-интерфейса:

```
[ComVisible(true)]
[Guid("572860E1-E4C6-4120-B3DC-78C0A03F7445")]
[ClassInterface(ClassInterfaceType.None)]
public sealed partial class TestPropertyPage : NavPropertyPageControl
```

Расширение типов карточек

Тип NavExtensionTypes.CardTypes.

Данное расширение предоставляет информацию о пользовательских подтипах (видах) карточек. Это может быть полезно в случае разработки собственного, отличного от стандартного, справочника типов.

Информация о подтипах, которую вернет расширение, будет доступна в контекстном меню создания новой карточки, а также на странице свойств папки с ограничением на типы карточек.

Для создания расширения требуется переопределить два метода: `PopulateCardTypes` и `LookupCardTypes`.

Расширение типов папок

Тип `NavExtensionTypes.FolderTypes`.

Такое расширение предоставляет информацию о пользовательских подтипах папок. Это может быть полезно в случае разработки собственного, отличного от стандартного, справочника типов папок.

Требуется реализовать единственный метод: `PopulateFolderTypes`. Метод должен сформировать и вернуть коллекцию объектов `NavFolderType`, описывающих тип папки. Свойства этого объекта практически полностью идентичны соответствующим свойствам папки, за исключением следующих:

- `FolderCardLocation` — идентификатор карточки папок.
- `FolderTypes` — дочерние типы папок.

Расширение дерева папок Windows-клиента

Тип `NavExtensionTypes.FolderTree`.

Данное расширение предназначено для добавления ссылок на папки в дерево папок Windows-клиента или в список избранных папок без участия папок-делегатов.

Необходимо дополнительно реализовать интерфейс `INavFolderTreeExtension`. Метод `GetTreeExtensionFolders` должен возвращать специально подготовленный массив папок, предназначенных для размещения в интерфейсе Windows-клиента.

Серверные расширения

Данный вид расширения может быть востребован при необходимости выполнения каких-то действий в контексте основного процесса сервера (с его

привилегиями), которые не могут быть выполнены с использованием стандартных методов интерфейса сервера. Механизм расширений позволяет интегрировать в контекст сервера дополнительные методы, а также обеспечивает возможность их вызова с клиента через общую объектную модель (ObjectManager).

Серверное расширение представляет собой .NET сборку (Class Library), в которой должен быть реализован как минимум один класс, унаследованный от `DocsVision.Platform.StorageServer.Extensibility.StorageServerExtension`.

Этот базовый класс уже содержит в себе всю необходимую инфраструктуру для реализации модуля расширения. Разработчику остается только определить сигнатуру методов, реализуемых данным модулем расширения. Такие методы необходимо пометить специальным атрибутом: `ExtensionMethod`.

Атрибут `ExtensionMethod` имеет дополнительное свойство `AllowedUserRoles`, которое можно использовать для ограничения возможности вызова данного метода расширения в зависимости от роли пользователя (членства в соответствующей группе безопасности). Пример использования свойства (метод будет доступен только для членов группы **DocsVision Administrators**): `ExtensionMethod(UserRoles.Administrator)`.

Пример кода реализации серверного расширения, где метод GetBaseName возвращает имя рабочей базы данных:

```
using System;
using System.IO;
using DocsVision.Platform.StorageServer.Methods;
using DocsVision.Platform.StorageServer.Extensibility;

namespace DocsVision.Samples
{
 public sealed class SampleExtension : StorageServerExtension
 {
  public SampleExtension(){ }

  [ExtensionMethod]
  public string GetBaseName()
  {
   return this.Request.Base.Name;
  }
 }
```

```
}
```

В качестве возвращаемого значения метод может вернуть либо скаляр, либо набор данных (коллекцию). Во втором случае лучше воспользоваться специальным типом возвращаемого значения — `DocsVision.Platform.StorageServer.CursorInfo`. Этот объект позволяет создать серверный курсор, из которого потом можно читать данные на клиенте.

Объект `CursorInfo` можно создать двумя способами:

- с помощью метода `GetCursorBuilder`, который возвращает объект `CursorBuilder` и позволяет программно заполнять серверный курсор.
- с помощью метода `ExecuteCursorCommand`, который выполняет SQL-команду, сохраняет её результаты в курсоре, и возвращает готовый `CursorInfo`.

Пример реализации серверного метода, возвращающего набор значений:

```
[ExtensionMethod]
public CursorInfo GetSectionsSize()
{
 using (DatabaseCommand Command = base.DbRequest.DataLayer.Connection.CreateCommand
("dvsys_help_show_section_size", CommandType.StoredProcedure))
 {
  return base.ExecuteCursorCommand(Command);
 }
}
```

Готовый модуль расширения необходимо зарегистрировать в качестве расширения на сервере. Для этого нужно создать файл формата `.json` с именем `extension-уникальное-название` в каталоге `/usr/lib/docsvision/platform/config/extension-уникальное-название.json`, например:

```
{
  "DocsVision": {
    "Platform": {
      "Server": {
        "Extensions": {
          "YourExtension"="DocsVision.Your.ServerExtension.Extension,
DocsVision.Your.ServerExtension, Version=6.0.0.0, Culture=neutral,
PublicKeyToken=7e990517f9148af9" ①
        }
      }
```

```
    }
  }
}
```

① Укажите псевдоним расширения и полное название типа.

Для вызова методов модуля расширения на клиенте необходимо воспользоваться специальным объектом — менеджером расширений (`ExtensionManager`). Менеджер расширений содержит метод для вызова у указанного расширения определённого метода: `GetExtensionMethod`.

Если такого расширения или метода на сервере не зарегистрировано, функция вернет ошибку. В противном случае будет возвращен специальный объект — `ExtensionMethod`, с помощью которого уже можно выполнять непосредственные вызовы метода серверного расширения.

Параметры вызова серверного метода задаются через коллекцию Parameters, в которую их необходимо явно добавить в виде объектов `ExtensionMethodParameter`. Этот объект очень прост — он описывает название, тип и значение параметра через одноименные свойства.

Пример кода вызова метода серверного расширения:

```
ExtensionMethod method = session.ExtensionManager.GetExtensionMethod
("BackOfficeExtension", "GetFileContent"); ①

method.Parameters.AddNew("FilePath", ParameterValueType.String, ".\\Test.txt"); ②

string content = method.Execute().ToString(); ③
```

① Получаем метод расширения. Первым параметром указывается имя расширения, вторым — название метода.

② Добавляем параметр вызываемого метода:

- Первый аргумент — название параметра метода.
- Второй аргумент-- тип параметра.
- Третий аргумент — значение параметра.

③ Вызов метода

Модуль расширения Консоли настройки

В ряде случаев задача установки серверной части решения может быть

усложнена необходимостью выполнения каких-то дополнительных действий, которые сложно (или невозможно) выполнить в рамках ограниченных возможностей инсталляционной программы.

Такими действиями, в частности, могут являться:

- Работа с базой данных.
- Установка или конфигурирование NT-сервисов.
- Взаимодействие с другими компонентами или решениями.
- Изменение параметров работы решения после окончания установки.
- Прочие действия.

Чтобы упростить решение этих задач, предоставляется возможность разработки специального программного компонента — модуля расширения (Snap-In) для программы Консоль настройки Docsvision.

Консоль настройки Docsvision — универсальный инструмент для конфигурирования и администрирования сервера Docsvision. Подробнее его использование рассмотрено в документации по /dv6/platform/6.1/console/settings-console/[администрированию модуля Платформа].

Механизм модулей расширения (Snap-Ins) является способом динамического расширения функциональности Консоли настройки новыми возможностями, которые могут появляться при установке в системе дополнительных модулей или решений. Аналогами подобного механизма могут выступать оснастки консоли администрирования (MMC) в Microsoft Windows или механизм плагинов (Plug-ins) в других программных средах.

В частности, средства конфигурирования стандартных решений "Управление процессами" и "Процессы WWF" также реализованы в виде модулей расширения для Консоли настройки — их можно рассматривать в качестве примера реализации.

Консоль настройки может работать в двух режимах:

- **Режим конфигурирования** (Мастера) — этот режим активируется при первичной установке или удалении конкретного решения (или сервера Docsvision в целом).
- **Режим администрирования** — этот режим активируется на рабочем сервере

для изменения конкретных параметров работы сервера и/или конкретных решений.

Соответственно, каждый Snap-In для Консоли настройки также имеет возможность использовать один или оба режима. Например, можно реализовать Snap-In для собственного решения, который будет активироваться как при его установке (для выполнения каких-то сложных действий по регистрации компонент), так и позволит администратору системы менять какие-то параметры в процессе его работы.

В качестве Snap-In Консоли настройки может выступать любой программный компонент, реализующий ряд специальных интерфейсов (по аналогии с компонентами карточек Docsvision).

В коде компонента Snap-In могут быть реализованы следующие интерфейсы (все интерфейсы объявлены в сборке `DocsVision.Tools.ServerConsole.Interfaces.dll`, которую необходимо подключить к проекту):

`DocsVision.Tools.ServerConsole.ISnapIn` — обязательный интерфейс для любого модуля расширения.

Предполагает реализацию следующих свойств и методов:

- `Name` — свойство, которое должно возвращать отображаемое имя модуля расширения.
- `Initialize(IEnvironment environment)` — метод, который вызывается при первичной инициализации модуля расширения. На вход метода передается объект, описывающий контекст исполнения (Консоль Настройки).
- `QueryInterface(SnapInInterfacesEnum itfType)` — метод, который вызывается при запросе конкретных функций модуля расширения.

  Модуль расширения может выполнять следующие функции:

  - `SnapInInterfacesEnum.CONFIGURATOR` — первичная инсталляция решения.
  - `SnapInInterfacesEnum.WIZARD_CONTROL` — инсталляция в режиме мастера.
  - `SnapInInterfacesEnum.CONSOLE_CONTROL` — режим администрирования (настройки).
  - `SnapInInterfacesEnum.TOOLS` — дополнительные инструменты.
  - `SnapInInterfacesEnum.UNINSTALL_SNAP_IN` — режим деинсталляции решения.

- `SnapInInterfacesEnum.DB_INFORMATION` — режим обновления базы данных.

  При запросе конкретной функции модуль расширения может либо вернуть ссылку на объект, реализующий соответствующий интерфейс (если он выполняет эту функцию), либо вернуть `null` (если он не выполняет эту функцию).

`DocsVision.Tools.ServerConsole.IDBInformation` — данный интерфейс является обязательным для модулей расширения, которые выполняют функции конфигурации или удаления решения. Методы данного интерфейса предназначены для загрузки каких-либо значений в базу данных (это необходимо, например, при установке новых библиотек карточек, или шлюзов Workflow).

Обратите внимание на то, что для работы (например, обновление базы данных) с собственной библиотекой карточек из программы Консоль настройки Docsvision, необходимо разработать и загрузить в Консоль настройки Docsvision расширение Windows-клиента, которое реализует интерфейс `IDBInformation`. Данное расширение должно возвращать схему разработанной библиотеки карточек.

Интерфейс предполагает реализацию следующих свойств и методов:

- `GetScript(ScriptTypeEnum type)` — Консоль настройки Docsvision может вызвать данный метод при выполнении каких-либо действий с базой данных.

  В зависимости от типа действия, модуль расширения должен вернуть одно из следующих значений:

  - `INSTALL_TABLES` — путь к файлу SQL-скрипта создания таблиц (для библиотек карточек).
  - `INSTALL_ACTIONS` — путь к файлу SQL-скрипта создания хранимых процедур (для библиотек карточек).
  - `INSTALL_FT_CATALOGS` — путь к файлу SQL-скрипта создания полнотекстовых каталогов.
  - `UNINSTALL_FT_CATALOGS` — путь к файлу SQL-скрипта удаления полнотекстовых каталогов.
  - `UNREGISTER_CARD_LIBRARY` — путь к файлу SQL-скрипта удаления библиотеки карточек.
  - `UNINSTALL_ALL` — путь к файлу SQL-скрипта удаления других данных решения.

- `GetCardPackage()` — возвращает путь к файлу — пакету инсталлируемых XML-описаний карточек. Это может потребоваться в случае, если для корректной работы решения необходимо наличие в базе данных каких-то предопределённых значений — заполненных справочников, шаблонов карточек и прочих.

  Файл с описанием пакета имеет следующий формат:

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <CardPackage>
   <Card Path="xmlpath" ID="cardid" Replace="1"/> ①
   ...
  </CardPackage>
  ```

  ① `xmlpath` — Путь к файлу `.xml`.

  `cardid` — ID карточки.

  `Replace="1"` — если карточка должна заменить существующую, `Replace="0"` — если дополнить.

- `ContainsCardLib` — признак, содержит ли данное решение библиотеку карточек (нужно ли запрашивать SQL-скрипты для создания таблиц и хранимых процедур).

`DocsVision.Tools.ServerConsole.IConfigurator` — необязательный интерфейс. Может быть реализован в модуле расширения, выполняющем функции первичной конфигурации решения (`SnapInInterfacesEnum.CONFIGURATOR`).

Интерфейс предполагает реализацию следующего метода:

- `Execute()` — единственный метод, который должен выполнять все задачи по корректной инсталляции решения. Возвращаемое значение показывает успешность (`true`) или неудачу (`false`) установки.

`DocsVision.Tools.ServerConsole.IUninstallSnapIn` — необязательный интерфейс. Может быть реализован в модуле расширения, выполняющем функции удаления решения (`SnapInInterfacesEnum.UNINSTALL_SNAP_IN`).

Интерфейс предполагает реализацию следующего метода:

- `Uninstall(Boolean)` — единственный метод, который должен выполнять все задачи по корректному удалению решения. Входящий параметр указывает на необходимость удалить (`true`) или сохранить (`false`) настройки решения.

`DocsVision.Tools.ServerConsole.ITools` — необязательный интерфейс. Может быть реализован в модуле расширения, дополняющим Консоль настройки Docsvision специфическими Инструментами (`SnapInInterfacesEnum.TOOLS`).

Интерфейс предполагает реализацию следующего свойства:

- `Controls` — возвращает массив элементов управления для конкретных инструментов. Элемент управления для реализации каждого инструмента должен реализовывать интерфейс `IControl (IControl2)`.

`DocsVision.Tools.ServerConsole.IConsoleControl` — необязательный интерфейс. Может быть реализован в элементе управления, который будет отображаться пользователю при установке в режиме мастера (`SnapInInterfacesEnum.WIZARD_CONTROL`) или в режиме администрирования (`SnapInInterfacesEnum.CONSOLE_CONTROL`).

Интерфейс предполагает реализацию следующих свойств и методов:

- `ControlChanged` — событие, которое элемент управления должен инициировать при изменении данных.
- `Caption` — свойство, возвращающее отображаемое имя элемента управления.
- `Instance` — свойство, возвращающее ссылку на элемент управления `WinForms`.
- `Changed` — признак изменения настроек решения.
- `Valid` — признак корректности указанных настроек решения.
- `Initialize()` — метод первичной инициализации элемента управления.
- `Execute()` — метод, вызываемый при завершении конфигурирования.

Пример кода модуля расширения с реализацией этих интерфейсов:

```
namespace DocsVision.Sample.SnapIn
{
 public class SnapIn : ISnapIn, IConfigurator, IUninstallSnapIn
 {
  private IEnvironment _environment;

  public SnapIn() { } ①

  public string Name
  {
   get { return "My Snap-In"; }
  }

  public string LibraryID
```

```
  {
   get { return "00000000-0000-0000-0000-000000000000"; }
  }

  public void Initialize(IEnvironment environment)
  {
   _environment = environment;
  }

  public object QueryInterface(SnapInInterfacesEnum itfType)
  {
   object result = null;
   switch (itfType)
   {
    case SnapInInterfacesEnum.CONFIGURATOR:
    case SnapInInterfacesEnum.UNINSTALL_SNAP_IN:
     result = this;
     break;
   }
   return result;
  }

  public bool Execute() ②
  {

   return true; ③
  }

  public void Uninstall(bool removeSettings) ④
  {
⑤
  }
 }
}
```

① Реализация интерфейса `ISnapIn`.

② Реализация интерфейса `IConfigurator`.

③ Регистрация компонент решения.

④ Реализация интерфейса `IUninstallSnapIn`.

⑤ Разрегистрация компонент решения.

В коде модуля расширения, можно обращаться к различным вспомогательным сервисам, которые предоставляет Консоль настройки Docsvision для упрощения решения типовых задач. Чтобы обратиться к сервисам, воспользоваться ссылкой

на объект контекста (IEnvironment), ссылка передается модулю расширения при инициализации. Этот объект имеет единственный метод: QueryService(EnvironmentServiceEnum service) — возвращающий ссылку на конкретный вспомогательный сервис, запрошенный в параметре.

Доступны следующие сервисы:

- EnvironmentServiceEnum.LOg — возвращает ссылку на сервис ILog, позволяющий записывать сообщения в общий журнал работы Консоли настройки.
- EnvironmentServiceEnum.COMMON_SETTINGS — возвращает ссылку на сервис ICommonSettings2, позволяющий прочитать и/или изменить основные настройки сервера Docsvision.
- EnvironmentServiceEnum.WORKER_PROCESS — возвращает ссылку на сервис IWorkerProcess, позволяющий модулю расширения корректно функционировать при выполнении длительных операций (например, отображать индикатор прогресса выполнения). Для реализации таких операций, соответствующие объекты модуля расширения должны реализовывать интерфейс ILengthyOperation3.
- EnvironmentServiceEnum.DB_INSTALLER — возвращает ссылку на сервис IDbInstaller, позволяющий выполнять операции с базой данных (например исполнить сценарий SQL из строки или из файла).
- EnvironmentServiceEnum.CARD_LIB_CONFIGURATOR — возвращает ссылку на сервис ICardLibConfigurator2, позволяющий корректно установить или удалить описание библиотеки карточек.
- EnvironmentServiceEnum.CARD_IMPORTER — возвращает ссылку на сервис ICardImporter, позволяющий загрузить в базу данных предопределённые значения (экспортированные в формат XML).
- EnvironmentServiceEnum.MANAGEMENT — возвращает ссылку на сервис IManagement, позволяющий управлять работой других сервисов и решений.

Пример использования вспомогательного сервиса Консоли настройки в модуле расширения для записи сообщения в журнал:

```
ILog log = (ILog)_environment.QueryService(EnvironmentServiceEnum.LOG);
log.WriteMessage("Конфигурирование решения успешно завершено");
```

Разработанный модуль расширения необходимо зарегистрировать на сервере в процессе инсталляции серверной части решения. Для этого программа

инсталляции должна создать в реестре ключ в ветке `HKEY_LOCAL_MACHINE\Software\DocsVision\ВЕРСИЯ\Console\Snap-Ins`.

Необходимо создать ключ с именем своего модуля расширения, в котором создать два строковых значения:

- `Path` — полный путь к сборке, в которой реализован модуль расширения.
- `TypeName` — имя основного класса, реализующего интерфейс `ISnapIn` в разработанном модуле расширения (например, `DocsVision.Sample.SnapIn.SnapIn`).

Чтобы запустить Консоль настройки Docsvision в режиме конфигурирования нового модуля расширения, необходимо запустить её исполняемый файл с ключами: `ServerConsole.exe /c /n ИМЯ_РЕШЕНИЯ`. Вызов этой команды можно сделать последним шагом программы инсталляции серверной части решения.

Разработка элемента управления для Конструктора разметок

Docsvision предлагает разработчикам возможность создавать собственные элементы управления и использовать их при проектировании разметки карточки в *Конструкторе разметок*.

Элемент управления, совместимый с *Конструктором разметок* — это сборка, включающая в себя три обязательных компонента:

- Собственно элемент управления, который предоставляет графический интерфейс и необходимую бизнес-логику.
- Класс-контейнер, который обеспечивает совместимость с *Конструктором разметок*, предоставляет необходимые для *Конструктора разметок* свойства, обеспечивает сохранение настроек элемента управления, а также передачу их значений в элемент управления.
- Контейнер расширения, содержащий информацию об элементах управления, реализованных в сборке (в одной сборке может быть определено несколько элементов управления).

В процессе загрузки разметки (в карточке, либо в *Конструкторе разметок*) механизмы *Конструктора разметок* обращаются с зарегистрированными сборкам в поисках класса контейнера расширения. Если в сборке обнаружен класс данного типа — вызывается его метод `GetControlExtensions`, который предоставляет данные о типах реализованных элементов управления. Каждая запись, возвращаемая методом `GetControlExtensions`, обязательно содержит тип класса-контейнера, а также может содержать типы вспомогательных сущностей,

которые реализуют механизмы настройки элемента управления.

После создания компонента карточки инициализируются элементы управления с загрузкой в них данных и настроек.

Далее рассматривается создание элемента управления на примере разработки простого списка, в который должны быть загружены подразделения из Справочника сотрудников.

Полный код проекта можно скачать по /dv6/programmer/dv6/_attachments/controlForStaff.zip[ссылке].

1. Создать новый проект типа `Windows Forms Class Library`.
2. В проект добавить элемент `XtraUserControl`, в котором будет реализован пользовательский интерфейс элемента управления.

Вместо `XtraUserControl` можно использоваться стандартный `UserControl`.

3. Реализовать обязательный интерфейс `IPropertyControl`, в котором определены ключевые функциональные возможности элемента управления: получение контекста объектов, установка и возвращение значения элемента управления, а также его базовые настройки, и др.

```
public partial class RealPropertyControl : XtraUserControl, IPropertyControl
{
 private IStaffService staffService; ①

 public RealPropertyControl()
 {
  this.InitializeComponent();
 }

②

 public bool Hierarchy ③
 {
  get;
  set;
 }

 public void LoadData() ④
 {
```

```
  if (ObjectContext = null) return; ⑤

  staffService = ObjectContext.GetService<IStaffService>();
  ComboBoxItemCollection coll = staffBox.Properties.Items;

  coll.BeginUpdate();
  foreach (var item in staffService.GetUnits(null, true, Hierarchy))
  {
   coll.Add(new StaffBoxItem(ObjectContext.GetObjectRef(item).Id, item.Name));
  }
  coll.EndUpdate();
 }

 private void staffBox_SelectedIndexChanged(object sender, EventArgs e) ⑥
 {
  if (staffBox.SelectedIndex = -1) ControlValue = null;
   else ControlValue = (staffBox.SelectedItem as StaffBoxItem).Id;
  ControlValueChanged(this, e);
 }
}
```

① Сервис для работы со Справочником сотрудников.

② Реализация интерфейса `IPropertyControl`.

③ Отображать в списке все подразделения или только подразделения верхнего уровня.

④ Загрузка данных из Справочника сотрудников в элемент управления.

⑤ Если контекст объектов не был передан, то загрузить данные не получится.

⑥ Выбор значения в элементе управления.
Приведенный выше код представляет собой часть содержимого класса `RealPropertyControl`. Метод `LoadData`, загружающий данные из *Справочника сотрудников*, вызывается только после присвоения значения контексту объектов (свойство `ObjectContext`), таким образом можно быть уверенным, что при загрузке данных будет доступное соединение с сервером Docsvision.

Загрузка данных может быть выполнена иным образом, к примеру при обращении пользователя или после изменения видимости элемента управления. Последний способ сработает, если элемент управления расположен на вкладке, не отображаемой при открытии карточки.

При загрузке данных учитывается значение свойства `Hierarchy`, которое может быть определено в настройках элемента управления в *Конструкторе*

разметок. Данное свойство получает значение из класса-контейнера (будет реализован далее).

4. Реализовать графический интерфейс пользователя для элемента управления.
5. Создаем класс, реализующий контейнер для создаваемого элемента управления, который делает данный компонент доступным для *Конструктора разметок*.

```
public class RealLayoutItem : FixedLayoutControlItem<RealPropertyControl>
{
 private bool hierarchy;

 public override string ItemTypeName ①
 {
  get
  {
   return "Собственный элемент управления";
  }
 }

 public override System.Drawing.Image CustomizationImage ②
 {
  get
  {
   return SampleControl.Properties.Resources.ButtonIcon.ToBitmap();
  }
 }

 public override LayoutsPropertyType PropertyType ③
 {
  get { return LayoutsPropertyType.DepartmentReference; } ④
 }

 public override FieldType[] GetSupportedFieldTypes() ⑤
 {
  return new FieldType[]
  {
   FieldType.RefId ⑥
  };
 }

 public override Control Control ⑦
 {
  get
  {
```

```
    return base.Control;
   }
   set
   {
    base.Control = value;
    if (value != null)
    {
     this.PropertyControl.Hierarchy = hierarchy;
    }
   }
  }

  [XtraSerializableProperty] ⑧
  public bool Hierarchy
  {
   get
   {
    if (base.PropertyControl != null)
     return base.PropertyControl.Hierarchy;
    return hierarchy;
   }
   set
   {
    if (this.PropertyControl != null)
     this.PropertyControl.Hierarchy = value;
    hierarchy = value;
   }
  }
 }
```

① Возвращает название элемента управления, отображаемое в Конструкторе разметок.

② Возвращает иконку для элемента управления, отображаемую в Конструкторе разметок.

③ Возвращает тип данные элемента управления, который используется при преобразовании для элемента управления его значения по умолчанию.

④ В данном случае — ссылка на подразделения.

⑤ Возвращает список типов полей, с которыми работает элемент управления.

⑥ В данном случае — ссылочное поле.

⑦ При установке элемента управления передаем настройки в него.

⑧ Обеспечение передачи значения свойства в класс элемента управления. Класс-контейнер наследуется от типа `FixedLayoutControlItem<T>`, где `T` — тип

контейнера элемента управления (был разработан ранее), и в самом простом случае должен содержать только переопределение свойства `PropertyType`. Свойство `PropertyType` должно возвращать тип данных элемента управления, что требуется для присвоения значения по умолчанию при создании новой карточки, в которой используется элемент управления.

Если элемент управления является настраиваемым, т.е. имеет дополнительные настройки в *Конструкторе разметок*, в приведенном классе должен быть реализован механизм передачи значений настроек в элемент управления, а также хранения присвоенных настройкам значений, через сериализации.

Для этого в класс добавляются свойства, аналогичные реализуемым настройкам с соответствующими типами. Свойства должны быть отмечены атрибутом `XtraSerializableProperty`. Для передачи значения настройки в элемент управления, переопределяется свойство `Control`, в котором присваивается значение свойству.

Помимо указанных функций, в класс-контейнере можно указать название элемента управления (иначе будет использовано название класса), отображаемое в *Конструкторе разметок*, а также его иконку. Помимо этого переопределить метод `GetSupportedFieldTypes`, который предоставляет список типов полей. С этими полями может работать элемент управления.

6. Реализуем класс-обертку, предоставляющий *Конструктору разметок* дополнительную информацию о дополнительных настройках элемента управления. В данном примере реализует единственное дополнительное свойство, определяющее логику загрузки подразделений из *Справочника сотрудников*. Класс должен наследовать от типа `SpecialPropertyWrapper<T>`.

   В параметре типа указывается класс-контейнер, реализованный ранее:

```
public class RealWrapper : SpecialPropertyWrapper<RealLayoutItem>
{
 [Category("Дополнительные настройки"), DisplayName("Все подразделения"), Description
("Выводить все подразделения или только первый уровень")]
 [TypeConverter(typeof(BooleanTypeConverter))]
 public bool Hierarchy
 {
  get {
   return this.Item.Hierarchy;
  }
```

```
  set
  {
   this.Item.Hierarchy = value;
  }
 }
}
```

Свойство помечается атрибутами, определяющим его название и категорию, в которой оно размещается:

- `Category` — категория свойств, в которой размещается собственная настройка.
- `DisplayName` — название свойства, отображаемое в Конструкторе разметок.
- `Description` — дополнительное описание.

  Также здесь установлен атрибут конвертера (`TypeConverter`), который формирует из значения свойства текстовое обозначение, отображаемое в *Конструкторе разметок*:

```
internal sealed class BooleanTypeConverter : BooleanConverter
{
 public override object ConvertTo(ITypeDescriptorContext context, CultureInfo
culture, object value, Type destType)
 {
  return (bool)value ? "Да" : "Нет";
 }

 public override object ConvertFrom(ITypeDescriptorContext context, CultureInfo
culture, object value)
 {
  return string.Compare((string)value, "Да", StringComparison.OrdinalIgnoreCase) =
0;
 }
}
```

7. Создать обязательный класс, унаследованный от `ControlExtensionInfoPackage`, который возвращает список всех элементов управления, реализованных в сборке:

```
public sealed class ExtensionPackage : ControlExtensionInfoPackage
{
 public override ControlExtensionInfo[] GetControlExtensions() ①
 {
```

```
    return new ControlExtensionInfo[]
    {
     new ControlExtensionInfo(typeof(RealLayoutItem), typeof(RealWrapper))
    };
   }
  }
```

① Переопределяем единственный метод, возвращающий список элементов управления.

Конструктор `ControlExtensionInfo` принимает тип контейнера элемента управления, а также может принимать, как в приведенном коде, тип обертки для свойств и тип формы, реализующей страницу настроек.

8. После получения готовой сборки, её необходимо зарегистрировать на всех компьютерах в ветке реестра:

   - `HKEY_CURRENT_USER\Software\DocsVision\BackOffice\Client\PropertyControls` — для текущего пользователя.
   - `HKEY_LOCAL_MACHINE\Software\DocsVision\BackOffice\Client\PropertyControls` — для всех пользователей.

   В ветку требуется добавить строковый параметр, значение которого должно содержать полный путь к сборке, либо полное название класса, если сборка зарегистрирована в GAC.

   Ветка реестра может отличаться от приведенной при отличной разрядности операционной системы.

В случае успешной реализации и регистрации элемента управления в список элементов управления *Конструктора разметок* будет добавлен реализованный компонент. Чтобы компонент отобразился в конструкторе разметок, необходимо перезапустить Windows-клиент.

Созданный компонент можно использовать для создания интерфейса карточки:

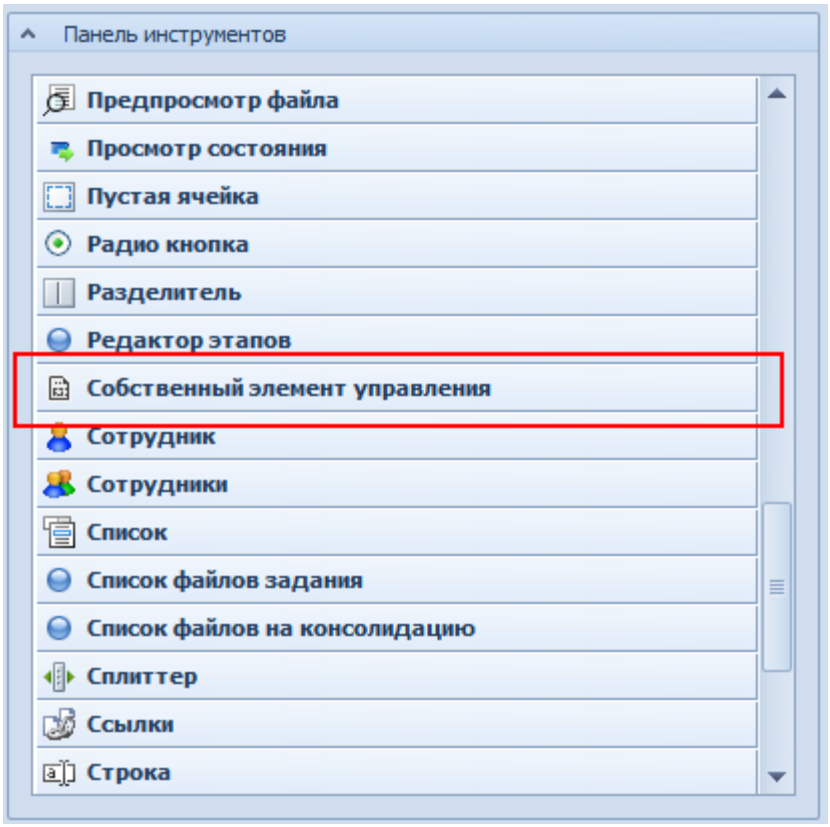


Рисунок 34. Собственный элемент управления в общем списке

Указанный элемент управления имеет дополнительную настройку, определяющую вариант загрузки списка подразделений из *Справочника сотрудников*.

Настройка доступна из свойства элемента управления:

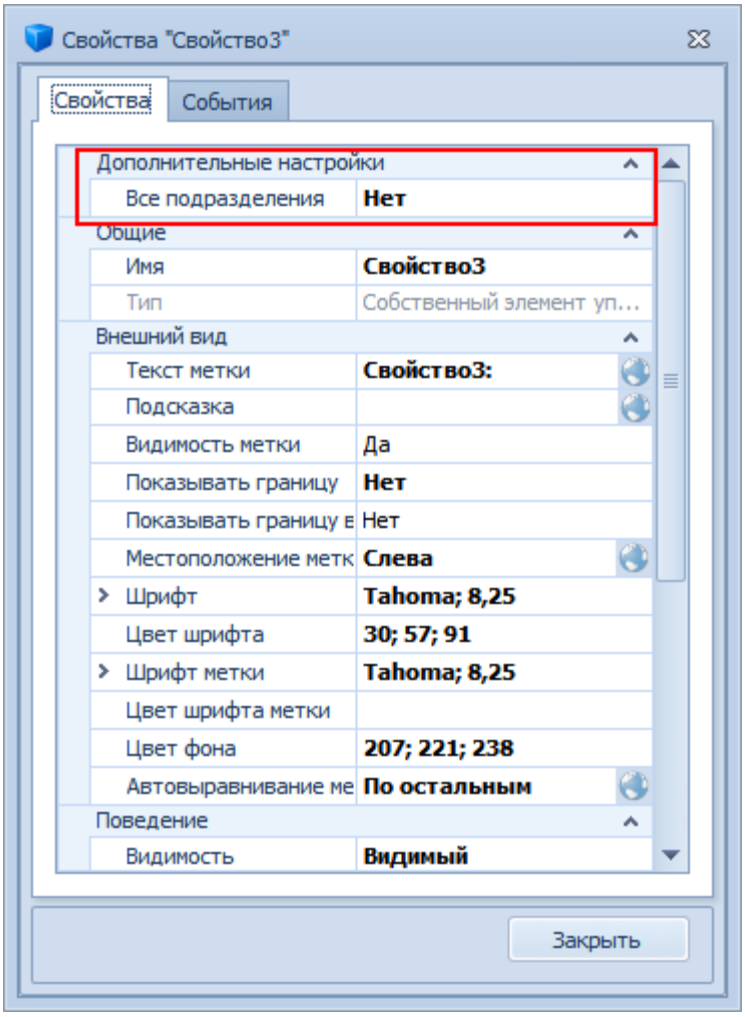


Рисунок 35. Свойства элемента управления

Распространение решения

После завершения разработки решения необходимо позаботиться о его распространении и развертывании на платформе.

Установка готового решения включает в себя два этапа:

- Обновление серверной части платформы.
- Инсталляция на клиентские рабочие места.

Практический пример рассмотрен в Сборнике примеров (подразделе Пример разработки решения).

В данном разделе

- Инсталляция серверной части решения
- Инсталляция клиентской части решения

Инсталляция серверной части решения

Если решение включает в себя какие-либо карточки (новые или модификацию имеющихся), то первоочередным шагом при его распространении является загрузка описаний этих карточек в базу данных на выбранном рабочем сервере.

Если решение предполагается установить на один конкретный сервер (разовая операция), то это действие можно выполнить вручную при помощи программы "CardManager" из Resource Kit, или запуском SQL-скрипта, предварительно сгенерированного "CardManager" для этой библиотеки карточек (команда *Сгенерировать сценарий SQL для библиотеки карточек*). Оба способа приведут к одинаковому результату — загрузке описаний карточек решения в конкретную базу данных.

При каждом обновлении версии платформы на сервере все дополнительные решения необходимо заново загружать в базу данных. В противном случае карточки этих решений будут считаться устаревшими и станут недоступны для использования (данные не будут потеряны, просто карточки решения не будут видны в Windows-клиенте).

Если же решение планируется тиражировать (устанавливать более чем на одной площадке), то выполнять обновление базы данных каждый раз вручную будет нецелесообразно. В этом случае рекомендуется создание отдельной инсталляционной программы для серверной части решения, с помощью которой обновление базы данных будет выполняться в автоматическом режиме.

Такая инсталляционная программа может быть создана при помощи любых специализированных средств (Install Shield, Visual Studio 2003 или 2005, WISE и т.п.), предназначенных для создания инсталляций. В состав этой инсталляционной программы можно будет включить специальный шаг, который будет исполнять SQL-скрипт по загрузке карточек решения в базу данных (выбор конкретной базы данных можно реализовать в диалоге в рамках этой же инсталляционной программы или прочитать имя текущей рабочей базы данных из реестра).

Содержание данного шага зависит от конкретной среды создания инсталляционных программ. InstallShield, например, предлагает встроенный инструмент для исполнения SQL-сценариев. В других средах может понадобиться написать код для исполнения SQL самостоятельно и т.д..

Кроме этого, в число задач серверной установки входит размещение

инсталляционного пакета клиентской части решения (о создании этого пакета рассказывается ниже) в определённой папке на сервере, откуда этот пакет может быть загружен всеми клиентами.

Пакет достаточно просто скопировать в заданную папку, которая обычно расположена по пути: `C:\ProgramFiles\DocsVision\Platform\ВЕРСИЯ_ПЛАТФОРМЫ\Server\Site\Setup\ИМЯ_РЕШЕНИЯ`. Тогда этот инсталляционный пакет будет доступен с клиентских машин по URL: `http://ИМЯ_СЕРВЕРА/ВИРТУАЛЬНАЯ_ПАПКА/Setup/ИМЯ_РЕШЕНИЯ/НАЗВАНИЕ_ПАКЕТА`. Например: `http://localhost/DocsVision/Setup/TakeOfficeClient/TakeOfficeClient.msi`

Возможны также какие-то дополнительные действия, которые должны быть выполнены в процессе установки серверной части решения — это может быть регистрация и запуск каких-то специализированных сервисов, установка прикладных бизнес-процессов. В этом случае необходимые файлы этих дополнительных компонент решения рекомендуется размещать по пути: `C:\ProgramFiles\DocsVision\ИМЯ_РЕШЕНИЯ\ВЕРСИЯ\Server\`.

А необходимые для работы решения дополнительные настройки рекомендуется сохранять в реестре по пути: `HKEY_LOCAL_MACHINE\DocsVision\ИМЯ_РЕШЕНИЯ\ВЕРСИЯ\`.

Инсталляция клиентской части решения

Для корректной работы разработанного решения на клиентских компьютерах там должны быть установлены и зарегистрированы компоненты карточек, библиотеки карточек, а также всех вспомогательных библиотек, использованных при разработке.

Платформа Docsvision предусматривает их автоматическую установку при первом обращении к серверу Docsvision при помощи специализированной программы инсталляции в формате Windows Installer (MSI), которую также должен создать разработчик решения.

Создать инсталляционный MSI-файл можно при помощи специализированных приложений, предназначенных для создания программ инсталляции. Например, это может InstallShield (Windows Installer Edition), либо можно воспользоваться средствами подготовки инсталляций в составе Microsoft Visual Studio (2003 или 2005). В процессе создания MSI-файла в него нужно включить компонент библиотеки и компонент(ы) карточек. Для всех COM-компонентов следует указать на необходимость их регистрации (extract-at-build или self-register).

В процессе создания инсталляционного пакета необходимо указать следующие его параметры:

- Название, описание, производитель.
- Версия, желательно, чтобы она совпадала с версией библиотеки карточек.
- Код продукта (`ProductCode`) — уникальный код инсталляционного пакета (будет сгененрирован автоматически).
- Базовый путь для установки файлов.

Путь для установки файлов обычно зависит от выбранной области установки. При установке "Только для текущего пользователя" это обычно путь: `$\КАТАЛОГ_ПРОФИЛЯ_ПОЛЬЗОВАТЕЛЯ\Local Settings\DocsVision\РЕШЕНИЕ\ВЕРСИЯ\Client\`, а при установке в область "Для всех пользователей" — путь: `$\Program Files\DocsVision\РЕШЕНИЕ\Client\`.

> Особенность работы Windows Installer такова, что даже при установке в режиме "только для текущего пользователя", ключи реестра для библиотек типов (*TypeLib*) будут создаваться в общей ветке реестра (`HKEY_LOCAL_MACHINE`).
>
> 
>
> В случае отсутствия у пользователя административных привилегий на компьютере, это может привести к ошибкам в процессе установки и некорректному функционированию решения. Для избежания подобных ошибок желательно отказаться от регистрации библиотек типов (если они не нужны) — для этого нужно проследить, чтобы в таблице *TypeLib* инсталляционного пакета не было никаких записей.

После создания инсталляционного пакета его необходимо сопоставить с библиотекой карточек. Это делается при помощи утилиты "CardManager" — в описании библиотеки на вкладке *Installers*:

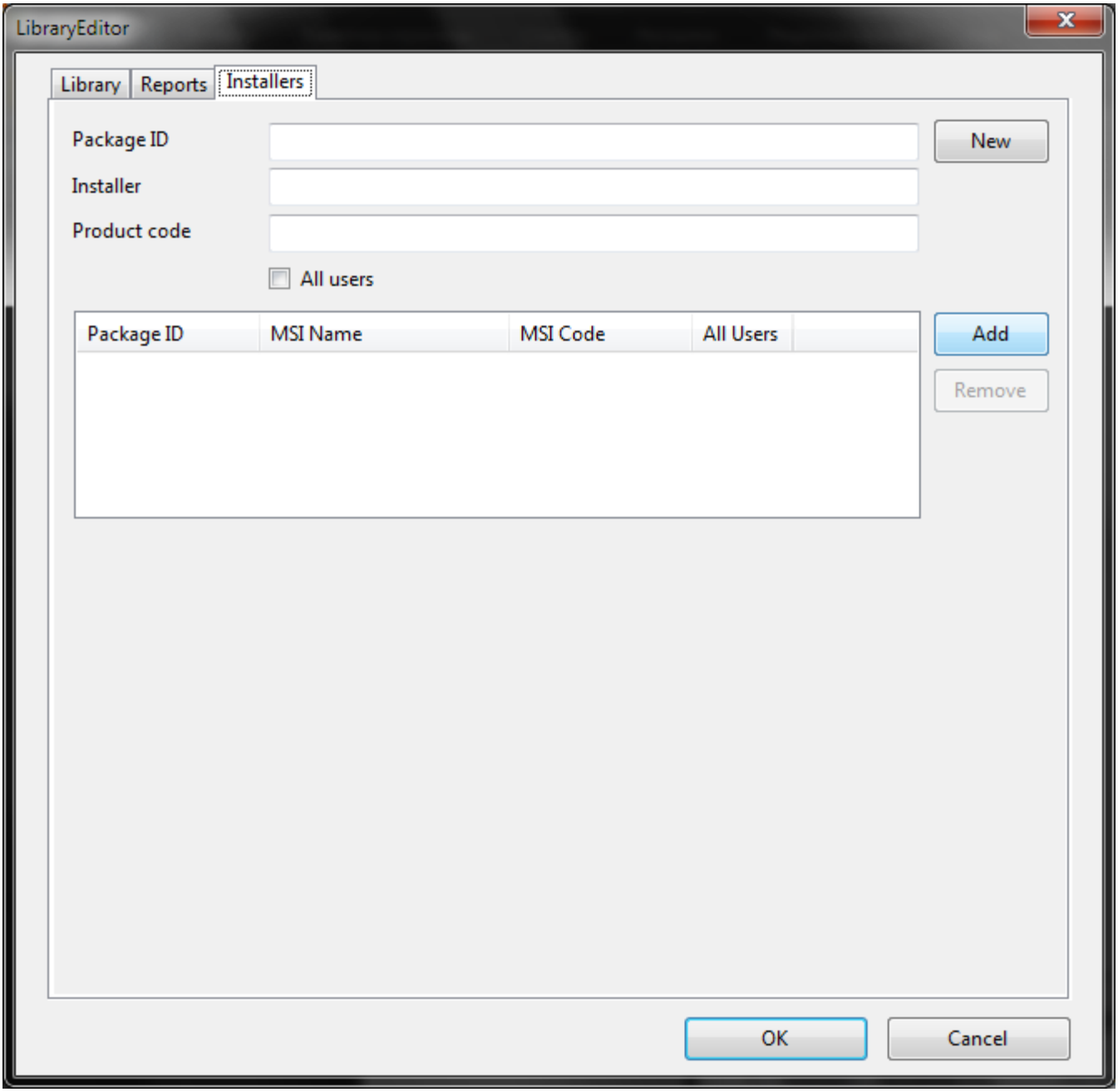


Рисунок 36. Вкладка "Installers"

Поле *PackageID* (значение генерируется автоматически) содержит уникальный идентификатор данной программы инсталляции (у библиотеки может быть несколько инсталляционных программ). В поле *ProductCode* необходимо скопировать идентификатор программы установки, сгенерированный при её создании, а в поле *Installer* — имя файла с относительным путем, по которому на сервере будет расположен инсталляционный пакет.

Готовый клиентский инсталляционный пакет необходимо расположить на сервере (вручную или при помощи программы инсталляции серверной части решения), где он будет доступен для скачивания и установки всем клиентам (см.

выше).

При обновлении версии решения на сервере клиентские компоненты будут обновлены на новую версию автоматически после первого обращения клиента к обновленному серверу.

Для корректного обновления решения на клиентских машинах при изменении версии на сервере в инсталляционном пакете новой версии решения необходимо сгенерировать новый *ProductCode*, но оставить прежнее значение *UpgradeCode*.

Отладка и тестирование

При разработке карточек Docsvision перед разработчиками встает задача отладки и тестирования разработанных компонент.

Процесс отладки и тестирования включает в себя следующие подзадачи:

- Проверка правильной загрузки библиотек и схем карточек в базу данных и соответствия созданных структур, данных их описаниям.
- Мониторинг состояния объектов системы.
- Проверка функционирования клиентского компонента карточки в различных режимах.
- Поиск и локализация ошибок в работе компонент карточек.
- Прямое воздействие на данные карточек в системе.
- Другие задачи.

Разработка скриптов карточки

Помимо разработки новых типов карточек, а также модификации компонентов существующих, новая бизнес-логика может быть добавлена в карточку с помощью скрипта карточки. В скриптах используется стандартный API Docsvision (базовый и объектная модель), а средой разработки служит "/dv6/backoffice/6.1/desdirs/layouts/designer/[Конструктор разметок"] и /dv6/backoffice/6.1/desdirs/scripts/designer/["Конструктор скриптов]", описание работы с которыми приведено в документации для модуля *Базовые объекты*.

В данном разделе приведена информация о специальных возможностях API Docsvision, которые могут быть использованы в скриптах карточки.

В этом разделе:

- Взаимодействие с контейнером карточек
- Подписка на события
- Доступ к подчиненным карточкам из карточки BackOffice
- Доступ к элементам ленты карточки

Взаимодействие с контейнером карточек

При разработке визуального интерфейса карточек или приложений, работающих с Docsvision, часто возникает необходимость взаимодействия с контейнером карточек (Windows-клиентом) для решения одной из следующих задач:

- Выбор карточки.
- Выбор папки.
- Отображение конкретной карточки.
- Навигация по дереву папок.
- Другие задачи.

Для решения этих задач можно использовать методы объекта — контейнера карточек (объект должен реализовывать интерфейс `DocsVision.Platform.CardHost.ICardHost`).

Ссылку на этот объект все карточки получают в момент инициализации, и впоследствии она доступна в любой момент жизненного цикла карточки через соответствующее свойство базового класса — `CardControl.CardHost`.

В случае разработки внешних приложений и утилит, которые работают вне контекста Windows-клиента, этот объект необходимо создавать явно. Для этого предназначен метод `CreateInstance`. На вход метода достаточно передать ссылку на текущую сессию.

Пример создания объекта-контейнера карточек:

```
SessionManager manager = SessionManager.CreateInstance();
manager.Connect("http://localhost/DocsVision/StorageServer/StorageServerService.asmx",
string.Empty);
UserSession session = manager.CreateSession();
ICardHost host = CardHost.CreateInstance(session);
```

Помимо контейнера, в карточку также передается ссылка на другой объект — текущее окно, в котором открыта карточка (`CardControl.CardFrame`).

Этот объект позволяет выполнять действия с окном:

- Закрыть окно.
- Задать заголовок окна.
- Открыть карточку в новом окне.
- Прочие действия.

При разработке внешних приложений и утилит получить ссылку на объект `CardFrame` невозможно, т.к. они работают вне контекста контейнера (Windows-клиента).

Пример кода работы с окном, где заголовок окна изменяется по мере ввода текста в некую форму:

```
private void txtName_TextChanged(object sender, EventArgs e)
{
 CardFrame.Caption = txtName.Text;
 isChanged = true;
}
```

Справочники платформы Docsvision поддерживают возможность программного открытия и позиционирования на конкретной записи, для чего предусмотрено использование метода `ShowCard` с передачей в качестве параметра `activateParams` массива управляющих объектов (см. Параметры активации справочников).

В зависимости от типа справочника передаваемый массив может содержать различное число объектов. При открытии возможно осуществлять позиционирование курсора на конкретной записи, открытие записи на просмотр, либо на редактирование. Примеры смотрите по ссылке Программное открытие справочников.

Программное открытие справочников

Во всех следующих примерах, описывающих методы открытия различных видов справочников:

- `selectionMode` — определяет метод открытия записи и соответствует значению из перечисления `DocsVision.BackOffice.WinForms.Controls.ItemSelectionMode` приведенному к `System.Int32`.

  Допустимые значения:

  - `0` — позиционирование на записи.
  - `1` — открытие записи для редактирования.

- `2` — открытие записи для чтения.

- `multiselect` — возможность выбора нескольких записей, соответствует типу `System.Boolean`.

  Значение `true`(множественный выбор разрешён) допустимо при `selectionMode` установленном в `0`, иначе должно быть `false` .

Методы в примере вызываются из некого скрипта, привязанного с кнопке в *Конструкторе разметок*. В компоненте карточки, унаследованном от `DocsVision.Platform.WinForms.CardControl` или `BaseCardControl`, код будет выглядеть аналогично. Единственное отличие в том, что параметр `CardHost` будет доступен уже из базового класса.

Открытие записи из "Справочника категорий" в режиме редактирования

```
private void RefCategories_Click(object sender, EventArgs e)
{
 string categoryID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant(); ①
 int selectionMode = 1;
 bool multiselect = false;

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefCategories.ID; ②

 object activateParams = new object[] { System.Reflection.Missing.Value, categoryID,
multiselect, System.Reflection.Missing.Value,
     selectionMode }; ③

 base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, DocsVision.Platform.CardHost
.ActivateMode.Edit, DocsVision.Platform.CardHost.ActivateFlags.None,
     activateParams); ④
}
```

① Идентификатор выбираемой категории в Справочнике категорий.

② Идентификатор Справочника категорий.

③ Параметры активации.

④ Вызов метода отображения карточки в режиме редактирования.

Открытие записи из "Справочника ссылок" в режиме редактирования

```
private void RefLinks_Click(object sender, EventArgs e)
{
```

```
 string linkID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant(); ①
 int selectionMode = 1;
 bool multiselect = false;

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefLinks.ID; ②

 object activateParams = new object[] { System.Reflection.Missing.Value, linkID,
multiselect, System.Reflection.Missing.Value,
     System.Reflection.Missing.Value, selectionMode }; ③

 base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, DocsVision.Platform.CardHost
.ActivateMode.Edit, DocsVision.Platform.CardHost.ActivateFlags.None,
     activateParams); ④
}
```

① Идентификатор выбираемой ссылки в Справочнике ссылок.

② Идентификатор Справочника ссылок.

③ Параметры активации.

④ Вызов метода показа карточки в режиме редактирования.

Открытие записи сотрудника из "Справочника сотрудников" в режиме редактирования

```
private void RefStaffEmployees_Click(object sender, EventArgs e)
{
 string employeeID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant(); ①
 int selectionMode = 1;
 bool multiselect = false;

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefStaff.ID; ②

 string sectionID = DocsVision.BackOffice.CardLib.CardDefs.RefStaff.Employees.ID.
ToString("B").ToUpperInvariant(); ③

 object activateParams = new object[] { sectionID, employeeID, System.Reflection.Missing
.Value, System.Reflection.Missing.Value,
     System.Reflection.Missing.Value, multiselect, false, false, 0, System.Reflection
.Missing.Value, true, true, selectionMode }; ④

 base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, DocsVision.Platform.CardHost
.ActivateMode.Edit, DocsVision.Platform.CardHost.ActivateFlags.None,
     activateParams);
}
```

① Идентификатор выбираемого сотрудника.

② Идентификатор Справочника сотрудников.

③ Идентификатор выбираемой секции справочника — сотрудники.

④ Параметры активации.

Открытие записи группы из "Справочника сотрудников" в режиме редактирования

```
private void RefStaffGroups_Click(object sender, EventArgs e)
{
 string groupID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant();
 int selectionMode = 1;
 bool multiselect = false; ①

 string sectionID = DocsVision.BackOffice.CardLib.CardDefs.RefStaff.AlternateHierarchy.
ID.ToString("B").ToUpperInvariant(); ②

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefStaff.ID; ③

 object activateParams = new object[] { sectionID, groupID, System.Reflection.Missing
.Value, System.Reflection.Missing.Value,
     System.Reflection.Missing.Value, multiselect, false, false, 0, System.Reflection
.Missing.Value, true, true, selectionMode }; ④

 base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, DocsVision.Platform.CardHost
.ActivateMode.Edit, DocsVision.Platform.CardHost.ActivateFlags.None,
     activateParams); ⑤
}
```

① Идентификатор выбираемой группы.

② Идентификатор выбираемой секции справочника — группы.

③ Идентификатор Справочника Сотрудников.

④ Параметры активации.

Открытие записи организации из "Справочника сотрудников" в режиме редактирования

```
private void RefStaffUnits_Click(object sender, EventArgs e)
{
 string unitID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant();
 int selectionMode = 1;
 bool multiselect = false; ①
```

```
 string sectionID = DocsVision.BackOffice.CardLib.CardDefs.RefStaff.Units.ID.ToString(
"B").ToUpperInvariant(); ②

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefStaff.ID; ③

 object activateParams = new object[] { sectionID, unitID, System.Reflection.Missing
.Value, System.Reflection.Missing.Value, System.Reflection.Missing.Value,
     multiselect, false, false, 0, System.Reflection.Missing.Value, true, true,
selectionMode }; ④

 base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, DocsVision.Platform.CardHost
.ActivateMode.Edit, DocsVision.Platform.CardHost.ActivateFlags.None,
     activateParams); ⑤
}
```

① Идентификатор выбираемой организации.

② Идентификатор выбираемой секции справочника — организации.

③ Идентификатор Справочника Сотрудников.

④ Параметры активации.

⑤ Вызов метода показа карточки в режиме редактирования.

Открытие записи сотрудника из "Справочника контрагентов" в режиме редактирования

```
private void RefPartnersEmployees_Click(object sender, EventArgs e)
{
 string employeeID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant();
 int selectionMode = 1; ①
 bool multiselect = false;

 string sectionID = DocsVision.BackOffice.CardLib.CardDefs.RefPartners.Employees.ID
.ToString("B").ToUpperInvariant(); ②

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefPartners.ID; ③

 object activateParams = new object[] { sectionID, employeeID, System.Reflection.Missing
.Value, System.Reflection.Missing.Value, System.Reflection.Missing.Value,
     multiselect, System.Reflection.Missing.Value, false, true, true, selectionMode }; ④

 base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, DocsVision.Platform.CardHost
.ActivateMode.Edit, DocsVision.Platform.CardHost.ActivateFlags.None,
     activateParams); ⑤
}
```

① Идентификатор выбираемого сотрудника.

② Идентификатор выбираемой секции справочника — сотрудники.

③ Идентификатор Справочника контрагентов.

④ Параметры активации.

⑤ Вызов метода показа карточки в режиме редактирования.

Открытие записи группы из "Справочника контрагентов" в режиме редактирования

```
private void RefPartnersGroups_Click(object sender, EventArgs e)
{
 string groupID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant();
 int selectionMode = 1; ①
 bool multiselect = false;

 string sectionID = DocsVision.BackOffice.CardLib.CardDefs.RefPartners.Groups.ID.
ToString("B").ToUpperInvariant(); ②

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefPartners.ID; ③

 object activateParams = new object[] { sectionID, groupID, System.Reflection.Missing
.Value, System.Reflection.Missing.Value, System.Reflection.Missing.Value,
     multiselect, System.Reflection.Missing.Value, false, true, true, selectionMode }; ④

 base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, DocsVision.Platform.CardHost
.ActivateMode.Edit, DocsVision.Platform.CardHost.ActivateFlags.None,
     activateParams); ⑤
}
```

① Идентификатор выбираемой группы.

② Идентификатор выбираемой секции справочника — группы.

③ Идентификатор Справочника контрагентов.

④ Параметры активации.

⑤ Вызов метода показа карточки в режиме редактирования.

Открытие записи организации из "Справочника контрагентов" в режиме редактирования

```
private void RefPartnersCompanies_Click(object sender, EventArgs e)
{
 string unitID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant();
```

```
 int selectionMode = 1; ①
 bool multiselect = false;

 string sectionID = DocsVision.BackOffice.CardLib.CardDefs.RefPartners.Companies.ID
.ToString("B").ToUpperInvariant(); ②

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefPartners.ID; ③

 object activateParams = new object[] { sectionID, unitID, System.Reflection.Missing
.Value, System.Reflection.Missing.Value, System.Reflection.Missing.Value,
     multiselect, System.Reflection.Missing.Value, false, true, true, selectionMode }; ④

 base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, DocsVision.Platform.CardHost
.ActivateMode.Edit, DocsVision.Platform.CardHost.ActivateFlags.None,
     activateParams); ⑤
}
```

① Идентификатор выбираемой организации.

② Идентификатор выбираемой секции справочника — организации.

③ Идентификатор Справочника контрагентов.

④ Параметры активации.

⑤ Вызов метода показа карточки в режиме редактирования.

Открытие записи из "Справочника видов карточек" в режиме позиционирования

```
private void RefKindsCompanies_Click(object sender, EventArgs e)
{
 string rootKindID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant(); ①

 string kindID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant(); ②

 string sectionID = DocsVision.BackOffice.CardLib.CardDefs.RefKinds.CardKinds.ID.
ToString("B").ToUpperInvariant(); ③

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefKinds.ID; ④

 object activateParams = new object[] { sectionID, rootKindID, System.Reflection.Missing
.Value, true, System.Reflection.Missing.Value, kindID,
     System.Reflection.Missing.Value, System.Reflection.Missing.Value, System.Reflection
.Missing.Value };

 base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, DocsVision.Platform.CardHost
```

```
.ActivateMode.Edit, DocsVision.Platform.CardHost.ActivateFlags.None,
     activateParams);
}
```

① Идентификатор типа карточки (например, Документ).

② Идентификатор выбираемого вида карточки (например, Внутренний).

③ Идентификатор выбираемой секции справочника.

④ Идентификатор Справочника видов карточек.

Справочник видов карточек поддерживает исключительно позиционирование на записи.

Открытие записи узла из "Конструктора справочников" в режиме редактирования

```
private void RefBaseUniversalNode_Click(object sender, EventArgs e)
{
 string nodeID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant(); ①
 int selectionMode = 1;

 string sectionID = new Guid("A1DCE6C1-DB96-4666-B418-5A075CDB02C9").ToString("B"
).ToUpperInvariant(); ②

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefBaseUniversal.ID; ③

 object activateParams = new object[] { sectionID, nodeID, System.Reflection.Missing
.Value, selectionMode }; ④

 base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, DocsVision.Platform.CardHost
.ActivateMode.Edit, DocsVision.Platform.CardHost.ActivateFlags.None,
     activateParams); ⑤
}
```

① Идентификатор выбираемого узла.

② Идентификатор выбираемой секции справочника — Узлы справочника.

③ Идентификатор Конструктора справочников.

④ Параметры активации.

⑤ Вызов метода показа карточки в режиме редактирования.

Открытие записи строки из "Конструктора справочников" в режиме редактирования

Пример открытия на редактирование строки "Вид ОРД / Приказ":

```
private void RefBaseUniversalString_Click(object sender, EventArgs e)
{
 Guid stringID = new Guid("285A31D9-2433-415A-AEC7-BAD45E705389");
 int selectionMode = 1; ①

 Guid nodeID = new Guid("B8B58477-CCC3-4213-8190-EFC66FEF86C1"); ②

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefBaseUniversal.ID; ③

 object activateParams = new object[] { RefBaseUniversal.Items.ID, stringID, nodeID,
selectionMode }; ④

 base.CardControl.CardHost.ShowCard(RefBaseUniversal.ID, Guid.Empty, ActivateMode.Edit,
ActivateFlags.None, activateParams); ⑤
}
```

① Идентификатор выбираемой строки.

② Идентификатор узла содержащего выбираемую строку.

③ Идентификатор Конструктора справочников.

④ Параметры активации.

⑤ Вызов метода показа карточки в режиме редактирования.

Открытие записи из "Справочника серверов" в режиме редактирования

```
private void RefBaseUniversalString_Click(object sender, EventArgs e)
{
 string serverID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant(); ①
 int selectionMode = 1;

 Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefServers.ID; ②

 object activateParams = new object[] { System.Reflection.Missing.Value, serverID,
selectionMode }; ③

 base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, DocsVision.Platform.CardHost
.ActivateMode.Edit, DocsVision.Platform.CardHost.ActivateFlags.None,
     activateParams); ④
}
```

① Идентификатор выбираемого сервера.

② Идентификатор Справочника серверов.

③ Параметры активации.

④ Вызов метода показа карточки в режиме редактирования.

Подписка на события

Элементы управления, используемые при создании пользовательского интерфейса карточек библиотеки *Базовые объекты*, разрешают подписываться на события, наследуемые от оригинальных компонентов.

Если подписка на событие выполняется из кода скрипта карточки (а не настроена в Конструкторе разметок), необходимо добавить в скрипт карточки код, отписывающий от события. Код должен выполняться при закрытии карточки.

В противном случае из-за использования механизма повторного использования компонента карточки обработчик события может вызываться многократно (по числу повторных использований компонента карточки).

События элемента управления "Ссылки"

Стандартный элемент управления "Ссылки" предоставляет возможность подписываться на события вызова настраиваемых команд (`CustomizeCommands`), которые выбираются из контекстного меню элемента управления "Ссылки". Для подписки на подобные события необходимо знать идентификатор соответствующей команды, которые собраны в статическом классе `DocsVision.BackOffice.WinForms.Controls.Commands.ReferenceListView.ReferenceListViewCommandIds` (сборка `DocsVision.BackOffice.WinForms.dll`):

- `ReferenceListViewCommandAddCardFromTemplateId` — добавление карточки по шаблону.
- `ReferenceListViewCommandAddCardId` — добавление карточки.
- `ReferenceListViewCommandAddDVFolderId` — добавление папки Docsvision.
- `ReferenceListViewCommandAddFileFromFileSystemId` — добавление файла.
- `ReferenceListViewCommandAddURLId` — добавление URL.
- `ReferenceListViewCommandCancelLockFileId` — отмена блокировки.
- `ReferenceListViewCommandCopyId` — копирование.

- `ReferenceListViewCommandCreateCardId` — создание карточки.
- `ReferenceListViewCommandDeleteId` — удаление.
- `ReferenceListViewCommandLockFileId` — блокировка для работы.
- `ReferenceListViewCommandOpenCardId` — открытие карточки.
- `ReferenceListViewCommandOpenFileAsReadOnlyId` — открытие файла в режиме чтения.
- `ReferenceListViewCommandOpenFileId` — открытие файла в режиме изменения.
- `ReferenceListViewCommandOpenId` — открытие карточки.
- `ReferenceListViewCommandRefreshId` — обновление.
- `ReferenceListViewCommandScanAndAddId` — сканирование и добавление.
- `ReferenceListViewCommandShowDescriptionId` — описание.
- `ReferenceListViewCommandShowGroupPanelId` — отображение области группировки.
- `ReferenceListViewCommandShowPreviewPanelId` — отображение области описания.
- `ReferenceListViewCommandUnlockFileId` — отображение области группировки.

Элемент управление "Ссылки" предоставляет единую точку входа для обработки всех этих событий — событие `CustomizeCommands`. В качестве примера, используем собственный обработчик события `CustomizeCommands` для обработки команд добавления URL-ссылки и открытия карточки из списка добавленных карточек:

```
private void Document_CardActivated(System.Object sender, DocsVision.Platform.WinForms
.CardActivatedEventArgs e) ①
{
 ReferenceListView referenceListView = ((ICustomizableControl)CardControl
).FindPropertyItem<ReferenceListView>("Свойство1"); ②

 referenceListView.CustomizeCommands += new EventHandler
<ReferenceListViewCustomizeCommandsEventArgs>(ReferenceListView_CustomizeCommands); ③
}
private void ReferenceListView_CustomizeCommands(object sender,
ReferenceListViewCustomizeCommandsEventArgs e) ④
{
 foreach (IReferenceListViewCommand command in e.Commands)
 {
  Guid commandId = ((IReferenceListViewCommand2)command).Id; ⑤

  if (commandId = ReferenceListViewCommandIds.ReferenceListViewCommandAddURLId) ⑥
  {
   command.BeforeCommandExecute += new EventHandler
```

```
<ReferenceListViewCommandExecuteEventArgs>(AddUrlCommand_BeforeCommandExecute); ⑦
  }
  else if (commandId = ReferenceListViewCommandIds.ReferenceListViewCommandOpenCardId)
  {
   command.AfterCommandExecute += new EventHandler
<ReferenceListViewCommandExecuteEventArgs>(OpenCardCommand_AfterCommandExecute); ⑧
  }
 }
}
private void AddUrlCommand_BeforeCommandExecute(object sender,
ReferenceListViewCommandExecuteEventArgs e) ⑨
{
 e.Cancel = true; ⑩
}
private void OpenCardCommand_AfterCommandExecute(object sender,
ReferenceListViewCommandExecuteEventArgs e) ⑪
{
 MessageBox.Show("Добавлена карточек";
}
```

① Организуем подписку в обработчике события активации карточки.

② Получаем элемент управления типа "Ссылки".

③ Добавляем обработчик для срабатывания настраиваемой команды.

④ Обрабатываем событие вызова настраиваемой команды.

⑤ Получаем идентификатор вызванной команды.

⑥ Выбираем команду добавления URL, либо открытия карточки.

⑦ Обрабатываем событие до непосредственного добавления URL.

⑧ Обрабатываем событие после открытия карточки.

⑨ Обработка команды добавления URL.

⑩ Прервать дальнейшее исполнение команды.

⑪ Обработка команды добавления карточки.

В обработчиках событий, срабатывающих до исполнения соответствующей команды, можно добавить логику прерывания команды, как в примере, установив `e.Cancel` в значение `true`.

События элемента управления файлами документа

Элемент управления файлами документа реализует интерфейс `IFilesViewWithPreview`, который добавляет подписку на события:

- `BeforeFileOpened`, `AfterFileOpened` — перед и после открытия основного или дополнительного файла.
- `BeforeMainFileAdded`, `MainFileAdded` — перед и после добавления основного файла.
- `BeforeAdditionalFileAdded`, `AdditionalFileAdded` — перед и после добавления дополнительного файла.
- `BeforeMainFileRemoved`, `MainFileRemoved` — перед и после удаления основного файла.
- `BeforeAdditionalFileRemoved`, `AdditionalFileRemoved` — перед/после удаления дополнительного файла.
- `MainFileSelected` — при выборе основного файла.
- `AdditionalFileSelected` — при выборе дополнительного файла.
- `DocumentUnlocking`, `DocumentUnlocked` — перед и после снятия блокировки с основного или дополнительного файла.

Для подписки на событие в скрипте карточки, достаточно получить из компонента карточки элемент управления файлами, возвращаемый из свойства `FilesViewWithPreview`. К примера подпишемся на событие, вызываемое перед открытием любого файла:

```
// Указываем дополнительные пространства имён
using DocsVision.BackOffice.ObjectModel.Services.Entities;
using DocsVision.BackOffice.Cards.CardDocument; ①

/* часть кода пропущена */

private void CardDocument_CardActivated(System.Object sender, DocsVision.Platform
.WinForms.CardActivatedEventArgs e) { ②

 var documentCardControl = this.CardControl as DocsVision.BackOffice.Cards.CardDocument
.MainControl; ③
 if (documentCardControl.FilesViewWithPreview != null) {
  IFilesViewWithPreview filesViewWithPreview = documentCardControl.FilesViewWithPreview;
④

  filesViewWithPreview.BeforeFileOpened += new EventHandler<DocumentOpenFileEventArgs
>(FilesViewWithPreview_BeforeFileOpened); ⑤
 }
}
```

```
private void FilesViewWithPreview_BeforeFileOpened(object sender,
DocumentOpenFileEventArgs e) {
 if (e.File.FileType = DocumentFileType.Main) {
  MessageBox.Show("Открытие файла прервано");

  e.Cancel = true; ⑥
 }
}
```

① Подписываемся на события элемента управления при активации карточки.

② Приводим тип компонента карточки к типу компонента карточки Документ.

③ Получаем элемент управления файлами.

④ Подписываемся на событие перед открытием файла .

⑤ Прерываем открытие.

Чтобы пример заработал нужно подключить дополнительную сборку: `DocsVision.BackOffice.Cards.dll`

В обработчиках событий, срабатывающих до исполнения соответствующей команды, можно добавить логику прерывания команды — установить `e.Cancel` в значение `true`.

Доступ к подчиненным карточкам из карточки BackOffice

Функциональность класса компонента карточки типа *Задание* позволяет получать доступ к связанным карточкам: документам или другим заданиям. Список возможностей доступа к связанным карточкам определяется интерфейсом `ICardView` который добавляет следующие свойства и методы:

- `SaveChildCardOnClosing` — задаёт или возвращает признак сохранения подчиненных карточек при их закрытии. Если значение `true`, то при закрытии подчиненной карточки изменения будут сохранение автоматически без запроса на сохранение.
- `ChildCardActivated` — событие, которое срабатывает при активации компонента какой-либо подчиненной карточки.
- `OnChildCardActivated(ActiveChildCardEventArgs)` — обработчик события активации подчиненной карточки.
- `GetChildCardIds` — возвращает список идентификаторов подчиненных карточек.

- `ActivateChildCard(Guid)` — активирует подчиненную карточку по её идентификатору.
- `GetActivatedChildCards` — возвращает массив компонентов активированных карточек. Активированной считается карточка, компонент которой был загружен в память, при этом сама карточка может быть закрыта.

Доступ к элементам ленты карточки

Лента карточки — сущность типа `RibbonControl`, доступ к которой обеспечивает одноименное свойство класса `ICustomizableControl`:

```
RibbonControl ribbonControl = (CardControl as ICustomizableControl).RibbonControl;
```

Класс `RibbonControl` предоставляет доступ к существующим элементам ленты (страницы, группы и элементы), а также позволяет добавлять новые.

В качестве примера использования данной сущности можно рассмотреть сценарий добавления на ленту карточки новой кнопки на страницу *Документ* в группу *Общие*.

Кнопка может добавляться, когда стандартных механизмов (например, ролевой модели) недостаточно для реализации сценария:

- При загрузке карточки,
- Если соблюдены определённые условия.
- Если выполняются определённые действия с карточкой.
- Если выполняется иная бизнес-логика. В приведенном далее примере добавление нового элемента — кнопки, выполняется при загрузке карточки.

Отладка скриптов карточек

Отладка скриптов карточек выполняется в Visual Studio.

1. Создайте в Visual Studio проект с типом Windows Forms Control Library (или Class library).

   В качестве целевой платформы укажите:

   - .NET Framework 4.0 — если используется система Docsvision версии 5.4,
   - .NET Framework 4.6.1 (не "Client profile") — если используется система Docsvision версии 5.5.
   - .NET Desktop Runtime (x86/x64) версии 8.0 и выше — если используется

система Docsvision версии 6.1.

2. Подключите к проекту сборки из папки установки Windows-клиента:
   - `DocsVision.Platform.ObjectManager.dll`.
   - `DocsVision.Platform.ObjectModel.dll`.
   - `DocsVision.Platform.WinForms.dll`.
   - `DocsVision.Platform.CardHost.dll`.
   - `DocsVision.BackOffice.ObjectModel.dll`.
   - `DocsVision.BackOffice.WinForms.dll`.
   - Другие необходимые сборки.
3. Добавьте в проект публичный класс с публичным методом, содержащим код отлаживаемого скрипта.

   В метод можно передать любые требуемые параметры, например, `BaseCardControl` даст доступ к классу карточки.

4. Постройте проект и подключите полученную сборку к скриптам карточки.
5. Измените реализацию отлаживаемого скрипта карточки, оставив только вызов функции, реализованной в отладочной сборке.
6. Подключитесь в Visual Studio к процессу Windown-клиента (`Navigator.exe`). Будет предоставлена возможность выполнить отладку кода в контексте работы карточки.

Visual Studio предоставляет средства, которые можно использовать для удалённой отладки — Visual Studio Remote Debugger.

7. После завершения отладки, рабочий код из проекта можно скопировать в скрипт карточки и отключить отладочную сборку.

Разработка в среде СУБП

Данный раздел содержит информацию, предназначенную для разработчиков, создающих решение на платформе Docsvision с использованием приложения *Управление процессами*. В разделе раскрываются вопросы связанные с созданием дополнительных *функций* СУБП, *сценариев*, *шлюзов* и других объектов системы, расширяющих её функциональные возможности по

реализации специфических бизнес-процессов.

В данном разделе освещаются следующие темы:

- Правила разработки пользовательских *функций* и *шлюзов*.
- Правила разработки, отладки и использования *сценариев*.

Общие сведения

Платформа Docsvision и приложение *Управление процессами* представляют собой систему описания сложных процессов обработки информации в организации и полнофункциональный Workflow-сервер. *СУБП* представляет собой решение на основе платформы Docsvision, предназначенное для создания бизнес-процессов любой сложности и управления ими.

Ключевыми объектами в терминологии *СУБП* являются *бизнес-процесс*, *переменные процесса*, *функции* и *шлюзы*.

Бизнес-процесс

Бизнес-процесс представляет собой формальное описание бизнес-процесса организации, которое строится из набора атомарных операций (*функций*), составляющих процесс, и связей между ними. Связи между функциями определяют порядок выполнения операций. *Функции бизнес-процесса* оперируют данными внешних систем с помощью *шлюзов* и обмениваются данными через *переменные процесса*.

Бизнес-процесс должен обязательно включать в себя специальную *функцию*, инициирующую выполнение процесса, а также одну или несколько *функций*, завершающих выполнение процесса. Остальные *функции* могут быть связаны между собой произвольным количеством связей, которые определяют порядок их выполнения в *бизнес-процессе*. Возможно наличие в процессе циклов, условных и безусловных переходов.

Физически *бизнес-процесс* состоит из набора данных, хранящихся в базе данных Docsvision, набора свойств и методов для операций с этими данными, а также карточки Docsvision типа *Бизнес-процесс*, реализующей пользовательский интерфейс для управления данными процесса.

Переменные процесса

Переменные процесса являются сущностями, которыми оперируют *функции* процесса. Переменные обеспечивают единый способ обмена информацией между *функциями* процесса.

Каждая переменная имеет уникальный идентификатор в формате строки. Индетификатор строки позволяет отличить её от остальных переменных *бизнес-процесса*: имя, значение, величину или набор данных, характеризующих переменную, и отображаемое значение, которое может как соответствовать значению, так и характеризовать переменную для пользователя более явным образом. Например, для переменной, представляющей карточку Docsvision, значением является идентификатор карточки, а отображаемым значением — её название.

Имена переменных в бизнес-процессе являются регистрозависимыми.

Все *переменные процесса* являются строго типизированными. К стандартным (системным) типам переменных относятся следующие типы:

- Строка.
- Целое.
- Дробное.
- Логическое значение ("Да/Нет").
- Дата/Время.
- Перечисление — при формировании шаблона процесса можно описать поля типа перечисления и задать все возможные значения. Все значения перечисления в процессе — строковые.

Кроме этих стандартных типов, в процесс могут быть добавлены переменные шлюзов. Каждый зарегистрированный в системе шлюз поставляет собственные типы переменных.

Автоматического преобразования типов не предусмотрено. Для преобразования экземпляра переменной из одного типа в другой (если это предусмотрено) необходимо использовать специальные функции, либо воспользоваться *сценарием*.

Функции

Описание процесса содержит набор принадлежащих ему *функций*, настройки их свойств. Свойства *функции* могут быть проинициализированы как непосредственным заданием значения, так и некой переменной процесса. Описание процесса также содержит описание связей между *функциями*. Каждая *функция* имеет собственный набор данных (обычно — в формате XML),

отображающих её настройки и любые дополнительные данные.

Функции и связи между ними могут иметь собственные названия и описания для улучшения читаемости *бизнес-процесса*.

Экземпляры каждой *функции* могут выполняться как параллельно при установке соответствующего свойства, так и последовательно, что означает использование одних и тех же данных для выполнения *функции* при её повторном запуске, например в циклах. Последовательное выполнение экземпляров *функции* подразумевает, что её повторный запуск не будет произведен, пока не будет завершено исполнение предыдущего экземпляра.

Функция может быть как независимой — выполняться без взаимодействия с внешними системами, так и привязанной к одной или нескольким внешним системам при помощи *шлюзов*. Привязка к *шлюзам* декларируется *функцией* на программном уровне и проверяется на этапе добавления *функции* в процесс.

Шлюзы

Шлюз представляет собой совокупность программных компонент, предназначенных для обеспечения взаимодействия *СУБП* с внешними системами.

Основными функциями шлюза являются:

- Обмен данными между прикладной системой и СУБП.
- Маршрутизация ссылки на объект прикладной системы в функциях СУБП.
- Специализированная обработка прикладного объекта в рамках процесса.

Шлюзы также предоставляют свой набор типов переменных, которыми можно оперировать в *СУБП* наряду с обычными переменными. Под переменными шлюза подразумеваются все возможные виды объектов, которыми может оперировать *шлюз*. Например, *шлюз* к файловой системе оперирует двумя типами объектов — файлами и папками файловой системы. Соответственно, данный *шлюз* предоставляет *бизнес-процессам* два типа переменных — `Папка файловой системы` и `Файл файловой системы`.

Выбор механизма использования уникальных значений и отображаемых значений переменных, поставляемых конкретным *шлюзом*, выбирается разработчиком *шлюза*. Например, в рассмотренном выше *шлюзе* к файловой системе, в качестве уникальных значений используются полные физические пути к файлам и папкам, а в качестве отображаемых значений — их частичные

имена.

Программные компоненты СУБП

Разработка в среде СУБП

Программные компоненты СУБП

Данный раздел содержит информацию, предназначенную для разработчиков, создающих решение на платформе Docsvision с использованием приложения *Управление процессами*.

- Шлюзы
- Функции
- Сценарии
- Больше подробностей в навигационном меню слева…

Шлюзы

Пользовательские *шлюзы* разрабатываются как дополнение к системе управления *бизнес-процессами* при решении задач интеграции *СУБП* с внешними системами, взаимодействие с которыми не предусмотрено в стандартной поставке Docsvision. К таким системам можно отнести сторонние системы документооборота, EPR-системы, бухгалтерские системы и прочие.

Все пользовательские *шлюзы* заносятся в *справочник шлюзов*. После занесения в справочник шлюз становится доступен в объектной модели и в карточке бизнес-процесса.

Реализация пользовательского шлюза включает в себя, как минимум, два основных компонента:

- Клиентский компонент, реализующий пользовательский интерфейс для отображения и выбора объектов *шлюза*.
- Серверный компонент, работающий в рамках сервиса *СУБП* и реализующий программный интерфейс для работы с объектами *шлюза*.

Каждый *шлюз* имеет собственный уникальный идентификатор, задаваемый на этапе разработки. Этот идентификатор заносится в справочник и используется в дальнейшем для обеспечения уникальности *шлюзов* в системе.

Клиентский компонент шлюза

Клиентский компонент *шлюза* представляет собой стандартный COM-компонент, реализующий ряд предопределённых интерфейсов, в частности:

- Интерфейс `IObjectUI` — стандартный для всех клиентских объектов *СУБП* (*функций* и *шлюзов*) интерфейс, который определяет наличие у объекта идентификатора, названия, иконки и хранимых данных.
- Интерфейс `IGateUI` — стандартный интерфейс клиентского компонента шлюза. Реализует свойства и методы для работы с доступными типами переменных (объектов шлюза). Например, шлюз к файловой системе может обрабатывать два типа переменных — файлы и папки файловой системы.
- Интерфейс `IDVObjectUI` (необязательный) — стандартный интерфейс для компонент, использующих объекты Docsvision. Позволяет получить доступ к текущей сессии и окну Windows-клиента.

Наиболее важными в реализации клиентского компонента шлюза являются следующие методы:

- `GetObjectTypes` — возвращает коллекцию описаний регистрируемых шлюзом типов переменных.
- `GetMethods` — возвращает коллекцию описаний методов, предоставляемых шлюзом для манипуляций над объектами.
- `SelectObject` — выбор значения переменной указанного типа. Обычно, вызов данного метода влечет за собой открытие одного или нескольких диалоговых окон.
- `GetObjectDisplayValue` — возвращает отображаемое значение переменной указанного типа по её числовому значению.

Шлюз может иметь собственные хранимые данные (например, настройки соединения с внешней системой или иные необходимые настройки), которые могут меняться в зависимости от того, в каком процессе используется *шлюз*. В таком случае эти данные передаются и возвращаются при помощи свойства `Data` интерфейса `IObjectUI` и сохраняются в базе данных Docsvision. Пользовательский интерфейс для изменения этих настроек (в случае необходимости) также реализуется клиентским компонентом *шлюза* с помощью страниц свойств.

Для передачи *шлюзу* настроек, общих для всех экземпляров *шлюза* одного типа, также предусмотрен интерфейс `IGateInit`. В существующих шлюзах он используется для установки xml-описания *шлюза*.

Для разработки страниц свойств рекомендуется использовать компонент

Property Page Designer (DDPropPageD.ocx), входящий в комплект поставки Docsvision.

Настройки (данные) *шлюза* используются в дальнейшем в серверном компоненте *шлюза* при его инициализации.

При занесении шлюза в справочник необходимо указать программный идентификатор (ProgID) разработанного клиентского компонента.

Клиентский компонент может быть разработан с использованием Visual Basic или Visual C++.

Серверный компонент шлюза

Серверный компонент *шлюза* реализуется в виде .NET-сборки, и используется сервисом *СУБП* при работе с объектами *шлюза* (переменными). Компонент реализует следующую функциональность:

- Предоставление программного интерфейса для обработки объектов *шлюза*.
- Создание экземпляров объектов.
- Удаление экземпляров объектов.
- Сравнение объектов.
- Дополнительные функции (свои для каждого шлюза).

Компонент реализует стандартный интерфейс IGate, содержащий методы, вызываемые сервисом *СУБП* или *функциями* процесса.

Кроме этого интерфейса, серверный компонент *шлюза* определяет собственные интерфейсы для всех типов переменных, реализуемых *шлюзом*. Каждый объект *шлюза* реализует стандартный интерфейс IGateVariable для совместимости с другими *шлюзами*, а также, возможно, некоторые специфические для данного типа методы. Например, тип Пользователь может, кроме стандартных свойств, содержать такие специфические свойства как ФИО, Дата рождения и другие.

Серверный компонент шлюза реализуется на любом из языков платформы .NET — C# или Visual Basic .NET.

Функции

Пользовательские *функции* дополняют систему управления бизнес-процессами при решении задач расширения функциональности *СУБП*, которые не могут быть решены при помощи сценариев. Чаще всего, пользовательские *функции*

реализуют механизмы работы со специфическими (пользовательскими) *шлюзами* и поставляются вместе с ними.

Все пользовательские *функции* заносятся в *справочник функций*. После добавления в справочник функция становится доступной в объектной модели и появляется на панели инструментов в карточке бизнес-процесса.

Реализация пользовательской функции включает в себя два основных компонента:

- Клиентский компонент, реализующий пользовательский интерфейс для настройки параметров *функции* (если таковые предусмотрены).
- Серверный компонент, работающий в рамках сервиса *СУБП* и реализующий логику работы *функции*.

Функция может быть независимой от *шлюза* (глобальной функцией *СУБП*), либо требовать для своего исполнения наличия в процессе одного (либо нескольких) предустановленных *шлюзов*. Эти требования определяются на этапе разработки *функции*. Например, функция копирования файлов может требовать для своей работы наличия в процессе *шлюза* к файловой системе.

Каждая *функция* имеет собственный уникальный идентификатор (`ID`), задаваемый на этапе разработки. Этот идентификатор заносится в справочник и используется в дальнейшем для обеспечения уникальности *функций* в системе.

Клиентский компонент функции

Клиентский компонент *функции* представляет собой COM-компонент, имеющий пользовательский интерфейс для настройки её свойств.

Клиентский компонент функции реализует ряд предопределённых интерфейсов:

- Интерфейс `IObjectUI` — стандартный для всех клиентских объектов СУБП — содержит свойства, возвращающие идентификатор объекта, его название, иконку и хранимые данные.
- Интерфейс `IFunctionUI` — стандартный интерфейс клиентского компонента функции — определяет свойства и методы, вызываемые карточкой бизнес-процесса при использовании функции.
- Интерфейс `IExportableFunctionUI` (необязательный) — интерфейс, предназначенный для *функций*, обладающих специфическим механизмом экспорта/импорта. Методы данного интерфейса используются для корректной работы функции при загрузке и выгрузке процесса в XML.
- Интерфейс `IStateFunctionUI` (необязательный) — данный интерфейс должен

быть реализован в *функциях*, имеющих собственный набор состояний (например, таких, как *функция Задание*). Свойства и методы данного интерфейса позволяют определить для каждого состояния функции собственный цвет, название и иконку, которые будут отображаться на графе бизнес-процесса.

Наиболее важными в реализации клиентского компонента функции являются следующие методы:

- `GetRequiredGates` — возвращает список необходимых для работы *функции шлюзов* (идентификаторы *шлюзов*, разделенные запятой). При разработке *функции*, данные идентификаторы должны быть известны.
- `AllowExecute` — разрешает или запрещает исполнение *функции*. В зависимости от того, все ли необходимые для запуска функции параметры указаны, функция может разрешить или запретить создание экземпляра процесса. Данный метод вызывается из карточки бизнес-процесса.
- `Convert` — позволяет преобразовать настройки *функции* при преобразовании процесса к более поздней версии.

Все настройки *функции* сохраняются в строчном формате, передаются и возвращаются при помощи свойства `Data` интерфейса `IObjectUI` и сохраняются в базе данных Docsvision. Для удобства сохранения настроек функции рекомендуется использовать формат XML.

При реализации визуального интерфейса настройки параметров *функции* рекомендуется использовать *механизм страниц свойств* (Property Pages). Тогда уникальные страницы свойств *функции* будут добавлены к стандартным страницам свойств в карточке бизнес-процесса. При разработке страниц свойств рекомендуется использовать компонент Property Page Designer (`DDPropPageD.ocx`), входящий в комплект поставки Docsvision.

При занесении *функции* в справочник необходимо указать программный идентификатор (`ProgID`) разработанного клиентского компонента.

Клиентский компонент функции может быть разработан с использованием Visual Basic или Visual C++.

Серверный компонент функции

Серверный компонент *функции* реализуется в виде .NET-сборки и определяет внутреннюю логику работы *функции*.

Компонент реализует следующую функциональность:

- Доступ к настройкам (параметрам) *функции*.
- Проверка возможности выполнения *функции*.
- Реализация действий, требуемых от данной *функции*.

Серверный компонент *функции* реализует интерфейс `IFunction`, методы которого вызываются сервисом *СУБП* во время исполнения *функции*.

До начала непосредственного исполнения *функция* получает заданные пользователем настройки в виде строки (свойство `Data`) и производится повторная проверка на их достаточность и полноту (метод `AllowExecute`). Далее вызывается метод `Execute`, реализующий непосредственно логику *функции*.

Серверный компонент *функции* реализуется на любом из языков платформы .NET — C# или Visual Basic .NET.

Сценарии

Сценарии представляют собой мощный механизм расширения функциональности *СУБП*, используемый в тех случаях, когда требуемые действия слишком сложно (или невозможно) выполнить при помощи стандартных *функций*.

Типовые применения сценариев следующие:

- Преобразование типов и форматов данных.
- Динамическое изменение данных процесса во время его работы: добавление и удаление переменных процесса, изменение значений переменных.
- Сложные операции обработки данных, которые невозможно осуществить стандартными средствами, например работа со справочниками и специфическими карточками Docsvision.
- Взаимодействие с внешними системами, которое не предусмотрено *функциями* или *шлюзами* этих систем.

Функция сценария (`Script`) позволяет выполнить в рамках процесса произвольный код на языке платформы .NET (C# или VB.NET).

Разработка сценариев

Разработка сценариев включает в себя следующие этапы:

- Написание кода *сценария*.
- Отладка кода *сценария*.

- Сохранение готового *сценария* в процессе или подпроцессе.

Код сценария может быть написан непосредственно в диалоговом окне *Редактирование скрипта*, в диалоговом окне *функции сценария*, в любом другом редакторе или при помощи Visual Studio.NET — с последующей вставкой полученного кода в *функцию* процесса.

К сценарию автоматически добавляется коллекция необходимых сборок:

- `Mscorlib.dll`
- `System.dll`
- `System.Core.dll`
- `System.Data.dll`
- `System.ServiceModel.dll`
- `System.Xml.dll`
- `DocsVision.Platform.dll`
- `DocsVision.Platform.CardLib.dll`
- `DocsVision.Platform.ObjectManager.dll`
- `DocsVision.Platform.ObjectModel.dll`
- `DocsVision.ObjectManager.Interop.dll`

  Данная библиотека может быть пропущена. Если в настройках сценария будет выставлен флаг **`Не добавлять ссылку на DocsVision.ObjectManager.Interop.dll`**, будет подключена сборка `DocsVision.Platform.ObjectManager.dll`.

- `DocsVision.Workflow.Interfaces.dll`
- `DocsVision.Workflow.Objects.dll`
- `DocsVision.Workflow.Runtime.dll`
- `DocsVision.Workflow.Functions.dll`
- `DocsVision.Workflow.Gates.dll`
- `DocsVision.BackOffice.ObjectModel.dll`
- `DocsVision.SecurityManager.Interop.dll`
- `DocsVision.HelperAPI.Interop.dll`

В сценариях, приведенных далее, будет использоваться сборка `DocsVision.Platform.ObjectManager.dll` (в параметрах сценария установлен флаг **`Не добавлять ссылку на DocsVision.ObjectManager.Interop.dll`**) — нужно учитывать при указании типа переменных.

При добавлении в карточку бизнес-процесса сценария, автоматически формируется стандартная структура класса сценария — `DVScript`, со стандартной точкой входа — метод `Execute`:

```
class DVScript
{
 public void Execute (ProcessInfo process, PassState passInfo) ①
 {
  try
  {
   ProcessVariable oVar = process.GetVariableByName("Var1"); ②

   oVar.Value = "Scripted"; ③

   process.LogMessage ("Значение переменной изменено"); ④
  }
  catch (Exception ex)
  {
   process.LogMessage("Ошибка выполнения скрипта:" + ex.Message); ⑤
  }
  return;
 }
}
```

① Стандартная функция, которая будет вызвана подсистемой СУБП.

Входные параметры функции:

- `process` — информация о процессе. Содержит коллекции:
  - `Gates` — коллекция шлюзов процесса
  - `Variables` — коллекция переменных процесса
- `passInfo` — информация о текущем проходе

② Пример — получение переменной процесса с именем `Var1`.

③ Присвоение переменной нового значения.

④ Запись в журнал процесса.

⑤ Запись в журнал ошибки исполнения.

Если в сценарии идет обращение к контексту объектов, то вместо его создания из пользовательской сессии достаточно изменить точку входа. Также необходимо подключить пространство имён `DocsVision.Platform.ObjectModel`:

```
public void Execute (ProcessInfo process, PassState passInfo, ObjectContext
objectContext)
```

Помимо приведенных сигнатур метода, можно использовать точку входа, возвращающую результат выполнения:

```
public ExecResultEnum Execute(ProcessInfo process, PassState passInfo)
```

Вариант с `ExecResultEnum` позволяет после завершения сценария оставить функцию активной или ожидающей какого-то события — сценарий будет запущен повторно после наступления события или (в зависимости от `ExecResultEnum`) при следующем проходе.

В ранних версиях Docsvision подобное поведение можно было организовать при помощи вызова в сценарии исключения типа `FunctionResultException`.

Сигнатура приведенного метода не является жесткой — список параметров может формироваться разработчиком *сценария*. Допускается использование параметров определённых типов, указанных ниже.

Тип	Привязка к объекту
`ProcessInfo`	Расширенная объектная модель текущего бизнес-процесса.
`Process`	Объектная модель текущего бизнес-процесса.
`ProcFunction`	Объектная модель над данными текущей выполняемой функции.
`PassState`	Объектная модель текущего прохода исполняемой функции.
`ScriptFunction`	Исполняемая функция "Сценарий".
`IFunction`	Интерфейс `IFunction`# текущей исполняемой функции.
`IFunction2`	Интерфейс `IFunction2`# текущей исполняемой функции.

Тип	Привязка к объекту
`IWorkflowRuntime`	Объект представляющий текущий `WorkflowRuntime`. В первую очередь рекомендуется использовать его при необходимости максимально эффективно запустить новый бизнес-процесс.
`UserSession`	Пользовательская сессия, в контексте которой работает текущий бизнес-процесс. Эта сессия не имеет отношения к сессии, хранящейся в шлюзе к Docsvision, текущего бизнес-процесса.
`ObjectContext`	Проинициализированный контекст для работы с объектной моделью над данными, доступной через библиотеку `DocsVision.Platform.ObjectModel`. В передаваемом `ObjectContext` будут добавлены сервисы, достаточные для работы с объектами `BackOffice`.
`DVGate`	Шлюз к Docsvision текущего бизнес-процесса.
`ExGate`	Шлюз к почте текущего бизнес-процесса.
`FSGate`	Шлюз к файловой системе текущего бизнес-процесса.
`BasicGate`	Шлюз к базовым типам текущего бизнес-процесса.
`AxaptaGate`	Шлюз к `Axapta\Dynamics Ax` текущего бизнес-процесса.
`SPGate`	Шлюз к `SharePoint` текущего бизнес-процесса.
`1CGate`	Шлюз к 1С текущего бизнес-процесса.
Тип стороннего *шлюза*	Любой сторонний шлюз, который реализует интерфейс `IGate` и доступен в *бизнес-процессе*.

Кроме того, предусмотрено использование переменной типа `ProcessVariable` — переменная процесса.

В этом случае алгоритм получения значения переменной следующий:

1. У параметра осуществляется поиск атрибута типа `VariableNameAtrribute`, в значении которого должно быть указано название переменной бизнес-процесса. Переменную требуется передать в выполняемый *сценарий*.

   Если такой атрибут у параметра отсутствует, то в качестве имени переменной принимается имя самого параметра метода.

2. Если переменная с полученным именем не найдена в *бизнес-процессе*, то в качестве значения параметра будет передан `null`.
3. Если значение параметра определить не удалось, то в качестве его значения принимается `DefaultValue` данного параметра. Если `DefaultValue` равно `DBValue.Null`, то оно принимается равным `null`.

Использование нескольких параметров, в том числе переменных процессов, позволяет реализовать следующий сценарий:

```
using System;
using System.Xml; ①

using DocsVision.Workflow.Objects;
using DocsVision.Workflow.Runtime;
using DocsVision.Workflow.Gates;
using DocsVision.Platform.HelperAPI; ②

using DocsVision.Workflow.Functions;
using DocsVision.Platform.ObjectManager; ③

namespace DVScriptHost
{
 class DVScript
 {
  public ExecResultEnum Execute( ④
   ProcessInfo process ⑤
   , PassState passInfo ⑥
   , UserSession session ⑦
   , DVGate gate ⑧
   , [VariableName("Input card")] ProcessVariable inputCard ⑨
   , ProcessVariable outputCard ⑩
  )
  {


   return ExecResultEnum.Done; ⑪
  }
 }
}
```

① Подключение системных библиотек.

② Подключение библиотек СУБП.

③ Подключение дополнительных пространств имён.

④ Стандартная функция, которая будет вызвана подсистемой СУБП.

⑤ Текущий процесс.

⑥ Описание состояния функции в процессе.

⑦ Сессия текущего процесса.

⑧ Шлюз к Docsvision текущего процесса.

⑨ Переменная текущего БП с именем "Input card", т.к. присутствует атрибут `VariableName`.

⑩ Переменная текущего БП с именем "outputCard".

⑪ Выполнение операций, предусмотренных бизнес-логикой.

В приведенном примере метод `Execute` принимает несколько параметров, два из которых (`inputCard`, `outputCard`) будут получены их переменных процесса. Переменные процесса будут получены по схеме приведенной выше.

Сам метод `Execute` будет вызван при запуске *функции*, в которой он определён. После выполнения операций, предусмотренных бизнес-логикой, метод должен вернуть результат своей работы, например, `ExecResultEnum.Done` — "Функция успешно выполнилась". Параметр `passInfo` используется для передачи в метод контекста выполнения (см. описание типа `PassState`) *функции*. При повторном вызове *функции* в метод будут переданы новые данные.

Взаимодействие с объектной моделью Workflow

Основные объекты, которыми оперирует функция сценария, — это внутренние объекты сервиса *СУБП*, специфические для исполняемого процесса. К ним относятся *переменные процесса*, *шлюзы*, журнал процесса и др. Необходимые типы данных собраны в пространстве имён DocsVision.Workflow.Runtime.

Главный способ получения данных бизнес-процесса — это обращение к соответствующим свойствам объекта типа `ProcessInfo`, который содержит данные о текущем процессе:

- Коллекция шлюзов, доступных процессу.
- Коллекция переменных процесса.
- Ссылка на справочники.
- Ссылка на карточку процесса.
- Журнал процесса.

Кроме того, `ProcessInfo` содержит методы:

- Получение шлюза с указанным именем.
- Получение переменной с указанным именем.
- Запись информационного сообщения в журнал процесса.

Экземпляр данного класса выступает в качестве входного параметра *функции сценария*.

Работа со шлюзами

Коллекция *шлюзов* процесса (переменная `ProcessInfo.Gates`) содержит типизированные экземпляры серверных классов *шлюзов*, реализующих общий интерфейс `IGate`. Для получения конкретного *шлюза* из коллекции, необходимо знать его название или тип.

Тип шлюза (уникальный идентификатор) задаётся в процессе его разработки и в дальнейшем не должен изменяться.

Для организации взаимодействия с объектами внешней системы *сценарий* может общаться с этой системой напрямую (при помощи её собственного API), либо с помощью *шлюза*, который может содержать вспомогательные классы для работы с каждым из объектов внешней системы.

Например, при работе со *Справочником сотрудников* разработчик *сценария* может работать напрямую с данными справочника, либо использовать соответствующие объекты *шлюза* к Docsvision. Однако чаще всего объекты *шлюза* служат внутренним целям (для организации работы стандартных функций), и могут не иметь всех свойств и методов по сравнению с API Docsvision. Поэтому по возможности, рекомендуется работать с внешней системой через API Docsvision.

Пример работы с объектами из сценария с использованием API Docsvision:

```
public void Execute (ProcessInfo process, PassState passInfo)
{
 DVGate dvGate = (DVGate)process.Gates[DVGate.GateID]; ①

 UserSession userSession = dvGate.UserSession; ②

 CardData data = userSession.CardManager.GetCardData(new Guid("00000000-0000-0000-0000-
000000000000")); ③

 // ...
```

```
}
```

① Получение шлюза к Docsvision.

② Получение сессии для доступа к API.

③ Получение данных карточки с идентификатором 00000000-0000-0000-0000-000000000000.

Сессия может быть получена из самого сценария:

```
public void Execute (ProcessInfo process, PassState passInfo)
{
 UserSession userSession = process.Session; ①

   // ...
}
```

① Получение сессии процесса.

Работа с переменными

Переменные процесса могут быть получены из переменной `ProcessInfo.Variables`, которая содержит объекты типа `ProcessVariable`.

Данный тип предоставляет следующие данные:

- Уникальный идентификатор переменной, её название, тип и значение.
- Отображаемое значение объекта, хранящегося в переменной.
- Идентификатор шлюза, которому принадлежит переменная.
- Коллекция строчных значений (для переменных типа "Перечисление").

При работе с переменными простых типов ("Строка", "Целое", "Дробное" и др.), значение переменной может быть получено непосредственно из свойства `ProcessVariable.Value`. Для прочих типов (переменные шлюзов) в качестве значения переменной будет возвращена ссылка на соответствующий типизированный объект шлюза:

```
ProcessVariable varStr = process.GetVariableByName("ПростоСтрока");
string strValue = (string)varStr.Value; ①

ProcessVariable varDoc = process.GetVariableByName("Карточка");
DVCard card = (DVCard)varDoc.Value; ②
```

① Получение переменной простого типа (тип "Строка").

② Получение переменной, тип которой предоставлен шлюзом.

Присвоение значений переменным выполняется аналогично: переменные простых типов получают непосредственное значение, тогда как шлюзовые переменные в качестве нового значения могут получать только объекты шлюза:

```
ProcessVariable varStr = process.GetVariableByName("ПростоСтрока");
varStr.Value = "Новое значение"; ①

DVGate dvGate = (DVGate)process.Gates[DVGate.GateID]; ②

ProcessVariable varDoc = process.GetVariableByName("Карточка"); ③
varDoc.Value = dvGate.GetVariable((int)DVVariableType.DOCUMENT, "ID_карточки");
```

① Получение переменной простого типа (тип "Строка") и присвоение ей значения.

② Получение шлюза к Docsvision.

③ Получение переменной, тип которой предоставлен шлюзом, и присвоение ей значения.

Отладка сценариев

Отладка *сценариев* представляет собой процесс тестирования разработанного кода *сценария* с последующим включением его в разработанный *бизнес-процесс*. Отладку можно условно разделить на две стадии:

- Отладка на стадии компиляции.
- Отладка на стадии исполнения.

При обработке экземпляра *СУБП* исполнение всех *функций сценария* проходит две фазы: компиляция и непосредственное исполнение. Компиляция производится при первом обращении к *функции* и осуществляется при помощи стандартного компилятора, входящего в состав Microsoft .NET Framework.

Если компиляция кода *сценария* завершилась с одной (или несколькими) ошибками, то дальнейшее выполнение *бизнес-процесса* считается невозможным, и работа *бизнес-процесса* завершается с ошибкой. Подробное описание возникших ошибок можно найти в журнале соответствующего экземпляра *бизнес-процесса* (они будут помечены именем функции сценария, компиляция которой не удалась) или в общем журнале работы *СУБП*, если такой журнал

ведется.

Проверить компиляцию сценария можно непосредственно из окна ввода текста сценария, нажатием кнопки **Компилировать**.

В случае, если компиляция завершена успешно, исполняется *сценарий*. Исполнение производится последовательно, если структурой кода не предусмотрено иное. После завершения исполнения всего кода *сценария* сервис *СУБП* переходит к обработке следующей функции *бизнес-процесса*.

Для упрощения отладки *бизнес-процесса* на этапе исполнения можно использовать механизм сообщений. *Сценарий* может выводить произвольные сообщения в журнал *бизнес-процесса* при помощи функции `LogMessage` класса `ProcessInfo`. Сообщения могут носить декларативный характер (например, сообщать о корректном выполнении тех или иных участков кода) или информировать о состоянии данных (например, значениях переменных).

Запись сообщений, необязательных для работы *сценария*, после тестирования предпочтительно удалить из кода — это значительно упростит поиск информации в журнале в будущем.

Сценарии могут использовать для организации корректной работы стандартный механизм исключений. Исключения могут возникать в результате ошибок исполнения *сценария*, либо принудительно вызываться в коде для выполнения необходимых действий. Все возможные исключения *сценарий* должен обрабатывать самостоятельно и в зависимости от этого реализовывать тот или иной вариант работы (например, записать информацию об исключении в журнал или выполнить другой участок кода).

Если исключение не было обработано в *сценарии*, то оно обрабатывается сервисом *СУБП*, и *сценарий* считается завершённым с непредвиденной ошибкой. Обработка процесса в этом случае прерывается на *функцию*, вызвавшей исключение.

Сценарий считается завершённым успешно (исполнение процесса продолжается), если:

- Компиляция сценария завершена без ошибок.
- Все возникшие во время исполнения исключения обработаны самим сценарием.

Сценарий считается завершённым с ошибкой (исполнение бизнес-процесса прерывается), если:

- Возникла одна или более ошибок при компиляции кода сценария (синтаксические, семантические ошибки или не удалось подключить дополнительные библиотеки).
- Были необработанные исключения.

Сохранение сценария в виде сборки

Начиная с версии Docsvision 4.5 в *функцию* "Сценарий" добавлена возможность компиляции кода для получения готовой сборки (DLL), которую можно повторно использовать в бизнес-процессах.

Последовательность получения сборки следующая:

1. Создайте новый бизнес-процесс или используйте имеющийся.
2. Добавьте в *бизнес-процесс* функцию "Сценарий".

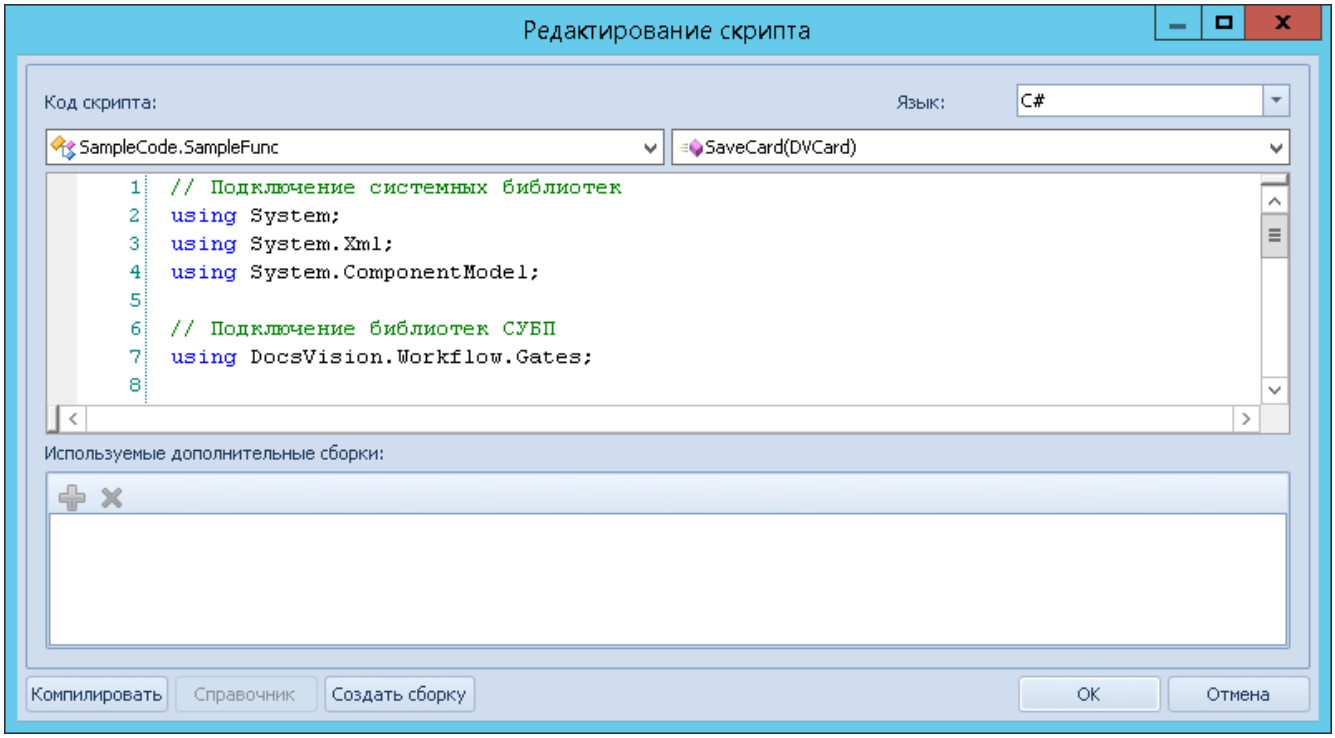


Рисунок 37. Компиляция скрипта из Docsvision

3. В *сценарии* напишите скрипт, который будет содержать необходимую функциональность и будет соответствовать следующим требованиям:
   a. Должен содержать минимум один публичный (`public`) класс.
   b. Должны присутствовать публичные статические (`public static`) методы. В рамках одного *сценария* возможно наличие произвольного числа таких методов — каждый из них будет выступать как отдельная *функция* в "Универсальной функции".

Аргументы данного метода, так же как и возвращаемое значение (если предусмотрено) должны иметь тип, реализованный шлюзами Workflow (включая шлюз к базовым типам).

Ниже приведён пример скрипта, в котором доступен один метод `SaveCard`, подразумевающий выполнение действий над переданной карточкой `card`.

```
using System;
using System.Xml;
using System.ComponentModel; ①

using DocsVision.Workflow.Gates; ②

namespace SampleCode
{
 public class SampleFunc
 {
  public static void SaveCard(DVCard card)
  {
③
  }
 }
}
```

① Подключение системных библиотек.

② Подключение библиотек СУБП.

③ Выполнение действие с объектом `card`.

4. Выполните проверку корректности скрипта (кнопка **Компилировать**).
5. Если ошибок при компиляции не обнаружено, нажмите кнопку **Создать сборку**.

   Система предложит выбрать место для сохранения библиотеки. Нужно учитывать, что итоговая сборка должна быть доступна сервису Workflow. Для этого её достаточно поместить в каталог приложения Workflow (исполняемый файл `ExecLogic.exe`), либо зарегистрировать в GAC.

6. Выполните сохранение, нажмите **OK**.

   Ссылка на сборку будет автоматически добавлена в список сборок Workflow (справочника *Системных настроек*). Это позволит в дальнейшем использовать

данный *сценарий* в других процессах в виде *функции*, без необходимости копирования его исходного кода.

Чтобы обратиться к методам полученной сборки:

1. Создайте новый бизнес-процесс или используйте имеющийся.
2. Добавьте в процесс *функцию* "Универсальная функция".
3. Выберите *Тип*.

   При выборе ориентируйтесь на название сборки, созданной ранее. В качестве типа может быть выбран один из классов, реализованных в скрипте.

4. Выберите *Функцию*.

   Для выбора доступны статические публичные методы класса, выбранного ранее.

5. При необходимости укажите значения *Параметров* функции.

   Параметры функции соответствую параметрам выбранного метода.

6. Сохраните настройки нажатием кнопки **ОК**.

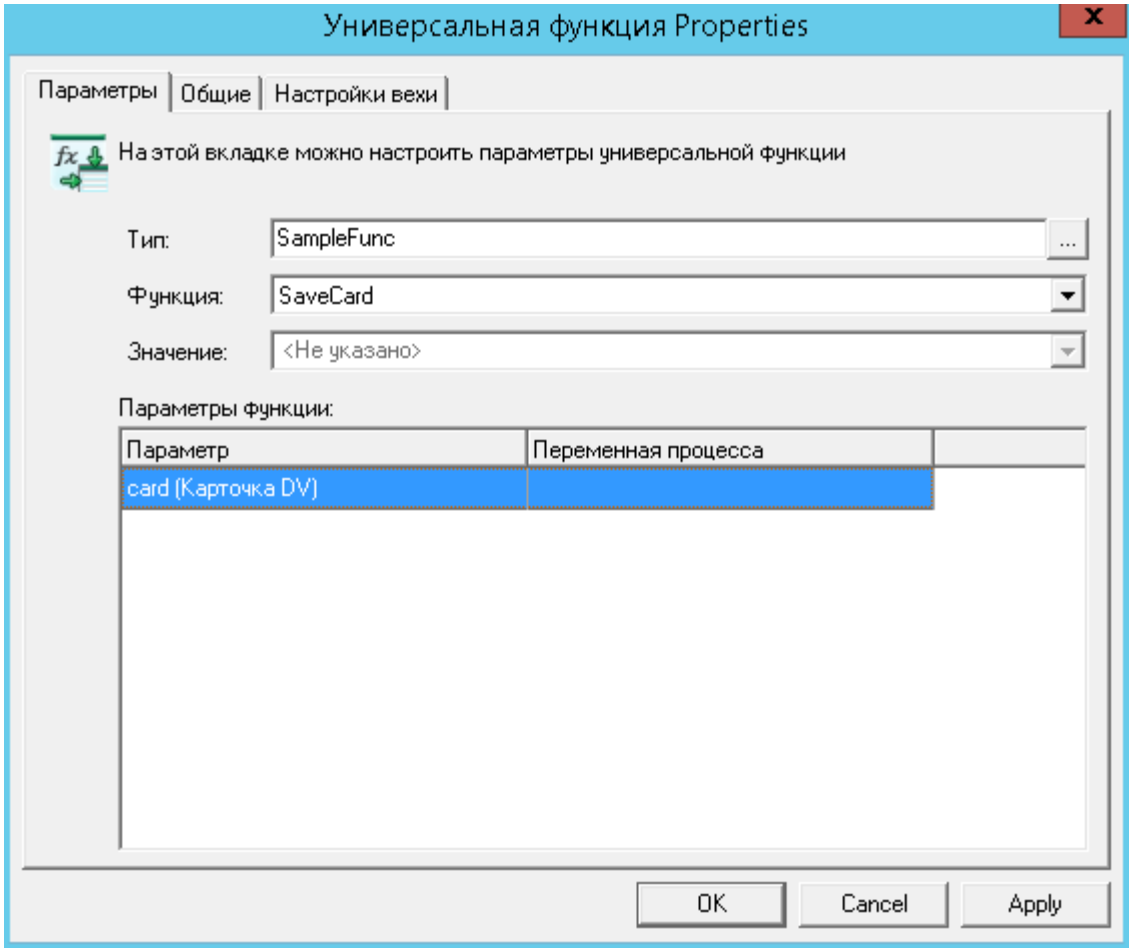


Рисунок 38. Созданный скрипт в области "Параметры функции"

Чтобы реализованные классы и методы имели локализованное название, при отображении в окне параметров "Универсальной функции", пометьте эти сущности специальным атрибутом `ResName`.

```
using System;
using System.Xml;
using System.ComponentModel; ①

using DocsVision.Workflow.Gates; ②

namespace SampleCode
{
 [ResName("Пример функции")]
 public class SampleFunc
 {
  [ResName("Пример метода функции")]
  public static void SaveCard([ResName("Переменная")] DVCard card)
  {
③
```

```
  }
 }

 // internal sealed class ResNameAttribute : DescriptionAttribute
}
```

① Подключение системных библиотек.

② Подключение библиотек СУБП.

③ Выполнение действий с объектом card.

Класс `ResNameAttribute` *требуется реализовать самостоятельно по следующему примеру:*

```
[System.AttributeUsage(System.AttributeTargets.All)]
internal sealed class ResNameAttribute : DescriptionAttribute
{
 public ResNameAttribute(string name) : base(name) { }
}
```

Данный класс может быть включён в сборку с получаемым *сценарием*, либо вынесен в отдельную библиотеку.

После получения библиотеки с локализованными названиями, окно настройки "Универсальной функции" будет как показано ниже.

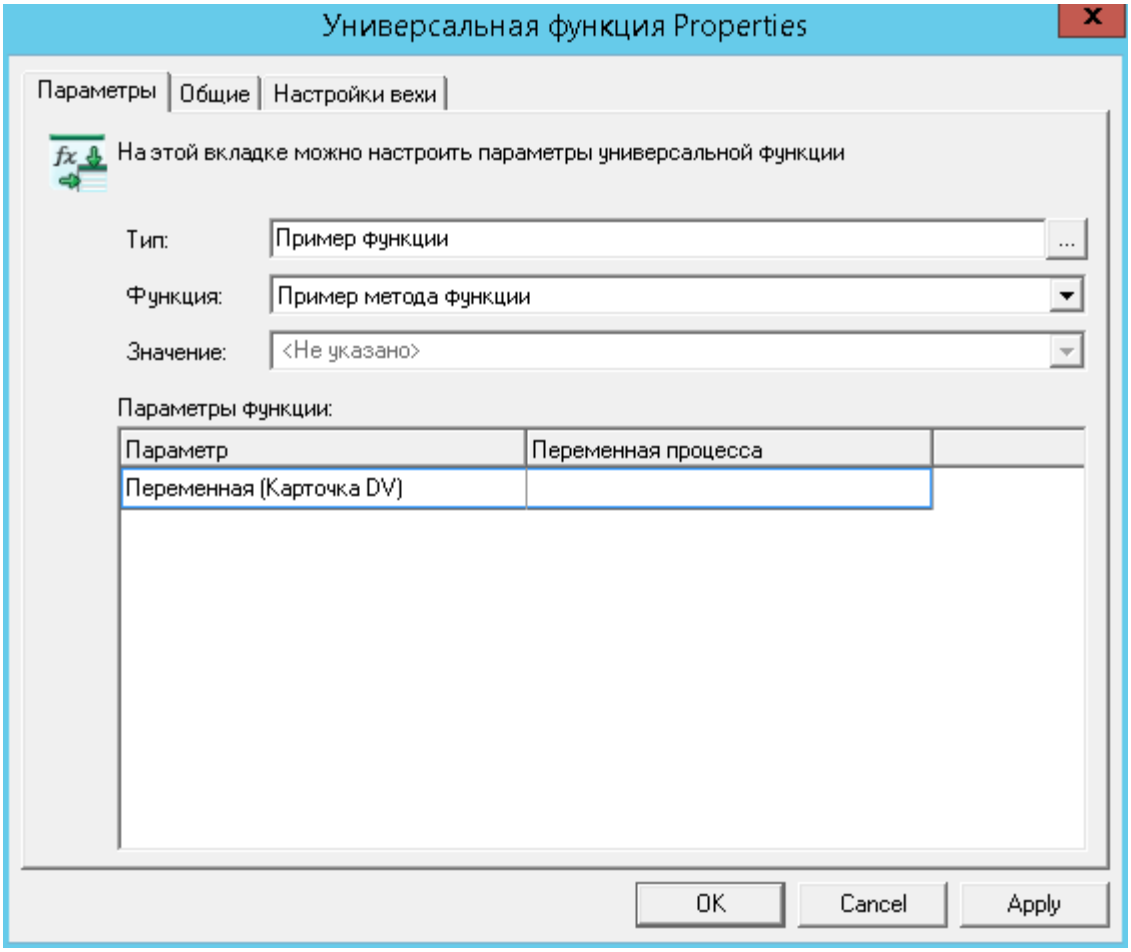


Рисунок 39. Параметры универсальной функции

Помимо сборок, созданных на базе *сценариев*, возможно подключение в таком качестве любых произвольных сборок, отвечающих тем же требованиям. Данные библиотеки должны быть самостоятельно зарегистрированы в качестве модуля Workflow. Регистрация осуществляется в категории "Настройки Workflow/Сборки" справочника "Системные настройки" (папка "Конструкторы и справочники") приложения **Docsvision Windows-клиент**. Сборка должна быть доступна сервису Workflow (см. выше).

Список COM интерфейсов

IObjectUI

Интерфейс, описывающий свойства клиентских объектов *СУБП* (*функций*, *шлюзов*).

Свойства и методы	Тип	Параметры	Описание
Data	string		Задаёт или возвращает данные объекта (функции, шлюза).
Icon	IPictureDisp		Возвращает иконку объекта.
TypeName	string		Возвращает название данного типа объектов.
ID	string		Возвращает идентификатор объекта.

IDVObjectUI

Данный интерфейс реализуется всеми объектами, использующими Docsvision. Предоставляет доступ к сессии.

Свойства и методы	Тип	Параметры	Описание
UserSession	IDispatch		Задаёт или возвращает сессию.
Host	IDispatch		Задаёт или возвращает хост Docsvision.

IObjectLocalize

Дополнительный интерфейс для объектов, поддерживающих локализацию.

Свойства и методы	Тип	Параметры	Описание
`GetLocaleID`	`long`		Получает идентификатор текущей локализации.
`SetLocaleID`		`(long) LocaleID`	Устанавливает идентификатор текущей локализации.

IGateInit

Интерфейс клиентского компонента *шлюза*. Определяет возможность инициализации компонента шлюза с передачей в него начальных данных.

Свойства и методы	Тип	Параметры	Описание
`Initialize`		`(string) Data` — данные шлюза	Инициализирует шлюз, передавая данные, сохранённые в настройках шлюза в справочнике шлюза, как правило, xml -описание шлюза.

IGateUI

Интерфейс клиентского компонента *шлюза*. Реализует свойства и методы для работы с доступными типами переменных.

Свойства и методы	Тип	Параметры	Описание
`GetObjectTypes`	`IobjectType Collection`		Коллекция описаний типов объектов, поддерживаемых шлюзом.
`GetMethods`	`Imethod Collection`		Коллекция описаний методов, поддерживаемых шлюзом.
`SelectObject`	`bool`	`(long) hWnd` — идентификатор родительского окна `(long) TypeID` — идентификатор типа объекта `(string) ObjectID` — выходной параметр, идентификатор и значение объекта в шлюзе	Выбор объекта в шлюзе. Метод позволяет выбрать и вернуть идентификатор объекта указанного типа. Измененное значение возвращается в параметре `ObjectID`. Метод возвращает `true`, если значение было выбрано, иначе `false`.

Свойства и методы	Тип	Параметры	Описание
`ShowObject`		`(long) hWnd` — идентификатор родительского окна `(long) TypeID` — идентификатор типа объекта `(string) ObjectID` — выходной параметр, идентификатор\значение объекта в шлюзе	Отображение объекта в шлюзе. Метод позволяет активизировать объект по его идентификатору и типу. Может быть поддержано не для всех типов объектов.
`GetObjectDisplayValue`	`string`	`(long) TypeID` — идентификатор типа объекта `(string) ObjectID` — выходной параметр, идентификатор и значение объекта в шлюзе	Возвращает отображаемое значение объекта по его типу и идентификатору.

Свойства и методы	Тип	Параметры	Описание
`GetObjectReference`	`string`	`(long) hWnd` — идентификатор родительского окна `(long) TypeID` — идентификатор типа объекта `(string) ObjectID` — выходной параметр, идентификатор\значение объекта в шлюзе	Возвращает строку-ссылку на объект в шлюзе.
`GetQuickSearchValues`	`IDispatch`	`(long) TypeID` — идентификатор типа объекта `(string) ObjectID` — выходной параметр, идентификатор и значение объекта в шлюзе	Возвращает коллекцию идентификаторов объектов, найденных по строке, заданной параметром `SearchFor`.

IFunctionUI

Интерфейс клиентского компонента *шлюза*. Реализует методы, вызываемые карточкой бизнес-процесса при работе с *функцией*.

Свойства и методы	Тип	Параметры	Описание
`GetRequiredGates`	`string`		Возвращает список идентификаторов необходимых шлюзов (разделенных запятой).
`AllowExecute`	`string`		Выполняет проверку на наличие всех необходимых для запуска функции данных. Возвращает `true`, если функция готова к выполнению, иначе `false`.

IExportableFunctionUI

Клиентский интерфейс для *функций*, обладающих собственным механизмом экспорта/импорта в XML.

Свойства и методы	Тип	Параметры	Описание
`Export`	`string`		Вызывается при экспорте функции. Возвращает данные для экспорта в виде строки.
`Import`		`(string) Data` — данные функции	Вызывается при импорте функции.

IStateFunctionUI

Клиентский интерфейс для *функций*, обладающих собственным набором

состояний.

Свойства и методы	Тип	Параметры	Описание
`CurrentIcon`	`IPictureDisp`		Возвращает иконку для текущего состояния.
`CurrentName`	`string`		Возвращает локализованное название текущего состояния.
`CurrentColor`	`long`		Возвращает цвет иконки для текущего состояния (в формате OLE Color).
`CurrentFunction`	`IDispatch`		Задаёт или возвращает данные о состоянии функции.

Подписка на события шлюза

Workflow предоставляет разработчикам механизм формирования подписок на определённые события шлюза, к примеру, если функция должна продолжить работу, только после разблокировки определённой карточки.

Механизм формирование подписок и сообщений

Шлюз, чей серверный компонент реализует специальный интерфейс `IGateMessageSource` (сборка `DocsVision.Workflow.Interfaces.dll`), предоставляет канал для формирования подписок. Доступ к нему может быть получен через публичное свойство шлюза — `SubscriptionChannel` (тип `ISubscriptionChannel`), к примеру так:

```
DVGate dvGate = (DVGate)process.Gates[DVGate.GateID]; ①
```

```
SubscriptionChannel subscriptionChannel = dvGate.SubscriptionChannel; ②
```

① Получение шлюза к Docsvision.

② Получение канала подписок.

После получения канала подписок, разработчик может выполнить подписку на определённое событие, например, подписаться на событие разблокировки определённой карточки Docsvision:

```
subscriptionChannel.Subscribe((int)DocsVisionSubscriptionType.ObjectUnlock, process.ID,
passInfo.FunctionID, cardId); ①
```

① `cardId` — идентификатор карточки, у которой ожидается разблокировка.

Метод `Subscribe` принимает (по порядку передачи параметров):

1. Тип ожидаемого события шлюза. Список событий индивидуален для каждого шлюза (см. далее по тексту), который в дальнейшем отвечает за обработку подписок, а также генерирует сообщения для функций, осуществивших подписку.
2. Идентификатор процесса (бизнес-процесса), который подписывается на события. Этот процесс должен получить сообщение о наступлении события шлюза.
3. Идентификатор функции, которая выполняет подписку на событие.

Также может быть передан (как в примере) дополнительный параметр (с типом `Guid` или `String`), тип и значение которого зависит от типа ожидаемого события шлюза (см. далее).

Если функция выполняет подписку на событие, то об этом необходимо проинформировать Workflow, вернув значение `ExecResultEnum.WaitForMessage`.

После подписки на событие, функция переходит в режиме ожидания и активируется при появлении сообщения соответствующего типа.

В общем случае, сценарий подписки и оповещения выглядит следующим образом:

1. *Менеджер процессов* (часть `ExecLogic`) вызывает метод `Execute` текущей функции (к примеру, *Функции N*), которая должна быть запущена.
2. *Функции N*, если это необходимо, оформляет подписку на событие в определённом шлюзе (к примеру, *Шлюз T*), посредством вызова метода

`Subscribe`, и возвращает значение `ExecResultEnum.WaitForMessage`.

3. *Шлюз T*, проверяет наличие подписок в своем хранилище (например, в базе данных) и организует проверку наступления соответствующего события.
4. При наступлении события *Шлюз T* добавляет сообщение для *Функции N*, через вызов метода `SendMessage`. Сообщение добавляется в очередь сообщений конкретного процесса.
5. *Менеджер процессов* проверяет очередь сообщений и, в случае наличия сообщения для *Функции N*, вызывает метод `Execute` *Функции N*.

Схематично данный сценарий можно представить следующим образом:

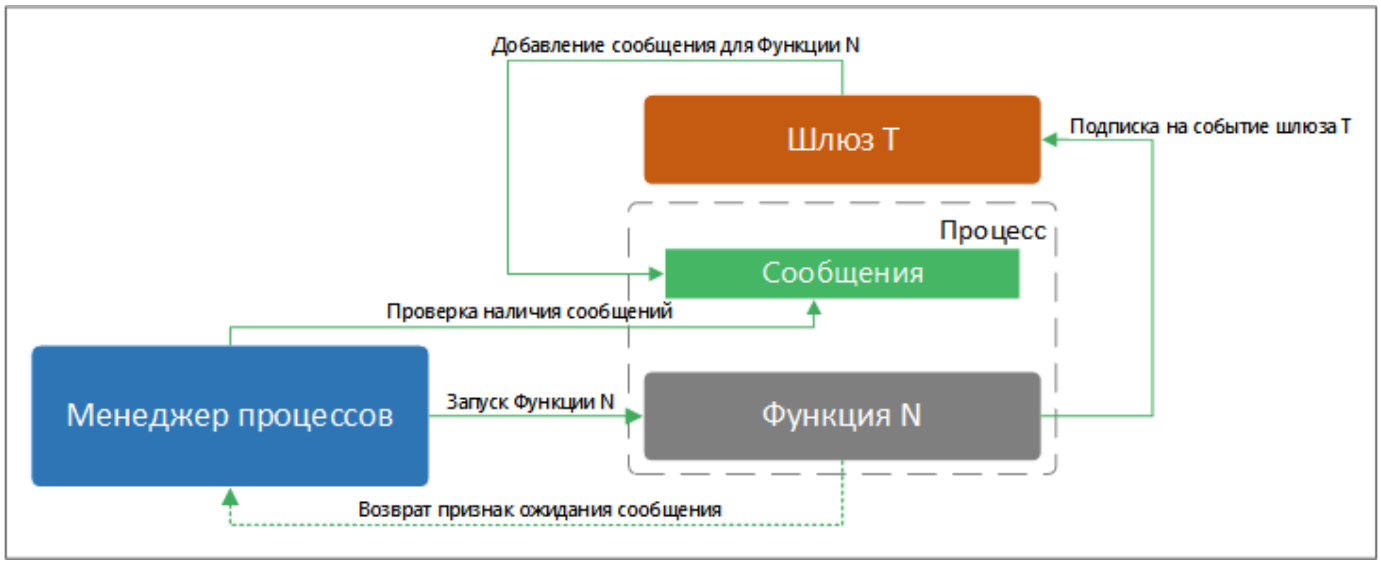


Рисунок 40. Сценарий подписки на событие шлюза и формирования сообщения о его наступлении

События шлюза к Docsvision

Шлюз к Docsvision разрешает подписку на следующие события (см. описание `DocsVisionSubscriptionType`):

- `CardChange` — изменение заданной карточки. При вызове `Subscribe` необходимо передать (четвертый параметр) идентификатор отслеживаемой карточки.
- `AnyCardChange` — изменение любой карточки заданного типа. При вызове `Subscribe` необходимо передать (четвертый параметр) идентификатор типа отслеживаемых карточек.
- `ObjectUnlock` — разблокировка заданной карточки. При вызове `Subscribe` необходимо передать (четвертый параметр) идентификатор соответствующей карточки.

- `ObjectLock` — блокировка заданной карточки. При вызове `Subscribe` необходимо передать (четвертый параметр) идентификатор соответствующей карточки.

События шлюза к файловой системе

Шлюз к файловой системе обеспечивает возможность подписки на события (см. описание `FSSubscriptionType`):

- `NewFileInFolder` — добавление файла в указанную папку. При вызове Subscribe необходимо передать (четвертый параметр) сериализованный (XML-сериализация) объект типа `FSSubscriptionInfo` — параметры подписки.
- `FileChange` — изменение указанного файла. При вызове Subscribe необходимо передать (четвертый параметр) сериализованный объект типа `FSSubscriptionInfo`.

События шлюза к почтовой системе

Шлюз к почтовой системе обеспечивает возможность подписки на события (см. описание `MailSubscriptionType`):

- NewMessage — поступление нового сообщения. При вызове Subscribe необходимо передать (четвертый параметр) сериализованный объект типа `MailFilter` — фильтр для новых сообщений.
- `MessageWithSubstringInSubject` — поступление нового сообщения с заданным тестом в заголовке письма. При вызове `Subscribe` необходимо передать (четвертый параметр) текст, который должен содержаться в заголовке ожидаемого письма.

Подписка на события шлюзов функциями приложения Workflow

Некоторые функции приложения *Workflow* используют подписку в своей работе. Ниже приведён полный список функций приложения *Workflow*, а также события, на которые они могут быть подписаны. Конкретный тип события, на которое подписывается функция, может зависеть от её конкретных настроек.

Функция	**События, на которые возможна подписка**
Базовые функции:	
Начальная функция	Не подписывается.
Конечная функция	Не подписывается.

Функция	События, на которые возможна подписка
Объединение И	Не подписывается, но может ожидать завершения другой функции.
Объединение Или	Не подписывается.
Разветвление	Не подписывается.
Подпроцесс	Не подписывается, но ожидает завершения другого подпроцесса.
Условие	Не подписывается.
Счетчик	Не подписывается.
Обработка коллекции	Не подписывается.
Удаление объекта	Не подписывается, но ожидает завершения удаления объекта.
Сценарий	Может подписываться на любые события шлюзов, если это реализовано разработчиком.
Обмен данными между переменными	Не подписывается.
Расписание	Подписывается на событие от таймера (Шлюз к простым типам).
Универсальный обмен данными	Не подписывается, но ожидает завершения обмена.
Универсальная функция	Может подписываться на любые события шлюзов, если это реализовано разработчиком выбранной функции.
Обработчик ошибок	Не подписывается.
Функции шлюза к Docsvision:	
Мониторинг Docsvision	Может подписываться на изменение определённой карточки, либо карточек определённого типа.
Задание	Может подписываться на поступление письма (Шлюз к почтовой системе), на событие от таймера (Шлюз к простым типам), на изменение или разблокировку карточек.

Функция	События, на которые возможна подписка
Управление заданием	Не подписывается, но остается активным, если карточка заблокирована.
Ярлык	Не подписывается.
Функция рассылки согласований	Может подписываться на разблокировку карточки, поступление письма (Шлюз к почтовой системе), а также ожидать завершения подпроцесса.
Функция рассылки заданий	Может подписываться на разблокировку карточки, а также ожидать завершения подпроцесса.
Задание 5	Может подписываться на поступление письма (Шлюз к почтовой системе), на событие от таймера (Шлюз к простым типам), на изменение или разблокировку карточек.
Параллельное задание	Может подписываться на разблокировку родительской и связанных карточек, а также ожидать завершения подпроцесса.
Последовательное задание	Может подписываться на разблокировку родительской и связанных карточек, а также ожидать завершения подпроцесса.
Функции шлюза к файловой системе:	
Мониторинг файловой системы	Может подписываться на появление нового файла и изменение файла.
Функции шлюза к почтовой системе:	
Мониторинг сообщений	Может подписываться на появление нового сообщения (и сообщения с определённой темой).
Сообщения задания 5	Может подписываться на появление нового сообщения, а также на изменение и разблокировку конкретного задания или всех карточек типа задание.

Внешние хранилища

В данном разделе рассмотрены примеры разработки компонентов для настройки внешних хранилищ:

- Провайдера к внешнему хранилищу — позволяет добавить новый тип внешних хранилищ.
- Новый тип правил помещения в хранилище.

Разработка провайдера к внешнему хранилищу

Система Docsvision предоставляет возможность хранить бинарные данные загружаемых файлов в основной БД Docsvision, а также во внешних хранилищах:

- Стандартных: файловая система, внешняя БД или FileStream.
- Собственных, для взаимодействия с которыми необходимо разработать специальные провайдеры.

Для использования собственного внешнего хранилища необходима опция лицензии *Docsvision модуль интеграции с произвольными файловыми хранилищами*.

Компонент провайдера используется только на сервере Docsvision.

Основной класс провайдера к хранилищу

Для возможности работы с собственными типами хранилищ разработчик должен реализовать специальный компонент — провайдер к внешнему хранилищу (далее — просто провайдер). Провайдер представляет собой библиотеку с публичным классом, в котором реализуются определённые программные интерфейсы. Набор интерфейсов определяется возможностями внешнего хранилища.

В простейшем случае, провайдер должен реализовывать два интерфейса: `IBinaryStorage` и `IExtensionInitialize`.

```
public class ExternalStorage : IBinaryStorage, IExtensionInitialize
```

Интерфейс `IBinaryStorage` определяет основные параметры хранилища и методы провайдера: идентификатор хранилища, поддерживающий по умолчанию разделы, методы создание, чтения, удаления файлов и др. См. описание интерфейса `IBinaryStorage`. Реализация методов интерфейса `IBinaryStorage` определяется способом работы с внешним хранилищем.

Интерфейс `IExtensionInitialize` определяет метод инициализации провайдера

(`Initialize`), принимающий в сериализованном виде настройки хранилища. Настройки хранилища устанавливаются с помощью графического интерфейса настройки (рассмотрено далее).

Следует обратить внимание, что сервер Docsvision может в течении пользовательской сессии многократно создавать экземпляры класса провайдера и инициализировать его, что может привести к росту потребления ресурсов, если при инициализации создаются дополнительные объекты. Данную особенность следует учитывать при создании класса провайдера.

Для более эффективной работы сервера Docsvision с хранилищем, провайдер может поддерживать интерфейс `IStreamedBinaryStorage`, позволяющий серверу работать напрямую с потоками чтения/записи к хранилищу. Также может быть поддержан интерфейс `IEnumerableBinaryStorage`, если провайдер может предоставить идентификаторы всех файлов, хранящихся в определённом разделе хранилища.

```
public class ExternalStorage : IBinaryStorage,
                               IExtensionInitialize,
                               IStreamedBinaryStorage,
                               IEnumerableBinaryStorage
```

Класс провайдера должен быть помечен атрибутом `DisplayName`, содержащим название хранилища.

```
[DisplayName("External storage name")]
public class ExternalStorage : IBinaryStorage, IExtensionInitialize
```

Для использования локализованного отображаемого названия можно добавить дополнительный класс, возвращающий название из ресурсов:

```
class LocalizedDisplayNameAttribute : DisplayNameAttribute
{
  private readonly string resourceName;
  public LocalizedDisplayNameAttribute(string resourceName)
     : base()
  {
     this.resourceName = resourceName;
  }

  public override string DisplayName
```

```
  {
     get
     {
        return Resource.ResourceManager.GetString(this.resourceName);
     }
  }
}
```

Тогда класс провайдера необходимо пометить атрибутом `LocalizedDisplayName` с указанием ключа локализованного ресурса:

```
[LocalizedDisplayName("ExternalStorageName")]
public class ExternalStorage : IBinaryStorage, IExtensionInitialize
```

Чтобы предоставить возможность настройки провайдера с помощью Консоли настройки Docsvision, должна быть разработана графическая форма. Информация о данном компоненте приведена в разделе Графический интерфейс настройки. Название компонента графической формы должно быть указано в атрибуте `Editor` класса провайдера:

```
[DisplayNameAttribute("ExternalStorageName")]
[Editor("ExternalStorageProvider.ProviderConfigurationPropertyControl,
ExternalStorageProvider, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=e955275a11279434, processorArchitecture=MSIL",
        "DocsVision.Platform.WinForms.Controls.IExtensionPropertiesControl,
DocsVision.Platform.WinForms"
)]
public class ExternalStorage : IBinaryStorage, IExtensionInitialize
```

В атрибуте `Editor` указываются:

1. Полное название класса, в котором реализована графическая форма.
2. Название и сборка класса `IExtensionPropertiesControl`, как в примере.

Если возможность настройки не требуется, то атрибут `Editor` не указывается.

Графический интерфейс настройки

Графический интерфейс настройки провайдера к внешнему хранилищу представляет собой компонент типа `System.Windows.Forms.UserControl` с реализованным программным интерфейсом `IExtensionPropertiesControl`.

```
public partial class ProviderConfigurationPropertyControl : UserControl,
IExtensionPropertiesControl
```

К реализации графического компонента настройки не предъявляется специальных требований (за исключением реализации интерфейса `IExtensionPropertiesControl`). Программист может реализовать любой графический интерфейс, необходимый для настройки провайдера.

Реализованный компонент добавляется в блок свойства панели настройки хранилища:

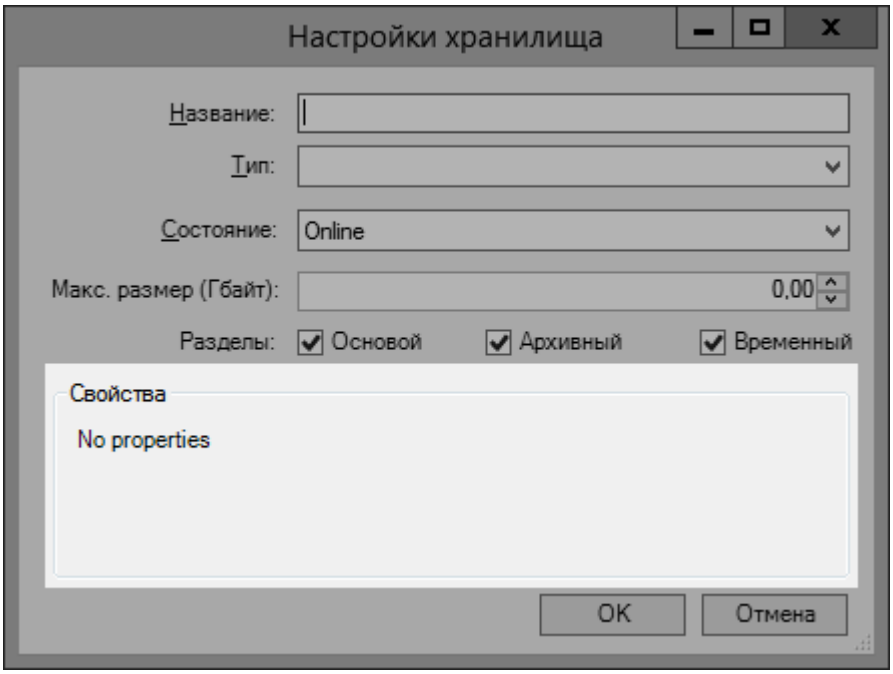


Рисунок 41. Блок настройки провайдера к хранилищу на панели настройки хранилища

Настройки могут включать любые элементы, необходимые для настройки подключения к хранилищу или работы с ним.

Программисту необходимо обеспечить:

1. Сериализацию значений настроек в строку (любым способом) в методе `IExtensionPropertiesControl.Save`. Сериализованное значение должно быть сохранено в переменную `Settings`.
2. Десериализовать настройки из строки в методе `Initialize`. При необходимости присвоить значения элементам графической формы.

Подключение провайдера к внешнему хранилищу к системе Docsvision

1. Сборки с классом провайдера и классом графического интерфейса настройки

должны быть подписаны. Допускается реализация класса провайдера и класса графического интерфейса в одной сборке.

2. Сборки должны быть загружены в GAC на сервере Docsvision.

Чтобы добавить новый тип хранилища в Docsvision (требуется опция лицензии Docsvision модуль интеграции с произвольными файловыми хранилищами):

3. Откройте страницу настройки внешних хранилищ в Консоли настройки Docsvision.
4. Нажмите кнопку добавления нового типа хранилища и выберите в списке *Тип* пункт ***Добавить из сборки***.
5. Выберите сборку с классом провайдера (в папке GAC). В список *Тип* будет добавлен новый пункт, соответствующий названию хранилища, указанному в `DisplayNameAttribute` класса хранилища.
6. Настройте хранилище, группу и правила помещения стандартным образом.

Пример провайдера к внешнему хранилищу Google Drive

В данном пункте рассмотрен пример создания провайдера к внешнему хранилищу Google Drive для хранения бинарных данных файлов Docsvision.

Данный пример не является готовым решением для использования Google Drive в качестве внешнего хранилища, а только демонстрирует способ создания провайдера к нему.

Облачное хранилище Google Drive предоставляет возможность и API для работы с файлами из собственного приложения. Инструкция по работе с Google Drive API приведена на этой странице.

Для работы с Google Drive в данном примере используется предоставляемая реализация обертки для работы с Google Drive API в .NET (NuGet пакет).

Исходный код примера доступен по /dv6/programmer/dv6/_attachments/gDriveStorage.zip[ссылке].

Пример содержит два проекта:

1. `GoogleDriveStorageProvider` — проект компонента провайдера к файловому хранилищу с классами:
   a. `GoogleDriveStorage` — класс провайдера с реализацией интерфейсов `IBinaryStorage`, `IExtensionInitialize`, `IStreamedBinaryStorage`, `IEnumerableBinaryStorage`.

b. `GoogleDriveProviderPropertyControl` — класс графического интерфейса настройки провайдера.

c. `GoogleDriveProxy` — класс-посредник для работы с Google Drive API.

d. `ProviderSettings` — вспомогательный класс с методами сериализации и десериализации настроек провайдера.

e. `LogWriter` — вспомогательный класс журналирования работы провайдера.

f. `LocalizedDisplayNameAttribute` — класс для получения локализованных ресурсов в атрибутах класса.

2. `AskToken` — проект консольного приложения для получения токена для работы с Google Drive.

Требования примера

На компьютере программиста должны быть установлены:

1. Microsoft Visual Studio 2015 или выше.
2. Microsoft .NET Framework 4.6.1.

Пример совместим с системой Docsvision версии 5.5.1.

Последовательность работы с примером

1. Распакуйте /dv6/programmer/dv6/_attachments/gDriveStorage.zip[архив] примера на локальный диск.
2. Получите файл с конфигурацией клиента (`credentials.json`) по инструкции в шаге 1 на странице. Потребуется авторизоваться в Google.
3. Загрузите файл `credentials.json` в папку `GoogleDriveStorageProvider\AskToken\bin\Debug\` распакованного архива примера.
4. Включите в Visual Studio настройку Allow Nuget to download missing package для автоматической загрузки необходимых пакетов библиотеки Google Drive API.
5. Повторно подключите зависимости `DocsVision.Platform.StorageServer` и `DocsVision.Platform.WinForms` в проекте `GoogleDriveStorageProvider`. К проекту должны быть подключены компоненты, используемые на сервере Docsvision.
6. Соберите проект `GoogleDriveStorageProvider`.
7. Соберите и запустите проект `AskToken`. В процессе работы программы `AskToken` будет вызван код из класса `GoogleDriveProxy` (проект `GoogleDriveStorageProvider`):

```
using (var stream =
                new FileStream(credentialsFile, FileMode.Open, FileAccess.Read))
            {
                credential = GoogleWebAuthorizationBroker.AuthorizeAsync(
                    GoogleClientSecrets.Load(stream).Secrets,
                    new string[] { DriveService.Scope.Drive },
                    "user",
                    CancellationToken.None,
                    new FileDataStore(tokenFolder, true)).Result;

                logWriter.Write("Токен получен из папки: " + tokenFolder);
            }
```

Данный код сформирует файл с токеном, который необходим для работы с API Google Drive (см. описание в шаге 3 данного примера). Для получения токена нужно разрешить приложению (название приложения указывается при выполнении п. 2) доступ к аккаунту Google.

Токен будет сохранен в папку `GoogleDriveStorageProvider\AskToken\bin\Debug\token.json\`.

8. Скопируйте на сервер Docsvision файлы из папки `GoogleDriveStorageProvider\bin\Debug\`:
   - `GoogleDriveStorageProvider.dll`,
   - `ru\GoogleDriveStorageProvider.resources.dll`,
   - `Google.Apis.Drive.v3.dll`,
   - `Google.Apis.Auth.PlatformServices.dll`,
   - `Google.Apis.Auth.dll`,
   - `Google.Apis.PlatformServices.dll`,
   - `Google.Apis.Core.dll`,
   - `Google.Apis.dll`,
   - `Newtonsoft.Json.dll`.
9. Зарегистрируйте в GAC скопированные файлы.
10. Скопируйте на сервер Docsvision папку `AskToken\bin\Debug\token.json\` и файл `AskToken\bin\Debug\credentials.json`. Данные файлы потребуются для работы провайдера к Google Drive. Файл и папку нужно скопировать в каталог доступный для сервера Docsvision, но недоступный рядовым пользователям

(из-за наличия конфиденциальных данных в файлах).

11. Подключите новый тип хранилища к Docsvision.

Проверяйте работу компонентов на тестовом сервере Docsvision.

a. Откройте Консоль настройки Docsvision.

b. Перейдите в раздел **Настройки сервер** › **Базы данных**. Выберите настраиваемую БД и нажмите кнопку **Настройка**., чтобы открыть окно *Свойства и управление базой данных*.

c. Перейдите на вкладку *Внешние хранилища*.

d. Добавьте новое хранилище. В окне настройки хранилища в поле *Тип* нужно выбрать вариант ***Добавить из сборки*** и выбрать файл `GoogleDriveStorageProvider.dll` из каталога GAC. В список типов хранилищ будет добавлено хранилище с названием "Хранилище в Google Drive".

e. Настройте новое хранилище с типом "Хранилище в Google Drive", группу и правило помещения в хранилище следуя инструкции, приведенной в /dv6/platform/6.1/console/storage/[документации модуля Платформа].

   При настройке хранилища необходимо указать: путь к файлу `credentials.json` и папке `token.json` (загружены в п. 10), путь к файлу журнала работы провайдера.

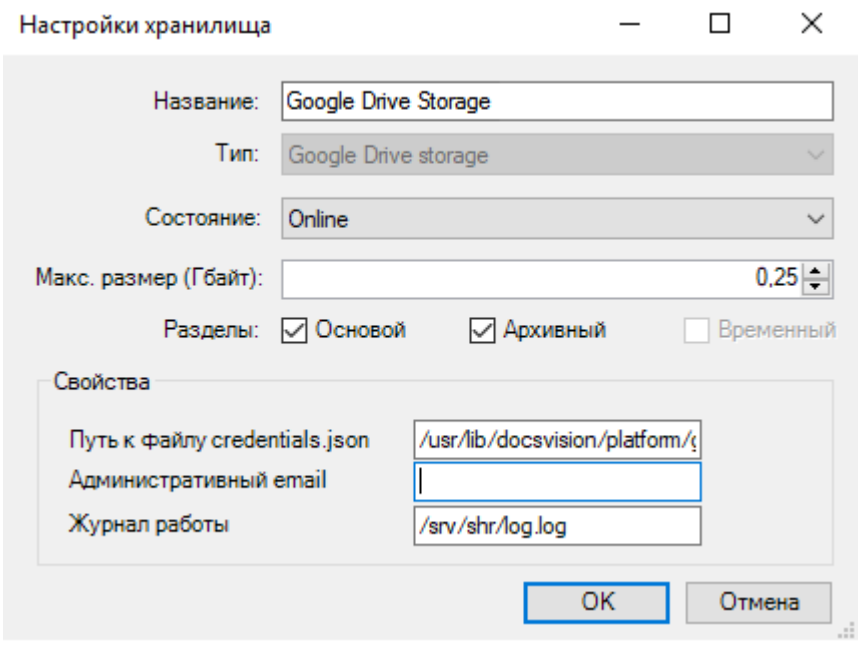


Рисунок 42. Пример настройки хранилища в Google Drive

После перезапуска сервера Docsvision (будет выполнен после добавления хранилища) в Google Drive будут созданы две папки:

- `DV_PrimaryPart` — представляет раздел для основных файлов.
- `DV_ArchivePart` — представляет раздел для архивных файлов.

  Проверить работу провайдера можно настроив правило на помещение в хранилище файлов с расширением `.pdf`. При добавлении в карточку Docsvision файла с типом `.pdf`, соответствующий файл будет загружен в Google Drive.

Создание нового типа правил добавления в хранилище

Администратор может настроить правила помещения бинарных данных файлов в определённые группы хранилищ, в зависимости от типа файла, его размера или принадлежности к справочникам Docsvision. Разработчику предоставляется возможность реализовать собственные типы правил для настройки условий помещения в хранилища.

Для использования собственного типа правил добавление в хранилище необходима опция лицензии *Docsvision модуль интеграции с произвольными файловыми хранилищами*.

Компонент, реализующий правило добавления в хранилище, представляет собой библиотеку `.dll` с классом, реализующим интерфейсы `IBinaryStorageRule` и `IExtensionInitialize`.

Интерфейс `IBinaryStorageRule` определяет метод проверки выполнения условий помещения в хранилище. Интерфейс `IExtensionInitialize` определяет метод инициализации провайдера (`Initialize`), принимающий в сериализованном виде настройки хранилища. Настройки хранилища устанавливаются с помощью графического интерфейса настройки (рассмотрено далее).

Стандартной реализацией интерфейсов `IBinaryStorageRule` и `IExtensionInitialize` является класс `BaseBinaryStorageRule`, который можно использовать в качестве базового класса разрабатываемого компонента с правилом добавления в хранилище.

Также может быть реализован компонент настройки правила, если предполагается предоставлять возможность настройки правила с помощью Консоли настройки Docsvision.

Пример

Далее демонстрируется пример реализации правила помещения в хранилища в

зависимости от установленного в настройках шаблона файлов. Исходный код примера доступен по /dv6/programmer/dv6/_attachments/ruleByFileName.zip[ссылке].

1. Основной класс.

   В данном примере основной класс правила базируется на классе `BaseBinaryStorageRule`. Класс должен быть помечен атрибутом `System.ComponentModel.DisplayName` с указанием отображаемого названия типа правила — отображается в Консоли настройки Docsvision.

   Реализация компонента правила предполагает возможность его настройки с помощью Консоли настройки Docsvision. Для этого класс должен быть помечен атрибутом `System.ComponentModel.Editor`, в котором нужно указать полное название класс, реализующего форму настройки (описание будет приведено далее).

```
using DocsVision.Platform.StorageServer.BinaryStorages.Rules;
using DocsVision.Platform.StorageServer.Files;
using System;
using System.ComponentModel;
using System.Text.RegularExpressions;

namespace RuleByFileName
{
    [DisplayName("Шаблон файла")]
    [Editor(
        "RuleByFileName.FileNamePatternPropertyControl, RuleByFileName,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=774759c67c4f8865,
processorArchitecture=MSIL"
        , "DocsVision.Platform.WinForms.Controls.IExtensionPropertiesControl,
DocsVision.Platform.WinForms"
    )]
    public class FileNameStorageRule : BaseBinaryStorageRule
    {
        string fileNamePattern;

        public FileNameStorageRule() ①
        {
        }

        public FileNameStorageRule(IServiceProvider serviceProvider) : base
(serviceProvider, null)
        {
        }
```

```
        public override bool IsRuleSatisfied(IFileInfo fileInfo) ②
        {
            Regex regex = new Regex(fileNamePattern, RegexOptions.IgnoreCase); ③
            return regex.IsMatch(fileInfo.Name);
        }

        protected override void ReadSettings() ④
        {
            fileNamePattern = Settings.Replace(@"\", @"\\").Replace(".", @"\."
).Replace("*", ".*"); ⑤
            fileNamePattern = "^" + fileNamePattern + "$";
        }
    }
}
```

① Обязательный конструктор без аргументов.

② Метод, осуществляющий проверку соответствия файла условиям данного правила.

③ Файл проверяется по маске с помощью регулярного выражения `fileNamePattern`.

④ Загрузка настроек.
По умолчанию настройки, установленные в Консоли настройки Docsvision, загружаются в строковую переменную `Settings`. Предполагается, что пользователь будет указывать шаблон файла в виде `Акт*.doc*`, а для проверки соответствия имени файла шаблону будет использоваться регулярное выражение. Чтобы привести установленный пользователем шаблон к виду шаблона регулярного выражения был переопределён стандартный метод загрузки настроек `ReadSettings`.

2. Компонент настройки.

   При создании нового типа правил помещения в хранилище можно предоставлять возможность настроить его через Консоль настройки Docsvision, или хранить настройки непосредственно в классе, или предоставлять иные способы настройки.

   В данном примере предусмотрен стандартный способ настройки правила с помощью Консоли настройки Docsvision, который предполагает возможность ввода шаблона на странице настройки внешних хранилищ.

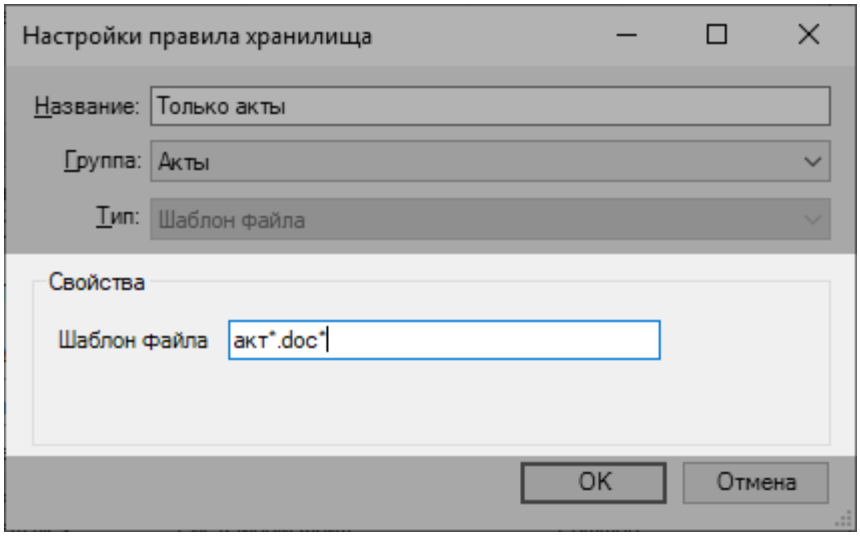


Рисунок 43. Настройки правил хранилища

Для реализации данной функции нужно:

3. Создать класс с формой настройки.
4. Указать данный класс в атрибуте `Editor` основного класса правила помещения в хранилище.

   Класс с формой настройки представляется собой компонент типа `System.Windows.Forms.UserControl` с реализованным программным интерфейсом `IExtensionPropertiesControl`. Далее приведён код примера данного класса.

```
using DocsVision.Platform.WinForms.Controls;
using System;
using System.Windows.Forms;

namespace RuleByUser
{
    public partial class FileNamePatternPropertyControl : UserControl,
IExtensionPropertiesControl
    {

        public FileNamePatternPropertyControl()
        {
            InitializeComponent();
        }

        public string Settings
        {
            get;
            private set;
        }
```

```
        public event EventHandler OnPropertiesChanged;

        public void Initialize(string settings) ①
        {
            Settings = settings ?? "*.*"; ②

            PatternBox.Text = Settings; ③
        }

        public bool Save() ④
        {
            Settings = PatternBox.Text; ⑤

            return true;
        }

        private void PatternBox_TextChanged(object sender, EventArgs e)
        {
            OnPropertiesChanged?.Invoke(this, e);
        }
    }
}
```

① Вызывается при загрузке компонента.

② Если строка настройки отсутствует, используется стандартный шаблон `*.*`.

③ Отображаем загруженный шаблон в графическом интерфейсе.

④ Вызывается при сохранении настроек.

⑤ Шаблон, установленный пользователем, должен быть сохранён в переменную `Settings`.

Рисунок 44. Пример формы настройки

У класса формы две основных задачи:

- Показать пользователю текущие настройки при открытии формы настройки при вызове метода `Initialize`;
- Сохранить настройки пользователя в переменную `Settings` при вызове метода `Save`.

  При вызове `Save` можно осуществить проверку настроек пользователя: если настройки не содержат ошибки нужно вернуть `true`, иначе — `false`.

5. Подписание сборки.

   Разработанный компонент должен быть подписан. Это необходимо для формирования полного имени сборки и регистрации компонента в GAC.

6. Регистрация компонента в GAC.

   Собранный компонент, включающий основной класс и класс формы настройки, должен быть зарегистрирован в GAC сервера Docsvision. Для регистрации используйте `gacutil`.

Проверка

Настройка нового правила помещения в хранилище:

1. Откройте Консоль настройки Docsvision.
2. Перейдите в раздел **Базы данных** › **Настройки базы данных** › **Внешние хранилища**.
3. В секции "Правила помещения в хранилища" добавьте новые правила.
   a. При добавлении в параметре *Тип* выберите вариант ***Добавить из сборки*** и выберите сборку с реализованным новым правилом. В список типов правил добавьте строку "Шаблон файла".
   b. Выберите тип "Шаблон файла" и укажите шаблон файла.
   c. Укажите группу хранилищ, в которую будет помещаться файлы по данному правилу.
4. Переместите правило в начало списка правил, чтобы оно проверялось первым при добавлении файла.
5. Если требуется, настройте другие параметры хранилищ.
6. Сохраните настройки.
7. Создайте карточку и добавьте файл с названием, соответствующим указанному в правиле "Шаблон файла" шаблону. Файл будет сохранён в хранилище из группы хранилищ, указанных при настройке правила "Шаблон файла".

Приложение

В этом разделе содержится справочная информация общего назначения.

В этом разделе:

- Битовые флаги стандартных прав доступа
- Параметры активации справочников
- Параметры соединения с сервером Docsvision
- Сервисы объектной модели
- Список свойств пользовательской сессии
- Параметры активации страницы свойств карточки
- Методы веб-сервиса Docsvision
- Элементы управления

Битовые флаги стандартных прав доступа

Описание прав	Маска	Расшифровка
Полный доступ	`0x000F001F`	*Получена сложением прав доступа:* • Чтение данных объекта. • Изменение данных объекта. • Создание дочерних объектов. • Удаление дочерних объектов. • Копирование объекта. • Удаление объекта. • Чтение разрешений. • Изменение разрешений. • Смена владельца.

Описание прав	Маска	Расшифровка
Чтение данных объекта	`0x00000001`	
Изменение данных объекта	`0x00000002`	
Создание дочерних объектов	`0x00000004`	
Удаление дочерних объектов	`0x00000008`	
Копирование объекта	`0x00000010`	
Удаление объекта	`0x00010000`	
Чтение разрешений	`0x00020000`	
Изменение разрешений	`0x00040000`	
Смена владельца	`0x00080000`	
Чтение	`0x00020001`	*Получена сложением прав:* • Чтение данных объекта. • Чтение разрешений.
Изменение	`0x0000001E`	*Получена сложением прав:* • Изменение данных объекта. • Создание дочерних объектов. • Удаление дочерних объектов. • Копирование объекта.
Владение	`0x000C0000`	*Получена сложением прав:* • Изменение разрешений. • Смена владельца.

Параметры активации справочников

Открытие некоторых справочников Docsvision может сопровождаться установкой

курсора (открытием) на заранее определённой записи справочника, что уменьшает количество переходов в интерфейсе при редактировании записи. Для установки курсора на определённый записи достаточно передать в метод `CardHost.ShowCard` специальный объект, содержащий параметры активации карточки.

Далее приведён пример скрипта карточки, в котором используется функциональность открытия *Справочника сотрудников* с установкой курсора на определённом сотруднике.

```
string employeeID = new Guid("00000000-0000-0000-0000-000000000000").ToString("B"
).ToUpperInvariant(); ①

Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefStaff.ID; ②

string sectionID = DocsVision.BackOffice.CardLib.CardDefs.RefStaff.Employees.ID.ToString
("B").ToUpperInvariant(); ③

object activateParams = new object[] { sectionID, employeeID, System.Reflection.Missing
.Value, System.Reflection.Missing.Value,
 System.Reflection.Missing.Value, false, false, false, 0, System.Reflection.Missing.
Value, true, true, 0 }; ④

base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, ActivateMode.Select,
ActivateFlags.None, activateParams); ⑤
```

① Приведение идентификатора сотрудника с требуемому формату.

② Получение идентификатора Справочника сотрудников.

③ Получение идентификатора секции (в примере, секция Сотрудники), из которой осуществляется выбор.

④ Установка параметров активации.

⑤ Отображение справочника.

Список допустимых параметров активации для разных типов справочников будет различным. Ниже приведены допустимые параметры и их назначение, сгруппированные по типу справочника.

Справочник сотрудников

Номер параметра	Описание
0	*Идентификатор секции, из которой производится выбор:* • Сотрудники (`DocsVision.BackOffice.CardLib.CardDefs.RefStaff.Employees.Id`). • Подразделения (`DocsVision.BackOffice.CardLib.CardDefs.RefStaff.Units.Id`). • Роли (`DocsVision.BackOffice.CardLib.CardDefs.RefStaff.Roles.Id`). • Группы (`DocsVision.BackOffice.CardLib.CardDefs.RefStaff.Group.Id`). • Должности (`DocsVision.BackOffice.CardLib.CardDefs.RefStaff.Positions.Id`).
1	Идентификатор объекта, выбираемого при открытии справочника.
3	Массив идентификаторов выбранных объектов.
5	При `true` разрешается множественный выбор. Если указан `false`, то, например, метод `SelectFromCard` вернет один результат независимо от количества выбранных записей.
6	Если `true`, то метод возвращает идентификаторы безопасности, а не идентификаторы сотрудников, подразделений и т.д.
8	*Включает ограничение на выбор:* • 0 — Ограничения отключены. • 1 — Выбор разрешён только из организаций. • 2 — Выбор разрешён только из групп.
9	Если задействован параметр 8, то указывается идентификатор той группы или подразделения, из которого будет разрешён выбор. Остальные подразделения или группы будут скрыты.
10	Не разрешать выбор подразделений, если `true`.
11	Не разрешать выбор организаций, если `true`.
12	Режим открытия записи (см. `ItemSelectionMode`) приведенный к типу `int`.
13	Не отображать записи с флагом `NotAvailable`.

Номер параметра	Описание
14	Не отображать записи с флагом `NotSearchable`.

Справочник контрагентов

Номер параметра	Описание
0	Идентификатор секции, из которой производится выбор: • Сотрудники (`CardLib.CardDefs.RefPartners.Employees.Id`). • Подразделения (`CardLib.CardDefs.RefPartners.Companies.Id`). • Группы (`CardLib.CardDefs.RefPartners.Groups.Id`). • Должности (`CardLib.CardDefs.RefPartners.Positions.Id`).
1	Идентификатор объекта, выбираемого при открытии справочника.
3	Массив идентификаторов выбранных объектов.
5	При `true` разрешается множественный выбор.
8	Не разрешать выбор подразделений, если `true`.
9	Не разрешать выбор организаций, если `true`.
10	Режим открытия записи (см. `ItemSelectionMode`) приведенный к типу `int`.

Конструктор справочников

Номер параметра	Описание
0	Идентификатор секции, из которой производится выбор: • Узлы справочника (`RefBaseUniversal.ItemTypes.Id`) • Строки справочника (`RefBaseUniversal.Items.Id`)

Номер параметра	Описание
1	Идентификатор открываемой/выбираемой строки. Может использоваться только при открытии в режиме чтения или редактирования *Конструктора справочников* методом `CardHost.ShowCard`. Должен быть указан узел (параметр 2), из которого выбирается строка.
2	Идентификатор узла из которого осуществляется выбор. В режиме выбора строк ограничивает узел (отображается только указанный узел и его дочерние узлы), из которого могут быть выбраны строки. Не предназначен для ограничения области выбора в режиме выбора узла.
3	Режим открытия записи (см. `ItemSelectionMode`) приведенный к типу `int`.

К примеру, чтобы выбрать строку (без открытия записи) из узла *Конструктора справочников*, нужно выполнить скрипт:

```
Guid cardID = DocsVision.BackOffice.CardLib.CardDefs.RefBaseUniversal.ID; ①

Guid idNode = Guid.Parse("B8B58477-CCC3-4213-8190-EFC66FEF86C1"); ②

Guid idRow = Guid.Parse("C0C164D7-2C0F-48E9-A642-3ABC8D2CEFBB"); ③


object activateParams = new object[] { DocsVision.BackOffice.CardLib.CardDefs
.RefBaseUniversal.Items.ID, idRow, idNode, (int)ItemSelectionMode.Positioning}; ④

base.CardControl.CardHost.ShowCard(cardID, Guid.Empty, ActivateMode.ReadOnly,
ActivateFlags.None, activateParams); ⑤
```

① Идентификатор Конструктора справочников.

② Идентификатор узла, из которого выбирается строка.

③ Идентификатор выбираемой строки.

④ Установка параметров активации.

⑤ Отображение справочника с позиционированием курсора на нужной строке.

Справочник ссылок

Номер параметра	Описание
0	ID секции, из которой выбираем (только типы ссылок)
1	ранее выбранный элемент
3	True, позволять выбор нескольких элементов
6	*Режим открытия записи:* • 0 — Позиционирование. • 1 — Редактирование. • 2 — Чтение.

Справочник категорий

Номер параметра	Описание
1	Идентификатор объекта, выбираемого при открытии справочника.
2	При `true` разрешается множественный выбор.
3	Массив идентификаторов выбранных объектов.
4	Режим открытия записи (см. `ItemSelectionMode`) приведенный к типу `int`.

Параметры соединения с сервером Docsvision

- `ConnectAddress` — адрес сервера. В зависимости от используемого протокола, может принимать одно из следующих значений (адрес однозначно определяет протокол, и наоборот):
  - `Web service (HTTP + SOAP)`: `http://SERVERNAME/FOLDERNAME/StorageServer/StorageServerService.asmx`
  - `WCF service (HTTP + SOAP)`: `http://SERVERNAME/StorageServer/StorageServerService.svc`

    - WCF TCP Service: net.tcp://SERVERNAME:8080/StorageServer/StorageServerService.svc
    - WCF Named Pipes: net.pipe://SERVERNAME/StorageServer/StorageServerService.svc

- BaseName — псевдоним базы данных (имя, под которым она подключена к серверу). Данный параметр является опциональным — если его значение не указать, то будет использована база данных по умолчанию.
- UserName — имя пользователя (при использовании базовой аутентификации).
- Password — пароль (при использовании базовой аутентификации).
- Proxy — адрес прокси-сервера (при использовании прокси). Если адрес прокси-сервера **не** указан, то будут использованы настройки Internet Explorer.
- ProxyBypass– список имён серверов, при соединении с которыми не будет использован прокси-сервер (таким образом, если нужно соединяться с сервером Docsvision в обход прокси, то его имеет смысл добавить сюда).
- AllowAutoProxy — использовать скрипт автоматического определения прокси, заданный в настройках Internet Explorer (используется для протокола HTTP).
- Timeout — таймаут ожидания ответа сервера (при установке соединения, а также при выполнении всех последующих операций). Задаётся в миллисекундах. Значение 0 указывает на необходимость использовать значение по умолчанию, которое равно 3 минуты.
- IgnoreInvalidCert — игнорировать неправильный сертификат сервера при использовании SSL.
- ClientCertificate — клиентский сертификат (если используется SSL с требованием клиентского сертификата).

Список свойств пользовательской сессии

В данном разделе перечислены свойства пользовательской сессии, которые включают: сведения о текущем пользователе, сервере Docsvision, лицензии и др.

Следующий код демонстрирует пример получения имени учетной записи текущего пользователя из пользовательской сессии.

```
①
```

```
string accountName = userSession.Properties["AccountName"].Value.ToString() ②
```

① Инициализация контекста объектов.

② Получаем из массива `Properties` — свойства, объект `AccountName` — имя учетной записи.

В следующей таблице перечислены стандартные свойства пользовательской сессии, список которых может отличаться в зависимости от текущего Решения.

Название свойства	**Описание**
`AccountName`	Имя учетной записи текущего пользователя.
`AppID`	Идентификатор приложения, из которого установлено соединение. Например, идентификатор `{8F47FCAF-6FC8-458E-910C-5FCC4FFE6AF7}` соответствует Windows-клиенту Docsvision.
`BaseName`	Псевдоним базы данных, к которой выполнено подключение.
`BaseReadOnly`	Флаг, указывающий (при `true`), что БД работает в режиме "только чтение".
`BaseVersion`	Версия базы данных из **dvsys_global_info**.
`CacheMode`	Режим кэширования данных карточек на клиенте.
`CardPoolSize`	Количество карточек в кэше `ObjectManager`.
`ClearCacheFrom`	Дата последней очистки клиентского кэша.
`ClientVersion`	Версия клиентских компонент. Определяется по версии сборки `ObjectManager.dll`.

Название свойства	Описание
`Compression`	Флаг, указывающий используется или нет сжатие при передаче данных по сети.
`ComputerAddress`	IP-адрес клиентского компьютера.
`ComputerName`	Имя клиентского компьютера.
`ConnectAddress`	Адрес сервера Docsvision, к которому выполнено подключение.
`CurrentUserID`	Идентификатор сотрудника, под чьей учетной записью выполнено подключение к серверу.
`CustomOperationsTimeout`	Время отклика сервера для сложной операции. Проверяется при включённом контроле сети.
`DatabaseType`	Тип базы данных: `0` — Microsoft SQL; `1` — PostgreSQL.
`EmployeeID`	Идентификатор сотрудника, под чьей учетной записью выполнено подключение к серверу.
`ExpirationDate`	Дата окончания срока действия лицензии для Docsvision.
`ExtendedMetadata`	Флаг, указывающий, что база данных использует расширенные метаданные.
`Features`	Список идентификаторов разрешённых в лицензии дополнительных модулей Docsvision.
`FileBlockSize`	Размер блока данных (в KB) для передачи файлов.
`FolderUnreadCountLimit`	Максимальное количество непрочитанных карточек, учитываемое при вычислении количества непрочитанных карточек.

Название свойства	Описание
`FullTextSearchEnabled`	Флаг, указывающий, что полнотекстовый поиск включён в базе данных.
`IsAdmin`	Флаг, указывающий, что текущий пользователь является администратором Docsvision — входит в группу **DocsVision Administrators** на сервере Docsvision.
`IsArchiveOperator`	Флаг, указывающий, что текущий пользователь может работать с долговременным архивом — входит в группу **DocsVision Archive Operators** на сервере Docsvision.
`IsPowerUser`	Флаг, указывающий, что текущий пользователь является привилегированным пользователем — входит в группу **DocsVision Power Users** на сервере Docsvision.
`IsSecurityAdmin`	Флаг, указывающий, что текущий пользователь обладает всеми правами **DocsVision Administrators**, а также может изменять уровни доступа для субъектов и объектов системы — входит в группу **DocsVision Administrators** на сервере Docsvision.
`IsSQCreator`	Флаг, указывающий, что текущий пользователь может создавать новые поисковые запросы — входит в группу **Docsvision Search Query Creators** на сервере Docsvision.
`Languages`	Список поддерживаемых языков пользовательского интерфейса.
`License`	Полный XML-текст лицензионного ключа.

Название свойства	Описание
`LocaleID`	Числовой идентификатор текущего языка, на котором будут возвращаться все строковые значения и описания ошибок.
`Logging`	Уровень журналирования.
`MinCompressSize`	Минимальный блок данных (в Кб), который нужно сжимать.
`NetworkControlEnabled`	Флаг, указывающий, что используется контроль сети.
`PageSize`	Размер блока данных (в Кб) для пересылки данных карточек.
`ReplState`	Флаг, указывающий, что база данных находится под репликацией.
`ServerID`	Идентификатор текущего сервера Docsvision.
`ServerName`	Имя компьютера, на котором расположен сервер Docsvision.
`ServerTime`	Текущее время на сервере Docsvision.
`ServerUrl`	URL к корню WEB-сервера.
`ServerVersion`	Версия сервера Docsvision. Определяется по версии сборки `DocsVision.StorageServer.dll`.
`SimpleOperationsTimeout`	Время отклика сервера для стандартной операции. Проверяется при включённом контроле сети.
`Timeout`	Величина максимального времени (в мс) ожидания ответа сервера Docsvision при установке соединения или выполнении операций.
`UpdateLimit`	Дата окончания срока, после которого возможность обновление сервера становится невозможным.

Название свойства	Описание
`UserID`	Идентификатор текущего пользователя.
`UseSectionDelayedRefresh`	Флаг, указывающий, что включено отложенное обновлений секций.
`UseStreaming`	Флаг, указывающий на использования потокового способа работы с файлами.

Параметры активации страницы свойств карточки

Таблица, приведенная ниже, содержит список допустимых параметров активации страницы свойств карточки, которые могут быть приняты расширением типа "Страницы свойств".

Таблица 2. Параметры активации

Тип страницы свойств	Параметры::
`NavPropertyPageTypes.Folder`	• `ReadOnly` — признак "только для чтения". • `Folder` — папка.
`NavPropertyPageTypes.Card`	• `ReadOnly` — признак "только для чтения". • `CardData` — данные карточки. • `CardID` — идентификатор карточки. • `CardTypeID` — идентификатор типа карточки. • `CardDesc` — описание карточки. • `FolderID` — идентификатор папки, где лежит карточка. • `ShortcutID` — идентификатор ярлыка. • `Archived` — признак архивности. • `Template` — является шаблоном. • `CreateDate` — дата создания. • `ChangeDate` — дата изменения.

Тип страницы свойств	Параметры::
`NavPropertyPageTypes.CardType`	• `ReadOnly` — признак "только для чтения". • `CardType` — тип карточки.

Методы веб-сервиса Docsvision

Взаимодействие с сервером Docsvision, помимо использования базового API, может быть организовано через SOAP.

Адрес сервера для подключения SOAP-клиента указан в настройках сервера Docsvision и обычно соответствует виду: `http://DOCSVISION_SERVER.COM/DocsVision/StorageServer/StorageServerService.asmx`.

Описание методов веб-сервиса доступно по адресу: `http://DOCSVISION_SERVER.COM/DocsVision/StorageServer/StorageServerService.asmx?WSDL`.

Для передачи данных в формате SOAP на адрес сервера Docsvision используется метод HTTP POST.

Описание протокола SOAP доступно по ссылке.

Конкретный способ подключения к серверу Docsvision и отправка сообщения зависят от языка программирования и используемой библиотеки: suds (Python), SOAPClient (PHP), SOAP::Lite for Perl и пр.

Для параметров всех методов справедливо: если текст передается в формате `.xml`, то спецсимволы, содержащиеся в нем, должны быть экранированы. Это применимо и к SOAP-ответу.

В данном подразделе, помимо описания методов веб-сервиса, приведены примеры использования данных методов на языке Python (версии 3); SOAP-клиент не используется.

Во всех примерах, приведенных далее, для подключения к веб-сервису Docsvision используется общий шаблон:

```
import xml.etree.ElementTree as ElementTree
import requests
```

```
from requests_ntlm import HttpNtlmAuth
import base64

url = 'http://DOCSVISION_SERVER.COM/DocsVision/StorageServer/StorageServerService.asmx'
①

username = 'Domain\User.Name'
password = 'User.Password' ②

session = requests.Session()
session.auth = HttpNtlmAuth(username, password, session)
session.headers['Content-Type'] =  'text/xml; charset=utf-8'
```

① Адрес веб-сервиса.

② Учетные данные пользователя Docsvision.

Модуль python-ntlm может быть получен по ссылке.

Программные компоненты СУБП

Методы веб-сервиса Docsvision

Программные компоненты СУБП

Методы программных компонентов Системы Управления Бизнес Процессами.

- SessionGetList
- SessionGetSettings
- SessionKeepAlive
- Больше подробностей в навигационном меню слева…

SessionGetList

Возвращает список открытых подключений к серверу.

Для получения списка сессий, необходимы права администратора Docsvision.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
```

```
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionGetList"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd
="http://www.w3.org/2001/XMLSchema" xmlns:soap
="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SessionGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <baseName>string</baseName>
    </SessionGetList>
  </soap:Body>
</soap:Envelope>
```

Параметры

baseName

Название базы данных. Если не указана, возвращается список подключений к базе данных по умолчанию.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd
="http://www.w3.org/2001/XMLSchema" xmlns:soap
="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SessionGetListResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessions>string</sessions>
    </SessionGetListResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит сведения о всех подключениях к базе данных и предоставляет следующую информацию:

- Идентификатор сессии.

- Учетная запись сотрудника, выполнившего подключение.
- Дата и время подключения, а также последнего запроса к серверу.
- Название и адрес компьютера, с которого выполнено подключение, если эти данные были переданы при открытии сессии.

Пример запроса

```
①

session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionGetList" ②

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <baseName></baseName>
    </SessionGetList>
  </soap:Body>
</soap:Envelope>""" ③

response = session.post(url, data=body) ④

print(response.content) ⑤
```

① Подключение модулей Python и открытие сессии к серверу Docsvision (см. код по ссылке).

② Изменяем цель запроса в заголовке запроса.

③ SOAP-сообщение.

④ Отправляем SOAP-сообщение.

⑤ Ответ SOAP.

SessionGetSettings

Возвращает данные пользовательской сессии.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
```

```
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionGetSettings"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionGetSettings xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
    </SessionGetSettings>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии, для которой запрашиваются сведения

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionGetSettingsResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <settings>string</settings>
    </SessionGetSettingsResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит полный набор сведений о пользовательской сессии (см. Список свойств пользовательской сессии).

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionGetSettings" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionGetSettings xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>9aef2367-7792-e511-9417-90e6ba57b9f8</sessionId>
    </SessionGetSettings>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

SessionKeepAlive

Обновляет штамп времени подключения клиента к серверу.

Периодический вызов данного метода, если другие методы не вызываются, поможет избежать принудительного отключения от сервера, что обусловлено настройками сервера Docsvision.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionKeepAlive"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
```

```
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionKeepAlive xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
    </SessionKeepAlive>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionKeepAliveResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

SessionLogin

Создаёт новую пользовательскую сессию.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionLogin"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
```

```
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionLogin xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <baseName>string</baseName>
      <sessionSettings>string</sessionSettings>
    </SessionLogin>
  </soap:Body>
</soap:Envelope>
```

Параметры

baseName

Название базы данных. Если не указана, использует база данных по умолчанию

sessionSettings

Параметры сессии

Строка `sessionSettings` *должна соответствовать схеме элемента:*

```
<xs:element name="Settings">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Setting" maxOccurs="unbounded" type="xs:string">
        <xs:complexType>
          <xs:attribute name="Name" type="xs:string"></xs:attribute>
          <xs:attribute name="Type" type="xs:string"></xs:attribute>
          <xs:attribute name="ReadOnly" type="xs:int"></xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Могут быть установлены следующие параметры:

- `AppID (string)` — идентификатор приложения, из которого установлено соединение.
- `BaseName (string)` — псевдоним базы данных.
- `ClientVersion (int)` — версия клиента.
- `Compression (boolean)` — необходимость использовать сжатие при передаче данных по сети.

- `ComputerAddress (string)` — IP-адрес клиента.
- `ComputerName (string)` — имя компьютера-клиента.
- `FileBlockSize (int)` — размер блока данных (в KB) для пересылки файлов.
- `LocaleID (int)` — числовой идентификатор используемого языка.
- `MinCompressSize (int)` — минимальных блок данных (в KB) который нужно сжимать.

Рекомендуется всегда передавать: `ComputerAddress`, `ComputerName` и `LocaleID`.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionLoginResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <userSettings>string</userSettings>
    </SessionLoginResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор открытой сессии, а также все её параметры.

Пример запроса

```
session.headers['SOAPAction'] = 'http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionLogin' ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionLogin xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <baseName></baseName>
```

```
      <sessionSettings>&lt;Settings&gt;&lt;Setting Name=&quot;ComputerAddress&quot;
Type=&quot;string&quot; ReadOnly=&quot;1&quot;&gt;192.168.0.2&lt;/Setting&gt;&lt;Setting
Name=&quot;ComputerName&quot;
Type=&quot;string&quot;&gt;connectedComputer&lt;/Setting&gt;&lt;Setting
Name=&quot;LocaleID&quot;
Type=&quot;int&quot;&gt;1049&lt;/Setting&gt;&lt;/Settings&gt;</sessionSettings>
    </SessionLogin>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext('.//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}sessionId')) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор открытой сессии.

SessionLoginEx

Создаёт новую пользовательскую сессию.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionLoginEx"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionLoginEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <baseName>string</baseName>
      <sessionSettings>string</sessionSettings>
      <flags>int</flags>
    </SessionLoginEx>
```

```
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sessionSettings

Параметры сессии

flags

Флаги, определяющие необходимость включения в SOAP-ответ дополнительной информации:

- `0` — без флага.
- `1` — включить информацию по библиотекам карточек.
- `2` — включить информацию по типам карточек.
- `4` — включить информацию по справочниками.
- `8` — включить информацию по пакетам установки.
- `16` — включить схему безопасности.

Строка `sessionSettings` должна соответствовать схеме элемента:

```
<xs:element name="Settings">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Setting" maxOccurs="unbounded" type="xs:string">
        <xs:complexType>
          <xs:attribute name="Name" type="xs:string"></xs:attribute>
          <xs:attribute name="Type" type="xs:string"></xs:attribute>
          <xs:attribute name="ReadOnly" type="xs:int"></xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Могут быть установлены следующие параметры:

- `AppID (string)` — идентификатор приложения, из которого установлено соединение.

- `BaseName (string)` — псевдоним базы данных.
- `ClientVersion (int)` — версия клиента.
- `Compression (boolean)` — необходимость использовать сжатие при передаче данных по сети.
- `ComputerAddress (string)` — IP-адрес клиента.
- `ComputerName (string)` — имя компьютера-клиента.
- `FileBlockSize (int)` — размер блока данных (в Кб) для пересылки файлов.
- `LocaleID (int)` — числовой идентификатор используемого языка.
- `MinCompressSize (int)` — минимальных блок данных (в Кб) который нужно сжимать.

Рекомендуется всегда передавать: `ComputerAddress`, `ComputerName` и `LocaleID`.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionLoginExResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <sessionLoginResponse>
        <ID>guid</ID>
        <PageCount>unsignedInt</PageCount>
        <RowCount>unsignedInt</RowCount>
        <FirstPage>string</FirstPage>
        <UsingCompression>boolean</UsingCompression>
      </sessionLoginResponse>
    </SessionLoginExResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор открытой сессии, а также её параметры. Серверный курсор будет открыт при необходимости.

Пример запроса

```
session.headers['SOAPAction'] = 'http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionLoginEx' ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionLoginEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <baseName></baseName>
      <sessionSettings>&lt;Settings&gt;&lt;Setting Name=&quot;ComputerAddress&quot;
Type=&quot;string&quot; ReadOnly=&quot;1&quot;&gt;192.168.0.2&lt;/Setting&gt;&lt;Setting
Name=&quot;ComputerName&quot;
Type=&quot;string&quot;&gt;connectedComputer&lt;/Setting&gt;&lt;Setting
Name=&quot;LocaleID&quot;
Type=&quot;int&quot;&gt;1049&lt;/Setting&gt;&lt;/Settings&gt;</sessionSettings>
      <flags>16</flags>
    </SessionLoginEx>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext('.//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}sessionId')) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор открытой сессии.

SessionLogout

Закрывает открытую пользовательскую сессию.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionLogout"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionLogout xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
    </SessionLogout>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionLogoutResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = 'http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionLogout' ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
```

```
    <SessionLogout xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>c906d7e8-8692-e511-9417-90e6ba57b9f8</sessionId>
    </SessionLogout>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

SessionSetSettings

Обновляет параметры пользовательской сессии.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionSetSettings"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionSetSettings xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <settings>string</settings>
    </SessionSetSettings>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

settings

Параметры сессии

Строка `settings` *должна соответствовать схеме элемента:*

```
<xs:element name="Settings">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Setting" maxOccurs="unbounded" type="xs:string">
        <xs:complexType>
          <xs:attribute name="Name" type="xs:string"></xs:attribute>
          <xs:attribute name="Type" type="xs:string"></xs:attribute>
          <xs:attribute name="ReadOnly" type="xs:int"></xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionSetSettingsResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

SessionTerminate

Завершает открытую пользовательскую сессию.

Для закрытия чужой сессии необходимы права администратора Docsvision.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
```

```
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionTerminate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionTerminate xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <baseName>string</baseName>
      <sessionId>guid</sessionId>
    </SessionTerminate>
  </soap:Body>
</soap:Envelope>
```

Параметры

baseName

Название базы данных

sessionId

Идентификатор пользовательской сессии

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionTerminateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

SessionUpdateOfflineState

Изменяет состояние пользовательской сессии.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionUpdateOfflineState"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionUpdateOfflineState xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <offline>boolean</offline>
    </SessionUpdateOfflineState>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

offline

Если `true`, то сессия переводится в режим "Офлайн", иначе в режим "Онлайн"

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionUpdateOfflineStateResponse xmlns=
"http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/" />
  </soap:Body>
```

```
</soap:Envelope>
```

Карточки

Методы веб-сервиса Docsvision

Карточки

В разделе представлены методы для работы с карточками.

- CardArchive
- CardArchiveGroup
- CardCopy
- Больше подробностей в навигационном меню слева...

CardArchive

Помещает карточку в архив.

Карточки, у которых установлен запрет на архивирование (в метаданных типа карточки установлен флаг "Non-archival"), не могут быть архивированы.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/CardArchive"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardArchive xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
      <options>int</options>
    </CardArchive>
```

```
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор архивируемой карточки

options

Флаг архивирования карточки:

- `0` — без параметров
- `1` — режим отложенной архивации
- `2` — архивировать связанные карточки
- `4` — архивировать связанные файлы

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardArchiveResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardArchive" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
```

```
">
  <soap:Body>
    <CardArchive xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>337ae630-568c-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>d30ba878-578c-e511-9417-90e6ba57b9f8</instanceID>
      <options>0</options>
    </CardArchive>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardArchiveGroup

Помещает несколько карточек в архив.

Карточки, у которых установлен запрет на архивирование (в метаданных типа карточки установлен флаг "Non-archival"), не могут быть архивированы.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardArchiveGroup"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardArchiveGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <items>string</items>
      <options>int</options>
    </CardArchiveGroup>
```

```
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

items

Идентификаторы архивируемых карточек

options

Флаг архивирования карточек. См. описание параметра options в `CardArchive`

Строка `items` должна соответствовать формату элемента `GroupOperation`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="GroupOperation">
  <xsd:complexType>
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Item">
        <xsd:complexType>
          <xsd:attribute name="ID" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardArchiveGroupResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <failedItems>string</failedItems>
    </CardArchiveGroupResponse>
```

```
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификаторы карточек, которые не были перемещены в архив.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardArchiveGroup" ①

body = """<?xml version="1.0" encoding="utf-8"?>
 <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
   <soap:Body>
     <CardArchiveGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
       <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
       <items>&lt;GroupOperation&gt;&lt;Item ID=&quot;DC35AB36-628C-E511-9417-
90E6BA57B9F8&quot;/&gt;&lt;Item ID=&quot;D30BA878-578C-E511-9417-
90E6BA57B9F8&quot;/&gt;&lt;/GroupOperation&gt;</items>
       <options>0</options>
     </CardArchiveGroup>
   </soap:Body>
 </soap:Envelope>""" ②

response = session.post(url,  data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardCopy

Копирует карточку.

Карточки с установленным запретом на копирование — в метаданных типа карточки установлен флаг "Instances can be copied",- не могут быть скопированы.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
```

```
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/CardCopy"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardCopy xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
    </CardCopy>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор копируемой карточки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardCopyResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <newInstanceId>guid</newInstanceId>
    </CardCopyResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор копии карточки.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardCopy" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardCopy xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>817E8CC7-F28C-E511-9417-90E6BA57B9F8</instanceID>
    </CardCopy>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext( ".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}newInstanceId")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор скопированной карточки.

CardCreate

Создаёт новый экземпляр карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/CardCreate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
```

```
    <CardCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <cardId>guid</cardId>
      <newInstanceId>guid</newInstanceId>
    </CardCreate>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardId

Идентификатор типа карточки

newInstanceId

Идентификатор создаваемой карточки. В данной реализации метода не используется — достаточно передать любой идентификатор.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardCreateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <newInstanceId>guid</newInstanceId>
    </CardCreateResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор (`newInstanceId`) созданной карточки.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardCreate" ①
```

```
body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <cardId>B9F7BFD7-7429-455E-A3F1-94FFB569C794</cardId>
      <newInstanceId>00000000-0000-0000-0000-000000000000</newInstanceId>
    </CardCreate>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}newInstanceId")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор новой карточки.

CardCreateEx

Создаёт новый экземпляр карточки.

Карточка создаётся пустой.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardCreateEx"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardCreateEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
```

```
03/StorageServer/">
      <sessionId>guid</sessionId>
      <cardId>guid</cardId>
      <newInstanceId>guid</newInstanceId>
    </CardCreateEx>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardId

Идентификатор типа карточки

newInstanceId

Собственный идентификатор новой карточки. Если идентификатор должен быть сформирован системой, то необходимо передать "00000000-0000-0000-0000-000000000000"

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardCreateExResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <newInstanceId>guid</newInstanceId>
      <cardInfo>string</cardInfo>
    </CardCreateExResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор (`newInstanceId`) созданной карточки, а также её атрибуты (`cardInfo`).

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardCreateEx" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardCreateEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <cardId>B9F7BFD7-7429-455E-A3F1-94FFB569C794</cardId>
      <newInstanceId>00000000-0000-0000-0000-000000000000</newInstanceId>
    </CardCreateEx>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}newInstanceId")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор созданной карточки.

CardDearchive

Извлекает карточку из архива.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardDearchive"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
```

```
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardDearchive xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
      <options>int</options>
    </CardDearchive>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

options

Флаг разархивирования карточки:

- 0 — без параметров
- 1 — режим отложенной разархивации
- 2 — извлечь из архива связанные карточки
- 4 — извлечь из архива связанные файлы

Флаги можно комбинировать, например: 6(2&4) — будут извлечены связанные карточки и файлы.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardDearchiveResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
```

```
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardDearchive" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardDearchive xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>DC35AB36-628C-E511-9417-90E6BA57B9F8</instanceID>
      <options>0</options>
    </CardDearchive>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardDearchiveGroup

Извлекает из архива несколько карточек.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardDearchiveGroup"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
```

```
">
  <soap:Body>
    <CardDearchiveGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <items>string</items>
      <options>int</options>
    </CardDearchiveGroup>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

items

Список идентификаторов карточек, извлекаемых из архива

options

Флаг разархивирования карточки:

- `0` — без параметров.
- `1` — режим отложенной разархивации.
- `2` — извлечь из архива связанные карточки.
- `4` — извлечь из архива связанные файлы.

Строка `items` должна соответствовать формату элемента `GroupOperation`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="GroupOperation">
  <xsd:complexType>
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Item">
        <xsd:complexType>
          <xsd:attribute name="ID" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardDearchiveGroupResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <failedItems>string</failedItems>
    </CardDearchiveGroupResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификаторы карточек, которые не были извлечены из архива.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardDearchiveGroup" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardDearchiveGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <items>&lt;GroupOperation&gt;&lt;Item ID=&quot;693DDF1A-F48C-E511-9417-
90E6BA57B9F8&quot;/&gt;&lt;Item ID=&quot;4E3C2F21-F38C-E511-9417-
90E6BA57B9F8&quot;/&gt;&lt;/GroupOperation&gt;</items>
      <options>0</options>
    </CardDearchiveGroup>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardDelete

Удаляет карточку.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/CardDelete"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardDelete xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
      <permanent>boolean</permanent>
    </CardDelete>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

permanent

Режим удаления. Если `0` (`False`), то с возможностью восстановления карточки, иначе без такой возможности.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardDeleteResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardDelete" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardDelete xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>693DDF1A-F48C-E511-9417-90E6BA57B9F8</instanceID>
      <permanent>1</permanent>
    </CardDelete>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardGetInfo

Возвращает атрибуты карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/CardGetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
    </CardGetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardGetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <cardInfo>string</cardInfo>
    </CardGetInfoResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит:

- `CardTypeID` — идентификатор типа.
- `Description` — описание.

- `CreationDateTime` — дата создания.
- `ChangeDateTime` — дата изменения.
- `timestamp` — отметка времени.
- `Deleted` — карточка помечена к удалению.
- `Template` — карточка является шаблоном.
- `WasRead` — карточка прочтена пользователем.
- `ArchiveState` — карточка находится в архиве.
- `Barcode` — штрих-код.
- `Status` — состояние карточки: 0 — обычный режим, 1 — карточка записываться.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardGetInfo" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>D5C4008F-0B89-E511-9417-90E6BA57B9F8</instanceID>
    </CardGetInfo>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

CardGetList

Возвращает список карточек определённого типа.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/CardGetList"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <cardId>guid</cardId>
    </CardGetList>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardId

Идентификатор типа возвращаемых карточек

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardGetListResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
```

```
      <cardInfo>string</cardInfo>
    </CardGetListResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит: идентификатор карточки, идентификатор типа, дата создания и изменения карточки, а также строка её описание.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardGetList" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <cardId>B9F7BFD7-7429-455E-A3F1-94FFB569C794</cardId>
    </CardGetList>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

CardGetState

Возвращает состояние карточки.

Не путайте с состоянием карточки, предоставляемым *Конструктором состояний*.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardGetState"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardGetState xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
    </CardGetState>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardGetStateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <state>CARD_STATE_NOT_EXISTING or CARD_STATE_EXISTING or CARD_STATE_DELETED</state>
    </CardGetStateResponse>
  </soap:Body>
```

```
</soap:Envelope>
```

Ответ может содержать одно из состояний:

- `CARD_STATE_NOT_EXISTING` — карточка не существует.
- `CARD_STATE_EXISTING` — карточка существует.
- `CARD_STATE_DELETED` — карточка отмечена как удалённая.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardGetState" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardGetState xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>D5C4008F-0B89-E511-9417-90E6BA57B9F8</instanceID>
    </CardGetState>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}state")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим состояние карточки.

CardPurge

Окончательно удаляет карточки, отмеченные к удалению.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
```

```
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/CardPurge"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardPurge xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <cardId>guid</cardId>
      <maxDate>double</maxDate>
    </CardPurge>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardId

Идентификатор типа удаляемых карточек

maxDate

Дата OLE-автоматизации, которой ограничивается удаление карточек. Будут удалены карточки, время удаления которых меньше указанного значения.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardPurgeResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
```

```
</soap:Envelope>
```

Пример запроса

К примеру, удаление карточек типа "Документ", помеченных к удалению ранее 01.10.2015:

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardPurge" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardPurge xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <cardId>B9F7BFD7-7429-455E-A3F1-94FFB569C794</cardId>
      <maxDate>42278</maxDate>
    </CardPurge>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardRestore

Восстанавливает карточку, помеченную к удалению.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/CardRestore"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
```

```
  <soap:Body>
    <CardRestore xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
    </CardRestore>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardRestoreResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardRestore" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardRestore xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
```

```
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>D5C4008F-0B89-E511-9417-90E6BA57B9F8</instanceID>
    </CardRestore>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardRestoreGroup

Восстанавливает несколько карточек, помеченных к удалению.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardRestoreGroup"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd
="http://www.w3.org/2001/XMLSchema" xmlns:soap
="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <CardRestoreGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <items>string</items>
    </CardRestoreGroup>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

items

Список идентификаторов восстанавливаемых карточек

Строка `items` должна соответствовать формату элемента `GroupOperation`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="GroupOperation">
  <xsd:annotation>
    <xsd:documentation>Group operation items</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Item">
        <xsd:complexType>
          <xsd:attribute name="ID" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardRestoreGroupResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <failedItems>string</failedItems>
    </CardRestoreGroupResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификаторы карточек, восстановление которых недоступно.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardRestoreGroup" ①

body = """<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardRestoreGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <items>&lt;GroupOperation&gt;&lt;Item ID=&quot;D5C4008F-0B89-E511-9417-
90E6BA57B9F8&quot;/&gt;&lt;Item ID=&quot;DF84039F-5288-E511-9417-
90E6BA57B9F8&quot;/&gt;&lt;/GroupOperation&gt;</items>
    </CardRestoreGroup>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardSetInfo

Обновляет атрибуты карточек.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/CardSetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
      <description>string</description>
      <template>boolean</template>
      <topic>string</topic>
    </CardSetInfo>
  </soap:Body>
```

```
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

description

Описание карточки

template

Карточка является шаблоном

topic

Тема карточки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardSetInfo" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
```

```
  <soap:Body>
    <CardSetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>D5C4008F-0B89-E511-9417-90E6BA57B9F8</instanceID>
      <description>New description</description>
      <template>0</template>
      <topic></topic>
    </CardSetInfo>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardSetInfoEx

Обновляет атрибуты карточек.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardSetInfoEx"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetInfoEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
      <description>string</description>
      <template>boolean</template>
      <topic>string</topic>
      <barcode>string</barcode>
    </CardSetInfoEx>
  </soap:Body>
```

```
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

description

Описание карточки

template

Карточка является шаблоном

topic

Тема карточки

barcode

Штрих-код

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetInfoExResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardSetInfoEx" ①

body = """<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetInfoEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>272AFAA6-238D-E511-9417-90E6BA57B9F8</instanceID>
      <description>New description</description>
      <template>0</template>
      <topic></topic>
      <barcode>999</barcode>
    </CardSetInfoEx>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardSetReadStatus

Обновляет признак того, что карточка была прочитана.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardSetReadStatus"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetReadStatus xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
      <wasRead>boolean</wasRead>
    </CardSetReadStatus>
```

```
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

wasRead

Признак прочтения карточки. Если `true`, то карточка считается прочитанной пользователем, иначе не прочитанной.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetReadStatusResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardSetReadStatus" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetReadStatus xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>272AFAA6-238D-E511-9417-90E6BA57B9F8</instanceID>
```

```
      <wasRead>true</wasRead>
    </CardSetReadStatus>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardSetStatus

Изменяет состояние карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardSetStatus"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetStatus xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceId>guid</instanceId>
      <recordStatus>OBJECT_STATUS_OPERATIVE or OBJECT_STATUS_RECORD or
OBJECT_STATUS_WITHDRAWN</recordStatus>
    </CardSetStatus>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceId

Идентификатор карточки

recordStatus

Устанавливаемое состояние карточки:

- `OBJECT_STATUS_OPERATIVE` — обычный режим.
- `OBJECT_STATUS_RECORD` — карточка записываться. Действует ограничение на удаление карточки в данном состоянии.
- `OBJECT_STATUS_WITHDRAWN`. Режим не используется.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetStatusResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardSetStatus" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetStatus xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <instanceId>272AFAA6-238D-E511-9417-90E6BA57B9F8</instanceId>
      <recordStatus>OBJECT_STATUS_WITHDRAWN</recordStatus>
    </CardSetStatus>
  </soap:Body>
```

```
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CardTypeGetInfo

Возвращает информацию о типе карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardTypeGetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardTypeGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <typeId>guid</typeId>
      <infoType>CARD_TYPE_SCHEMA or CARD_TYPE_ICON or CARD_TYPE_XSD</infoType>
    </CardTypeGetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

typeId

Идентификатор типа карточки

infoType

Тип запрашиваемых данных:

- `CARD_TYPE_SCHEMA` — возвращает схему.
- `CARD_TYPE_ICON` — возвращает иконку.
- `CARD_TYPE_XSD`.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardTypeGetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <info>string</info>
      <usingCompression>boolean</usingCompression>
    </CardTypeGetInfoResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит данные типа карточки, которые определены значением `infoType`.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardTypeGetInfo" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardTypeGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <typeId>B9F7BFD7-7429-455E-A3F1-94FFB569C794</typeId>
      <infoType>CARD_TYPE_SCHEMA</infoType>
    </CardTypeGetInfo>
```

```
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

CardTypeGetList

Возвращает список типов карточек, зарегистрированных на сервере.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardTypeGetList"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardTypeGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
    </CardTypeGetList>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

Ответ

```
HTTP/1.1 200 OK
```

```
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardTypeGetListResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <cardTypeInfo>string</cardTypeInfo>
      <usingCompression>boolean</usingCompression>
    </CardTypeGetListResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardTypeGetList" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardTypeGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
    </CardTypeGetList>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

CardTypeSetOptions

Обновляет дополнительные флаги, установленные для типа карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardTypeSetOptions"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardTypeSetOptions xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <cardId>guid</cardId>
      <options>int</options>
    </CardTypeSetOptions>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardId

Идентификатор типа карточки

options

Дополнительные флаги. Значение может быть получено сложением:

- `0` — Not specified.
- `1` — Card is a dictionary (only one instance allowed at all).
- `2` — Card instance cannot be created directly.
- `4` — Card instances cannot be displayed in views.
- `8` — Card instances cannot be found by search.
- `0x10` — Card instance can be open from another card by link.
- `0x20` — Card would not be automatically locked while opening by user (developer have to manage locks by himself).

- `0x40` — Specific card type to implement Navigator extensions.
- `0x80` — Hard shortcut would not be created automatically while creating a new card instance in a folder.
- `0x100` — Specific card type which stores information about folders.
- `0x200` — Card instances cannot be marked as unread.
- `0x400` — Card instances can be copyed.
- `0x800` — Specifies whether a server should store own security descriptors for card sections and rows, or not.
- `0x1000` — Card instances can be marked as templates.
- `0x2000` — Card instances cannot be deleted directly.
- `0x4000` — Specifies whether card instances can be opened in selection mode (specific to dictionaries).
- `0x8000` — Card instances cannot be exported to XML.
- `0x10000` — Card creation can be denied by administrator.
- `0x20000` — Card deletion can be denied by administrator.
- `0x1000000` — Card instances cannot be archived.
- `0x2000000` — Card instances cannot be replicated to another database.
- `0x4000000` — Only instances marked as templates can be replicated to another database.
- `0x8000000` — Determines whether card type is marked for replication.
- `0x10000000` — Card provide method to get timestamp info for optimizing client cache update.
- `0x20000000` — Try to use server extension while access check.
- `0x40000000` — Caching on disk is restricted.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
```

```
">
  <soap:Body>
    <CardTypeSetOptionsResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <options>int</options>
    </CardTypeSetOptionsResponse>
  </soap:Body>
</soap:Envelope>
```

Строки секций

Методы веб-сервиса Docsvision

Строки секций

В разделе представлены методы для работы со строками секций.

- RowCopy
- RowCreate
- RowDelete
- Больше подробностей в навигационном меню слева...

RowCopy

Копирует строку секции.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/RowCopy"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowCopy xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
      <rowId>guid</rowId>
```

```
      <newRowId>guid</newRowId>
      <newInstanceId>guid</newInstanceId>
      <newParentRowId>guid</newParentRowId>
      <newParentTreeRowId>guid</newParentTreeRowId>
    </RowCopy>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

rowId

Идентификатор копируемой строки

newRowId

Идентификатор копии строки

newInstanceId

Идентификатор экземпляра карточки, в которую копируется строка

newParentRowId

Идентификатор родительской строки, если копируемая строка должна быть подчиненной

newParentTreeRowId

Идентификатор родительской строки, если копируемая строка должна быть подчиненной в иерархической секции

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
```

```
    <RowCopyResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <newServerRowId>guid</newServerRowId>
    </RowCopyResponse>
  </soap:Body>
</soap:Envelope>
```

RowCreate

Создаёт новую строку секции.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/RowCreate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
      <rowId>guid</rowId>
      <instanceID>guid</instanceID>
      <parentRowId>guid</parentRowId>
      <parentTreeRowId>guid</parentTreeRowId>
    </RowCreate>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

rowId

Идентификатор создаваемой строки

instanceID

Идентификатор карточки

parentRowId

Идентификатор родительской строки, если создаваемая строка должна быть подчиненной

parentTreeRowId

Идентификатор родительской строки, если создаваемая строка должна быть подчиненной в иерархической секции

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowCreateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <newRowId>guid</newRowId>
    </RowCreateResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/RowCreate" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <sectionId>2FDE03C2-FF87-4E42-A8C2-7CED181977FB</sectionId>
      <rowId>00000000-0000-0000-0000-000000000000</rowId>
      <instanceID>272AFAA6-238D-E511-9417-90E6BA57B9F8</instanceID>
```

```
      <parentRowId>00000000-0000-0000-0000-000000000000</parentRowId>
      <parentTreeRowId>00000000-0000-0000-0000-000000000000</parentTreeRowId>
    </RowCreate>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
newRowId = responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}newRowId") ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор новой строки.

RowDelete

Удаляет строку секции.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/RowDelete"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowDelete xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
      <rowId>guid</rowId>
    </RowDelete>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

rowId

Идентификатор удаляемой строки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowDeleteResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

RowExists

Проверяет существование строки секции.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/RowExists"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowExists xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
```

```
      <rowId>guid</rowId>
    </RowExists>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

rowId

Идентификатор проверяемой строки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowExistsResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <exists>boolean</exists>
    </RowExistsResponse>
  </soap:Body>
</soap:Envelope>
```

RowGetData

Возвращает содержимое строки секции карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/RowGetData"
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowGetData xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
      <rowId>guid</rowId>
      <instanceID>guid</instanceID>
    </RowGetData>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

rowId

Идентификатор строки

instanceID

Идентификатор карточки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowGetDataResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <rowData>string</rowData>
      <usingCompression>boolean</usingCompression>
    </RowGetDataResponse>
```

```
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/RowGetData" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowGetData xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <sectionId>7473F07F-11ED-4762-9F1E-7FF10808DDD1</sectionId>
      <rowId>23E73F8B-5748-4759-9548-A2B79D680024</rowId>
      <instanceID>6710B92A-E148-4363-8A6F-1AA0EB18936C</instanceID>
    </RowGetData>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

RowGetDataByInstanceID

Возвращает содержимое строки плоской секции карточки.

Метод возвращает данные только последней строки секции, поэтому не подходит для иерархических или коллекционных секций.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/RowGetDataByInstanceID"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowGetDataByInstanceID xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
      <sectionId>guid</sectionId>
    </RowGetDataByInstanceID>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

sectionId

Идентификатор секции

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowGetDataByInstanceIDResponse xmlns="http://schemas.docsvision.com/Platform/2009-
02-03/StorageServer/">
      <rowData>string</rowData>
      <usingCompression>boolean</usingCompression>
    </RowGetDataByInstanceIDResponse>
  </soap:Body>
```

```
</soap:Envelope>
```

RowGetHierarchy

Возвращает идентификаторы всех родительских строк в секции для подчиненной строки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/RowGetHierarchy"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowGetHierarchy xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
      <rowId>guid</rowId>
    </RowGetHierarchy>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

rowId

Идентификатор строки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowGetHierarchyResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <hierarchyData>string</hierarchyData>
    </RowGetHierarchyResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/RowGetHierarchy" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowGetHierarchy xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <sectionId>7473F07F-11ED-4762-9F1E-7FF10808DDD1</sectionId>
      <rowId>3A63F7B9-2106-4853-8100-CBDC19CB2841</rowId>
    </RowGetHierarchy>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

RowGetInfo

Возвращает данные о строке секции.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/RowGetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
      <rowId>guid</rowId>
    </RowGetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

rowId

Идентификатор строки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowGetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <instanceID>guid</instanceID>
```

```
      <parentRowId>guid</parentRowId>
      <parentTreeRowId>guid</parentTreeRowId>
    </RowGetInfoResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор карточки, в которой содержится строка, идентификатор родительской строки и идентификатор родительской строки в иерархической секции.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/RowGetInfo" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <sectionId>7473F07F-11ED-4762-9F1E-7FF10808DDD1</sectionId>
      <rowId>3A63F7B9-2106-4853-8100-CBDC19CB2841</rowId>
    </RowGetInfo>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

RowMove

Перемещает строку секции карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
```

```
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/RowMove"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowMove xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
      <rowId>guid</rowId>
      <newParentRowId>guid</newParentRowId>
      <newParentTreeRowId>guid</newParentTreeRowId>
    </RowMove>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

rowId

Идентификатор строки

newParentRowId

Идентификатор родительской строки, если копируемая строка должна быть подчиненной

newParentTreeRowId

Идентификатор родительской строки, если копируемая строка должна быть подчиненной в иерархической секции

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowMoveResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

RowSetData

Устанавливает содержимое строки секции карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/RowSetData"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowSetData xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
      <rowId>guid</rowId>
      <rowData>string</rowData>
    </RowSetData>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

rowId

Идентификатор строки

rowData

Содержимое строки

Строка `rowData` *является xml-документом, содержащим название и значение изменяемого поля:*

```
<RowData>
  <row FieldName="Содержимое поля"/>
</RowData>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowSetDataResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/RowSetData" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <RowSetData xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <sectionId>DBC8AE9D-C1D2-4D5E-978B-339D22B32482</sectionId>
      <rowId>AC700CB2-6B9D-4312-BD97-EBB3F96846E4</rowId>
      <rowData>&lt;RowData&gt;&lt;row DisplayString=&quot;New DisplayString&quot;
FirstName=&quot;New FirstName&quot;/&gt;&lt;/RowData&gt;</rowData>
    </RowSetData>
```

```
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

Нумераторы

Методы веб-сервиса Docsvision

Нумераторы

В разделе представлены методы для работы с нумераторами.

- NumAllocateNumber
- NumAllocateNumbers
- NumChangeLeftBound
- Больше подробностей в навигационном меню слева...

NumAllocateNumber

Занимает номер в зоне нумератора.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumAllocateNumber"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumAllocateNumber xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
```

```
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <userId>guid</userId>
      <zoneName>string</zoneName>
      <number>int</number>
    </NumAllocateNumber>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор карточки нумератора

userId

Идентификатор сотрудника (*справочник сотрудников*), занимающего номер

zoneName

Название зоны, которой принадлежит занимаемый

number

Занимаемый номер

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumAllocateNumberResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <NumberId>guid</NumberId>
    </NumAllocateNumberResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор занятого номера.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumAllocateNumber" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumAllocateNumber xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <numeratorId>A16E9792-5B88-E511-9417-90E6BA57B9F8</numeratorId>
      <userId>C9449204-39AD-4DD8-B212-7C38580BD2BB</userId>
      <zoneName>SimpleZone</zoneName>
      <number>5</number>
    </NumAllocateNumber>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}NumberId")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор занятого номера.

NumAllocateNumbers

Занимает диапазон номеров в зоне нумератора.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
```

```
03/StorageServer/NumAllocateNumbers"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumAllocateNumbers xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <userId>guid</userId>
      <zoneName>string</zoneName>
      <firstNumber>int</firstNumber>
      <lastNumber>int</lastNumber>
    </NumAllocateNumbers>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор карточки нумератора

userId

Идентификатор сотрудника (*справочник сотрудников*), занимающего номер

zoneName

Название зоны, которой принадлежит занимаемый

firstNumber

Начальный занимаемый номер

lastNumber

Конечный занимаемый номер

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumAllocateNumbersResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumAllocateNumbers" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumAllocateNumbers xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <numeratorId>A16E9792-5B88-E511-9417-90E6BA57B9F8</numeratorId>
      <userId>C9449204-39AD-4DD8-B212-7C38580BD2BB</userId>
      <zoneName>Test</zoneName>
      <firstNumber>6</firstNumber>
      <lastNumber>8</lastNumber>
    </NumAllocateNumbers>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

NumChangeLeftBound

Изменяет начальное значение границы нумератора.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
```

```
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumChangeLeftBound"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumChangeLeftBound xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <newLeftBound>int</newLeftBound>
      <forceChange>boolean</forceChange>
    </NumChangeLeftBound>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор карточки нумератора

newLeftBound

Новое начальное значение

forceChange

Игнорировать наличие занятых номеров

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
```

```
  <soap:Body>
    <NumChangeLeftBoundResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

NumChangeRightBound

Изменяет конечное значение границы нумератора.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumChangeRightBound"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumChangeRightBound xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <newRightBound>int</newRightBound>
      <forceChange>boolean</forceChange>
    </NumChangeRightBound>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор нумератора

newRightBound

Конечное начальное значение

forceChange

Игнорировать наличие занятых номеров

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumChangeRightBoundResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

NumGetFirstFree

Занимает первый свободный номер.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumGetFirstFree"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetFirstFree xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <zoneName>string</zoneName>
      <userId>guid</userId>
    </NumGetFirstFree>
  </soap:Body>
```

```
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор карточки нумератора

zoneName

Название зоны нумератора

userId

Идентификатор сотрудника (*справочник сотрудников*), занимающего номер

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetFirstFreeResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <number>int</number>
      <NumberId>guid</NumberId>
    </NumGetFirstFreeResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumGetFirstFree" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
```

```
  <soap:Body>
    <NumGetFirstFree xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <numeratorId>A16E9792-5B88-E511-9417-90E6BA57B9F8</numeratorId>
      <zoneName>SimpleZone</zoneName>
      <userId>C9449204-39AD-4DD8-B212-7C38580BD2BB</userId>
    </NumGetFirstFree>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}NumberId")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор занятого номера.

NumGetLastFree

Занимает последний свободный номер.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumGetLastFree"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetLastFree xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <zoneName>string</zoneName>
```

```
      <userId>guid</userId>
    </NumGetLastFree>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор карточки нумератора

zoneName

Название зоны нумератора

userId

Идентификатор сотрудника (*справочник сотрудников*), занимающего номер

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetLastFreeResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <number>int</number>
      <NumberId>guid</NumberId>
    </NumGetLastFreeResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumGetLastFree" ①

body = """<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetLastFree xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
   <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <numeratorId>A16E9792-5B88-E511-9417-90E6BA57B9F8</numeratorId>
      <zoneName>Test</zoneName>
      <userId>C9449204-39AD-4DD8-B212-7C38580BD2BB</userId>
    </NumGetLastFree>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}NumberId")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор занятого номера.

NumGetNumberByID

Возвращает занятый номер нумератора по его идентификатору.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumGetNumberByID"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetNumberByID xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
```

```
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <zoneName>string</zoneName>
      <NumberId>guid</NumberId>
    </NumGetNumberByID>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор номера

zoneName

Название зоны нумератора

NumberId

Идентификатор номера

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetNumberByIDResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <number>int</number>
    </NumGetNumberByIDResponse>
  </soap:Body>
</soap:Envelope>
```

NumGetNumberID

Возвращает идентификатор занятого номера нумератора.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumGetNumberID"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetNumberID xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <zoneId>guid</zoneId>
      <number>int</number>
    </NumGetNumberID>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор нумератора

zoneId

Идентификатор зоны нумератора

number

Занятый номер, для которого получаем идентификатор

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetNumberIDResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <NumberId>guid</NumberId>
    </NumGetNumberIDResponse>
  </soap:Body>
</soap:Envelope>
```

NumGetNumberStatus

Возвращает статус номера в нумераторе.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumGetNumberStatus"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetNumberStatus xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <zoneName>string</zoneName>
      <number>int</number>
    </NumGetNumberStatus>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор номера

zoneName

Название зоны нумератора

number

Проверяемый номер

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetNumberStatusResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <numberStatus>NUMBER_STATUS_FREE or NUMBER_STATUS_IN_USE or NUMBER_STATUS_RESERVED
or NUMBER_STATUS_NOT_EXISTS</numberStatus>
    </NumGetNumberStatusResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ может содержать значение:

- NUMBER_STATUS_FREE — не занят.
- NUMBER_STATUS_IN_USE — занят.
- NUMBER_STATUS_RESERVED — зарезервирован.
- NUMBER_STATUS_NOT_EXISTS — не существует, выходит за диапазон номеров зоны, указанной в zoneName.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumGetNumberStatus" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetNumberStatus xmlns="http://schemas.docsvision.com/Platform/2009-02-
```

```
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <numeratorId>A16E9792-5B88-E511-9417-90E6BA57B9F8</numeratorId>
      <zoneName>SimpleZone</zoneName>
      <number>5</number>
    </NumGetNumberStatus>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}numberStatus")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим статус номера.

NumGetNumbersStatus

Возвращает статусы номеров в диапазоне номеров нумератора.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumGetNumbersStatus"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetNumbersStatus xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <zoneName>string</zoneName>
      <firstNumber>int</firstNumber>
      <lastNumber>int</lastNumber>
```

```
    </NumGetNumbersStatus>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор нумератора

zoneName

Название зоны нумератора

firstNumber

Начальный номер проверяемого диапазона

lastNumber

Конечный номер диапазона

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumGetNumbersStatusResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <cursorInfo>
        <ID>guid</ID>
        <PageCount>unsignedInt</PageCount>
        <RowCount>unsignedInt</RowCount>
        <FirstPage>string</FirstPage>
        <UsingCompression>boolean</UsingCompression>
      </cursorInfo>
    </NumGetNumbersStatusResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит номер нумератора и статус номера, который может иметь значение:

- `0` — не занят.
- `1` — занят.
- `2` — зарезервирован.
- `3` — не существует (выходит за пределы диапазона номеров зоны нумератора).

NumReleaseNumber

Освобождает занятый номер.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumReleaseNumber"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumReleaseNumber xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <zoneName>string</zoneName>
      <number>int</number>
    </NumReleaseNumber>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор номера

zoneName

Название зоны нумератора

number

Освобождаемый номер

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumReleaseNumberResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumReleaseNumber" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumReleaseNumber xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <numeratorId>A16E9792-5B88-E511-9417-90E6BA57B9F8</numeratorId>
      <zoneName>SimpleZone</zoneName>
      <number>5</number>
    </NumReleaseNumber>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

NumReleaseNumberByID

Освобождает занятый номер по его идентификатору в нумераторе.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumReleaseNumberByID"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumReleaseNumberByID xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <NumberId>guid</NumberId>
    </NumReleaseNumberByID>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор номера

NumberId

Идентификатор занятого номера

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumReleaseNumberByIDResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

NumReleaseNumbers

Освобождает несколько занятых номеров нумератора.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/NumReleaseNumbers"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumReleaseNumbers xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <numeratorId>guid</numeratorId>
      <zoneName>string</zoneName>
      <firstNumber>int</firstNumber>
      <lastNumbers>int</lastNumbers>
    </NumReleaseNumbers>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

numeratorId

Идентификатор номера

zoneName

Название зоны нумератора

firstNumber

Первый номер освобождаемого диапазона

lastNumbers

Последний номер освобождаемого диапазона

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <NumReleaseNumbersResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Файлы

Методы веб-сервиса Docsvision

Файлы

В разделе представлены методы для работы с файлами.

- FileArchive
- FileArchiveGroup
- FileBringOnLine
- Больше подробностей в навигационном меню слева...

FileArchive

Перемещает файл в архив.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileArchive"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileArchive xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <fileId>guid</fileId>
      <delay>boolean</delay>
    </FileArchive>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileId

Идентификатор файла

delay

Если true, то метод отметит файл, как перемещаемый в архив, а непосредственное перемещение будет выполнено позднее

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
```

```
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileArchiveResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

FileArchiveGroup

Перемещает несколько файлов в архив.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileArchiveGroup"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileArchiveGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <items>string</items>
      <delay>boolean</delay>
    </FileArchiveGroup>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

items

Список идентификаторов архивируемых файлов

delay

Если true, то метод отметит файл, как перемещаемый в архив, а

непосредственное перемещение будет выполнено позднее

Строка **items** должна соответствовать формату элемента **GroupOperation**, приведенному в схеме ServerRequests.xsd:

```
<xsd:element name="GroupOperation">
  <xsd:annotation>
    <xsd:documentation>Group operation items</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Item">
        <xsd:complexType>
          <xsd:attribute name="ID" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileArchiveGroupResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <failedItems>string</failedItems>
    </FileArchiveGroupResponse>
  </soap:Body>
</soap:Envelope>
```

FileBringOnLine

Восстанавливает файл из файлового хранилища.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
```

```
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileBringOnLine"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileBringOnLine xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <fileId>guid</fileId>
    </FileBringOnLine>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileId

Идентификатор файла

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileBringOnLineResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

FileBringOnLineGroup

Восстанавливает несколько файлов из файлового хранилища.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileBringOnLineGroup"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileBringOnLineGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <items>string</items>
    </FileBringOnLineGroup>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

items

Список идентификаторов восстанавливаемых файлов

Строка **items** должна соответствовать формату элемента **GroupOperation**, приведенному в схеме ServerRequests.xsd:

```
<xsd:element name="GroupOperation">
  <xsd:complexType>
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Item">
        <xsd:complexType>
          <xsd:attribute name="ID" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileBringOnLineGroupResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <failedItems>string</failedItems>
    </FileBringOnLineGroupResponse>
  </soap:Body>
</soap:Envelope>
```

FileClose

Закрывает открытый файл.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileClose"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileClose xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <fileHandleId>guid</fileHandleId>
      <commitWrite>boolean</commitWrite>
    </FileClose>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileHandleId

Дескриптор открытого файла

commitWrite

Если true, то изменения файла будут сохранены, иначе файл будет закрыт без сохранения изменений

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileCloseResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

FileCopy

Копирует файл.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileCopy"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileCopy xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
```

```
      <fileId>guid</fileId>
    </FileCopy>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileId

Идентификатор копируемого файла

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileCopyResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <newFileId>guid</newFileId>
    </FileCopyResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор созданной копии файла.

FileCreate

Создаёт новый файл.

Данный метод не загружает содержимое файла в созданную в Docsvision сущность — для этого необходимо использовать метод FileWrite.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
```

```
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileCreate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <name>string</name>
    </FileCreate>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

name

Название файл

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileCreateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <fileId>guid</fileId>
    </FileCreateResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор файла.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileCreate" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <name>Z:\\upload_file.docx</name>
    </FileCreate>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}fileId")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор созданного файла.

FileCreateEx

Создаёт новый файл.

Данный метод не загружает содержимое файла в созданную в Docsvision сущность — для этого необходимо использовать метод FileWrite.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileCreateEx"
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileCreateEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <name>string</name>
      <newFileId>guid</newFileId>
    </FileCreateEx>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

name

Название файла

newFileId

Предопределённый идентификатор файл

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileCreateExResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <newFileId>guid</newFileId>
    </FileCreateExResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileCreateEx" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileCreateEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <name>Z:\\upload_file.docx</name>
      <newFileId>2ee9fc01-c28e-e511-9417-90e6ba57b9f1</newFileId>
    </FileCreateEx>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

FileDearchive

Извлекает файл из архива.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileDearchive"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
```

```
  <soap:Body>
    <FileDearchive xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <fileId>guid</fileId>
      <delay>boolean</delay>
    </FileDearchive>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileId

Идентификатор файла

delay

Если true, то метод отметит файл, как извлеченный из архива, а непосредственное извлечение выполнено позднее

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileDearchiveResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

FileDearchiveGroup

Извлекает несколько файлов из архива.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
```

```
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileDearchiveGroup"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileDearchiveGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <items>string</items>
      <delay>boolean</delay>
    </FileDearchiveGroup>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

items

Список идентификаторов архивируемых файлов

delay

Если true, то метод отметит файл, как извлеченный из архива, а непосредственное извлечение выполнено позднее

Строка `items` должна соответствовать формату элемента `GroupOperation`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="GroupOperation">
  <xsd:annotation>
    <xsd:documentation>Group operation items</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Item">
        <xsd:complexType>
          <xsd:attribute name="ID" type="xsd:string" use="required"/>
        </xsd:complexType>
```

```
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileDearchiveGroupResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <failedItems>string</failedItems>
    </FileDearchiveGroupResponse>
  </soap:Body>
</soap:Envelope>
```

FileDelete

Удаляет файл.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileDelete"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileDelete xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <fileId>guid</fileId>
    </FileDelete>
  </soap:Body>
```

```
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileId

Идентификатор файла

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileDeleteResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileDelete" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileDelete xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <fileId>63D891EB-C18E-E511-9417-90E6BA57B9F8</fileId>
    </FileDelete>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

FileExists

Определяет, существует ли указанный файл.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileExists"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileExists xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <fileId>guid</fileId>
    </FileExists>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileId

Идентификатор файла

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
```

```
">
  <soap:Body>
    <FileExistsResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <exists>boolean</exists>
    </FileExistsResponse>
  </soap:Body>
</soap:Envelope>
```

FileFind

Выполняет поиск файлов.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileFind"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileFind xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <searchFilter>string</searchFilter>
    </FileFind>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

searchFilter

Поисковый запрос

Строка `searchFilter` *должна соответствовать формату элемента* `FileSearch`*, приведенному в схеме* `ServerRequests.xsd`*:*

```
<xsd:element name="FileSearch">
  <xsd:complexType>
```

```
    <xsd:sequence>
      <xsd:element name="FullTextQuery" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Mask" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Date" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="After" type="xsd:dateTime"/>
          <xsd:attribute name="Before" type="xsd:dateTime"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Size" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="Min" type="xsd:int"/>
          <xsd:attribute name="Max" type="xsd:int"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileFindResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <foundFiles>string</foundFiles>
    </FileFindResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификаторы найденных файлов.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileFind" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
```

```
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileFind xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>

<searchFilter>&lt;FileSearch&gt;&lt;Mask&gt;*.doc*&lt;/Mask&gt;&lt;/FileSearch&gt;</searc
hFilter>
    </FileFind>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Выводим ответ.

FileGetInfo

Возвращает основную информацию о файле.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileGetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <fileId>guid</fileId>
    </FileGetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileId

Идентификатор файла

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileGetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <fileInfo>string</fileInfo>
    </FileGetInfoResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileGetInfo" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <fileId>2D6F7690-6B81-E511-9417-90E6BA57B9F8</fileId>
    </FileGetInfo>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

```
print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

FileOpen

Открывает файловый курсор.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileOpen"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileOpen xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <fileId>guid</fileId>
      <modify>boolean</modify>
    </FileOpen>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileId

Идентификатор файла

modify

Режим открытия курсора. Если true, то только запись, иначе чтение.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileOpenResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <fileHandleId>guid</fileHandleId>
    </FileOpenResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит указатель на открытый файл.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileOpen" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileOpen xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <fileId>2D6F7690-6B81-E511-9417-90E6BA57B9F8</fileId>
      <modify>false</modify>
    </FileOpen>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}fileHandleId")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор открытого курсора.

FileRead

Возвращает содержимое файла.

Файл, содержимое которого возвращается, должен быть открыт в режиме чтения.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileRead"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileRead xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <fileHandleId>guid</fileHandleId>
    </FileRead>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileHandleId

Указатель на открытый файл

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileReadResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <compressed>boolean</compressed>
      <fileData>base64Binary</fileData>
    </FileReadResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileRead" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileRead xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <fileHandleId>97a1b22d-548f-e511-9417-90e6ba57b9f8</fileHandleId>
    </FileRead>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

FileReplace

Заменяет один файл другим.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileReplace"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileReplace xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <fileId>guid</fileId>
      <newFileId>guid</newFileId>
    </FileReplace>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileId

Идентификатор файла

newFileId

Новый идентификатор файла

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileReplaceResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
```

```
  </soap:Body>
</soap:Envelope>
```

FileSetInfo

Обновляет основную информацию о файле.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileSetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileSetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <fileId>guid</fileId>
      <fileInfo>string</fileInfo>
    </FileSetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileId

Идентификатор файла

fileInfo

Основная информация

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileSetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

FileTakeOffLine

Перемещает файл в файловое хранилище.

Возможность переноса файла в хранилище должна быть активирована администратором Docsvision.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileTakeOffLine"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileTakeOffLine xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <fileId>guid</fileId>
      <autoRestore>boolean</autoRestore>
    </FileTakeOffLine>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileId

Идентификатор файла

autoRestore

Если true, то файл будет возвращен в базу данных по первому запросу к нему

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileTakeOffLineResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

FileTakeOffLineGroup

Переносит в файловое хранилище несколько файлов.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileTakeOffLineGroup"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileTakeOffLineGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <items>string</items>
```

```
      <autoRestore>boolean</autoRestore>
    </FileTakeOffLineGroup>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

items

Список файлов

autoRestore

Если true, то файл из группы будет возвращен в базу данных по первому запросу к нему

Строка `items` должна соответствовать формату элемента `GroupOperation`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="GroupOperation">
  <xsd:annotation>
    <xsd:documentation>Group operation items</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Item">
        <xsd:complexType>
          <xsd:attribute name="ID" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
```

```
">
  <soap:Body>
    <FileTakeOffLineGroupResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <failedItems>string</failedItems>
    </FileTakeOffLineGroupResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификаторы файлов, не перемещенных в хранилище.

FileWrite

Записывает данные в файл.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FileWrite"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileWrite xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <fileHandleId>guid</fileHandleId>
      <compressed>boolean</compressed>
      <fileData>base64Binary</fileData>
    </FileWrite>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fileHandleId

Указатель на открытый файл

compressed

Используется сжатие данных, переданных в `fileData`

fileData

Содержимое файла

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileWriteResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

Пример записи содержимого файла на диске в файл базы данных:

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileWrite" ①

with open("Z:\\Document.docx", "rb") as image_file:
    encoded_string = base64.b64encode(image_file.read()).decode('utf-8')

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileWrite xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <fileHandleId>c40c8af4-5a8f-e511-9417-90e6ba57b9f8</fileHandleId>
      <compressed>false</compressed>
      <fileData>"""+encoded_string+"""</fileData>
    </FileWrite>
  </soap:Body>
</soap:Envelope>""" ②
```

```
response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

Папки

Методы веб-сервиса Docsvision

Папки

В разделе представлены методы для работы с папками.

- RowCopy
- RowCreate
- RowDelete
- Больше подробностей в навигационном меню слева…

FolderCopyData

Копирует папку вместе с ярлыками с возможностью копирования подчиненных папок.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderCopyData"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderCopyData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
```

```
      <folderId>guid</folderId>
      <newParentFolderId>guid</newParentFolderId>
      <folderOptions>string</folderOptions>
    </FolderCopyData>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

folderId

Идентификатор копируемой папки

newParentFolderId

Папка, в которую выполняется копирование

folderOptions

Параметры копирование

Строка `folderOptions` должна соответствовать формату элемента `FolderOptions`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="FolderOptions">
  <xsd:complexType>
    <xsd:attribute name="NewFolderName" type ="xsd:string" use="optional"/>
    <xsd:attribute name="IncludeChildFolders" type ="xsd:boolean" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderCopyDataResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
```

```
      <newFolderId>guid</newFolderId>
    </FolderCopyDataResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор копии папки.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderCopyData" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderCopyData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <folderId>E3768B6E-CDD2-47E7-BF26-F86C9EEC2840</folderId>
      <newParentFolderId>B89B3747-4245-4AEF-94BA-7EB096A734E1</newParentFolderId>
      <folderOptions>&lt;FolderOptions IncludeChildFolders=&quot;1&quot;
NewFolderName=&quot;Copy-Folder&quot;/&gt;</folderOptions>
    </FolderCopyData>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}newFolderId")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор созданной папки.

FolderCopyEx

Копирует папку вместе со ссылками.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderCopyEx"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderCopyEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <folderId>guid</folderId>
      <targetFolderId>guid</targetFolderId>
      <newFolderName>string</newFolderName>
    </FolderCopyEx>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

folderId

Идентификатор копируемой папки

targetFolderId

Папка, в которую выполняется копирование

newFolderName

Название копии папки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderCopyExResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <newFolderId>guid</newFolderId>
    </FolderCopyExResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор созданной папки.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderCopyEx" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderCopyEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <folderId>E3768B6E-CDD2-47E7-BF26-F86C9EEC2840</folderId>
      <targetFolderId>B89B3747-4245-4AEF-94BA-7EB096A734E1</targetFolderId>
      <newFolderName>Folder-Copy</newFolderName>
    </FolderCopyEx>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}newFolderId")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор копии папки.

FolderDelete

Удаляет папку.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderDelete"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderDelete xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <folderId>guid</folderId>
      <permanent>boolean</permanent>
    </FolderDelete>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

folderId

Идентификатор папки

permanent

Если `false`, то папка только помечается как удалённая.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
```

```
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderDeleteResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

FolderFindHardLink

Возвращает идентификатор сильного ярлыка на карточку.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderFindHardLink"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderFindHardLink xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <folderCardId>guid</folderCardId>
      <cardId>guid</cardId>
    </FolderFindHardLink>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

folderCardId

Идентификатор карточки папок. Используется только для формирования ответа с ошибкой.

cardId

Идентификатор карточки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderFindHardLinkResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <shortcutId>guid</shortcutId>
    </FolderFindHardLinkResponse>
  </soap:Body>
</soap:Envelope>
```

FolderGetInfoGroup

Возвращает массив данных о нескольких папках.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderGetInfoGroup"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderGetInfoGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <folderIds>string</folderIds>
      <folderInfoOptions>string</folderInfoOptions>
    </FolderGetInfoGroup>
  </soap:Body>
```

```
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

folderIds

Идентификаторы папок

folderInfoOptions

Дополнительные параметры запроса

Строка `folderIds` должна соответствовать формату элемента `GroupOperation`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="GroupOperation">
  <xsd:complexType>
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Item">
        <xsd:complexType>
          <xsd:attribute name="ID" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Строка `folderInfoOptions` должна соответствовать формату элемента `FolderInfoOptions`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="FolderInfoOptions">
  <xsd:complexType>
    <xsd:attribute name="IncludeArchived" type="xsd:boolean"/>
    <xsd:attribute name="IncludeDeleted" type="xsd:boolean"/>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderGetInfoGroupResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <cursorInfo>
        <ID>guid</ID>
        <PageCount>unsignedInt</PageCount>
        <RowCount>unsignedInt</RowCount>
        <FirstPage>string</FirstPage>
        <UsingCompression>boolean</UsingCompression>
      </cursorInfo>
    </FolderGetInfoGroupResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит количество непрочитанных карточек и количество подчиненных папок. Серверный курсор будет открыт при необходимости.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderGetInfoGroup" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderGetInfoGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <folderIds>&lt;GroupOperation&gt;&lt;Item ID=&quot;E3768B6E-CDD2-47E7-BF26-
F86C9EEC2840&quot;/&gt;&lt;Item ID=&quot;B89B3747-4245-4AEF-94BA-
7EB096A734E1&quot;/&gt;&lt;/GroupOperation&gt;</folderIds>
      <folderInfoOptions>&lt;FolderInfoOptions IncludeArchived=&quot;false&quot;
IncludeDeleted=&quot;false&quot;/&gt;</folderInfoOptions>
    </FolderGetInfoGroup>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

```
print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

FolderGetShortcuts

Возвращает идентификаторы ярлыков на карточку.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderGetShortcuts"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderGetShortcuts xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <folderCardId>guid</folderCardId>
      <cardId>guid</cardId>
    </FolderGetShortcuts>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

folderCardId

Идентификатор карточки папок. Используется только для формирования ответа с ошибкой.

cardId

Идентификатор карточки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderGetShortcutsResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <shortcuts>string</shortcuts>
      <usingCompression>boolean</usingCompression>
    </FolderGetShortcutsResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderGetShortcuts" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderGetShortcuts xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <folderCardId>DA86FABF-4DD7-4A86-B6FF-C58C24D12DE2</folderCardId>
      <cardId>E15E24D5-5D8F-E511-9417-90E6BA57B9F8</cardId>
    </FolderGetShortcuts>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

FolderGetUnreadCount

Возвращает количество непрочитанных карточек в папке.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderGetUnreadCount"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderGetUnreadCount xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <folderId>guid</folderId>
    </FolderGetUnreadCount>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

folderId

Идентификатор папки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderGetUnreadCountResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <unread>int</unread>
    </FolderGetUnreadCountResponse>
  </soap:Body>
</soap:Envelope>
```

FolderGetUnreadCountEx

Возвращает количество непрочитанных карточек в папке.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderGetUnreadCountEx"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderGetUnreadCountEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <folderId>guid</folderId>
      <includeArchived>boolean</includeArchived>
    </FolderGetUnreadCountEx>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

folderId

Идентификатор папки

includeArchived

Учитывать архивированные карточки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderGetUnreadCountExResponse xmlns="http://schemas.docsvision.com/Platform/2009-
02-03/StorageServer/">
      <unread>int</unread>
    </FolderGetUnreadCountExResponse>
  </soap:Body>
</soap:Envelope>
```

FolderMakeHardLink

Преобразует слабый ярлык в сильный.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderMakeHardLink"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderMakeHardLink xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <cardId>guid</cardId>
```

```
      <shortcutId>guid</shortcutId>
    </FolderMakeHardLink>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardId

Идентификатор карточки

shortcutId

Идентификатор ярлыка

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderMakeHardLinkResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

FolderMarkAll

Отмечает все карточки в папке как прочитанные.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderMarkAll"
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderMarkAll xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <folderId>guid</folderId>
      <read>boolean</read>
    </FolderMarkAll>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

folderId

Идентификатор папки

folderId

Если true, то все карточки отмечаются как прочитанные, иначе отмечаются как не прочитанные

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderMarkAllResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

FolderPurge

Удаляет карточки, помеченные к удалению.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/FolderPurge"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderPurge xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <folderId>guid</folderId>
      <recursive>boolean</recursive>
      <maxDate>double</maxDate>
    </FolderPurge>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

folderId

Идентификатор папки, из которой удаляются папки

recursive

Не проверяется

maxDate

Дата OLE-автоматизации, которой ограничивается удаление карточек. Будут удалены карточки, время удаления которых меньше указанного значения.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderPurgeResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderPurge" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderPurge xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <folderId>E3768B6E-CDD2-47E7-BF26-F86C9EEC2840</folderId>
      <recursive>false</recursive>
      <maxDate>42370</maxDate>
    </FolderPurge>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

FolderSetCardDescriptor

Устанавливает дескриптор безопасности на карточку.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FolderSetCardDescriptor"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderSetCardDescriptor xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <cardId>guid</cardId>
      <descBytes>base64Binary</descBytes>
      <securityInfo>int</securityInfo>
      <flags>int</flags>
    </FolderSetCardDescriptor>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardId

Идентификатор карточки

descBytes

Дескриптор безопасности

securityInfo

Раздел дескриптора безопасности

flags

Дополнительные флаги:

- `0` — без флага.
- `1` — распространить политики на слабые ярлыки.
- `2` — распространить политики на сильные ярлыки.

Ответ

```
HTTP/1.1 200 OK
```

```
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FolderSetCardDescriptorResponse xmlns="http://schemas.docsvision.com/Platform/2009-
02-03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Поиск

Методы веб-сервиса Docsvision

Поиск

В разделе представлены методы для работы с поиском.

- SearchCards
- SearchCardsEx
- SearchRows

SearchCards

Возвращает результат выполнения поискового запроса.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/SearchCards"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SearchCards xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
```

```
">
      <sessionId>guid</sessionId>
      <filter>string</filter>
    </SearchCards>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

filter

Поисковый запрос в виде xml-документа. Сформировать запрос можно, в том числе, утилитой "Docsvision Explorer", входящей в Resource Kit.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SearchCardsResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <cards>string</cards>
      <usingCompression>boolean</usingCompression>
    </SearchCardsResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит список найденных карточек. Для каждой карточки указывается: идентификатор, тип, описание, дату создания и изменения, отметку об удалении и признак шаблона.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SearchCards" ①

body = """<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SearchCards xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <filter>&lt;Search AllCards=&quot;true&quot;&gt;&lt;Scope&gt;&lt;CardTypes
Enabled=&quot;true&quot;&gt;&lt;CardType&gt;{B9F7BFD7-7429-455E-A3F1-
94FFB569C794}&lt;/CardType&gt;&lt;/CardTypes&gt;&lt;/Scope&gt;&lt;/Search&gt;</filter>
    </SearchCards>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

SearchCardsEx

Возвращает результат выполнения поискового запроса.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SearchCardsEx"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SearchCardsEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <search>string</search>
```

```
      <flags>int</flags>
      <parameters>string</parameters>
    </SearchCardsEx>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

search

Поисковый запрос в виде xml-документ. Сформировать запрос можно, в том числе, утилитой "Docsvision Explorer" (входит в Resource Kit).

flags

Дополнительные флаг: 0 — отсутствует; 1- в `search` передан идентификатор сохранённого поискового запроса, а не его содержимое

parameters

Параметры запроса, если поисковый запрос использует их

Строка **parameters** должна соответствовать формату элемента **MethodCall**, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="MethodCall">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:simpleContent>
            <xsd:extension base="xsd:string">
              <xsd:attribute name="Name" type="xsd:string" use="optional"/>
              <xsd:attribute name="Type" use="required">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="bool"/>
                    <xsd:enumeration value="byte"/>
                    <xsd:enumeration value="short"/>
                    <xsd:enumeration value="integer"/>
                    <xsd:enumeration value="long"/>
                    <xsd:enumeration value="single"/>
                    <xsd:enumeration value="double"/>
                    <xsd:enumeration value="decimal"/>
```

```
                    <xsd:enumeration value="uniqueidentifier"/>
                    <xsd:enumeration value="char"/>
                    <xsd:enumeration value="string"/>
                    <xsd:enumeration value="datetime"/>
                    <xsd:enumeration value="text"/>
                    <xsd:enumeration value="binary"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:attribute>
            </xsd:extension>
          </xsd:simpleContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="MethodName" type="xsd:string"/>
    <xsd:attribute name="ObjectName" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SearchCardsExResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <cursorInfo>
        <ID>guid</ID>
        <PageCount>unsignedInt</PageCount>
        <RowCount>unsignedInt</RowCount>
        <FirstPage>string</FirstPage>
        <UsingCompression>boolean</UsingCompression>
      </cursorInfo>
    </SearchCardsExResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит список карточек. Серверный курсор будет открыт при необходимости.

Пример запроса

```
# Изменяем цель запроса в заголовке запроса
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SearchCardsEx"

# SOAP-сообщение
body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SearchCardsEx xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <search>&lt;Search
CombineResults=&quot;OR&quot;&gt;&lt;AttributiveSearch&gt;&lt;CardTypeQuery
CardTypeID=&quot;{00000000-0000-0000-0000-000000000001}&quot;&gt;&lt;SectionQuery
Version=&quot;4300&quot; SectionTypeID=&quot;{00000000-0000-0000-0000-
000000000002}&quot;&gt;&lt;ConditionGroup Alias=&quot;alias2&quot;
Operation=&quot;OR&quot;&gt;&lt;Condition Alias=&quot;alias3&quot;
ParameterAlias=&quot;paramalias1&quot; SystemField=&quot;true&quot;
Parameter=&quot;true&quot; ParameterID=&quot;{5A3336D7-F7C6-45C9-8628-9714412ABD50}&quot;
ParameterName=&quot;CardId&quot;&gt;&lt;Field
FieldType=&quot;string&quot;&gt;InstanceID&lt;/Field&gt;&lt;Op&gt;EQ&lt;/Op&gt;&lt;Value&
gt;&#39;&#39;&lt;/Value&gt;&lt;/Condition&gt;&lt;/ConditionGroup&gt;&lt;Options
Limit=&quot;-
1&quot;/&gt;&lt;/SectionQuery&gt;&lt;/CardTypeQuery&gt;&lt;/AttributiveSearch&gt;&lt;Scop
e/&gt;&lt;Params&gt;&lt;Param Alias=&quot;paramalias1&quot; Name=&quot;CardId&quot;
Type=&quot;string&quot; Flags=&quot;0&quot;
Value=&quot;&quot;/&gt;&lt;/Params&gt;&lt;/Search&gt;</search>
      <flags>0</flags>
      <parameters>&lt;MethodCall&gt;&lt;Parameter Name=&quot;paramalias1&quot;
Type=&quot;uniqueidentifier&quot;&gt;B289C6D4-8A8F-E511-9417-
90E6BA57B9F8&lt;/Parameter&gt;&lt;/MethodCall&gt;</parameters>
    </SearchCardsEx>
  </soap:Body>
</soap:Envelope>"""

# Отправляем SOAP-сообщение
response = session.post(url, data=body)

# Ответ SOAP
print(response.content)
```

SearchRows

Выполняет поиск строк в секции карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/SearchRows"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SearchRows xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
      <sectionId>guid</sectionId>
      <filter>string</filter>
    </SearchRows>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

sectionId

Идентификатор секции

filter

Поисковый запрос

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SearchRowsResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <rows>string</rows>
      <usingCompression>boolean</usingCompression>
    </SearchRowsResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит данные всех найденных строк.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SearchRows" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SearchRows xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>6710B92A-E148-4363-8A6F-1AA0EB18936C</instanceID>
      <sectionId>CFDFE60A-21A8-4010-84E9-9D2DF348508C</sectionId>
      <filter>&lt;SectionQuery&gt;&lt;ConditionGroup&gt;&lt;Condition&gt;&lt;Field
FieldType=&quot;unistring&quot;&gt;Name&lt;/Field&gt;&lt;Op&gt;EQ&lt;/Op&gt;&lt;Value&gt;
Analysts&lt;/Value&gt;&lt;/Condition&gt;&lt;/ConditionGroup&gt;&lt;/SectionQuery&gt;</fil
ter>
    </SearchRows>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

Блокировки

Методы веб-сервиса Docsvision

Блокировки

В разделе представлены методы для работы с блокировками.

- LockClear
- LockClearGroup
- LockGetInfo
- Больше подробностей в навигационном меню слева...

LockClear

Снимает блокировку с объекта.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/LockClear"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockClear xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <resourceId>guid</resourceId>
      <forceUnlock>boolean</forceUnlock>
    </LockClear>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

resourceId

Идентификатор объекта: карточки, файла или строки секции

forceUnlock

Для снятия блокировки, установленной в другой сессии, указать `true`. Для принудительной разблокировки необходимы права администратора Docsvision.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockClearResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = 'http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/LockClear' ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockClear xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>eccccc21-aa92-e511-9417-90e6ba57b9f8</sessionId>
      <resourceId>A08B9D3E-A892-E511-9417-90E6BA57B9F8</resourceId>
      <forceUnlock>true</forceUnlock>
    </LockClear>
  </soap:Body>
</soap:Envelope>""" ②
```

```
response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

LockClearGroup

Снимает блокировку с нескольких объектов.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/LockClearGroup"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockClearGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <items>string</items>
      <forceUnlock>boolean</forceUnlock>
    </LockClearGroup>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

items

Список объектов

forceUnlock

Для снятия блокировки, установленной в другой сессии, указать `true`. Для

принудительной разблокировки необходимы права администратора Docsvision.

Строка `items` должна соответствовать формату элемента `GroupOperation`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="GroupOperation">
  <xsd:complexType>
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Item">
        <xsd:complexType>
          <xsd:attribute name="ID" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockClearGroupResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <failedItems>string</failedItems>
    </LockClearGroupResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит список объектов, которые не были разблокированы из-за ошибки.

LockGetInfo

Возвращает детальную информацию о блокировке, установленной на объект.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/LockGetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <resourceId>guid</resourceId>
    </LockGetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

resourceId

Идентификатор объекта, для которого запрашивается информация

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockGetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <lockStatus>STATUS_FREE or STATUS_LOCKED or STATUS_CHECKED_OUT or
STATUS_OWNER_LOCKED or STATUS_OWNER_CHECKED_OUT</lockStatus>
      <ownerName>string</ownerName>
    </LockGetInfoResponse>
```

```
  </soap:Body>
</soap:Envelope>
```

LockGetList

Возвращает детальную информацию о всех блокировках объектов.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/LockGetList"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <currentUserOnly>boolean</currentUserOnly>
      <allTypes>boolean</allTypes>
      <resType>CUSTOM or CARD or FILE or ROW</resType>
    </LockGetList>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

currentUserOnly

Если true, то будут выбраны блокировки, установленные текущим пользователем

allTypes

Если true, то будут выбраны блокировки, установленные на все типы объектов, иначе на указанные в `resType`

resType

Тип заблокированного объекта: `CUSTOM` (неизвестен), `CARD` (карточка), `FILE` (файл), `ROW` (строка секции)

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockGetListResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <lockedObjects>string</lockedObjects>
      <usingCompression>boolean</usingCompression>
    </LockGetListResponse>
  </soap:Body>
</soap:Envelope>
```

LockSet

Устанавливает блокировку на объект.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/LockSet"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockSet xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <resourceId>guid</resourceId>
      <resType>CUSTOM or CARD or FILE or ROW</resType>
      <sectionId>guid</sectionId>
      <instanceID>guid</instanceID>
```

```
      <checkOut>boolean</checkOut>
    </LockSet>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

resourceId

Идентификатор блокируемого объекта: файл, карточка, либо строка секции

resType

Тип блокируемого объекта: `CUSTOM` (неизвестен), `CARD` (карточка), `FILE` (файл), `ROW` (строка секции)

sectionId

Идентификатор секции, если блокируется строка секции

instanceID

Идентификатор карточки, если блокируется строка секции

checkOut

Если `true`, то устанавливается постоянная блокировка, иначе временная

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockSetResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/LockSet" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LockSet xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <resourceId>B289C6D4-8A8F-E511-9417-90E6BA57B9F8</resourceId>
      <resType>CARD</resType>
      <sectionId>00000000-0000-0000-0000-000000000000</sectionId>
      <instanceID>00000000-0000-0000-0000-000000000000</instanceID>
      <checkOut>true</checkOut>
    </LockSet>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

Библиотеки карточек

Методы веб-сервиса Docsvision

Библиотеки карточек

В разделе представлены методы для работы с библиотеками карточек.

- CardLibGetInfo
- CardLibGetInfoInstall
- CardLibGetList

CardLibGetInfo

Возвращает информацию о библиотеки карточек.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardLibGetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardLibGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <cardLibId>guid</cardLibId>
      <infoType>CARD_TYPE_SCHEMA or CARD_TYPE_ICON or CARD_TYPE_XSD</infoType>
    </CardLibGetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardLibId

Идентификатор библиотеки карточек

infoType

Тип запрашиваемых данных:

- `CARD_TYPE_SCHEMA` — возвращает схему.
- `CARD_TYPE_ICON` — возвращает иконку библиотеки.
- `CARD_TYPE_XSD`.

Ответ

```
HTTP/1.1 200 OK
```

```
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardLibGetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <cardLibInfo>string</cardLibInfo>
      <usingCompression>boolean</usingCompression>
    </CardLibGetInfoResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит запрашиваемые данные (`cardLibInfo`), которые зависят от переданного `infoType`, а также признак сжатия (`usingCompression`) возвращенных данных.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardLibGetInfo" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardLibGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <cardLibId>DE9C2809-3207-42EF-BC32-B9AB2D1B5608</cardLibId>
      <infoType>CARD_TYPE_SCHEMA</infoType>
    </CardLibGetInfo>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

CardLibGetInfoInstall

Возвращает информацию о пакетах установки библиотеки карточек.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardLibGetInfoInstall"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardLibGetInfoInstall xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <cardLibId>guid</cardLibId>
    </CardLibGetInfoInstall>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardLibId

Идентификатор библиотеки карточек

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
```

```
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardLibGetInfoInstallResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <cardLibInfo>string</cardLibInfo>
      <usingCompression>boolean</usingCompression>
    </CardLibGetInfoInstallResponse>
  </soap:Body>
</soap:Envelope>
```

Метод возвращает метаданные (`cardLibInfo`) библиотеки карточек, указанные в разделе `Installers` библиотеки карточек, а также признак сжатия (`usingCompression`) возвращенных данных.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardLibGetInfoInstall" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardLibGetInfoInstall xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
      <cardLibId>26DBF2C0-E72E-4E9E-AB99-816758387AE2</cardLibId>
    </CardLibGetInfoInstall>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

CardLibGetList

Возвращает метаданные всех библиотек карточек, зарегистрированных на сервере.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardLibGetList"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardLibGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
    </CardLibGetList>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardLibGetListResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <cardLibInfo>string</cardLibInfo>
      <usingCompression>boolean</usingCompression>
    </CardLibGetListResponse>
```

```
  </soap:Body>
</soap:Envelope>
```

Ответ содержит метаданные (`cardLibInfo`) всех библиотек карточек, а также признак сжатия возвращенных данных.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardLibGetList" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardLibGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>1d29bba9-6c8c-e511-9417-90e6ba57b9f8</sessionId>
    </CardLibGetList>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

Цветовые метки карточки

Методы веб-сервиса Docsvision

Цветовые метки карточки

В разделе представлены методы для работы с цветовыми метками карточек.

- LabelClearCard

- LabelCreate
- LabelDelete
- Больше подробностей в навигационном меню слева...

LabelClearCard

Удаляет цветовую метку у карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/LabelClearCard"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelClearCard xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <cardId>guid</cardId>
    </LabelClearCard>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardId

Идентификатор карточки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelClearCardResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

LabelCreate

Создаёт новую цветовую метку.

Метка будет добавлена только текущему пользователю.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/LabelCreate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <xmlLabel>string</xmlLabel>
    </LabelCreate>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

xmlLabel

Данные добавляемой цветовой метки

Строка `xmlLabel` должна соответствовать формату элемента `CardLabelInfo`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:complexType name="CardLabelInfo">
  <xsd:sequence>
    <xsd:element name="Description" type="xsd:string" />
  </xsd:sequence>
  <xsd:attribute name="ID" type="ms:guid" use="required"/>
  <xsd:attribute name="Color" type="xsd:int" use="required"/>
  <xsd:attribute name="Name" type="xsd:string"/>
</xsd:complexType>
```

Значение `Color` выбирается из цветовой схемы ARGB.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelCreateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <labelId>guid</labelId>
    </LabelCreateResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор добавленной цветовой метки.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/LabelCreate" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
```

```
    <LabelCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <xmlLabel>&lt;CardLabelInfo Color=&quot;-16711937&quot;
Name=&quot;NewColor&quot;&gt;&lt;Description&gt;New
color&lt;/Description&gt;&lt;/CardLabelInfo&gt;</xmlLabel>
    </LabelCreate>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

LabelDelete

Удаляет цветовую метку.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/LabelDelete"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelDelete xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <labelId>guid</labelId>
    </LabelDelete>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

labelId

Идентификатор удаляемой цветовой метки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelDeleteResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

LabelGetList

Возвращает список цветовых меток, доступных пользователю.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/LabelGetList"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
    </LabelGetList>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelGetListResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <xmlLabels>string</xmlLabels>
    </LabelGetListResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор, цвет и название метки.

LabelSetCard

Устанавливает или снимает цветовую метку карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/LabelSetCard"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelSetCard xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <cardId>guid</cardId>
```

```
      <labelId>guid</labelId>
      <enable>boolean</enable>
    </LabelSetCard>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardId

Идентификатор карточки

labelId

Идентификатор цветовой метки

enable

Если `true`, то метка устанавливается, иначе — удаляется

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelSetCardResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

LabelSetInfo

Обновляет данные цветовой метки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/LabelSetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelSetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <xmlLabel>string</xmlLabel>
    </LabelSetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

xmlLabel

Новые данные цветовой метки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LabelSetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Расширенные метаданные

Методы веб-сервиса Docsvision

Расширенные метаданные

В разделе представлены методы для работы с расширенными метаданными.

- DynamicFieldCreate
- DynamicFieldDelete
- DynamicFieldGetInfo
- Больше подробностей в навигационном меню слева...

DynamicFieldCreate

Добавляет в секцию расширенное поле.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicFieldCreate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicFieldCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
      <fieldXml>string</fieldXml>
    </DynamicFieldCreate>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

fieldXml

Метаданные добавляемого поля

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicFieldCreateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <newFieldId>guid</newFieldId>
    </DynamicFieldCreateResponse>
  </soap:Body>
</soap:Envelope>
```

DynamicFieldDelete

Удаляет расширенное поле.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicFieldDelete"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
```

```
    <DynamicFieldDelete xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <fieldId>guid</fieldId>
    </DynamicFieldDelete>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fieldId

Идентификатор расширенного поля

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicFieldDeleteResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicFieldDelete" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicFieldDelete xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
```

```
      <fieldId>E0D0493F-1C6F-4947-87FE-45196BD9D793</fieldId>
    </DynamicFieldDelete>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

DynamicFieldGetInfo

Возвращает метаданные расширенного поля.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicFieldGetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicFieldGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <fieldId>guid</fieldId>
    </DynamicFieldGetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

fieldId

Идентификатор расширенного поля

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicFieldGetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <fieldXml>string</fieldXml>
    </DynamicFieldGetInfoResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicFieldGetInfo" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicFieldGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <fieldId>65DCA871-6237-49A9-9B01-525029D1510C</fieldId>
    </DynamicFieldGetInfo>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

DynamicFieldUpdate

Обновляет метаданные расширенного поля.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicFieldUpdate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicFieldUpdate xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <fieldXml>string</fieldXml>
    </DynamicFieldUpdate>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

fieldXml

Метаданные добавляемого поля

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
```

```
">
  <soap:Body>
    <DynamicFieldUpdateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

DynamicMetadataGetInfo

Возвращает расширенные метаданные для всех типов карточек.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicMetadataGetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicMetadataGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
    </DynamicMetadataGetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
```

```
">
  <soap:Body>
    <DynamicMetadataGetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-
02-03/StorageServer/">
      <metadataXml>string</metadataXml>
    </DynamicMetadataGetInfoResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicMetadataGetInfo" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicMetadataGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
    </DynamicMetadataGetInfo>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

DynamicMetadataUpdate

Обновляет расширенные метаданные.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicMetadataUpdate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicMetadataUpdate xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <metadataXml>string</metadataXml>
    </DynamicMetadataUpdate>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

metadataXml

Метаданные

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicMetadataUpdateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicMetadataUpdate" ①
```

```
body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicMetadataUpdate xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <metadataXml>&lt;?xml version=&quot;1.0&quot; encoding=&quot;utf-
16&quot;?&gt;&lt;DynamicMetadataDefinition
xmlns:xsi=&quot;http://www.w3.org/2001/XMLSchema-instance&quot;
xmlns:xsd=&quot;http://www.w3.org/2001/XMLSchema&quot;&gt;&lt;Card ID=&quot;b9f7bfd7-
7429-455e-a3f1-94ffb569c794&quot;&gt;&lt;Section Alias=&quot;SimpleSection&quot;
Type=&quot;struct&quot; ID=&quot;65BA351C-4FA1-487D-9B2C-FE89C96ACE1E&quot;
SimpleSecurity=&quot;true&quot;
Dynamic=&quot;true&quot;&gt;&lt;Name&gt;&lt;LocalizedString
Language=&quot;en&quot;&gt;SimpleSection&lt;/LocalizedString&gt;&lt;/Name&gt;&lt;Field
Alias=&quot;NewExtField&quot; ID=&quot;65DCA871-6237-49A9-9B01-525029D1510C&quot;
Type=&quot;int&quot; DefaultValue=&quot;&quot; CopyBehavior=&quot;Null&quot;
Dynamic=&quot;true&quot;&gt;&lt;Name&gt;&lt;LocalizedString
Language=&quot;en&quot;&gt;Desc NewExtField
Override&lt;/LocalizedString&gt;&lt;/Name&gt;&lt;Description
/&gt;&lt;/Field&gt;&lt;DisplayFields
/&gt;&lt;/Section&gt;&lt;/Card&gt;&lt;/DynamicMetadataDefinition&gt;</metadataXml>
    </DynamicMetadataUpdate>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

DynamicSectionCreate

Создаёт расширенную секцию в типе карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
```

```
03/StorageServer/DynamicSectionCreate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicSectionCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <cardTypeId>guid</cardTypeId>
      <parentSectionId>guid</parentSectionId>
      <sectionXml>string</sectionXml>
    </DynamicSectionCreate>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cardTypeId

Идентификатор типа карточки

parentSectionId

Идентификатор родительской секции

sectionXml

Метаданные расширенной секции

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicSectionCreateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <newSectionId>guid</newSectionId>
```

```
    </DynamicSectionCreateResponse>
  </soap:Body>
</soap:Envelope>
```

DynamicSectionDelete

Удаляет расширенную секцию.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicSectionDelete"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicSectionDelete xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
    </DynamicSectionDelete>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор расширенной секции

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
```

```
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicSectionDeleteResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

DynamicSectionGetInfo

Возвращает метаданные расширенной секции.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicSectionGetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicSectionGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
    </DynamicSectionGetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор расширенной секции

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicSectionGetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sectionXml>string</sectionXml>
    </DynamicSectionGetInfoResponse>
  </soap:Body>
</soap:Envelope>
```

DynamicSectionUpdate

Обновляет метаданные расширенной секции.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DynamicSectionUpdate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicSectionUpdate xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionXml>string</sectionXml>
    </DynamicSectionUpdate>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionXml

Метаданные секции

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DynamicSectionUpdateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Курсоры

Методы веб-сервиса Docsvision

Курсоры

В разделе представлены методы для работы с курсорами.

- CursorClose
- CursorGetPageCount
- CursorOpenCardData
- Больше подробностей в навигационном меню слева...

CursorClose

Закрывает открытый серверный курсор.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/CursorClose"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorClose xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <cursorId>guid</cursorId>
    </CursorClose>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cursorId

Идентификатор открытого серверного курсора

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorCloseResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorClose" ①

body = """<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorClose xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
    <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <cursorId>e31fb0ba-f88d-e511-9417-90e6ba57b9f8</cursorId>
    </CursorClose>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

CursorGetPageCount

Возвращает количество страниц в открытом серверном курсоре.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorGetPageCount"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorGetPageCount xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <cursorId>guid</cursorId>
    </CursorGetPageCount>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cursorId

Идентификатор открытого серверного курсора

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorGetPageCountResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <count>unsignedInt</count>
    </CursorGetPageCountResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorGetPageCount" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorGetPageCount xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <cursorId>7bcb0c2e-f78d-e511-9417-90e6ba57b9f8</cursorId>
    </CursorGetPageCount>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

```
responseParsed = ElementTree.fromstring(response.content)
print(responseParsed.findtext( ".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}count")) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим количество страницы.

CursorOpenCardData

Возвращает данные карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorOpenCardData"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorOpenCardData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
      <timestamp>long</timestamp>
    </CursorOpenCardData>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

timestamp

Метка времени

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorOpenCardDataResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <timestamp>long</timestamp>
      <cursorInfo>
        <ID>guid</ID>
        <PageCount>unsignedInt</PageCount>
        <RowCount>unsignedInt</RowCount>
        <FirstPage>string</FirstPage>
        <UsingCompression>boolean</UsingCompression>
      </cursorInfo>
    </CursorOpenCardDataResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит метку времени (timestamp) и данные карточки (FirstPage). При необходимости будет открыт серверный курсор.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorOpenCardData" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorOpenCardData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <instanceID>6710B92A-E148-4363-8A6F-1AA0EB18936C</instanceID>
```

```
      <timestamp>343451</timestamp>
    </CursorOpenCardData>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

CursorOpenSectionData

Возвращает данные секции карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorOpenSectionData"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorOpenSectionData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <objectInfo>
        <InstanceID>guid</InstanceID>
        <SectionTypeID>guid</SectionTypeID>
        <ParentID>guid</ParentID>
      </objectInfo>
      <readType>READ_SECTION or READ_SUBSECTION or READ_TREE or READ_TREE_LEVEL or
READ_FIRST_TREE_LEVEL or READ_SEARCH_RESULTS</readType>
      <timestamp>long</timestamp>
    </CursorOpenSectionData>
  </soap:Body>
```

```
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

InstanceID

Идентификатор карточки

SectionTypeID

Идентификатор карточки

ParentID

Идентификатор родительского, по отношению к выбираемым данным, объекта, ограничивающего выборку при чтении иерархической секции

readType

Тип чтения:

- `READ_SECTION` — читать всю секцию.
- `READ_SUBSECTION` — читать подсекцию.
- `READ_TREE` — читать всё дерево.
- `READ_TREE_LEVEL` — читать уровень дерева, находящийся не на первом уровне иерархии.
- `READ_FIRST_TREE_LEVEL` — читать первый уровень дерева.
- `READ_SEARCH_RESULTS` — читать результаты поиска.

timestamp

Метка времени

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
```

```
  <soap:Body>
    <CursorOpenSectionDataResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <timestamp>long</timestamp>
      <cursorInfo>
        <ID>guid</ID>
        <PageCount>unsignedInt</PageCount>
        <RowCount>unsignedInt</RowCount>
        <FirstPage>string</FirstPage>
        <UsingCompression>boolean</UsingCompression>
      </cursorInfo>
    </CursorOpenSectionDataResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит метку времени (`timestamp`) и данные секции (`FirstPage`). При необходимости будет открыт серверный курсор.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorOpenSectionData" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorOpenSectionData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <objectInfo>
        <InstanceID>DA86FABF-4DD7-4A86-B6FF-C58C24D12DE2</InstanceID>
        <SectionTypeID>FE27631D-EEEA-4E2E-A04C-D4351282FB55</SectionTypeID>
        <ParentID>FFFFFFFF-FFFF-0003-FFFF-000000000000</ParentID>
      </objectInfo>
      <readType>READ_TREE_LEVEL</readType>
      <timestamp>33451</timestamp>
    </CursorOpenSectionData>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

CursorReadPage

Читает страницу данных из открытого серверного курсора.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorReadPage"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorReadPage xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <cursorId>guid</cursorId>
      <pageNum>unsignedInt</pageNum>
    </CursorReadPage>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

cursorId

Идентификатор открытого курсора

pageNum

Номер запрашиваемой страницы

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorReadPageResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <data>string</data>
      <usingCompression>boolean</usingCompression>
    </CursorReadPageResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorReadPage" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorReadPage xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <cursorId>7bcb0c2e-f78d-e511-9417-90e6ba57b9f8</cursorId>
      <pageNum>2</pageNum>
    </CursorReadPage>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

CursorRefreshCardData

Возвращает данные карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorRefreshCardData"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorRefreshCardData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceId>guid</instanceId>
      <timestamp>long</timestamp>
    </CursorRefreshCardData>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceId

Идентификатор карточки

timestamp

Метка времени. Если `timestamp` соответствует метке времени на сервере, то данные возвращены не будут.

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorRefreshCardDataResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <timestamp>long</timestamp>
      <cursorInfo>
        <ID>guid</ID>
        <PageCount>unsignedInt</PageCount>
        <RowCount>unsignedInt</RowCount>
        <FirstPage>string</FirstPage>
        <UsingCompression>boolean</UsingCompression>
      </cursorInfo>
    </CursorRefreshCardDataResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит данные, если существующие в кэше данные устарели. Актуальность определяется на основе метки времени.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorRefreshCardData" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorRefreshCardData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <instanceId>6710B92A-E148-4363-8A6F-1AA0EB18936C</instanceId>
      <timestamp>377104</timestamp>
    </CursorRefreshCardData>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

CursorRefreshSectionData

Возвращает данные секции карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorRefreshSectionData"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorRefreshSectionData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <objectInfo>
        <InstanceID>guid</InstanceID>
        <SectionTypeID>guid</SectionTypeID>
        <ParentID>guid</ParentID>
      </objectInfo>
      <readType>READ_SECTION or READ_SUBSECTION or READ_TREE or READ_TREE_LEVEL or
READ_FIRST_TREE_LEVEL or READ_SEARCH_RESULTS</readType>
      <timestamp>long</timestamp>
      <hash>int</hash>
    </CursorRefreshSectionData>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

InstanceID

Идентификатор карточки

SectionTypeID

Идентификатор карточки

ParentID

Идентификатор родительского, по отношению к выбираемым данным, объекта, ограничивающего выборку при чтении иерархической секции

readType

Тип чтения:

- `READ_SECTION` — читать всю секцию.
- `READ_SUBSECTION` — читать подсекцию.
- `READ_TREE` — читать всё дерево.
- `READ_TREE_LEVEL` — читать уровень дерева, находящийся не на первом уровне иерархии.
- `READ_FIRST_TREE_LEVEL` — читать первый уровень дерева.
- `READ_SEARCH_RESULTS` — читать результаты поиска.

timestamp

Метка времени. Если `timestamp` и `hash` соответствуют аналогичным значениям на сервере, то данные возвращены не будут.

hash

Хеш данных

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorRefreshSectionDataResponse xmlns="http://schemas.docsvision.com/Platform/2009-
02-03/StorageServer/">
```

```
      <timestamp>long</timestamp>
      <hash>int</hash>
      <cursorInfo>
        <ID>guid</ID>
        <PageCount>unsignedInt</PageCount>
        <RowCount>unsignedInt</RowCount>
        <FirstPage>string</FirstPage>
        <UsingCompression>boolean</UsingCompression>
      </cursorInfo>
    </CursorRefreshSectionDataResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит данные, если существующие в кэше данные устарели. Актуальность определяется на основе метки времени.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CursorRefreshSectionData" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CursorRefreshSectionData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
       <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <objectInfo>
        <InstanceID>DA86FABF-4DD7-4A86-B6FF-C58C24D12DE2</InstanceID>
        <SectionTypeID>FE27631D-EEEA-4E2E-A04C-D4351282FB55</SectionTypeID>
        <ParentID>FFFFFFFF-FFFF-0003-FFFF-000000000000</ParentID>
      </objectInfo>
      <readType>READ_TREE_LEVEL</readType>
      <timestamp>371933</timestamp>
      <hash>-756518670</hash>
    </CursorRefreshSectionData>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

Иконки

Методы веб-сервиса Docsvision

Иконки

В разделе представлены методы для работы с иконками.

- IconCreate
- IconDelete
- IconGetInfoGroup
- Больше подробностей в навигационном меню слева…

IconCreate

Добавляет новую иконку.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/IconCreate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IconCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <xmlIcon>string</xmlIcon>
    </IconCreate>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

xmlIcon

Данные иконки

Строка `xmlIcon` должна соответствовать формату элемента `IconInfo`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:complexType name="IconInfo">
  <xsd:sequence>
    <xsd:element name="Data" type="xsd:base64Binary"/>
    <xsd:element name="Description" type="xsd:string" />
  </xsd:sequence>
  <xsd:attribute name="ID" type="ms:guid" use="required"/>
  <xsd:attribute name="Name" type="xsd:string"/>
</xsd:complexType>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IconCreateResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <newIconId>guid</newIconId>
    </IconCreateResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор добавленной сущности.

IconDelete

Удаляет иконку.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/IconDelete"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IconDelete xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <iconId>guid</iconId>
    </IconDelete>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

iconId

Идентификатор удаляемой иконки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IconDeleteResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

IconGetInfoGroup

Возвращает данные нескольких иконок.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/IconGetInfoGroup"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IconGetInfoGroup xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <xmlIconGroup>string</xmlIconGroup>
    </IconGetInfoGroup>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

xmlIconGroup

Список идентификатором иконок

Строка `xmlIconGroup` должна соответствовать формату элемента `GroupOperation`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="GroupOperation">
  <xsd:annotation>
    <xsd:documentation>Group operation items</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Item">
        <xsd:complexType>
          <xsd:attribute name="ID" type="xsd:string" use="required"/>
```

```
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IconGetInfoGroupResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <xmlIcons>string</xmlIcons>
      <failedItems>string</failedItems>
    </IconGetInfoGroupResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификаторы элементов, по которым из-за ошибки не были получены данные.

IconGetList

Возвращает данные всех иконок.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/IconGetList"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IconGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
```

```
">
      <sessionId>guid</sessionId>
      <pageItemsCount>int</pageItemsCount>
    </IconGetList>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

pageItemsCount

Максимальное количество элементов на возвращенной странице

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IconGetListResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <cursorInfo>
        <ID>guid</ID>
        <PageCount>unsignedInt</PageCount>
        <RowCount>unsignedInt</RowCount>
        <FirstPage>string</FirstPage>
        <UsingCompression>boolean</UsingCompression>
        <PageSize>unsignedInt</PageSize>
        <CurrentPage>unsignedInt</CurrentPage>
        <CursorType>CURSOR_TYPE_DEFAULT or CURSOR_TYPE_STANDARD or CURSOR_TYPE_KEYSET or
CURSOR_TYPE_ICON</CursorType>
      </cursorInfo>
    </IconGetListResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит данные всех иконок. При необходимости будет открыт серверный курсор.

IconSetInfo

Обновляет данные иконки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/IconSetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IconSetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>guid</sessionId>
      <xmlIcon>string</xmlIcon>
    </IconSetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

xmlIcon

Данные иконки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IconSetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
```

```
  </soap:Body>
</soap:Envelope>
```

Прочее

Методы веб-сервиса Docsvision

Прочее

В разделе представлены прочие методы, не попавшие ни в одну из вышеописанных категорий.

- RowCopy
- RowCreate
- RowDelete
- Больше подробностей в навигационном меню слева…

DecrementFeatureUsage

Уменьшает число активных соединений в счетчике дополнительной опции лицензии.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DecrementFeatureUsage"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DecrementFeatureUsage xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <featureId>guid</featureId>
    </DecrementFeatureUsage>
  </soap:Body>
```

```
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

featureId

Идентификатор дополнительной опции лицензии

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DecrementFeatureUsageResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/DecrementFeatureUsage" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <DecrementFeatureUsage xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <featureId>da212a06-edcd-461f-8134-c5e39f1559f7</featureId>
    </DecrementFeatureUsage>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

ExtensionExecuteCursorMethod

Вызывает метод серверного расширения.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/ExtensionExecuteCursorMethod"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ExtensionExecuteCursorMethod xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <methodCall>string</methodCall>
      <compressed>boolean</compressed>
    </ExtensionExecuteCursorMethod>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

methodCall

Вызываемый метод

compressed

Признак сжатия данных в `methodCall`

Строка `methodCall` должна соответствовать формату элемента `MethodCall`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="MethodCall">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:simpleContent>
            <xsd:extension base="xsd:string">
              <xsd:attribute name="Name" type="xsd:string" use="optional"/>
              <xsd:attribute name="Type" use="required">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="bool"/>
                    <xsd:enumeration value="byte"/>
                    <xsd:enumeration value="short"/>
                    <xsd:enumeration value="integer"/>
                    <xsd:enumeration value="long"/>
                    <xsd:enumeration value="single"/>
                    <xsd:enumeration value="double"/>
                    <xsd:enumeration value="decimal"/>
                    <xsd:enumeration value="uniqueidentifier"/>
                    <xsd:enumeration value="char"/>
                    <xsd:enumeration value="string"/>
                    <xsd:enumeration value="datetime"/>
                    <xsd:enumeration value="text"/>
                    <xsd:enumeration value="binary"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:attribute>
            </xsd:extension>
          </xsd:simpleContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="MethodName" type="xsd:string"/>
    <xsd:attribute name="ObjectName" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
```

```
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ExtensionExecuteCursorMethodResponse xmlns=
"http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <cursorInfo>
        <ID>guid</ID>
        <PageCount>unsignedInt</PageCount>
        <RowCount>unsignedInt</RowCount>
        <FirstPage>string</FirstPage>
        <UsingCompression>boolean</UsingCompression>
      </cursorInfo>
    </ExtensionExecuteCursorMethodResponse>
  </soap:Body>
</soap:Envelope>
```

При необходимости будет открыт серверный курсор на данные.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/ExtensionExecuteCursorMethod" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ExtensionExecuteCursorMethod xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <methodCall>&lt;MethodCall ObjectName=&quot;SampleExtension&quot;
MethodName=&quot;GetCard&quot;&gt;&lt;Parameter Name=&quot;cardId&quot;
Type=&quot;uniqueidentifier&quot; xmlns:dt=&quot;urn:schemas-microsoft-
com:datatypes&quot; dt:dt=&quot;string&quot;&gt;{B87B6FEC-E092-4D37-ABB2-
C13E2000B6E3}&lt;/Parameter&gt;&lt;/MethodCall&gt;</methodCall>
      <compressed>false</compressed>
    </ExtensionExecuteCursorMethod>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

ExtensionExecuteMethod

Вызывает метод серверного расширения.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/ExtensionExecuteMethod"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ExtensionExecuteMethod xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <methodCall>string</methodCall>
      <compressed>boolean</compressed>
    </ExtensionExecuteMethod>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

methodCall

Вызываемый метод

compressed

Признак сжатия данных в `methodCall`

Строка `methodCall` должна соответствовать формату элемента `MethodCall`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="MethodCall">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:simpleContent>
            <xsd:extension base="xsd:string">
              <xsd:attribute name="Name" type="xsd:string" use="optional"/>
              <xsd:attribute name="Type" use="required">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="bool"/>
                    <xsd:enumeration value="byte"/>
                    <xsd:enumeration value="short"/>
                    <xsd:enumeration value="integer"/>
                    <xsd:enumeration value="long"/>
                    <xsd:enumeration value="single"/>
                    <xsd:enumeration value="double"/>
                    <xsd:enumeration value="decimal"/>
                    <xsd:enumeration value="uniqueidentifier"/>
                    <xsd:enumeration value="char"/>
                    <xsd:enumeration value="string"/>
                    <xsd:enumeration value="datetime"/>
                    <xsd:enumeration value="text"/>
                    <xsd:enumeration value="binary"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:attribute>
            </xsd:extension>
          </xsd:simpleContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="MethodName" type="xsd:string"/>
    <xsd:attribute name="ObjectName" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
```

```
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ExtensionExecuteMethodResponse xmlns="http://schemas.docsvision.com/Platform/2009-
02-03/StorageServer/">
      <result>string</result>
    </ExtensionExecuteMethodResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/ExtensionExecuteMethod" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ExtensionExecuteMethod xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
    <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <methodCall>&lt;MethodCall ObjectName=&quot;SampleExtension&quot;
MethodName=&quot;GetCard&quot;&gt;&lt;Parameter Name=&quot;cardId&quot;
Type=&quot;uniqueidentifier&quot; xmlns:dt=&quot;urn:schemas-microsoft-
com:datatypes&quot; dt:dt=&quot;string&quot;&gt;{B87B6FEC-E092-4D37-ABB2-
C13E2000B6E3}&lt;/Parameter&gt;&lt;/MethodCall&gt;</methodCall>
      <compressed>false</compressed>
    </ExtensionExecuteMethod>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

GetServerDateTime

Возвращает серверное время.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/GetServerDateTime"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <GetServerDateTime xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <GetServerDateTimeResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <curDateTime>double</curDateTime>
    </GetServerDateTimeResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит дату OLE-автоматизации.

HealthCheck

Проверяет доступность сервиса.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
```

```
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/HealthCheck"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <HealthCheck xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/"
/>
  </soap:Body>
</soap:Envelope>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <HealthCheckResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <result>string</result>
    </HealthCheckResponse>
  </soap:Body>
</soap:Envelope>
```

Корректный ответ содержит строку `Running`.

IncrementFeatureUsage

Увеличивает число активных соединений в счетчике дополнительной опции лицензии.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/IncrementFeatureUsage"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IncrementFeatureUsage xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <featureId>guid</featureId>
    </IncrementFeatureUsage>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

featureId

Идентификатор дополнительной опции

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <IncrementFeatureUsageResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

LinkClear

Очищает значение ссылочного поля.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/LinkClear"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LinkClear xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>guid</sessionId>
      <sectionId>guid</sectionId>
      <rowId>guid</rowId>
      <fieldName>string</fieldName>
    </LinkClear>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

sectionId

Идентификатор секции

rowId

Идентификатор строки

fieldName

Псевдоним ссылочного поля

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
```

```
">
  <soap:Body>
    <LinkClearResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/" />
  </soap:Body>
</soap:Envelope>
```

LinkGetCardInfo

Возвращает данные о ссылках карточки.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/LinkGetCardInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LinkGetCardInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
      <toCard>boolean</toCard>
      <linkTypes>int</linkTypes>
      <recurseDepth>int</recurseDepth>
    </LinkGetCardInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

toCard

Если true, будут возвращены ссылки на карточку, иначе ссылки карточки

linkTypes

Тип ссылки:

- `0` — не указан.
- `1` — слабая.
- `2` — сильная.
- `4` — автоматическая.
- `255` — все типы.

recurseDepth

Уровень вложенности

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LinkGetCardInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <linksData>string</linksData>
      <usingCompression>boolean</usingCompression>
    </LinkGetCardInfoResponse>
  </soap:Body>
</soap:Envelope>
```

LinkGetParent

Возвращает информацию о владельце ссылки на карточку.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/LinkGetParent"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LinkGetParent xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <instanceID>guid</instanceID>
    </LinkGetParent>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

instanceID

Идентификатор карточки

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <LinkGetParentResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sourceInstanceId>guid</sourceInstanceId>
      <sourceRowId>guid</sourceRowId>
      <sourceFieldId>guid</sourceFieldId>
      <sourceSectionId>guid</sourceSectionId>
    </LinkGetParentResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор карточки, секции, строки и поля, в котором содержится ссылка на карточку.

ReportGetData

Получает результат выполнения отчёта.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/ReportGetData"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ReportGetData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <reportId>guid</reportId>
      <paramsData>string</paramsData>
    </ReportGetData>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

reportId

Идентификатор отчёта

paramsData

Параметры выполнения отчёта

Строка `paramsData` должна соответствовать формату элемента `MethodCall`, приведенному в схеме `ServerRequests.xsd`:

```
<xsd:element name="MethodCall">
```

```
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:simpleContent>
            <xsd:extension base="xsd:string">
              <xsd:attribute name="Name" type="xsd:string" use="optional"/>
              <xsd:attribute name="Type" use="required">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="bool"/>
                    <xsd:enumeration value="byte"/>
                    <xsd:enumeration value="short"/>
                    <xsd:enumeration value="integer"/>
                    <xsd:enumeration value="long"/>
                    <xsd:enumeration value="single"/>
                    <xsd:enumeration value="double"/>
                    <xsd:enumeration value="decimal"/>
                    <xsd:enumeration value="uniqueidentifier"/>
                    <xsd:enumeration value="char"/>
                    <xsd:enumeration value="string"/>
                    <xsd:enumeration value="datetime"/>
                    <xsd:enumeration value="text"/>
                    <xsd:enumeration value="binary"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:attribute>
            </xsd:extension>
          </xsd:simpleContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="MethodName" type="xsd:string"/>
    <xsd:attribute name="ObjectName" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
```

```
  <soap:Body>
    <ReportGetDataResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <cursorInfo>
        <ID>guid</ID>
        <PageCount>unsignedInt</PageCount>
        <RowCount>unsignedInt</RowCount>
        <FirstPage>string</FirstPage>
        <UsingCompression>boolean</UsingCompression>
      </cursorInfo>
    </ReportGetDataResponse>
  </soap:Body>
</soap:Envelope>
```

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/ReportGetData" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ReportGetData xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
      <reportId>31058C9C-B730-4C8B-88EA-D16080DD9460</reportId>
      <paramsData>&lt;MethodCall&gt;&lt;Parameter Name=&quot;EmployeeID&quot;
Type=&quot;uniqueidentifier&quot; xmlns:dt=&quot;urn:schemas-microsoft-
com:datatypes&quot; dt:dt=&quot;string&quot;&gt;{A81E1347-9CDE-405F-BA2F-
EB122DF74B8F}&lt;/Parameter&gt;&lt;/MethodCall&gt;</paramsData>
    </ReportGetData>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

ReportGetInfo

Возвращает список параметров отчёта.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/ReportGetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ReportGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <reportId>guid</reportId>
    </ReportGetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

reportId

Идентификатор отчёта

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ReportGetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
```

```
03/StorageServer/">
      <reportInfo>string</reportInfo>
    </ReportGetInfoResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит список параметров отчёта и их тип.

ReportGetList

Возвращает список всех отчётов.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/ReportGetList"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ReportGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
    </ReportGetList>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
```

```
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ReportGetListResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <reportList>string</reportList>
    </ReportGetListResponse>
  </soap:Body>
</soap:Envelope>
```

В ответ включается список параметров каждого отчёта и тип этих параметров.

Пример запроса

```
session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/ReportGetList" ①

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <ReportGetList xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>6565c75b-298d-e511-9417-90e6ba57b9f8</sessionId>
    </ReportGetList>
  </soap:Body>
</soap:Envelope>""" ②

response = session.post(url, data=body) ③

print(response.content) ④
```

① Изменяем цель запроса в заголовке запроса.

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Ответ SOAP.

UserProfileGetAccount

Возвращает имя учетной записи сотрудника, которому принадлежит указанный профиль.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/UserProfileGetAccount"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <UserProfileGetAccount xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <userRefId>guid</userRefId>
    </UserProfileGetAccount>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

userRefId

Идентификатор карточки настроек пользователя

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <UserProfileGetAccountResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <userAccount>string</userAccount>
    </UserProfileGetAccountResponse>
  </soap:Body>
```

```
</soap:Envelope>
```

UserProfileGetInfo

Возвращает описание пользователя.

Запрос

```
POST /DocsVision/StorageServer/StorageServerService.asmx HTTP/1.1
Host: DOCSVISION_SERVER.COM
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/UserProfileGetInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <UserProfileGetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <sessionId>guid</sessionId>
      <userName>string</userName>
    </UserProfileGetInfo>
  </soap:Body>
</soap:Envelope>
```

Параметры

sessionId

Идентификатор пользовательской сессии

userName

Учетная запись пользователя

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
```

```
  <soap:Body>
    <UserProfileGetInfoResponse xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <curUserName>string</curUserName>
      <userRefId>guid</userRefId>
    </UserProfileGetInfoResponse>
  </soap:Body>
</soap:Envelope>
```

Ответ содержит идентификатор карточки настроек пользователя.

Примеры использования

Подключение к серверу Docsvision

Практически каждый метод веб-сервиса требует идентификатор пользовательской сессии, поэтому первоначальной задачей является создание сессии пользователя. Для этого предназначен метод SessionLogin.

Код для открытия пользовательской сессии может выглядеть следующим образом:

```
import xml.etree.ElementTree as ElementTree
import requests
from requests_ntlm import HttpNtlmAuth

url = 'http://DOCSVISION_SERVER.COM/DocsVision/StorageServer/StorageServerService.asmx'
username = 'Domain\User.Name'
password = 'User.Password' ①

session = requests.Session()
session.auth = HttpNtlmAuth(username, password, session)
session.headers['Content-Type'] = 'text/xml; charset=utf-8'

session.headers['SOAPAction'] = 'http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/SessionLogin' ②

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <SessionLogin xmlns="http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/">
      <baseName></baseName>
      <sessionSettings>&lt;Settings&gt;&lt;Setting Name=&quot;ComputerAddress&quot;
Type=&quot;string&quot; ReadOnly=&quot;1&quot;&gt;192.168.0.2&lt;/Setting&gt;&lt;Setting
Name=&quot;ComputerName&quot;
```

```
Type=&quot;string&quot;&gt;connectedComputer&lt;/Setting&gt;&lt;Setting
Name=&quot;LocaleID&quot;
Type=&quot;int&quot;&gt;1049&lt;/Setting&gt;&lt;/Settings&gt;</sessionSettings>
    </SessionLogin>
  </soap:Body>
</soap:Envelope>""" ③

response = session.post(url, data=body) ④

responseParsed = ElementTree.fromstring(response.content)
sessionId = responseParsed.findtext('.//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}sessionId') ⑤
```

① Адрес и учетные данные пользователя для подключения к Docsvision.

② Устанавливаем цель запроса в заголовке запроса.

③ SOAP-сообщение, в котором передаем адрес и сетевое имя клиента.

④ Отправляем SOAP-сообщение.

⑤ Разбираем ответ и получаем идентификатор сессии.

Добавление файла в базу данных

После открытия сессии можно, к примеру, добавить файл (не путать с версионным файлом — *карточка файла с версиями*) в базу данных.

Чтобы добавить файл в базу данных:

1. Создать объект-файл в базе данных, используя метод `FileCreate`.
2. Открыть созданный объект для записи, используя метод `FileOpen`.
3. Записать содержимое файла с файловой системы в открытый объект-файл, используя метод `FileWrite`.
4. Закрыть с сохранением открытый объект, используя метод `FileClose`

Итоговый код может выглядеть следующим образом:

```
①

session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileCreate" ②

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
```

```
  <soap:Body>
    <FileCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>%s</sessionId>
      <name>Document_Order.docx</name>
    </FileCreate>
  </soap:Body>
</soap:Envelope>""" % sessionId ③

response = session.post(url, data=body) ④

responseParsed = ElementTree.fromstring(response.content)
fileId = responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/}fileId") ⑤


⑥

session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileOpen" ⑦

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileOpen xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>%s</sessionId>
      <fileId>%s</fileId>
      <modify>true</modify>
    </FileOpen>
  </soap:Body>
</soap:Envelope>""" % (sessionId, fileId) ⑧

response = session.post(url, data=body) ⑨

responseParsed = ElementTree.fromstring(response.content)
fileHandleId = responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-
02-03/StorageServer/}fileHandleId") ⑩


⑪

session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileWrite" ⑫

with open("Z:\\Order.docx", "rb") as document:
    encoded_string = base64.b64encode(document.read()).decode('utf-8')
```

```
body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileWrite xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>%s</sessionId>
      <fileHandleId>%s</fileHandleId>
      <compressed>false</compressed>
      <fileData>%s</fileData>
    </FileWrite>
  </soap:Body>
</soap:Envelope>""" % (sessionId, fileHandleId, encoded_string) ⑬

session.post(url, data=body) ⑭


⑮

session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/FileClose" ⑯

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <FileClose xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>%s</sessionId>
      <fileHandleId>%s</fileHandleId>
      <commitWrite>true</commitWrite>
    </FileClose>
  </soap:Body>
</soap:Envelope>""" % (sessionId, fileHandleId) ⑰

session.post(url, data=body) ⑱

print(fileId) ⑲
```

① Создаем объект-файл.

② Изменяем цель запроса в заголовке запроса.

③ SOAP-сообщение.

④ Отправляем SOAP-сообщение.

⑤ Разбираем ответ и получаем идентификатор объект-файла.

⑥ Открываем объект-файл.

⑦ Изменяем цель запроса в заголовке запроса.

⑧ SOAP-сообщение.

⑨ Отправляем SOAP-сообщение.

⑩ Разбираем ответ и выводим идентификатор новой карточки.

⑪ Запись данных в файл.

⑫ Изменяем цель запроса в заголовке запроса.

⑬ SOAP-сообщение.

⑭ Отправляем SOAP-сообщение.

⑮ Сохранение файла.

⑯ Изменяем цель запроса в заголовке запроса.

⑰ SOAP-сообщение.

⑱ Отправляем SOAP-сообщение.

⑲ Выводим идентификатор файла.

Приведенный пример может быть значительно упрощен, к примеру, с использованием soap-клиента, который может получить описание методов из WSDL.

Создание карточки файла с версиями

Чтобы создать карточки файла с версиями:

1. Создать карточку типа "Файл с версиями", используя метод `CardCreate`.
2. Создать строки в секциях "Версии" и "Общая информация", используя метод `RowCreate`.
3. Добавить, используя метод `RowSetData`, в строку секции "Версии" значения полей (см. /dv6/schemas/dv6/[Описание полей стандартной карточки]): `FileId` (используем идентификатор файла, полученный в предыдущем примере), `Version` и `VersionNumber`.
4. Добавить, используя метод `RowSetData`, в строку секции "Общая информация" значения полей: `CheckinDate`, `CurrentID` (указывается идентификатор строки версии, созданной ранее), `CurrentVersion`, `Name` и `NextVersion`.
5. Изменить описание карточки, используя `CardSetInfo`.

Итоговый код может выглядеть следующим образом:

```
①

session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardCreate" ②

body = """<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
      <sessionId>%s</sessionId>
      <cardId>6E39AD2B-E930-4D20-AAFA-C2ECF812C2B3</cardId>
      <newInstanceId>00000000-0000-0000-0000-000000000000</newInstanceId>
    </CardCreate>
  </soap:Body>
</soap:Envelope>""" % sessionId ③

response = session.post(url, data=body) ④

responseParsed = ElementTree.fromstring(response.content)
newInstanceId = responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-
02-03/StorageServer/}newInstanceId") ⑤


⑥

versions_row_id = create_row(newInstanceId, 'F831372E-8A76-4ABC-AF15-D86DC5FFBE12')
update_row('F831372E-8A76-4ABC-AF15-D86DC5FFBE12', versions_row_id,
'&lt;RowData&gt;&lt;row FileID=&quot;%s&quot; Version=&quot;1&quot;
VersionNumber=&quot;2&quot;/&gt;&lt;/RowData&gt;' % fileId) ⑦

maininfo_row_id = create_row(newInstanceId, '2FDE03C2-FF87-4E42-A8C2-7CED181977FB')
update_row('2FDE03C2-FF87-4E42-A8C2-7CED181977FB', maininfo_row_id,
           '&lt;RowData&gt;&lt;row CheckinDate=&quot;%s&quot; CurrentID=&quot;%s&quot;
CurrentVersion=&quot;1.1&quot; Name=&quot;Document_Order versionedFileCard&quot;
NextVersion=&quot;2&quot; /&gt;&lt;/RowData&gt;' %(datetime.datetime.now().isoformat(),
versions_row_id) )


⑧

session.headers['SOAPAction'] = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/CardSetInfo" ⑨

body = """<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <CardSetInfo xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
      <sessionId>%s</sessionId>
      <instanceID>%s</instanceID>
      <description>Document_Order versionedFileCard</description>
      <template>false</template>
      <topic></topic>
    </CardSetInfo>
  </soap:Body>
</soap:Envelope>""" % (sessionId, newInstanceId) ⑩

response = session.post(url, data=body) ⑪

print(newInstanceId) ⑫
```

① Создаём карточки типа "Файл" с версиями.

② Изменяем цель запроса в заголовке запроса.

③ SOAP-сообщение.

④ Отправляем SOAP-сообщение.

⑤ Разбираем ответ для получения идентификатора созданной карточки.

⑥ Создаем строки для секций "Общая информация" и "Версии", а также заполняем их поля.

⑦ Для простоты используем дополнительные методы `create_row` (создание строки) и `update_row` (запись данных в строку).

Реализация методов create_row и update_row приведена далее.

⑧ Добавляем описание.

⑨ Изменяем цель запроса в заголовке запроса.

⑩ SOAP-сообщение.

⑪ Отправляем SOAP-сообщение.

⑫ Выводим идентификатор созданной карточки.

Идентификатор полученной карточки версионного файла можно, к примеру, записать в секцию "Файлы" карточки типа "Документ", что соответствует прикрепление файла к документу.

Дополнительные методы, использованные выше, реализованы следующим образом:

```
def create_row(card_id, section_id):
    session.headers['SOAPAction'] =  "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/RowCreate" ①

body =  """<?xml version="1.0" encoding="utf-8"?>
     <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
     <soap:Body>
     <RowCreate xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/">
     <sessionId>%s</sessionId>
     <sectionId>%s</sectionId>
     <rowId>00000000-0000-0000-0000-000000000000</rowId>
     <instanceID>%s</instanceID>
     <parentRowId>00000000-0000-0000-0000-000000000000</parentRowId>
     <parentTreeRowId>00000000-0000-0000-0000-000000000000</parentTreeRowId>
     </RowCreate>
     </soap:Body>
     </soap:Envelope>""" % (sessionId, section_id, card_id) ②

response = session.post(url,  data=body) ③

responseParsed = ElementTree.fromstring(response.content)
    newRowId = responseParsed.findtext(".//{http://schemas.docsvision.com/Platform/2009-
02-03/StorageServer/}newRowId")
return newRowId ④

⑤

def update_row(sectionId, rowId, rowData):
session.headers[ 'SOAPAction'] =  "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/RowSetData" ⑥

body =  """<?xml version="1.0" encoding="utf-8"?>
     <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
     <soap:Body>
     <RowSetData xmlns="http://schemas.docsvision.com/Platform/2009-02-03/StorageServer/
">
     <sessionId>%s</sessionId>
     <sectionId>%s</sectionId>
     <rowId>%s</rowId>
     <rowData>%s</rowData>
```

```
      </RowSetData>
      </soap:Body>
      </soap:Envelope>""" % (sessionId, sectionId, rowId, rowData) ⑦

session.post(url,  data=body) ⑧
```

① Добавление пустой строки в секцию `section_id` карточки `card_id`

② SOAP-сообщение.

③ Отправляем SOAP-сообщение.

④ Разбираем ответ и выводим идентификатор новой строки.

⑤ Запись данных `rowData` в строку `rowId` секции `sectionId`.

⑥ Изменяем цель запроса в заголовке запроса.

⑦ SOAP-сообщение.

⑧ Отправляем SOAP-сообщение.

Элементы управления Docsvision

Данный раздел содержит описание элементов управления, которые используются в Backoffice (сборка `DocsVision.BackOffice.WinForms.dll`), но также могут быть использованы в собственном решении. Данные компоненты могут быть добавлены непосредственно на *Панель элементов* Visual Studio при подключении указанной сборки.

Для использования данных элементов управления, компонент карточки должен быть унаследован от базового типа `BaseCardControl`.

Элемент управления "CardChooseBox"

Элемент управления `CardChooseBox` предоставляет функциональность выбора карточки с возможностью фильтрации по типу, а также открытия ранее выбранной карточки. Соответствует элементу управления `Карточка` *Конструктора разметок*.

Рисунок 45. Элемент управления "CardChooseBox"

`CardChooseBox` *содержит следующие функциональные кнопки:*

- **...** — открывает окно выбора карточки.

- **X** — очищает результат выбора карточки.
-  — открывает выбранную карточку.

В списке задач элемента управления `CardChooseBox` можно изменить тип редактора для элемента управления. Тип редактора определяет возможности элемента управления.

Для полнофункциональной работы элемента управления `CardChooseBox` должен быть установлен тип `DocsVision.BackOffice.WinForms.Controls.ChooseBox` (значение по умолчанию).

Свойства

- `Caption` — определяет заголовок окна выбора карточки.
- `CardControl` — возвращает ссылку на компонент карточки (унаследован от `CardControl`), на котором расположен данный элемент управления.
- `CardOpenMode` — определяет режим открытия карточки при нажатии кнопки открытия карточки. Доступны следующие варианты:
  - `Standard` — разрешено редактирование карточки.
  - `ReadOnly` — карточка будет открыта в режиме чтения.
- `CardsFilter` — текстовый фильтр для отбора карточек.

Текстовый фильтр может быть составлен вручную или создан при помощи утилиты Docsvision Explorer, входящей в Resource Kit.

Примера запроса, выбирающего задания, в которых текущий сотрудник назначен контролёром:

```
<?xml version="1.0" encoding="UTF-8"?>
<Search Version="4300" Limit="100" CombineResults="OR">
 <AttributiveSearch>
   <CardTypeQuery CardTypeID="{C7B36F33-CDD4-4DA9-8444-600FE14111E4}">
     <SectionQuery Version="4300" SectionTypeID="{20D21193-9F7F-4B62-8D69-
272E78E1D6A8}">
       <ConditionGroup Alias="alias2" Operation="OR">
         <Condition Alias="alias3" ParameterAlias="SW_I" SearchWord="true">
           <Field FieldType="refid">Controller</Field>
           <Op>EQ</Op>
```

```
            <Value>'{9E3ED92F-9474-4725-9823-F2C1A7122CD3}'</Value>
          </Condition>
        </ConditionGroup>
      </SectionQuery>
    </CardTypeQuery>
  </AttributiveSearch>
  <Params>
    <Param Alias="SW_I" Name="" Type="uniqueid" Flags="3" Value="{9E3ED92F-9474-4725-
9823-F2C1A7122CD3}" ContextWord="{48293072-F090-47F5-8A9E-8C041884B6CA}" />
  </Params>
</Search>
```

Перед записью запроса в свойство элемента управления CardChooseBox, запрос необходимо перевести из многострочного варианта в однострочный.

- **CardsTypeID** — позволяет ограничить выбор карточек типом с заданным идентификатором.
- **CommandMenuStrip** — позволяет установить новое контекстное меню.
- **Enabled** — при значении `false` будет заблокирована возможность ввода теста в область ввода.
- **InheritedCaption** — задаёт строку, отображаемую в области ввода, если была выполнена очистка результат выбора карточки. Работает при установленном **InheritedValue**.
- **InheritedValue** — задаёт идентификатор предустановленной карточки. Данная карточка будет открыта при нажатии кнопки открытия карточки, если была выполнена очистка результатов выбора карточки.
- **ObjectContext** — *контекст объектов*. Рекомендуется самостоятельно передавать *контекст объектов* в CommunicativeChooseBox, к примеру, при инициализации карточки:

```
protected override void OnCardInitialized(EventArgs e)
{
 base.OnCardInitialized(e);
 InitializeComponent();

 this.cardChooseBox.ObjectContext = this.ObjectContext; ①
}
```

① Передача контекста объектов.

- `OpenCardButtonInvisible` — при значении `true`, кнопка открытия карточки будет скрыта.
- `SearchTemplate` — при значении `true`, запрещает вывод шаблонов в список карточек.
- `Session` — возвращает сессию пользователя.
- `SimpleView` — при установленном значении `true`, будут скрыты все кнопки управления.
- `Text` — значение, отображаемое в области ввода.
- `Value` — идентификатор выбранной карточки.
- `ValueResolved` — возвращает `true`, если карточка выбрана.

Методы

Наиболее интересные методы:

- `ChooseValue()` — открывает окно выбора карточки;
- `FilterCardKinds(IEnumerable<Guid>)` — при заданном `CardsTypeID`, ограничивает выбор карточек перечисленными видами.

Событие `ValueChanged` оповещает о том, что карточка была выбрана.

Элемент управления "CategoryListView"

Элемент `CategoryListView` предоставляет функциональность выбора категории из дерева категорий. Соответствует элементу управления **Категории** *Конструктора разметок*.

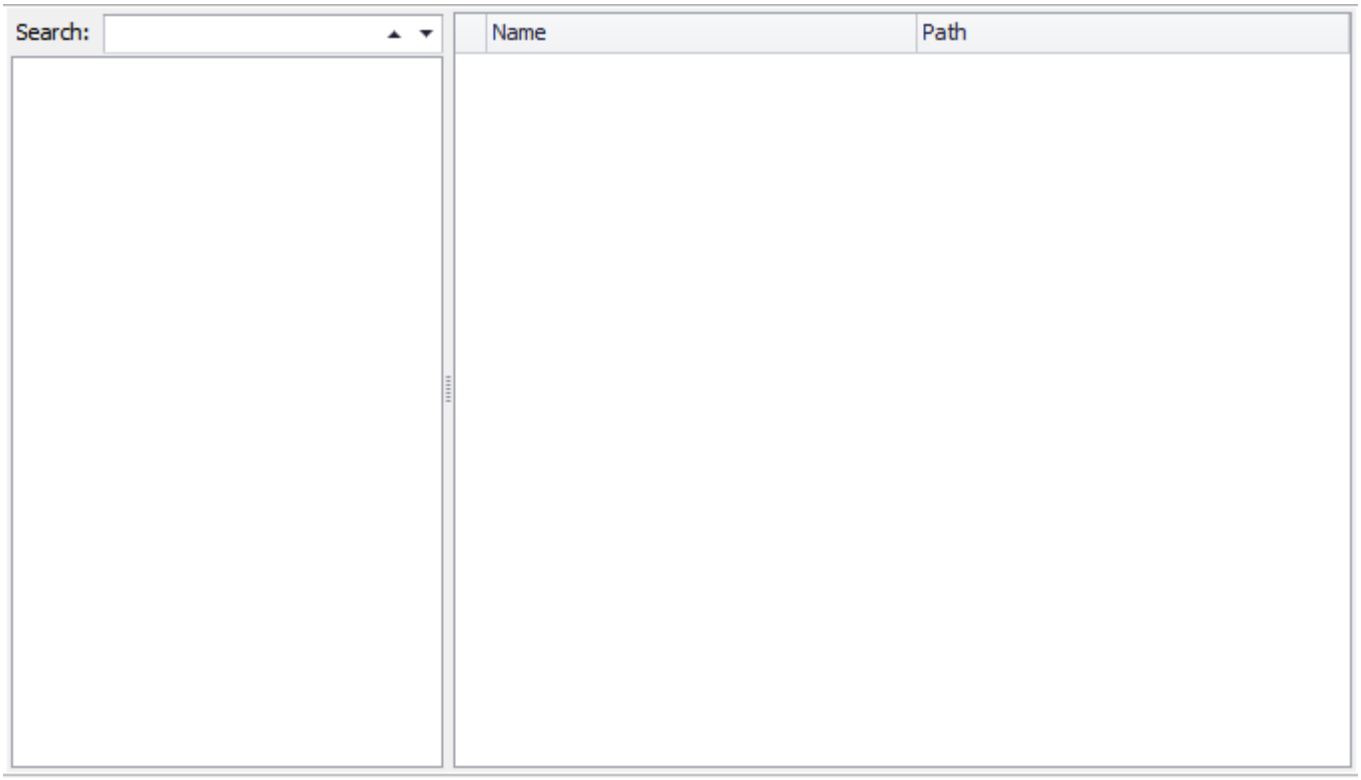


Рисунок 46. Элемент управления "CategoryListView"

Для работы `CategoryListView` необходимо наличие в карточке, к которой привязан данный элемент управления, поля, ссылающегося на экземпляр карточки *Список категорий*, содержащей список категорий. В качестве примера можно рассмотреть часть класса карточки, у которой описана специальное поле:

```
private Guid mainSectionID = new Guid("783CD108-BA4A-432D-9C12-B32E0577D53C"); ①

private string categoryListFieldName = "CategoryList"; ②
private CategoryList cardCategoryList;

protected override void OnCardInitialized(EventArgs e) ③
{
 base.OnCardInitialized(e); ④
 InitializeComponent();

 SectionData mainSection = this.CardData.Sections[mainSectionID]; ⑤
 RowData mainSectionRow = (mainSection.Rows.Count = 0) ? mainSection.CreateRow() :
mainSection.Rows[0];
 Guid? cardCategoryID = mainSectionRow.GetGuid(categoryListFieldName);
 if (cardCategoryID.HasValue = false)
 {
  cardCategoryID = ObjectContext.GetService<UserSession>().CardManager.CreateCardData(new
Guid("337CC874-BF87-4C70-A29B-F099F630B9F2")).Id;
  mainSectionRow.SetGuid(categoryListFieldName, cardCategoryID);
```

```
  }

  cardCategoryList = ObjectContext.GetObject<CategoryList>(cardCategoryID); ⑥

  this.categoryListView.Initialize(ObjectContext, cardCategoryList); ⑦
 }
```

① Идентификатор секции, содержащей ссылочное поле со ссылкой на карточку "Список категорий".

② Псевдоним ссылочного поля, которое ссылается на карточку "Список категорий".

③ Инициализация карточки.

④ Начало инициализации — стандартное.

⑤ Далее, получение и при необходимости установка ссылки в ссылочном поле.

⑥ Получение карточки "Список категорий".

⑦ Инициализация `CategoryListView`.

Свойства

- **`CategoryList`** — возвращает связанную карточку "Список категорий".
- **`CategoryService`** — возвращает сервис `ICategoryListService` для работы со списками категорий.
- **`Context`** — *контекст объектов*.
- **`RootCategory`** — определяет корневую категорию, начиная с которой будет построено дерево категорий.
- **`StatesOperations`** — предоставляет доступ к правам доступа на совершение типовых операций с `CategoryListView`.

Методы

- `Initialize(ObjectContext, CategoryList)` — данный метод используется для передачи элементу управления *контекста объектов* и списка категорий карточки. Должен быть вызван на этапе инициализации карточки (см. пример выше).
- `Rebind()` — выполняет перестроение дерева категорий.
- `RefreshData()` — повторно загружает данные для списка категорий.

События

- `ListChanged` — вызывается при любых изменениях (снятие/установка флажка на категории) в списке категорий.
- `CategoryChecked` — вызывает при установке флажка на категории.
- `CategoryUnchecked` — вызывает при снятии флажка с категории.

Первоначально срабатывает событие `CategoryChecked`/`CategoryUnchecked`, после чего срабатывает `ListChanged`.

Элемент управления "CommunicativeChooseBox"

Элемент управления CommunicativeChooseBox предоставляет функциональность выбора сотрудника из *Справочника сотрудников*. Соответствует элементу управления **Сотрудник** *Конструктора разметок*.

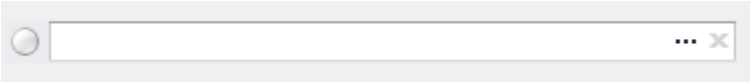


Рисунок 47. Элемент управления "CommunicativeChooseBox"

`CommunicativeChooseBox` *содержит следующие функциональные кнопки:*

- — флаг состояния выбранного сотрудника в Microsoft Lync Server. Открывает окно отправки сообщения сотруднику.
- **...** — открывает окно выбора сотрудника.
- **X** — очищает результат выбора сотрудника.

Связь с сотрудником возможна только при наличии на компьютере пользователя установленного Microsoft Lync.

Свойства

- `AccountName` — учетная запись выбранного сотрудника.
- `ActiveControl.Text` — позволяет получить имя сотрудника, отображенное на элементе управления.
- `ContactLoaded` — признак доступности сотрудника в сети Lync.
- `HideNotAvailable` — при значении `true` (значение по умолчанию) из быстрого поиска будут скрыты сотрудники с флагом `Не показывать при выборе`, установленным в *Справочнике сотрудников*.
- `ObjectContext` — *контекст объектов*. Элемент управления `CommunicativeChooseBox`

требует ручной передачи *контекста объектов*, к примеру, при инициализации карточки

```
protected override void OnCardInitialized(EventArgs e)
{
 base.OnCardInitialized(e);
 InitializeComponent();

 this.communicativeChooseBox.ObjectContext = this.ObjectContext; ①
}
```

① Передача контекста объектов.

- `SearchIndex` — определяет минимальное количество символов в строке поиска, при котором будет выполнен быстрый поиск с отображением результатов.
- `Value` — идентификатор выбранного сотрудника.
- `ValueResolved` — значение `true` символизирует завершение выбора сотрудника.

Методы

- `ChooseValue()` — вызывает окно выбора сотрудника.
- `GetValue<T>()` — возвращает объект с идентификатором `Value`, приведенный к указанному типу `t`. Например, после выбора сотрудника (из списка или окна поиска), его объектная модель может быть получена следующим образом:

```
private void GetSelectedEmployee_Click(object sender, EventArgs e)
{
 var employee = this.communicativeChooseBox.GetValue<DocsVision.BackOffice.
ObjectModel.StaffEmployee>();
}
```

- `SetValue(ObjectBase)` — переданный объект будет записан в результат выбора сотрудника. Данный метод может быть использован, если необходимо программно задать значение выбора сотрудника. В текстовое поле поиска будет записано отображаемое имя сотрудника.

События

- **`CustomChoosingValue`** — событие срабатывает при нажатии кнопки выбора сотрудника.

- `EmployeeChanged` — событие срабатывает после выбора сотрудника.
- `ButtonClick` — нажатие кнопки очистки строки поиска.
- `CustomizeSearchQuery` — событие срабатывает при выполнении быстрого поиска. Обработка события позволяет изменить поисковый запрос, результаты которого отображаются при вводе текста.

Событие `CustomizeSearchQuery` вызывается не каждый раз при вводе символов, а только при обращении к серверу Docsvision. При первом обращении результаты помещаются в локальный кэш, время жизни которого — 5 секунд. Если следующий поисковый запрос основан на текущем и кэш не старше 5 секунд, то обращений к серверу не будет и событие `CustomizeSearchQuery` не сработает (результаты будут выбраны из кэша).

Пример использования данного элемента управления — демонстрация фамилии сотрудника, подготовившего документ:

```
ICustomizableControl customizable = CardControl;

CommunicativeChooseBox regControl = customizable.FindPropertyItem<
CommunicativeChooseBox>("Registrar"); ①

if (regControl != null && regControl.Value != Guid.Empty)
{
 StaffEmployee employee = CardControl.ObjectContext.GetObject<StaffEmployee>(regControl
.Value); ②
 MessageBox.Show(string.Format("Фамилия: {0}, имя: {1}", employee.LastName, employee
.FirstName));
}
```

① Получение элемента управления "Подготовил".

② Возвращаем данные сотрудника.

Элемент управления CommunicativeMultiChooseBox

Элемент управления `CommunicativeMultiChooseBox` предоставляет функциональность выбора нескольких сотрудников из *Справочника сотрудников*. Соответствует элементу управления `Сотрудники` *Конструктора разметок*.

Рисунок 48. Элемент управления CommunicativeMultiChooseBox

`CommunicativeMultiChooseBox` содержит следующие функциональные кнопки:

- — флаг состояния выбранного сотрудника в Microsoft Lync Server. Открывает окно отправки сообщения сотрудникам.
- — открывает окно выбора сотрудников.
- **X** — очищает результат выбора сотрудников.

Связь с сотрудником возможна только при наличии на компьютере пользователя установленного Microsoft Lync.

Свойства:

- `ObjectContext` — *контекст объектов*. Элемент управления `CommunicativeMultiChooseBox` требует ручной передачи *контекста объектов*, к примеру, при инициализации карточки

```
protected override void OnCardInitialized(EventArgs e)
{
 base.OnCardInitialized(e);
 InitializeComponent();

 this.communicativeMultiChooseBox.ObjectContext = this.ObjectContext; ①
}
```

① Передача контекста объектов.

- `Text` — возвращает текст, отображаемый в строке поиска.

Методы

- `ChooseValue()` — открывает окно поиска.

События

- `CustomizeTypeSearchQuery` — событие срабатывает при вводе текста в область поиска.

Элемент управления "CommunicatorControl"

В текущей версии системы интеграция с Microsoft Lync и Skype For

Business не поддерживается.

Элемент управления `CommunicatorControl` отображает статус доступности сотрудника в Microsoft Lync. Является частью элемента управления `CommunicativeChooseBox`, но может быть использован отдельно. Не имеет аналога в *Конструкторе разметок*.

`CommunicatorControl` *содержит единственную функциональную кнопку:*

- — флаг состояния выбранного сотрудника в Microsoft Lync Server. Открывает окно отправки сообщения сотруднику.

Связь с сотрудником возможна только при наличии на компьютере пользователя установленного Microsoft Lync.

`CommunicatorControl` может быть привязан только к одному сотруднику в одно время.

В таком случае необходимо задать следующие свойства:

- `Email` — почтовый адрес сотрудника;
- `Sip` — адрес абонента в Lync. В обычном случае совпадает с `Email`.

Данные свойства могут быть установлены непосредственно из дизайнера Visual Studio.

Элемент управления "ExportDialog"

Элемент управления `ExportDialog` предоставляет функциональность экспорта и печати карточки. Не имеет аналогов в *Конструкторе разметок*.

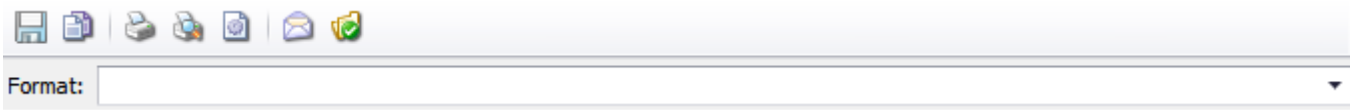


Рисунок 49. Элемент управления "ExportDialog"

Данный компонент не может быть использован в карточке как элемент управления, он должен вызываться по событию (см. описание класса `DocsVision.BackOffice.WinForms.Controls.ExportForm`).

Элемент управления "FieldSelector"

Элемент управления `FieldSelector` предназначен для выбора поля из схемы данных карточки. Соответствует элементу управления `Поле секции` *Конструктора разметок*.

Рисунок 50. Элемент управления "FieldSelector"

`FieldSelector` *содержит следующие функциональные кнопки:*

- **...** — открывает окно выбора поля;
- **X** — очищает результат выбора.

Свойства

- `CardTypeId` — идентификатор типа карточки, из схемы которой осуществляется выбор.
- `FieldTypes` — список типов полей, которые должен отображать элемент управления.
- `ObjectContext` — *контекст объектов*. Рекомендуется самостоятельно передавать *контекст объектов* в `FieldSelector`, к примеру, при инициализации карточки:

```
protected override void OnCardInitialized(EventArgs e)
{
 base.OnCardInitialized(e);
 InitializeComponent();

 this.fieldSelector.ObjectContext = this.ObjectContext; ①
 this.fieldSelector.CardTypeId = this.BaseObject.CardType.Id; ②
}
```

① Передача контекста объектов.

② Определение типа карточки, схема которой используется при выборке поля.

- `ShowDynamic` — признак, позволяющий осуществлять фильтрацию полей по признаку, является ли поле динамическим.
- `Value` — выбранное значение в формате `{Идентификатор секции}/{Псевдоним поля}`.

События

- `ControlValueChanged` — срабатывает при завершении выбора поля.

Элемент управления "FolderChooseBox"

Элемент управления `FolderChooseBox` предназначен для выбора папки. Соответствует элементу управления **Выбор папки** *Конструктора разметок*.

Рисунок 51. Элемент управления "FolderChooseBox"

`FolderChooseBox` *содержит следующие функциональные кнопки:*

- **...** — открывает окно выбора папки.
- **X** — очищает результат выбора.

Свойства

- `AllowedFolderTypes` — определяет типы папок, доступные для выбора из `FolderChooseBox`. Для других типов папок, кнопка завершения выбора недоступна.

Методы

- `ChooseValue` — открывает окно выбора папки.

Элемент управления "GridEx"

Элемент управления `GridEx` предназначен для организации табличного ввода данных. Соответствует элементу управления **Таблица** *Конструктора разметок*.

	Колонка 1	Колонка 2
	string	string
	string	string

Рисунок 52. Элемент управления "GridEx"

Свойства

- `AllowEdit` — разрешает или запрещает редактирование в таблице.
- `SourceCollection` — отображаемые данные в виде перечисления. При записи обновлении значения, будет вызвано перестроение таблицы.

Методы

- `RefreshDataSource` — перезагружает данные из источника данных.
- `RefreshSourceCollection` — перезагружает данные из источника данных, и выполняет перестроение таблицы.

Пример инициализации `GridEx` *с использованием подразделений из Справочника сотрудников в качестве источника:*

```
protected override void OnCardInitialized(EventArgs e)
{
```

```
 base.OnCardInitialized(e);
 InitializeComponent();

 IStaffService staffService = this.ObjectContext.GetService<IStaffService>();
 gridEx.DataSource = staffService.GetUnits(null, false);
}
```

Элемент управления "HistoryGrid"

Элемент управления "HistoryGrid" предназначен для просмотра записей в журнале операций, относящихся к данной карточке. Соответствует элементу управления `История` *Конструктора разметок*.

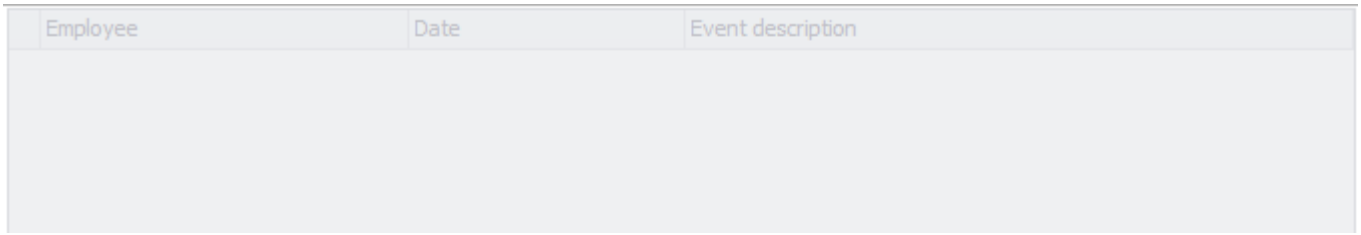


Рисунок 53. Элемент управления "HistoryGrid"

Свойства:

- `Context` — возвращает *контекст объектов*.
- `ReadOnly` — при значении `true`, `HistoryGrid` доступен на чтение.

Методы

- `Initialize(ObjectContext, BaseCard, BuiltInOperation)` — инициализация `HistoryGrid` с передачей операции открытия истории. Последний параметр может иметь значение `null`.

Пример инициализации `HistoryGrid` с передачей данных текущей карточки:

```
protected override void OnCardActivated(DocsVision.Platform.WinForms
.CardActivatedEventArgs e)
{
 base.OnCardActivated(e);

 historyGrid.Initialize(this.ObjectContext, this.BaseObject, Document
.ViewHistoryOperation); ①
}
```

① Передача контекста объектов.

Элемент управления "HtmlBrowser"

Элемент управления `HtmlBrowser` предназначен для отображения произвольного HTML-документа, либо XML-содержимого связанной карточки с (опционально) примененным XSL-преобразованием. Соответствует элементу управления `HTML браузер` *Конструктора разметок*.

`HtmlBrowser` не содержит элементов управления.

Свойства

- **`ContentType`** — тип отображаемого контента:
  - **`CardDataXml`** — `HtmlBrowser` отобразит содержимое карточки, полученной из ссылочного поля, если определены **`ReferenceCardFieldAlias`** и **`ReferenceCardSectionId`**.

    Если определено свойство `XslTransform`, будет применено соответствующее преобразование.

  - **`CustomHtml`** — предопределённый HTML-документ, который задан в значении `CustomData`.
  - **`CustomUrl`** — отображение HTML-страницы с URL. При установке значения `CustomData` необходимо соблюдать формат адреса, например: `http://www.docsvision.com` .
  - **`OwnCardDataXml`** — `HtmlBrowser` отобразит содержимое текущей карточки в формате XML, либо преобразованный документ, если определено свойство **`XslTransform`**.
- **`CustomData`** — URL-адрес для режима **`CustomUrl`**, либо HTML-документ для режима **`CustomHtml`**.
- **`IsContextMenuEnabled`** — при значении `true` будет доступно контекстное меню.
- **`ObjectContext`** — *контекст объектов*.
- **`ReferenceCardFieldAlias`** — псевдоним ссылочного поля карточки, из которой должен быть получен идентификатор карточки для отображения содержимого в режиме `CardDataXml`.
- **`ReferenceCardSectionId`** — идентификатор секции, в которой содержится поле `ReferenceCardFieldAlias`.
- **`XslTransform`** — XSLT-шаблон для преобразования содержимого карточки в режимах **`CardDataXml`** и **`OwnCardDataXml`**.

Пример инициализации `HtmlBrowser` с отображением содержимого карточки, идентификатор которой указан в ссылочном поле `CategoryList`:

```
protected override void OnCardInitialized(EventArgs e)
{
 base.OnCardInitialized(e);
 InitializeComponent();

 this.htmlBrowser.ContentType = DocsVision.BackOffice.WinForms.Controls
.HtmlBrowserContentType.CardDataXml;
 this.htmlBrowser.ReferenceCardSectionId = new Guid("783CD108-BA4A-432D-9C12-
B32E0577D53C");
 this.htmlBrowser.ReferenceCardFieldAlias = "CategoryList";
}
```

Элемент управления "Preview"

Элемент управления `Preview` предназначен для просмотра содержимого файла без открытия файла. Соответствует элементу управления `Предпросмотр файла` *Конструктора разметок*.

`Preview` не содержит элементов управления.

Свойства

- `FileName` — возвращает название открытого файла.

Методы

- `ClosePreview` — освобождает ресурсы и очищает область предварительного просмотра. Вызывается автоматически, перед вызовом ShowPreview.
- `SetText(String)` — передает текст для отображения в окно предварительного просмотра.
- `ShowPreview(String)` — формирует предварительный просмотр файла, расположенного по указанному пути.
- `ShowPreview(String, IStream)` — формирует предварительный просмотр файла, расположенного по указанному пути. Должен быть использован, если `SupportStreaming` возвращает `true`.
- `SupportStreaming(String)` — определяет поддержку обработчиком, формирующим предварительный просмотр файла, чтение файла из потока.

События

- PreviewCompleted — событие вызывается после отображения пользователю содержимого файла (используется метод ShowPreview).

Пример использования Preview для отображения содержимого документа Microsoft Word:

```
private void ShowButton_Click(object sender, EventArgs e)
{
 if (!this.preview.ShowPreview(@"X:\Sample.docx"))
  this.preview.SetText("Предварительный просмотр недоступен");
}
```

Для формирования предварительного просмотра, необходимо наличие на компьютере средств открытия файла.

Элемент управления "ReferenceListView"

Элемент управления ReferenceListView предназначен для формирования таблицы ссылок. Соответствует элементу управления Ссылки *Конструктора разметок*.

Элемент управления "RowChooseBox"

Элемент управления RowChooseBox предназначен для выбора значения из указанной в настройках элемента секции справочника. Соответствует элементу управления Значение из справочника *Конструктора разметок*.

Рисунок 54. Элемент управления "RowChooseBox"

RowChooseBox *содержит следующие функциональные кнопки:*

- **...** — открывает окно выбора строки справочника.
- **X** — очищает результат выбора.

Свойства

- ActivateParams — параметры активации справочника (см. Параметры активации справочников).
- ReferenceCardTypeId — идентификатор (типа) справочника, из которого осуществляется выбор.
- ReferenceSectionTypeId — идентификатор секции справочника, содержащего данные для выбора.

Методы

- `ChooseValue` — открывает окно выбора элемента из справочника.

События

- **`CustomChoosingValue`** — событие вызывается при открытии окна поиска. Позволяет изменить механизм выбора значения за счет изменения:
  - **`e.Search`** — флаг, показывающий, что это быстрый поиск, т.е. не надо показывать окно выбора значения из справочника, вместо этого надо найти все значения по `Text` и дать выбрать из них.
  - **`e.Chosen`** — если `true`, то считается, что значение было выбрано.
  - **`e.ValueId`** — идентификатор выбранной строки (`RowId`).
- **`CustomizeSearchQuery`** — событие срабатывает при выполнении быстрого поиска. Позволяет изменить поисковый запрос, результаты которого отображаются при вводе текста.

  См. дополнительную информацию в описании `CustomizeSearchQuery` в пункте Элемент управления `CommunicativeChooseBox`.

  Например. Нужно ограничить поиск по определённому типу записи при выборе значения из *Конструктора справочников*.

  Для этого достаточно добавить условие к существующему поисковому запросу:

```
private void MyUniversalChooseBox_CustomizeSearchQuery(object sender,
CustomizeSearchQueryEventArgs e)
{
 if (ItemTypeId != Guid.Empty)
 {
  e.SectionQuery.ConditionGroup.Conditions.AddNew(RowDataFields.ParentRow, FieldType
.UniqueId, ConditionOperation.Equals, ItemTypeId);
 }
}
```

- `CustomTextValue` — событие вызывается перед преобразованием идентификатора выбранной строки справочника в текст. Позволяет установить любой текст по выбранному значению.

  К примеру, для отображения в элементе управления полного названия организации контрагента (при его выборе) обработчик события может быть следующим:

```
private void Sender_CustomTextValue(System.Object sender, CustomTextValueEventArgs e)
{
 RowChooseBox control = sender as RowChooseBox;

 if (control.Value = Guid.Empty) ①
  return;

 PartnersCompany company = CardControl.ObjectContext.GetObject<PartnersCompany
>(control.Value); ②

 if (company = null)
  return;

 e.CustomData.Text = company.FullName; ③
}
```

① Проверка факта выбора.

② Получение организации контрагента.

③ Присвоение отображаемого значения.

Данное событие не сработает до завершения загрузки карточки. Если нужно поддержать обработку на более раннем этапе, добавьте в `CardActivated` код вида:

```
public override void CardActivated(CardActivatedEventArgs e)
{
 ICustomizableControl customizable = CardControl;
 ILayoutPropertyItem senderItem = customizable.FindPropertyItem
<ILayoutPropertyItem>("Sender"); ①
 senderItem.ControlValue = senderItem.ControlValue;
}
```

① `Sender` — название элемента управления типа `RowChooseBox`.

Пример инициализации `RowChooseBox` с возможностью выбора сотрудника из справочника сотрудников:

```
protected override void OnCardInitialized(EventArgs e)
{
 base.OnCardInitialized(e);
 InitializeComponent(); ①

 this.rowChooseBox.CardInstanceId = Guid.Empty;
```

```
 this.rowChooseBox.ReferenceCardTypeId = new Guid("6710B92A-E148-4363-8A6F-
1AA0EB18936C");
 this.rowChooseBox.ReferenceSectionTypeId = new Guid("DBC8AE9D-C1D2-4D5E-978B-
339D22B32482");
}
```

① Начало инициализации — стандартное.

Элемент управления "RowMultiChooseBox"

Элемент управления `RowMultiChooseBox` предназначен для выбора нескольких значений: из различных справочников или из различных секций одного справочника. Соответствует элементу управления `Коллекция значений` *Конструктора разметок*.

Рисунок 55. Элемент управления "RowMultiChooseBox"

- ▾ — открывает окно выбора строки справочника.
- **X** — очищает результат выбора.

Свойства

- `ObjectContext` — *контекст объектов*.

  Рекомендуется самостоятельно передавать контекст объектов в `FieldSelector`, к примеру, при инициализации карточки:

```
protected override void OnCardInitialized(EventArgs e)
{
 base.OnCardInitialized(e);
 InitializeComponent();

 this.fieldSelector.ObjectContext = this.ObjectContext; ①
 this.fieldSelector.CardTypeId = this.BaseObject.CardType.Id; ②
}
```

- `SectionTypeIds` — список идентификаторов справочников и их секций, из которых осуществляется выбор.

  Каждая строка списка должна соответствовать строковому формату: `"Идентификатор справочника;Идентификатор секции"` (см. пример ниже).

- `TypeIds` — сформированная коллекция из пары: `"Идентификатор справочника"`,

"Идентификатор секции".

Методы

- ChooseValue — открывает окно выбора элементов.

События

- CustomizeTypeSearchQuery — событие срабатывает при формировании поискового запроса при вводе символов в строку поиска, если не используется кэш. Если данные для вывода имеются в кэше, событие не срабатывает.

Пример инициализации RowMultiChooseBox, с возможностью выбора сотрудников из Справочника сотрудников и Справочника контрагентов

```
protected override void OnCardInitialized(EventArgs e)
{
 base.OnCardInitialized(e);
 InitializeComponent(); ①

 this.rowMultiChooseBox.SectionTypeIds.Add(string.Format("{0};{1}", "6710B92A-E148-4363-8A6F-1AA0EB18936C", "DBC8AE9D-C1D2-4D5E-978B-339D22B32482"));
 this.rowMultiChooseBox.SectionTypeIds.Add(string.Format("{0};{1}", "65FF9382-17DC-4E9F-8E93-84D6D3D8FE8C", "1A46BF0F-2D02-4AC9-8866-5ADF245921E8"));
}
```

① Начало инициализации — стандартное.

Элемент управления "TaskTreeView"

Элемент управления TaskTreeView предназначен для наглядного отображения жизненного цикла карточек *Задание* и *Группа заданий*. Соответствует элементу управления Дерево исполнения *Конструктора разметок*.

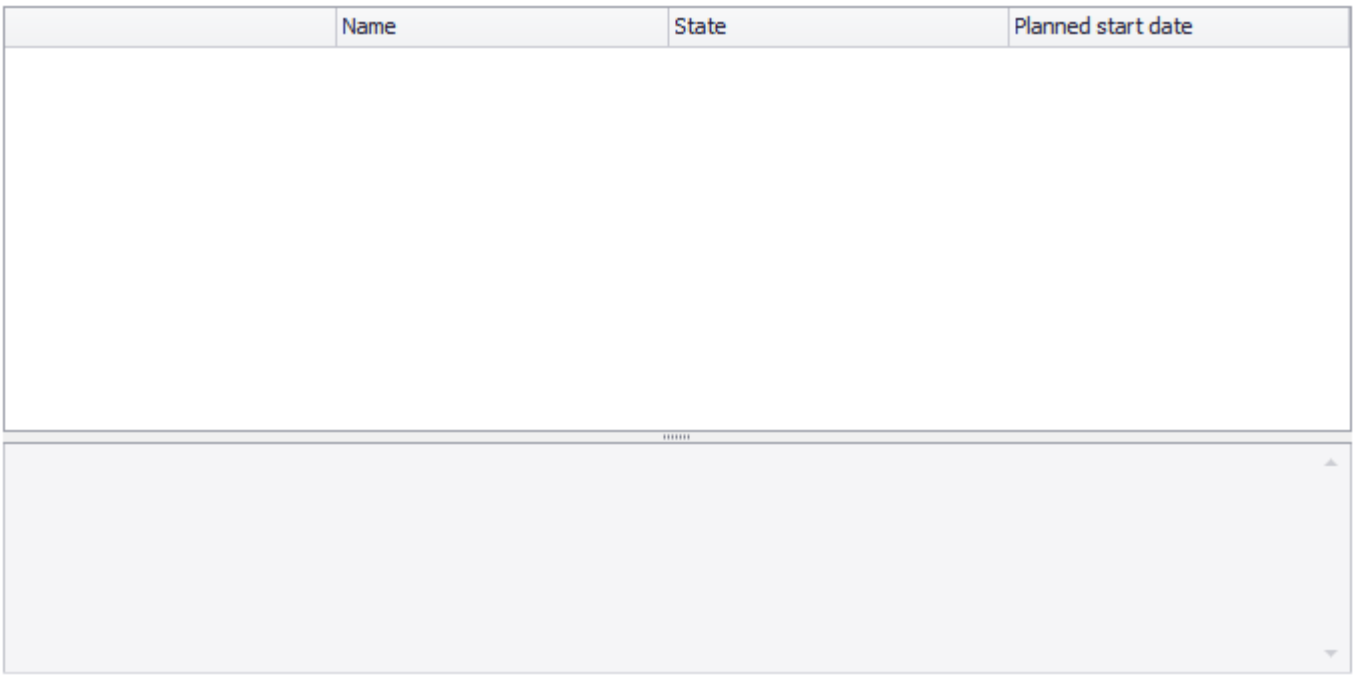


Рисунок 56. Элемент управления "TaskTreeView"

Свойства

- `AllowedOperations` — список операций (см. описание `TaskTreeViewOperation`), доступных из контекстного меню `TaskTreeView`.
- `Columns` — перечисление типов (см. описание перечисления `TaskTreeViewColumn`) отображаемых колонок.

  Например:

  ```
  this.taskTreeView.Columns = DocsVision.BackOffice.WinForms.Controls.
  TaskTreeViewColumn.Name|
  DocsVision.BackOffice.WinForms.Controls.TaskTreeViewColumn.State;
  ```

- `ExcludeDelegates` — при `true` исключает список делегирования делегированные задания списка заданий.
- `FocusedNodeInfo` — возвращает информацию по выделенному элементу.
- `NeedInitialize` — при `true`, требуется инициализация элемента управления. Используйте метод `Initialize(ObjectContext, TaskList, BaseCard)`.
- `ParentCard` — возвращает карточку, указанную при инициализации в качестве родительской.
- `ParentTask` — возвращает карточку, приведенную к типу `DocsVision.BackOffice.ObjectModel.Task` и указанную при инициализации в качестве родительской.

- **StatesOperations** — список операций с признаком доступности.
- **TaskList** — возвращает список заданий, связанный с TaskTreeView.

Методы

- AddChildTaskToSelectedTask — создаёт дочернее задание у выделенного элемента списка.
- ClearData — очищает список удалённых заданий.
- CopyResultsToParentTask(Task) — копирует результат выполнения дочернего задания в родительское.
- CopyTask — копирует выделенное задание.
- CreateChildTask(Task, KindsCardKind) — создаёт дочернее задание, указанного вида.
- CreateTaskGroup(Task, KindsCardKind) — создаёт дочернюю группу заданий, указанного вида.
- DataBind — перезагружает данные в TaskTreeView.
- EnsureColumns — формирует список колонок для отображения основываясь на значении свойства **Columns**.
- Initialize(ObjectContext, TaskList, BaseCard) — инициализация элемента управления.

  Пример инициализации TaskTreeView с отображением в дереве заданий, полученных из карточки:

```
protected override void OnCardActivated(DocsVision.Platform.WinForms
.CardActivatedEventArgs e)
{
 base.OnCardActivated(e);

 TaskGroup taskGroup = ObjectContext.GetObject<TaskGroup>(new Guid("00000000-0000-
0000-0000-000000000000"));
 this.taskTreeView.Initialize(ObjectContext, taskGroup.MainInfo.TaskList, taskGroup);
}
```

- RemoveChildTask(Task) — удаляет задание.
- SetControlsState — обновляет состояние элементов управления.
- ShowTask — открывает карточку выбранного задания.

Свойства

- `NodeCustomDisplayText` — событие вызывается перед формированием в интерфейсе единичного узла дерева заданий.
- `TaskCreated` — событие вызывается после создания задания.
- `TaskGroupCreated` — событие вызывается после создания группы заданий.

Элемент управления "UniqueRowChooseBox"

По функциональности аналогичен элементу управления `RowChooseBox`. Добавлена функциональность выделения всего текста в строке поиска.

Элемент управления "UniversalItemChooseBox"

Элемент управления `UniversalItemChooseBox` предназначен для добавления поля, в котором предполагается выбор строки из справочника, созданного в *Конструкторе справочников*. Соответствует элементу управления `Строка конструктора справочников` *Конструктора разметок*.

Рисунок 57. Элемент управления "UniversalItemChooseBox"

`CardChooseBox` *содержит следующие функциональные кнопки:*

- **...** — открывает окно выбора строки справочника.
- **X** — очищает результат выбора.

Свойства

- `ItemTypeId` — идентификатор узла справочника, ограничивающего выбор элементов.

  Если значение не задано, будет предоставлено все дерево элементов.

Пример инициализации `UniversalItemChooseBox` *с ограничением выбора элементов из определённого узла:*

```
protected override void OnCardActivated(DocsVision.Platform.WinForms
.CardActivatedEventArgs e)
{
 base.OnCardActivated(e);
 this.universalItemChooseBox1.ItemTypeId = new Guid("00000000-0000-0000-0000-
000000000000");
}
```

Устаревшие элементы управления Docsvision

На практике не все задачи могут быть решены при помощи технологии связывания данных Data Binding. Например, невозможно корректно отобразить иерархические данные в виде дерева, потому что стандартный элемент управления `TreeView` не поддерживает Data Binding.

Кроме того, область применения источников данных ограничена задачами просмотра и редактирования упорядоченных наборов данных (строк, карточек). Тогда как типичные практики предусматривают и другие способы взаимодействия с пользователем. Например, выбор каких-либо объектов при помощи специализированного браузера.

Для решения некоторых из этих задач, Docsvision предлагает несколько вспомогательных элементов управления собственной разработки.

В числе вспомогательных элементов управления:

- `BoundChooseBox` — универсальный элемент управления для выбора значений ссылочных полей.
- `CardChooseBox` — специализированный элемент управления для выбора карточек.
- `RowChooseBox` — элемент управления для выбора любых объектов.
- `BoundTreeView` — элемент управления для работы с иерархической секцией.
- `NavigationToolStrip` — навигационная панель инструментов.
- `WizardControl` — вспомогательный элемент управления для создания Мастеров.

Источники данных

Основная идея разработки визуальных форм карточек Docsvision заключается в максимальном использовании стандартных элементов управления Visual Studio (`TextBox`, `ComboBox`, `ListView` и т.д.) — вместо того, чтобы предлагать собственные элементы управления, которые разработчику понадобится дополнительно изучать.

Этот подход имеет свои преимущества:

- Стандартные элементы управления хорошо задокументированы (например, на MSDN).
- У разработчиков обычно есть навыки работы с ними.
- Совместимость элементов управления гарантируется Microsoft.

Однако недостатком подобного подхода является тот факт, что стандартные элементы управления ничего "не знают" о Docsvision. Соответственно, при их использовании разработчику придется каждый раз писать один и тот же код по загрузке данных из Docsvision в элемент управления и обратному сохранению данных из элемента управления в Docsvision.

Данную проблему предлагается решать при помощи Data Binding — технологии Microsoft для динамического связывания элемента управления с источником данных (базой данных, внешней системой, провайдером и т.д.).

Она дает следующие ключевые преимущества:

- Настройка в design-time (позволяет создать полнофункциональный элемент управления для обработки данных без написания какого-либо кода).
- Автоматизация рутинных операций по загрузке данных в элемент управления и сохранению изменений (разработчик избавлен от необходимости дублировать этот код для каждого элемента).
- Поддерживается подавляющим большинством стандартных элементов управления Visual Studio, а также элементами сторонних производителей.

Чтобы сделать подобное связывание возможным, требуется наличие специального объекта — источника данных. Библиотека элементов управления Docsvision предусматривает четыре таких объекта для разных типов данных в системе, а также один вспомогательный объект — `SessionSource`.

Объект "SessionSource"

Объект `SessionSource` является вспомогательным для других источников данных, и обеспечивает им соединение с сервером Docsvision. Его наличие является **обязательным** при использовании механизма связывания (Data Binding).

Для начала использования объекта `Session Source`, его нужно поместить с панели инструментов на форму карточки. При этом, откроется форма настройки, позволяющая удобно сконфигурировать параметры соединения с сервером:

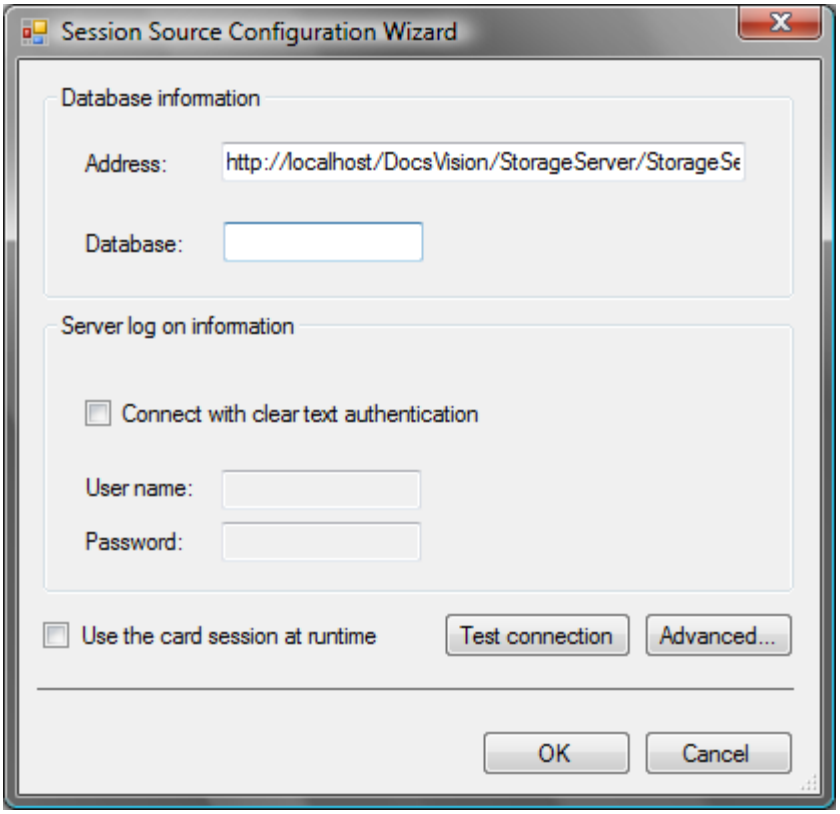


Рисунок 58. Мастер настройки Session Source

На форме можно указать следующие параметры:

- *Address* — адрес сервера Docsvision (SOAP или PIPE).
- *Database* — имя базы данных. Можно не указывать, в этом случае будет использована база данных по умолчанию.
- `Connect with clear text authentification` — признак использования базовой аутентификации вместо встроенной.
- *User name* — имя пользователя для базовой аутентификации.
- *Password* — пароль для базовой аутентификации.
- `Use the card session at runtime`. Если флаг не установлен, будет создана новая сессия с указанным в настройках сервером и базой данных. Если флаг установлен, параметры соединения с сервером будут проигнорированы, и будет использована сессия, которая передана в объект карточки (`CardControl.Session`).

  Таким образом, можно установить следующее простое правило:

  - При разработке карточек, флаг должен быть **установлен**.
  - При разработке внешних приложений и утилит, флаг должен быть **снят**.

Test connection позволяет проверить правильность введённых данных (будет

произведена попытка создания новой сессии с указанным сервером).

При нажатии на кнопку **Advanced**, откроется окно, позволяющее задать дополнительные настройки сессии (их также можно задать позже в окне свойств объекта `SessionSource`)

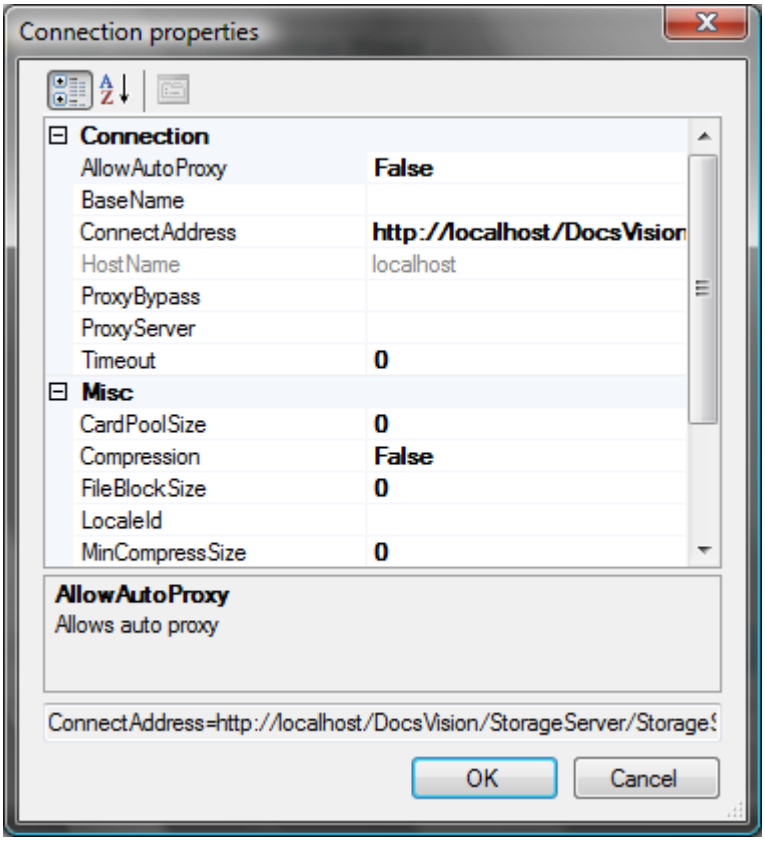


Рисунок 59. Дополнительные настройки сессии

Свойства объекта `SessionSource` соответствуют параметрам соединения.

Источник данных "CardDataSource"

Объект `CardDataSource` является уже непосредственным источником данных для элементов управления. Он позволяет работать с коллекцией данных карточек (`CardDataCollection`).

При первичном размещении этого источника данных на форме карточке, откроется специальный мастер, которые позволит сконфигурировать источник данных за несколько простых шагов.

Первый шаг мастера настройки позволяет указать сессию, с которой будет работать источник данных `CardDataSource`:

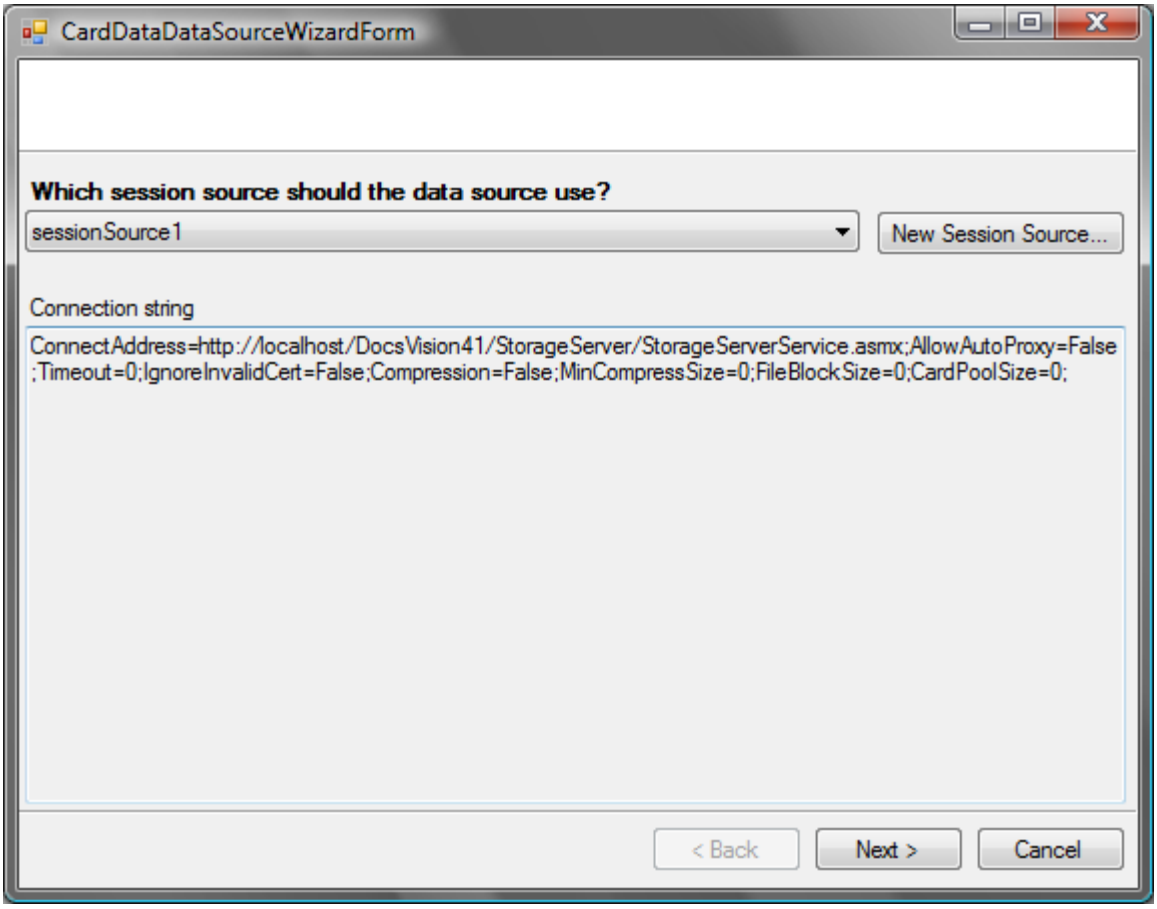


Рисунок 60. Мастер настройки "CardDataSource"

В выпадающем списке представлены все имеющиеся в проекте объекты-сессии (`SessionSource`), из которых нужно выбрать тот, который будет использован для получения данных. При выборе конкретного объекта сессии, в поле "Connection string" отображаются детальные сведения о выбранном соединении. Кнопка "**New session source**" позволяет создать новый объект сессии, если он ещё не добавлен в проект.

После выбора соединения и перехода на следующий шаг предлагается указать тип карточек, с которыми будет работать данный источник данных.

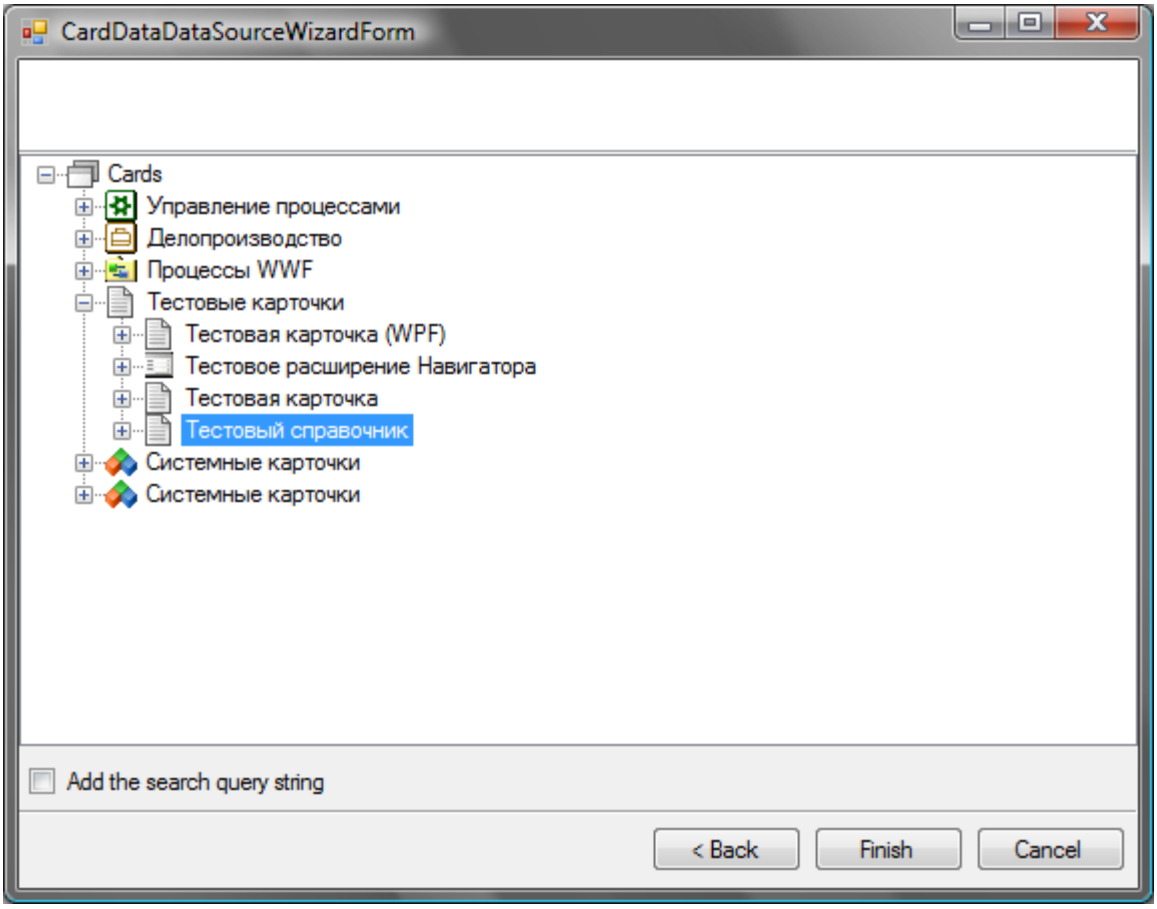


Рисунок 61. Выбор типа карточек

В дереве приведены все типы карточек, зарегистрированные в текущей базе данных, сгруппированные согласно своим библиотекам. Источник данных `CardDataSource` может работать только **с одним конкретным типом** единовременно.

Для выбранного типа карточек, источник данных `CardDataSource` может получить либо все имеющиеся в базе данных экземпляры карточек такого типа (по аналогии с системной веткой "Карточки" в Windows-клиенте), либо ограничить количество возвращаемых карточек по фильтру (поисковому запросу).

Следует всегда помнить о том, что Windows-клиент кэширует компонент карточки в памяти после первого обращения к нему, и впоследствии активирует этот же самый компонент при открытии всех карточек такого типа. Поэтому важно не забывать всегда инициализировать и очищать элементы управления, чтобы в них не осталось данных со времени предыдущей активации.

По умолчанию мастер настройки источника данных `CardDataSource` предполагает, что получаются все карточки указанного типа. Поэтому сразу после выбора

конкретного типа карточек становится доступной кнопка **Finish**, завершающая работу мастера.

Чтобы конкретизировать список возвращаемых карточек при помощи поискового запроса, нужно установить флаг `Add the search query string`. В этом случае становится доступным следующий шаг мастера, который позволяет указать поисковый запрос для отбора карточек:

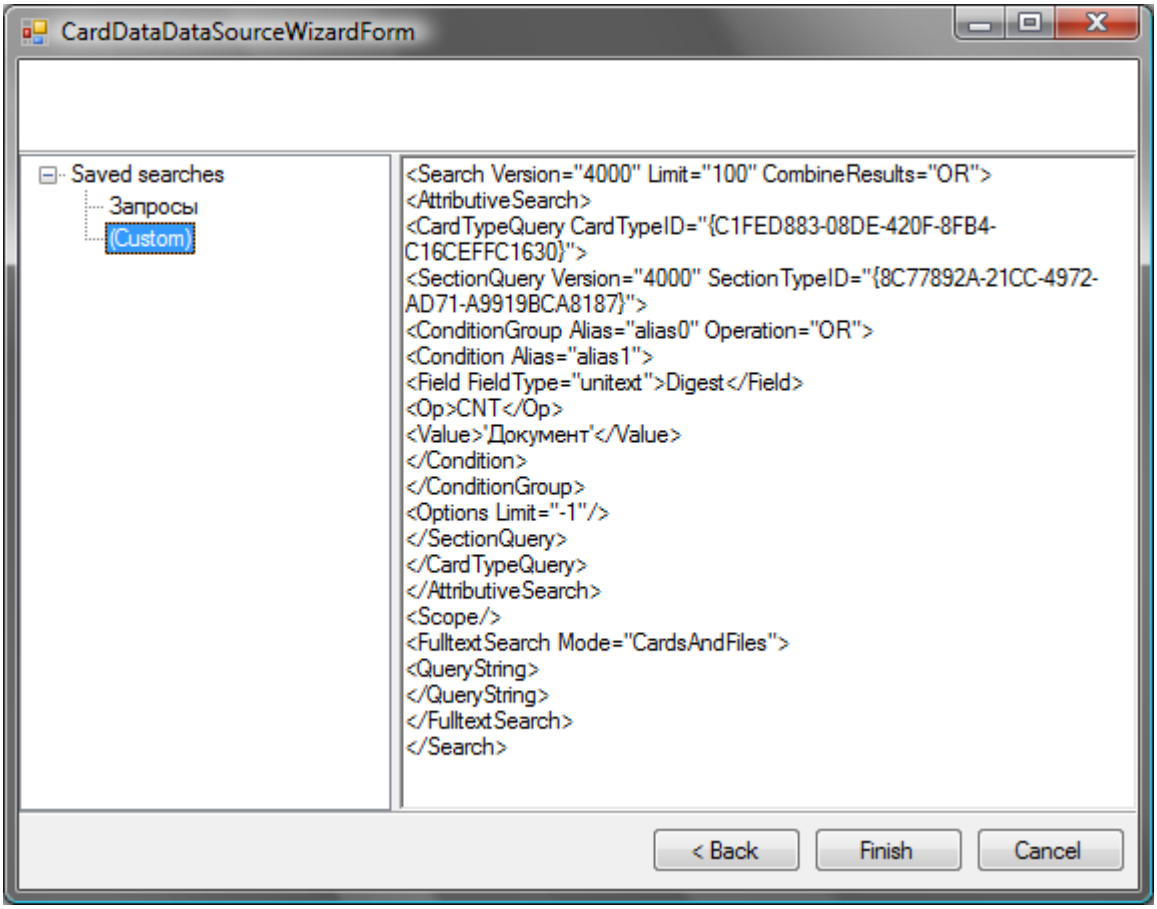


Рисунок 62. Указание поискового запроса для отбора карточек

Можно выбрать уже готовый запрос из числа сохранённых, или ввести непосредственно текст запроса в виде XML — для этого необходимо установить курсор на специальную ветку (Custom), тогда в правой части станет доступным элемент управления для редактирования поискового запроса.

При выборе сохранённого поискового запроса, источник данных копирует и сохраняет его непосредственный текст, а не ссылку. Поэтому последующие изменения этого сохранённого запроса в базе данных уже не отразятся на источнике данных — его придется заново явно конфигурировать.

После указания запроса и нажатия кнопки **Finish**, конфигурирование источника

данных завершается. Впоследствии все установленные при помощи мастера параметры можно изменить — в окне свойств (*Properties*) в Visual Studio, или в run-time в виде свойств объекта `CardDataSource`.

Доступен следующий набор свойств:

- **`CardTypeId (Guid)`** — идентификатор типа карточек.
- **`SessionSource`** — объект сессии (соединение) для источника данных.
- **`QueryString (string)`** — строка поискового запроса.

После создания и настройки источника данных можно приступать к привязке готового источника к элементу управления. Рассмотрим пример использования объекта `CardDataSource` в качестве источника данных для стандартного элемента управления `ComboBox`.

Прежде всего, необходимо поместить элемент управления на форму карточки, и задать значения основных атрибутов (название, расположение, и т.д.). После этого, можно приступить к связыванию элемента управления с источником данных

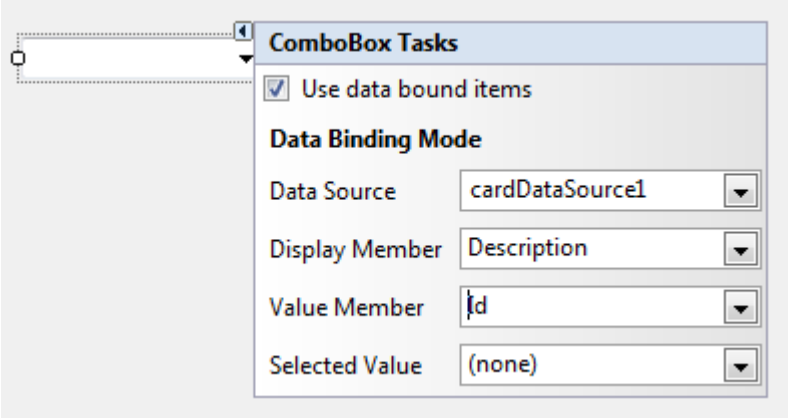


Рисунок 63. "ComboBox" и "CardDataSource"

Для этого требуется задать значения следующих свойств:

- **`Data Source`** — источник данных для элемента управления (выбираем созданный и настроенный `CardDataSource`).
- **`Dispay Member`** — поле источника данных, значение которого будет использоваться для отображения в элементе управления.

  Поскольку в данном случае используется источник данных карточек Docsvision (`CardDataSource`), то в качестве полей данных возвращается набор стандартных системных атрибутов карточек (описание, дата создания, дата изменения, и т.д.).

- `Value member` — поле источника данных, которое будет определять значение выбранного элемента. В случае `CardDataSource`, уместно в качестве такого поля использовать идентификатор карточки (ID).
- `Selected Value` — выбранное по умолчанию значение в списке.

После указания значений данных свойств, привязка элемента управления к источнику данных завершена, и он готов к работе. Если после этого сразу запустить разработанную карточку, то в выпадающем списке появятся карточки выбранного типа.

Источник данных "RowDataSource"

Источник данных `RowDataSource` предназначен для работы с набором строк секции карточки (`RowDataCollection`). При этом он позволяет не только просматривать, но и изменять, добавлять и удалять строки в секции.

Как и в случае источника `CardDataSource`, первичную конфигурацию источника данных можно выполнить при помощи мастера.

Первый шаг мастера позволяет указать сессию, с которой будет работать источник данных.

На втором шаге мастер предлагает указать секцию, строки из которой будет получать источник данных. Допускается работа с секциями любых типов (плоская, табличная, древовидная).

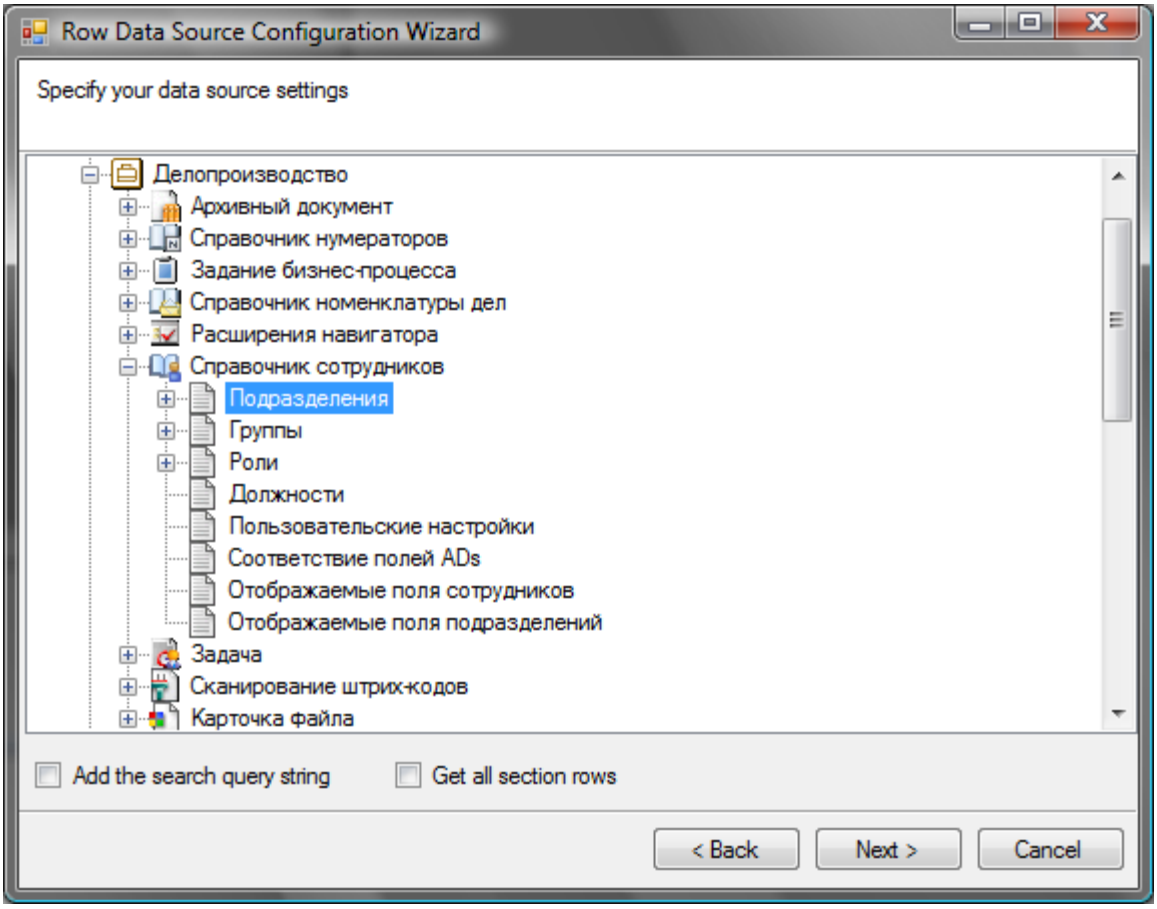


Рисунок 64. Мастер настройки "RowDataSource"

По умолчанию мастер предполагает, что будут использоваться все строки указанной секции. Если же требуется отфильтровать набор строк, необходимо установить флаг `Add the search query string`. Тогда следующий шаг мастера позволит указать поисковый запрос для отбора строк секции.

+ Дополнительный признак `Get all section rows` на этой же странице указывает на необходимость получения всех строк иерархической секции, без учета уровня иерархии (например, получить все подразделения и организации из справочника сотрудников).

При использовании этого признака, результирующий набор строк в источнике данных **не** будет доступен для добавления и удаления строк!

На следующем шаге предлагается выбрать конкретный экземпляр карточки, из которой будут получаться строки. Если выбрана секция справочника, которая всегда существует в системе в единственном экземпляре, в таком выборе нет необходимости и этот шаг можно пропустить.

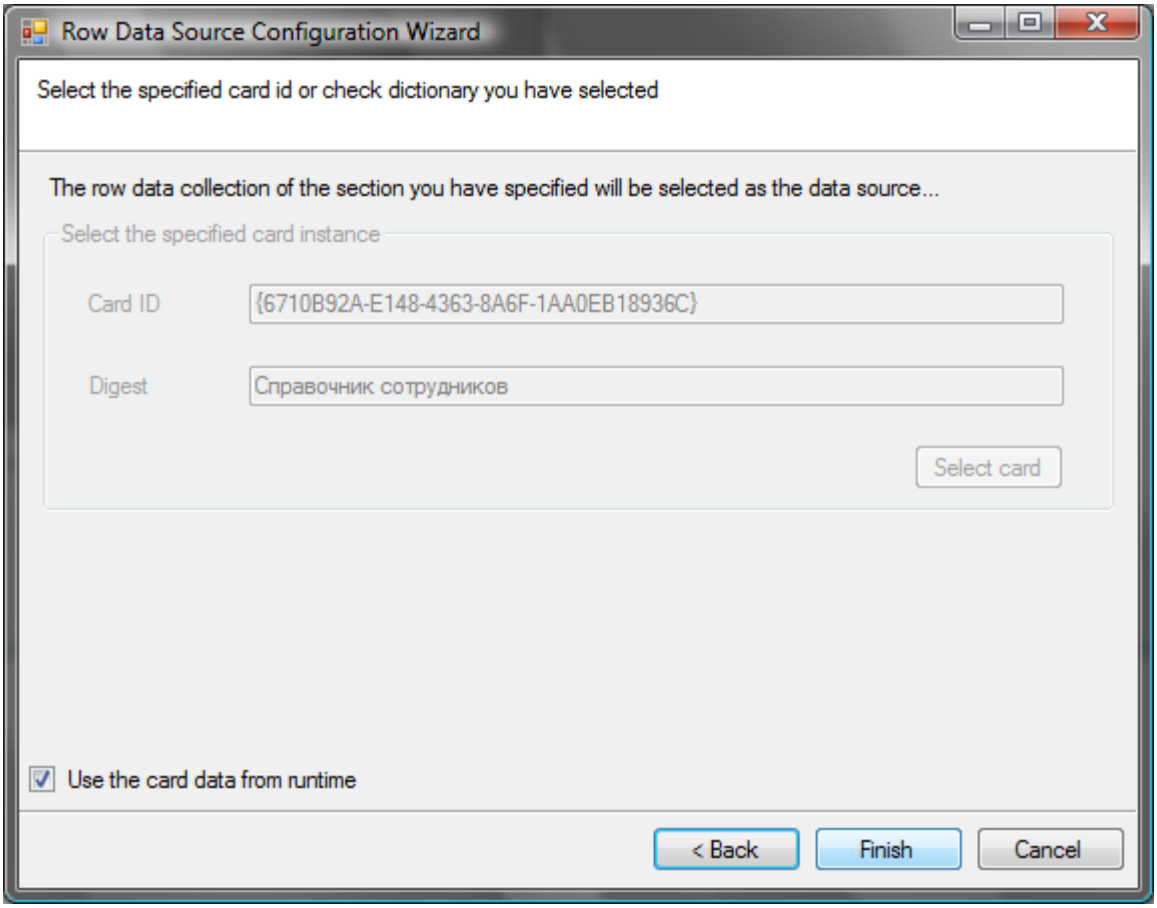


Рисунок 65. Выбор ID карточки

Для выбора конкретной карточки можно воспользоваться кнопкой **Select card**. Флаг `Use the card data from runtime` сигнализирует о том, что для получения данных будет использоваться динамический объект, доступный через свойство `CardData` в контексте данной формы. В случае разработки компонента карточки, это будут данные самой этой карточки.

Наконец, последний шаг предназначен для указания запроса фильтрации данных (он становится доступен при установке флага `Add the search query string`). Как и в случае `CardDataSource`, на этом шаге можно выбрать сохранённый поисковый запрос или указать текст запроса напрямую.

После этого работа мастера завершается, и источник данных готов к использованию.

Все параметры, установленные при помощи мастера для источника данных, можно впоследствии изменить через свойства объекта `RowDataSource`:

- *CardInstanceID* — идентификатор экземпляра карточки, из которой будут получаться строки.

- *CardTypeID* — идентификатор типа карточки, из которой будут получаться строки.
- *SectionTypeID* — идентификатор типа секции карточки, из которой будут получаться строки.
- *SearchQuery* — текст поискового запроса для фильтрации строк.
- *ParentRowID* — идентификатор родительской строки для подчиненной секции.
- *ParentTreeRowID* — идентификатор родительской строки в иерархической секции.
- *SessionSource* — объект сессии.

Пример использования источника данных `RowDataSource`: разместим на форме карточки таблицу со списком всех сотрудников из справочника сотрудников. Для этого необходимо настроить источник данных на соответствующую секцию, а также установить флаг **`Get all section rows`**, чтобы получить все строки секции независимо от уровня в иерархи.

В качестве элемента управления для работы со строками целесообразно использовать стандартный табличный элемент управления `DataGridView`, который позволяет не только просматривать строки, но и редактировать их. После указания источника данных для элемента управления, он автоматически получает заголовки столбцов как имена соответствующих полей секции:

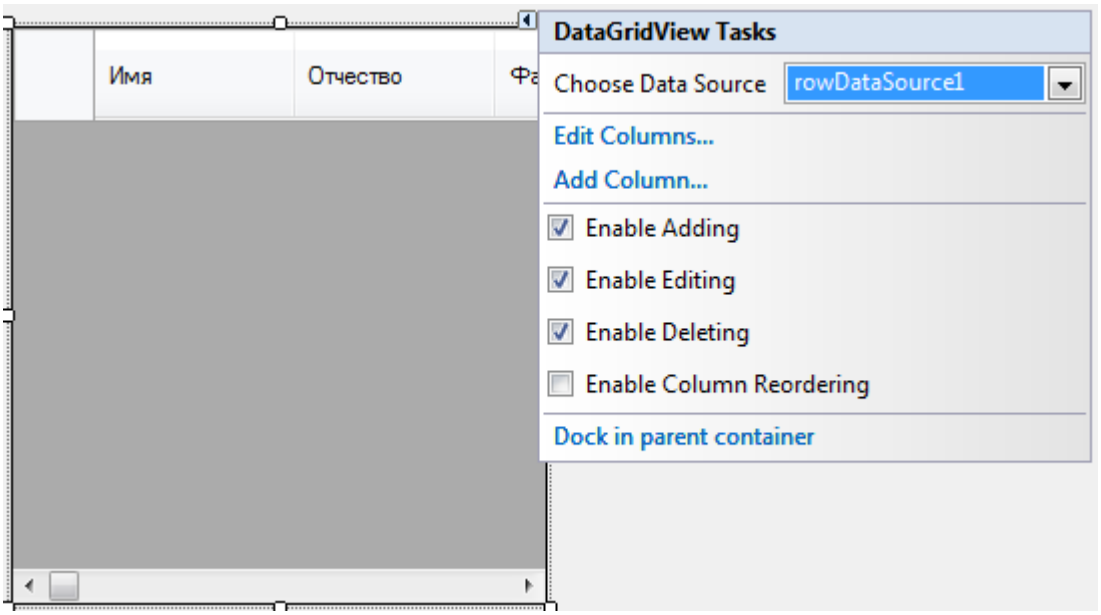


Рисунок 66. Мастер настройки RowDataSource

Сразу после этого элемент управления готов к работе, и позволяет просматривать данные справочника сотрудников.

Источник данных InfoRowDataSource

Источник данных `InfoRowDataSource` предназначен для работы с данными представлений. Он позволяет только просматривать данные и не предоставляет возможности их редактирования.

Конфигурирование этого источника данных осуществляется в режиме мастера. На первом шаге предлагается выбрать объект сессии для установки соединения. На втором шаге необходимо указать представление из числа сохранённых описаний представлений на данном сервере, или создать новое (`Custom`).

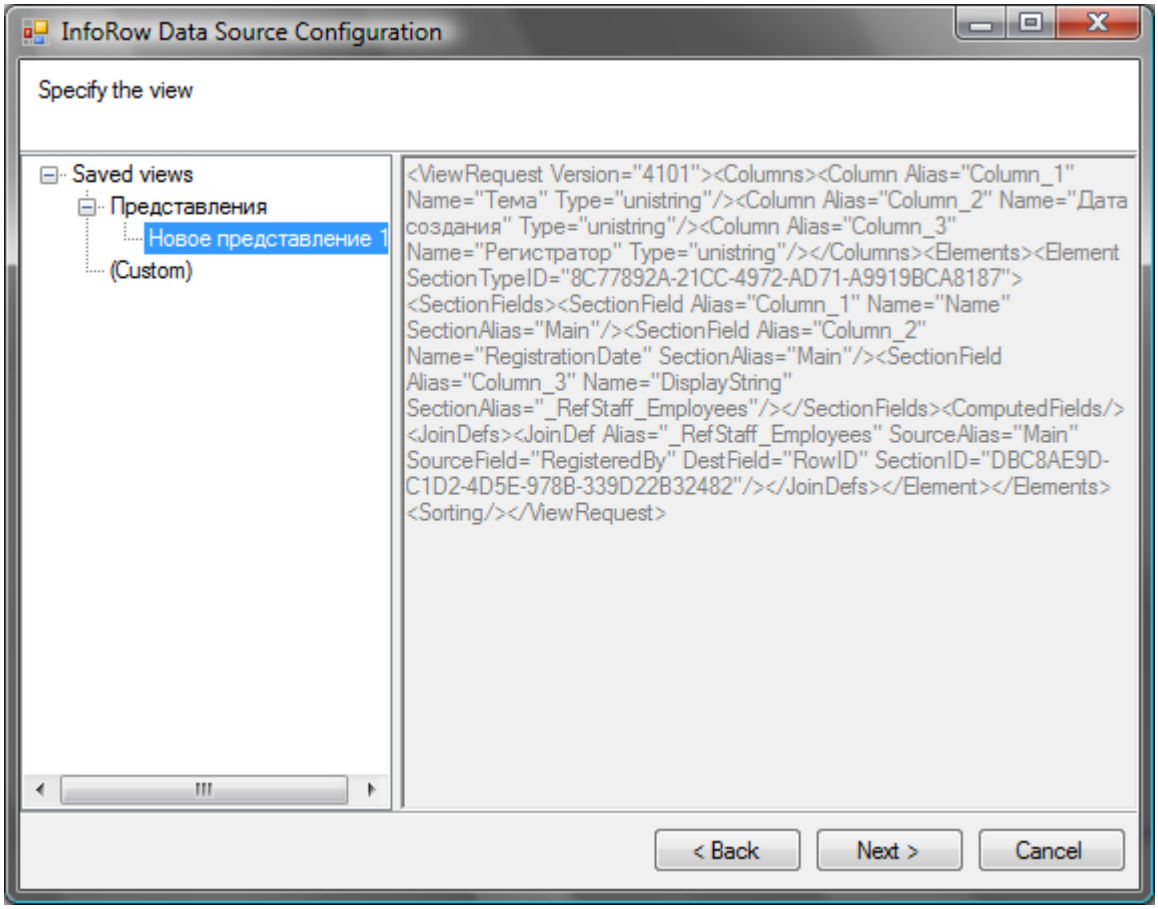


Рисунок 67. Мастер настройки "InfoRowDataSource"

Поскольку представление описывает всего лишь способ отображения данных, а не сами данные — то необходимо указать ещё `источник` данных для представления. Это можно сделать на следующем шаге мастера.

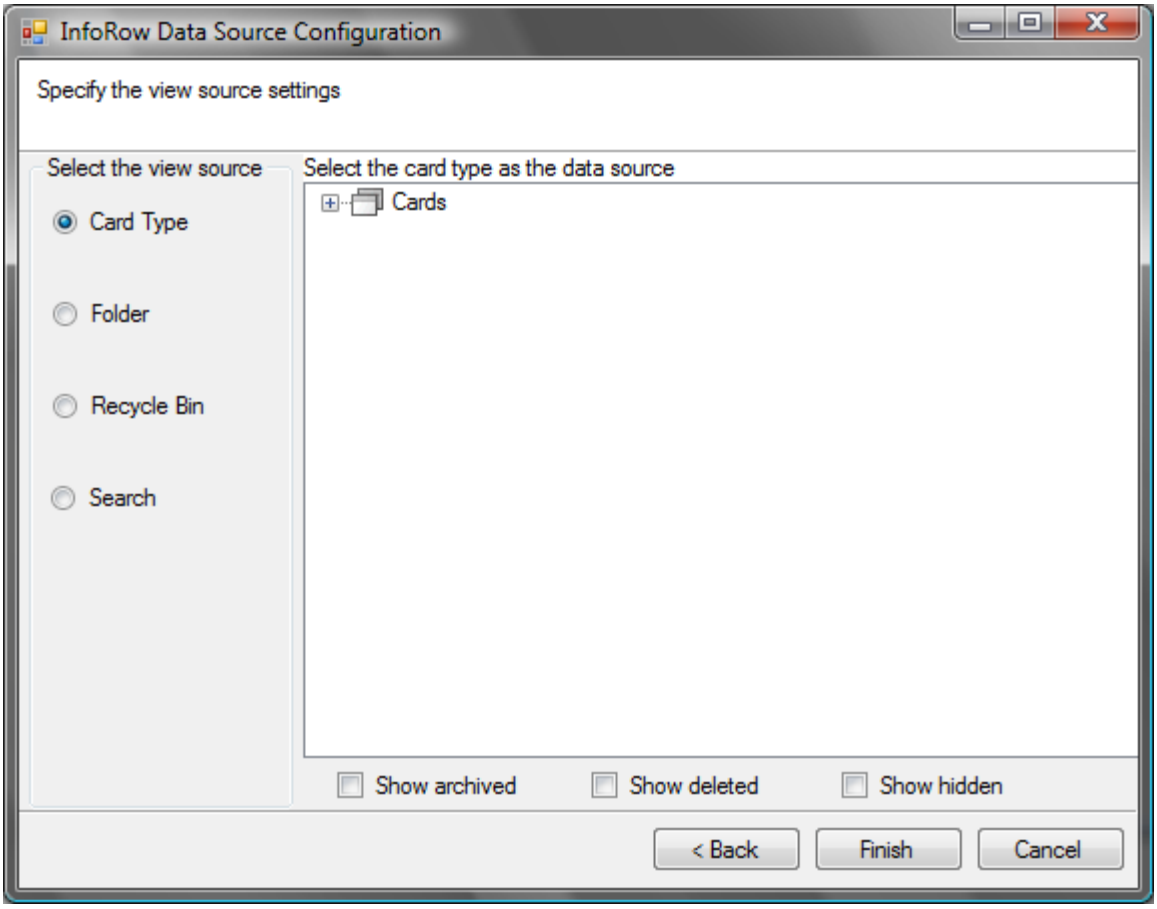


Рисунок 68. Настройка источника данных для представления

Возможны следующие варианты:

- **Card Type** — в представлении будут отображены все карточки указанного типа (необходимо выбрать конкретный тип карточки в дереве типов).
- **Folder** — в представлении будут отображены данные карточек, лежащих в конкретной папке (необходимо указать папку в дереве папок).
- **Recycle Bin** — отображает содержимое корзины (все удалённые карточки).
- **Search** — представление применяется к результатам выполнения поискового запроса (необходимо выбрать сохранённый поисковый запрос, или ввести его текст).

Здесь же можно установить дополнительные флаги, влияющие на формирование представления:

- `Show arc hived` — выводить в представлении архивные карточки.
- `Show deleted` — выводить также удалённые карточки.
- `Show hidden` — выводить скрытые и системные типы карточек.

Эти признаки могут быть установлены совместно в любых сочетаниях.

После определения данных представления, работа мастера заканчивается, и

источник данных готов к использованию. Все параметры также доступны для последующего изменения в виде свойств объекта `InfoRowDataSource:`

- **`ItemId`** — идентификатор выбранной папки, при использовании папки в качестве источника данных; либо идентификатор типа карточек.
- **`SearchQuery`** — текст поискового запроса, если НЕ используется сохранённый (при использовании поиска в качестве источника данных).
- **`SearchQueryID`** — идентификатор сохранённого поискового запроса для выборки карточек (при использовании поиска в качестве источника данных).
- **`ViewID`** — идентификатор сохранённого описания представления.
- **`ViewQuery`** — XML-описание представления, если НЕ используется сохранённое.
- **`ViewSourceType`** — тип данных в представлении (`Card Type`, `Folder`, `Recycle Bin`, `Search`).

Источник данных ReportDataSource

`ReportDataSource` предназначен для работы с данными хранимых процедур.

Первичное конфигурирование источника данных выполняется при помощи мастера. После выбора активного соединения на первом шаге, мастер предлагает указать хранимую процедуру из числа зарегистрированных в базе данных (отображаются хранимые процедуры сразу из всех загруженных в БД библиотек карточек!).

По умолчанию в базе данных присутствуют только системные хранимые процедуры из решения *Workflow*, названия которых приведены.

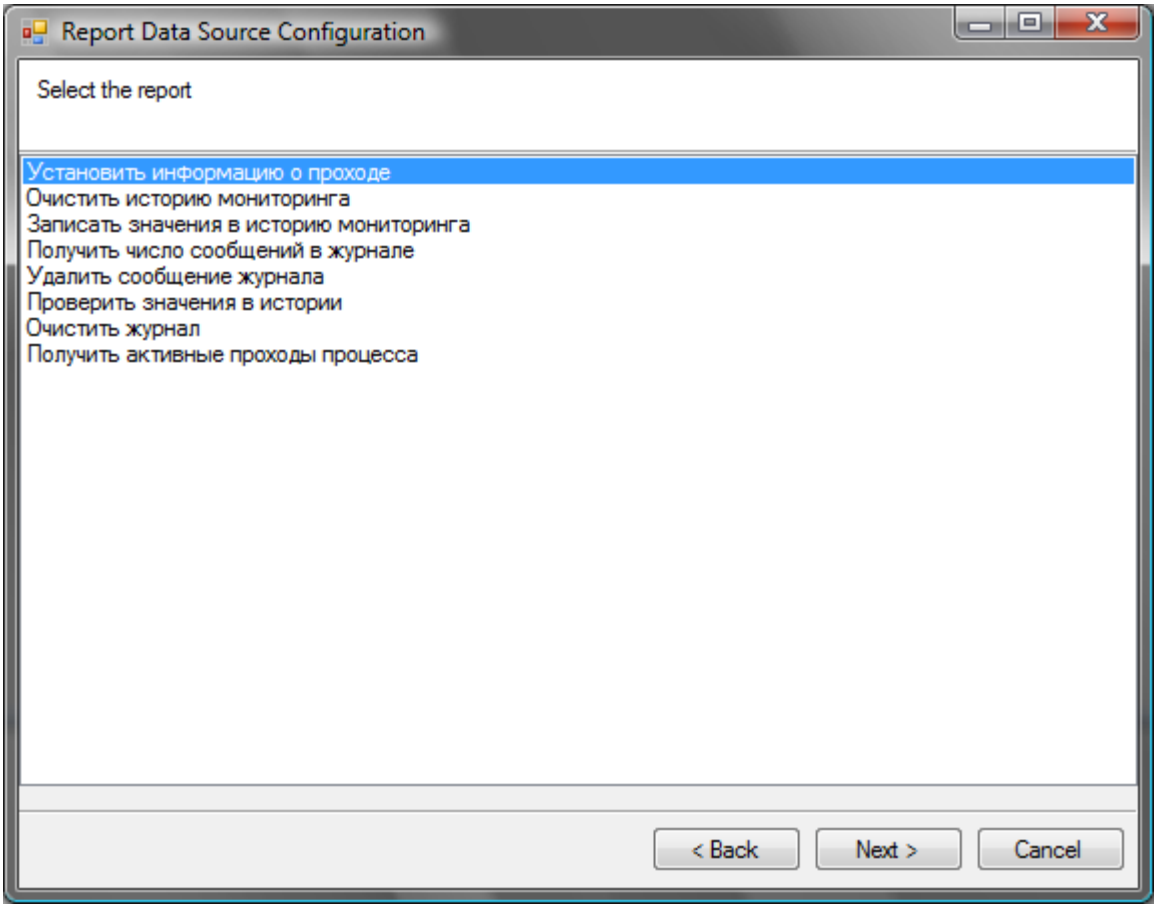


Рисунок 69. Мастер настройки "ReportDataSource"

После выбора процедуры, Мастер предлагает указать значения всех её параметров.

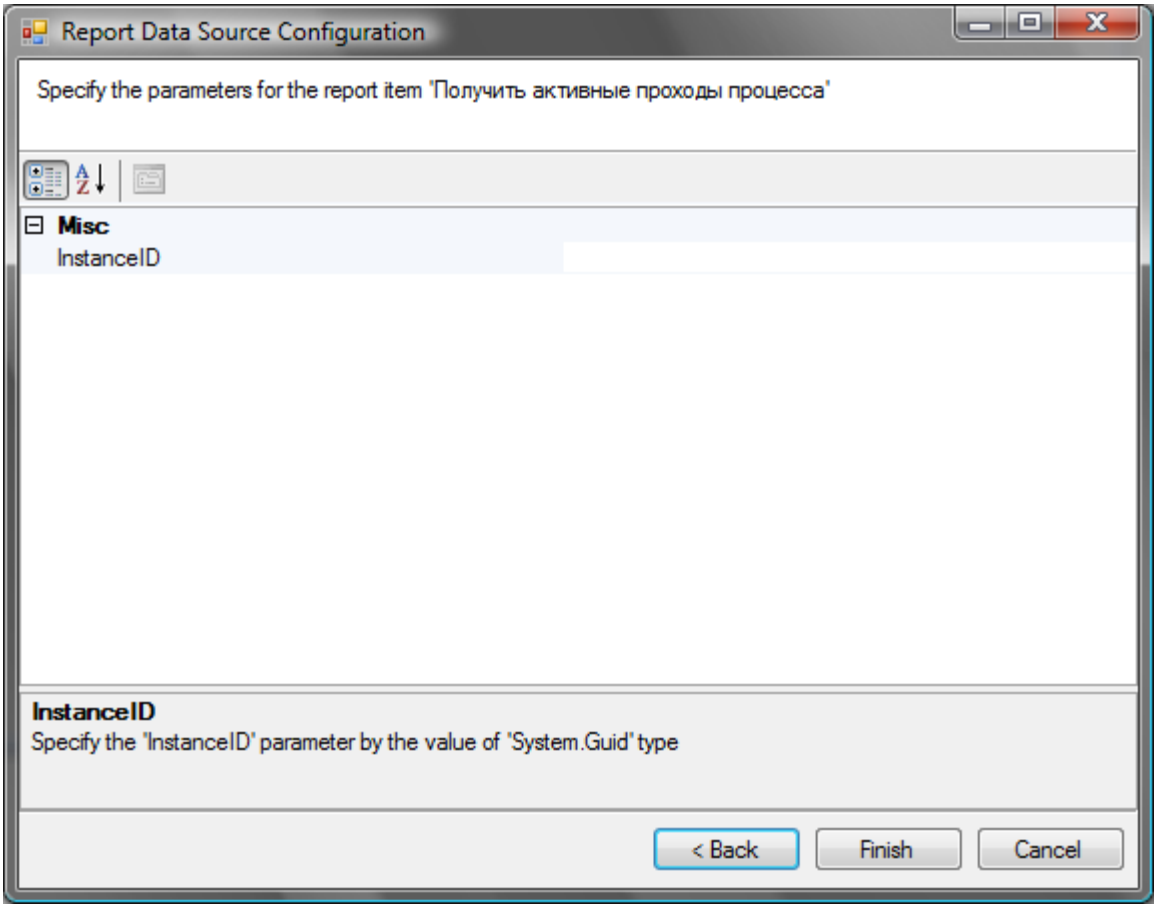


Рисунок 70. Ввод значений параметров хранимой процедуры

У каждой хранимой процедуры собственный уникальный набор параметров, определённый на этапе её разработки. Часть из них может быть обязательной, часть — нет. Для облегчения процесса заполнения параметров, мастер подсказывает тип каждого из них в строке статуса.

После ввода всех обязательных параметров и завершения работы мастера, источник данных готов к работе.

Все параметры также доступны для последующего изменения в виде свойств объекта `ReportDataSource`*:*

- `ReportID` — идентификатор хранимой процедуры
- `Parameters` — коллекция значений параметров процедуры

Элемент управления "BoundChooseBox"

Элемент управления `BoundChooseBox` предназначен для выбора значений ссылочных полей при помощи унифицированного интерфейса. В стандартном виде, элемент управления `ChooseBox` представляет из себя поле для отображения результатов выбора, и связанную с ним кнопку **...**, инициирующую новый выбор:

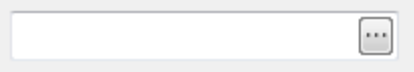


Рисунок 71. Элемент управления "BoundChooseBox

`BoundChooseBox` использует технологию Data Binding (что отражено в его названии) для автоматической привязки к одному из полей карточки ссылочного типа (`RefID` или `RefCardID`). При этом он автоматически определяет тип ссылки, и в зависимости от этого отображает соответствующий интерфейс для выбора значения:

- `RefCardID` — если поле является ссылкой на карточку, то для выбора значения `BoundChooseBox` отображает интерфейс Windows-клиента, в котором разрешено выбирать карточки.
- `RefID` — для типизированной ссылки на строку другой карточки (или справочника). На выбор открывается пользовательский интерфейс этой карточки. Например, если поле является ссылкой на строку секции справочника сотрудников, то на выбор будет открыт справочник сотрудников.

  Особым случаем является ссылка на строку секции карточки папок — в этом случае элемент управления откроет на выбор Windows-клиент, в котором разрешено выбирать папки.

Если поле `RefID` карточки является нетипизированным (не указаны значения типа и секции связанной карточки) — то `BoundChooseBox` не сможет работать с таким полем! В этом случае рекомендуется использовать элемент управления `CustomChooseBox`.

При работе в режиме связывания, элемент управления `BoundChooseBox` автоматически выполняет чтение данных из поля карточки и первичную инициализацию, а также автоматически сохраняет выбранное значение в поле карточки в процессе использования. Таким образом, разработчику нужно выполнить только предварительную настройку элемента управления в Design Time, и нет необходимости писать код обработчиков событий.

Кроме этого, элемент управления `BoundChooseBox` имеет дополнительные сервисные функции:

- Поддержка быстрого поиска и выбора объектов по нажатию горячих клавиш `Ctrl` + `K`.
- Возможность ассоциации дополнительного контекстного меню для поля

результатов выбора.

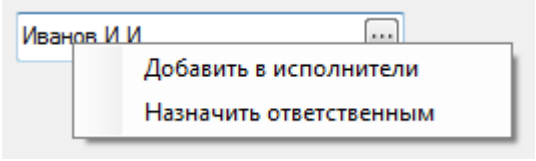


Рисунок 72. Контекстное меню "BoundChooseBox"

- Возможность ассоциации дополнительного контекстного меню для кнопки выбора

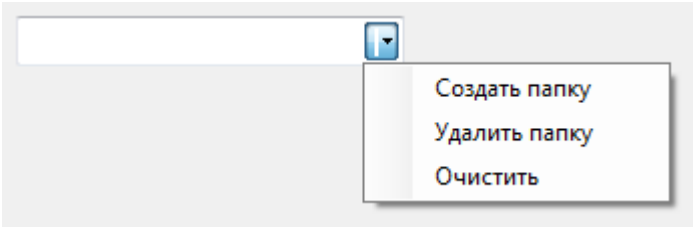


Рисунок 73. Контекстного меню кнопки выбора

Прежде чем приступить к использованию элемента управления BoundChooseBox, необходимо подготовить источник данных для него. В качестве источника данных может выступать объект RowDataSource, предназначенный для работы со строками секции карточки. Однако здесь необходимо учесть один нюанс: источник данных RowDataSource может возвращать сразу несколько строк (например, для секции коллекционного типа), а элемент управления BoundChooseBox может отображать только значение конкретного поля одной конкретной строки. Чтобы разрешить это противоречие, вводится ещё один промежуточный объект — BindingSource, который предназначен как раз, чтобы выбрать одну конкретную строку из набора, возвращаемого RowDataSource. Общую картину связывания данных в этом случае можно представить как на рисунке ниже.

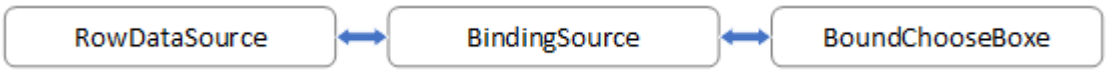


Рисунок 74. Алгоритм связывания данных

Объект BindingSource входит в число стандартных элементов управления Microsoft, и доступен на панели инструментов в группе Data. Использовать его достаточно просто, он содержит всего одно значимое свойство — **DataSource**, где нужно разместить ссылку на уже подготовленный источник данных RowDataSource.

После создания и настройки источников данных (RowDataSource и BindingSource) можно приступать к настройке собственно элемента управления BoundChooseBox.

В свойстве BindingSource необходимо указать ссылку на предварительно настроенный объект BindingSource. После этого, в свойстве **DataMember** появится возможность выбрать (или ввести вручную) название конкретного поля карточки, с которыми связывается элемент управления BoundDataSource. Именно описание этого поля в схеме карточки будет являться определяющим для дальнейшей работы элемента управления (что конкретно он будет выбирать).

Дополнительно можно указать свойство **FormatString** — определяющее, как именно BoundChooseBox будет отображать выбранное значение. Формировать отображаемое значение можно при помощи комбинации строковых констант и значений любых полей той строки, из которой производится выбор. Имена полей оформляются в строке отображения в фигурных скобках, а строковые константы — напрямую. Таким образом, пример строки отображения для вывода данных о сотруднике (строка секции Emploees из справочника сотрудникорв — RefStaff) может выглядеть так:

```
Фамилия:{LastName}_Имя:{FirstName}_Отчество:{MiddleName}
```

Свойства **CommandMenuStrip** и **ContextMenuStrip** позволяют сформировать контекстные меню для кнопки выбора и отображаемого значения соответственно. Для реализации самих контекстных меню используется стандартный элемент управления Microsoft **ContextMenuStrip**. Его необходимо предварительно добавить на форму карточки и настроить его свойства (добавить пункты меню), а также создать обработчики для событий выбора пунктов меню. Готовый и настроенный элемент управления можно указать в качестве значения свойства **CommandMenuStrip** или **ContextMenuStrip**.

В случае использования меню команд выбора *CommandMenuStrip*, стандартный значок кнопки выбора **...** будет заменён на иконку первого из пунктов контекстного меню.

Другие свойства элемента управления BoundChooseBox*:*

- **Caption** — заголовок окна выбора.
- **HideNotAvailable** — признак не показывать скрытые элементы для выбора.
- **SearchDelimiter** — разделительный символ для быстрого поиска.

  Элемент управления позволяет выполнять быстрый поиск по всем полям, указанным в строке отображения, в порядке их следования. Для разделения этих полей при вводе используется этот символ.

По умолчанию это пробел.

- `SearchSubstring` — для быстрого поиска по `Ctrl` + `K`. Признак, искать ли вхождение подстроки только сначала или в любом месте.

  По умолчанию значение равно `false`, то есть поиск ведется только по первым символам.

- `Text` — отображаемое значение элемента управления.
- `Value` — выбранное значение (идентификатор).

Методы

- `System.Guid ChooseValue()` — инициирует процесс выбора значения.
- `QuickSearch()` — инициирует быстрый поиск по введённому тексту.

Элемент управления "CardChooseBox"

Элемент управления `CardChooseBox` является специализированной версией `ChooseBox`, предназначенной **только** для выбора карточек.

Этот элемент управления не поддерживает технологию Data Binding, поэтому для его использования потребуется вручную написать код для инициализации существующих данных, и для сохранения результата выбора. Вместе с тем данный элемент управления предоставляет большую гибкость по сравнению с `BoundChooseBox`, за счет того, что позволяет разработчику управлять особенностями своей работы. `CardChooseBox` позволяет указывать поисковый запрос для фильтрации карточек, доступных на выбор. Это позволит, например, ограничить область выбора конкретным типом карточек.

Для указания фильтра используется свойство `CardsFilter`, которое доступно только программно (в runtime). В качестве значения свойства необходимо установить ссылку на предварительно созданный объект `SearchQuery` (подробнее об использовании поиска в соответствующей главе).

Остальные свойства элемента управления `CardChooseBox` соответствуют аналогичным свойствам `BoundChooseBox`.

Для чтения выбранного значения из элемента управления можно использовать свойства `Text` (отображаемое значение) и `Value` (идентификатор выбранного значения), а также событие `OnValueChanged(object sender, System.EventArgs e)` — которое инициируется после завершения нового выбора.

Пример инициализации элемента управления CardChooseBox ограничивающий возможность выбора только карточками файла:

```
const string FILE_CARD_TYPE = "{2BBD0A41-265E-4FF8-82D6-C6342F34B1AF}";
SearchQuery query = session.CreateSearchQuery();
query.AttributiveSearch.CardTypeQueries.AddNew(new Guid(FILE_CARD_TYPE));
cardChooseBox1.CardsFilter = query;
```

Элемент управления "RowChooseBox"

RowChooseBox, является специализированным средством для выбора строк из карточек (справочников). От BoundChooseBox его отличает отсутствие поддержки Data Binding, но большая гибкость в настройке и использовании за счет дополнительных свойств:

- System.GuidCardTypeId — идентификатор типа карточки, из которой нужно выбирать строки.
- System.GuidSectionTypeId — идентификатор типа секции, из которой нужно выбирать строки.
- System.GuidCardInstanceId — идентификатор конкретного экземпляра карточки, из которой нужно выбирать строки. Используется в случае если карточка **не** является справочником.

Значения этих свойств нужно указать в design-time, или программно при инициализации элемента управления, а также добавить обработчики для чтения/записи значений из элемента управления.

Элемент управления "BoundTreeView"

Элемент управления BoundTreeView предназначен для отображения данных иерархической секции. Как следует из его названия, этот элемент поддерживает технологию Data Binding — то есть позволяет настроить его без необходимости написания какого-либо кода. Это же объясняет и сам факт его появления — он вызван тем, что стандартный элемент управления TreeView не поддерживает технологию Data Binding.

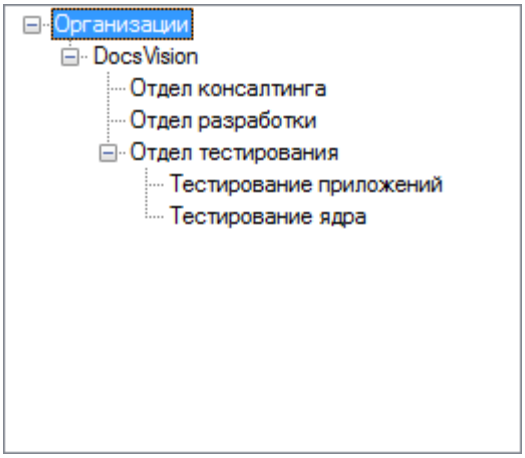


Рисунок 75. Элемент управления "BoundTreeView"

По способу использования, элемент управления `BoundTreeView` похож на `BoundChooseBox` — он привязывается к источнику данных секции (`RowDataSource`) с использованием вспомогательного объекта `BindingSource`.

Ссылка на `Binding Source` записывается в одноименное свойство объекта `BoundTreeView`.

Кроме этого, необходимо заполнить следующие свойства элемента управления:

- `DataMember` — имя поля секции, к которому будет привязываться элемент управления. Значение этого поля будет формировать значение (Value) каждого узла в дереве.
- `RootNodeText` — имя корневого узла (если это свойство не указать, то имя корневого узла будет пустым)
- `FormatString` — отображаемое значение имён узлов. Формировать отображаемое значение можно при помощи комбинации строковых констант и значений любых полей той строки, с которой связан элемент управления. Имена полей оформляются в строке отображения в фигурных скобках, а строковые константы — напрямую.

Элемент управления "WizardConrol"

Элемент управления `WizardControl` позволяет облегчить процесс создания стандартных мастеров (последовательность форм с возможностью навигации вперед/назад). Внешний вид элемента управления:

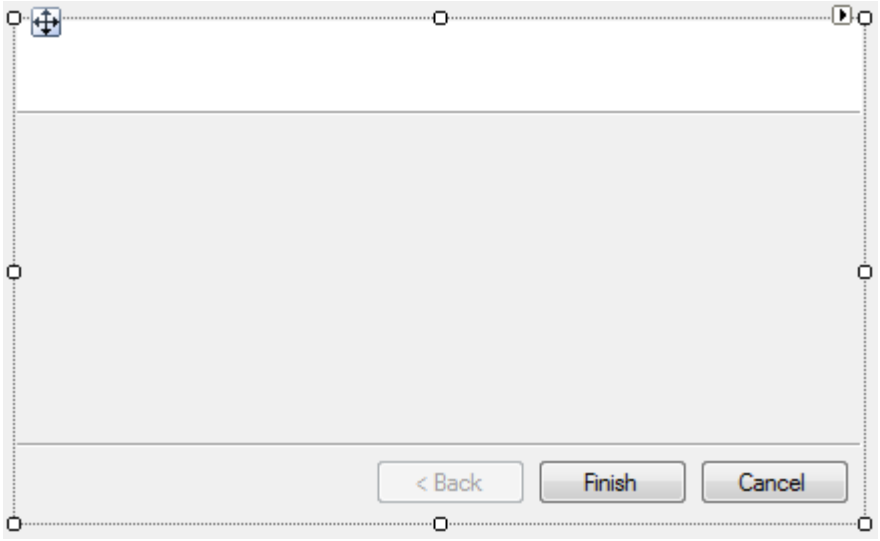


Рисунок 76. Элемент управления "WizardConrol"

Свойства

- `int ButtonHeight` — высота стандартных кнопок (**Back**, **Next**, **Cancel**, **Finish**).
- `int ButtonWidth` — ширина стандартных кнопок (**Back**, **Next**, **Cancel**, **Finish**).
- `int HorizontalSpacing` — расстояние между кнопками по горизонтали.
- `int VerticalSpacing` — расстояние между кнопками по вертикали.
- `string BackButtonText` — текст кнопки **Back**.
- `string CancelButtonText` — текст кнопки **Cancel**.
- `string FinishButtonText` — текст кнопки **Finish**.
- `string HelpButtonText` — текст кнопки **Help**.
- `System.Drawing.SizeDefaultSize` — начальный размер элемента управления.
- `System.Windows.Forms.FlatStyleFlatStyle` — стиль отрисовки элемента управления.
- `bool HelpButtonVisible` — признак, отображать ли кнопку помощи (**Help**).

Набор шагов мастера задаётся при помощи коллекции WizardPages — которую можно редактировать, используя специальный визуальный редактор:

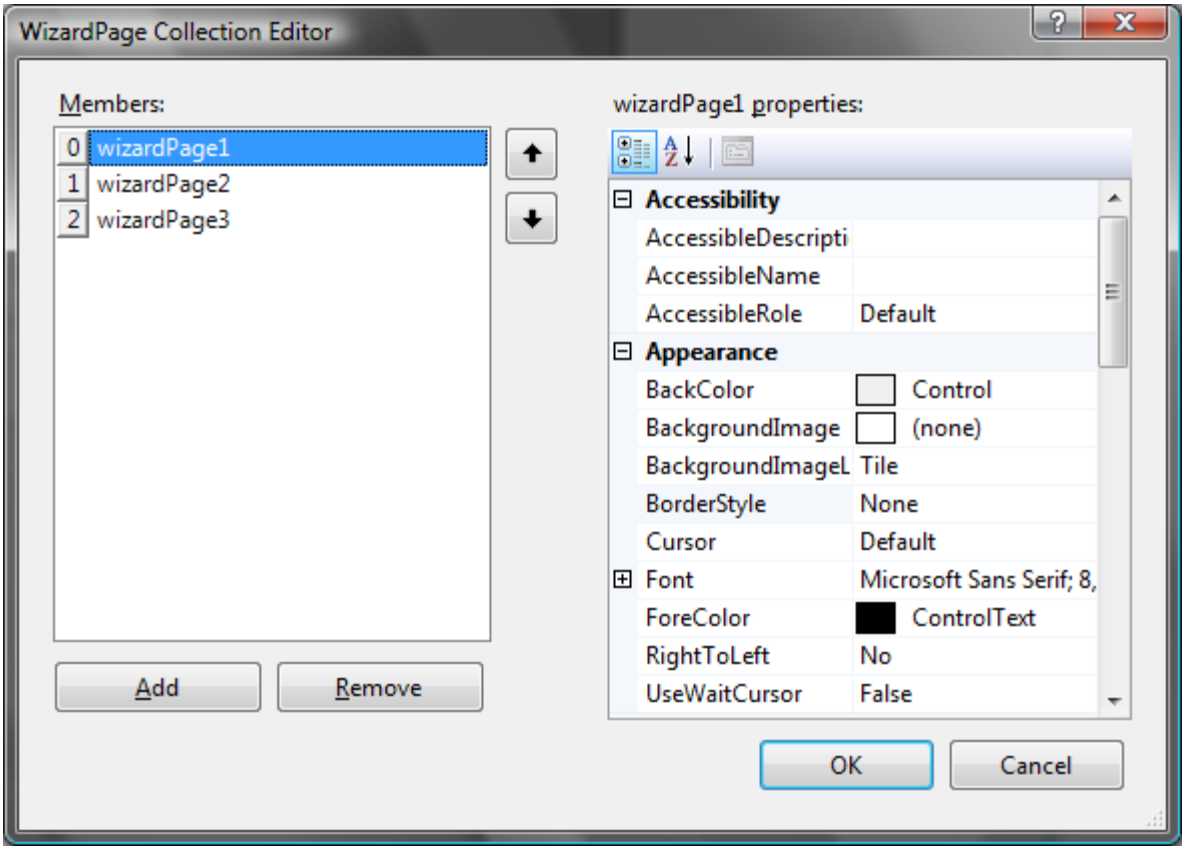


Рисунок 77. Визуальный редактор WizardControl

Редактор позволяет добавить произвольное количество страниц (шагов мастера), а также менять порядок их следования. Каждая страница имеет единственное значимое свойство — `Text`, которое позволяет определить заголовок страницы.

После определения всех страниц, они появляются в мастере, и можно добавлять интерактивные элементы управления непосредственно на них. Переход между страницами при этом осуществляется с помощью стандартных навигационных кнопок мастера.

Для завершения разработки мастера, остается добавить обработчики основных событий:

- `OnWizardFinished(System.EventArgs e)` — событие инициируется при нажатии на кнопку **Finish**, и позволяет реализовать необходимые действия по завершению его работы (сохранить результаты работы, выполнить какие-то действия, и т.д.).
- `OnWizardCanceled(System.EventArgs e)` — событие инициируется при нажатии на кнопку **Cancel**, и позволяет реализовать необходимые действия по аварийному прерыванию его работы (закрыть форму на которой расположен мастер).

- `OnSelectedIndexChanging(DocsVision.Platform.WinForms.Controls.WizardControl.SelectedIndexChangingEventArgs e)` — событие инициируется **до** смены вкладки (при перемещении вперед или назад), и позволяет в случае необходимости воспрепятствовать этому переходу. Это может быть полезно для реализации проверки ввода всех необходимых параметров, прежде чем разрешить переход к следующему шагу.
- `OnSelectedIndexChanged(System.EventArgs e)` — это событие возникает **после** перехода на новый шаг (вперед или назад), и позволяет выполнить какие-то действия по инициализации элементов управления на новом шаге.

Сборник примеров

Руководство разработчика Docsvision

Сборник примеров

В разделе представлены примеры работы с API Docsvision.

- Использование объектной модели уровня бизнес-логики
- Примеры реализации компонента карточек, библиотек карточек и расширений
- Примеры использования API Workflow
- Больше подробностей в навигационном меню слева…

Использование объектной модели уровня бизнес-логики

Руководство разработчика Docsvision

Использование объектной модели уровня бизнес-логики

В разделе представлены примеры использование объектной модели уровня бизнес-логики.

- Инициализация контекста объектов
- Добавление ссылки на другую карточку к карточке документа
- Изменение состояния карточки
- Больше подробностей в навигационном меню слева…

Инициализация контекста объектов

Если в качестве базового класса для карточки используется базовая карточка `DocsVision.BackOffice.WinForms.BaseCardControl`, создавать и инициализировать контекст объектов нет необходимости — он доступен через свойство `ObjectContext`, при этом в него уже добавлены необходимые сервисы для работы с

"Базовыми объектами".

Для класса, унаследованного от `DocsVision.BackOffice.WinForms.ScriptClassBase`, *контекст объектов* будет доступен через свойство `base.CardControl.ObjectContext`.

Пример инициализации контекста объектов, а также добавления в него основных сервисов и преобразователей данных приведён ниже.

```
using DocsVision.BackOffice.ObjectModel;
using DocsVision.BackOffice.ObjectModel.Mapping;
using DocsVision.BackOffice.ObjectModel.Services;
using DocsVision.Platform.Data.Metadata;
using DocsVision.Platform.ObjectManager;
using DocsVision.Platform.ObjectModel;
using DocsVision.Platform.ObjectModel.Mapping;
using DocsVision.Platform.ObjectModel.Persistence;
using DocsVision.Platform.SystemCards.ObjectModel.Mapping;
using DocsVision.Platform.SystemCards.ObjectModel.Services;

namespace ConnectDV
{
 class Program
 {
  static UserSession userSession; ①
  static ObjectContext objectContext; ②

  static void Main(string[] args)
  {
   CreateContext();
  }

  static void CreateContext()
  {

   string serverDocsvisionUrl =
"ConnectAddress=http://localhost/DocsVision/StorageServer/StorageServerService.asmx;BaseN
ame=SampleBase;UserName=IvanovII;Password=SamplePass"; ③

   SessionManager sessionManager = SessionManager.CreateInstance(serverDocsvisionUrl); ④
   userSession = sessionManager.CreateSession();

   var sessionContainer = new System.ComponentModel.Design.ServiceContainer(); ⑤
   sessionContainer.AddService(typeof(DocsVision.Platform.ObjectManager.UserSession),
userSession);
```

```
   objectContext = new ObjectContext(sessionContainer); ⑥

   IObjectMapperFactoryRegistry mapperFactoryRegistry = objectContext.GetService
<IObjectMapperFactoryRegistry>();
   mapperFactoryRegistry.RegisterFactory(typeof(SystemCardsMapperFactory)); ⑦
   mapperFactoryRegistry.RegisterFactory(typeof(BackOfficeMapperFactory));

   IServiceFactoryRegistry serviceFactoryRegistry = objectContext.GetService
<IServiceFactoryRegistry>();
   serviceFactoryRegistry.RegisterFactory(typeof(SystemCardsServiceFactory)); ⑧
   serviceFactoryRegistry.RegisterFactory(typeof(BackOfficeServiceFactory));

   objectContext.AddService<IPersistentStore>(DocsVisionObjectFactory
.CreatePersistentStore(new SessionProvider(userSession), null)); ⑨

   IMetadataProvider metadataProvider = DocsVisionObjectFactory.CreateMetadataProvider
(userSession); ⑩
   objectContext.AddService<IMetadataManager>(DocsVisionObjectFactory
.CreateMetadataManager(metadataProvider, userSession));
   objectContext.AddService<IMetadataProvider>(metadataProvider);
  }
 }
}
```

① Сессия пользователя.

② Контекст объектов.

③ Строка соединения с сервером Docsvision.

④ Подключение к серверу и открытие сессии.

⑤ Инициализация сервис-провайдера.

⑥ Инициализация контекста объектов.

В качестве контейнера может выступать компонент карточки, унаследованный от `DocsVision.Platform.WinForms.CardControl`.

⑦ Получение сервис-реестра и регистрация фабрик преобразователей.

⑧ Получение сервис-реестра и регистрация фабрик сервисов.

⑨ Регистрация сервиса для работы с хранилищем Docsvision.

⑩ Регистрация поставщика метаданных карточек.

Для корректной сборки необходимо подключить к проекту следующие библиотеки:

- `DocsVision.Platform.ObjectManager.dll`

- DocsVision.Platform.ObjectModel.dll
- DocsVision.Platform.SystemCards.ObjectModel.dll
- DocsVision.Platform.StorageServer.dll
- DocsVision.BackOffice.ObjectModel.dll

Добавление ссылки на другую карточку к карточке документа

Ниже приведён пример добавления ссылки на карточку библиотеки *Базовые объекты* в карточку типа Документ:

```
①

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ②

BaseCard card = objectContext.GetObject<BaseCard>(new Guid("00000000-0000-0000-0000-
000000000001")); ③

LinksLinkType linkType = objectContext.GetObject<LinksLinkType>(new Guid("502F7FE3-477F-
492F-9F43-ED2AA7CB32D9")); ④

IReferenceListService referenceListService = objectContext.GetService
<IReferenceListService>(); ⑤

ReferenceList referenceList = document.MainInfo.ReferenceList;
if (referenceList = null) ⑥
{
 referenceList = referenceListService.CreateReferenceList(); ⑦

 document.MainInfo.ReferenceList = referenceList;
}

referenceListService.CreateReference(referenceList, linkType, card, string.Empty, false);
⑧

objectContext.SaveObject<Document>(document); ⑨
```

① Инициализация контекста объектов.

② Получение документа, в который добавляется ссылка.

③ Идентификатор карточки, которая должна быть добавлена в виде ссылки.

④ Получение типа ссылки (в примере соответствует типу "В ответ на").

Идентификатор может быть получен из Справочника ссылок.

⑤ Получение сервиса для работы со списками ссылок.

⑥ Получение существующего списка ссылок из документа, либо создание нового.

⑦ Создаем список ссылок и сохраняем его в карточку документа.

⑧ Создание слабой ссылки на карточку.

⑨ Сохранение документа.

Изменение состояния карточки

Пример кода, переводящего карточку группы заданий в завершённое состояние:

```
ObjectContext context = this.LocalContext; ①

BaseCard taskGroupCard = GetParentCard(); ②

if (taskGroupCard != null)
{
    if (this.LocalContext.LockObject(taskGroupCard, false)) ③
    {
        ObjectModel.Services.IStateService stateService = context.GetService<ObjectModel
.Services.IStateService>(); ④

        StatesStateMachineBranch branch = stateService.FindBranchByBuiltIn(TaskGroup
.PerformanceToCompletedByCompleteBranch, taskGroupCard.SystemInfo.State); ⑤
        if (branch = null)
            return false;

        taskGroupCard.SystemInfo.State = branch.EndState; ⑥

        this.LocalContext.SaveObject<BaseCard>(taskGroupCard); ⑦

        this.LocalContext.UnlockObject(taskGroupCard); ⑧
    }
}
```

① Получаем контекст.

② Получаем карточку.

③ Блокируем карточку.

④ Получаем сервис состояний.

⑤ Получаем ветку перехода по встроенной ветке перехода.

⑥ Получаем конечное состояние веки и присваиваем его полю состояния в

карточке.

⑦ Сохраняем карточку группового задания.

⑧ Разблокируем карточку.

Добавление собственного кода при создании задания по группе заданий

При создании задания по группе заданий можно добавлять собственный код стандартным способом, по аналогии с другими случаями:

1. На уровне логики карточки группы заданий. Унаследовать `TaskGroupEventHandlerService`, в нём дореализовать нужную функциональность, определив свои версии нужных методов. Заменить в справочнике видов (в секции `Services` у нужного вида ГЗ) штатную регистрацию сервиса `TaskGroupEventHandlerService` на собственную.
2. На уровне логики карточки задания. Реализовать `ITaskEventHandlerService` или `IEventHandlerService` (можно унаследовать базовый класс `EventHandlerService`). Реализовать метод-обработчик смены состояния задания в котором при запуске задания выполнять нужную логику. Добавить в справочнике видов (в секции `Services` у нужного вида задания) регистрацию этого нового сервиса обработчика. Другой вариант — унаследовать `TaskEventHandlerService` и переопределять его логику, но тогда нужно заменять регистрацию стандартного сервиса в справочнике видов, а не добавлять параллельно (иначе одна и та же базовая логика будет отрабатывать 2 раза).

Создание дочерней карточки задания

Пример создания дочернего задания связанного с карточкой документа:

```
ObjectContext context = GetContext(); ①

Guid kindId = GetTaskKindId(); ②

KindsCardKind taskKind = context.GetObject<KindsCardKind>(kindId); ③

Guid parentCardId = GetParentCardId(); ④

Document document = context.GetObject<Document>(parentCardId); ⑤

TaskList taskList = document.MainInfo.Tasks; ⑥

if (taskList = null) ⑦
{
    ObjectModel.Services.ITaskListService taskListService = context.GetService
```

```
<ObjectModel.Services.ITaskListService>();
    taskList = taskListService.CreateTaskList(); ⑧

    document.MainInfo.Tasks = taskList; ⑨
}

Task parentTask = GetParentTask(); ⑩

ObjectModel.Services.ITaskService taskService = context.GetService<ObjectModel.Services
.ITaskService>(); ⑪

Task task = TaskService.CreateChildTask(parentTask, taskKind, document, taskList); ⑫

context.AcceptChanges(); ⑬
```

① Получаем контекст.

② Получаем идентификатор вида задания.

③ Получаем вид задания.

④ Получаем идентификатор родительской карточки.

⑤ Получаем документ.

⑥ Получаем список заданий документа.

⑦ Если список заданий в документе отсутствует — его необходимо создать.

⑧ Получаем сервис списков заданий.

⑨ Сохраняем список заданий в документ.

⑩ Получаем родительское задание, предполагается, что функция GetParentTask вернет задание, дочерним для которого будет является созданное.

⑪ Получаем сервис для работы с заданиями.

⑫ Создаем задание.

⑬ Сохраняем изменения.

Проверка допустимости выполнения операции над карточкой

Использование сервисов состояний и прав доступа для проверки допустимости выполнения операции над карточкой.

Пример проверки допустимости выполнения встроенной операции `TakeToWorkOperation` для карточки задания:

```
ObjectContext context = GetContext(); ①
```

```
Guid taskId = GetTaskId(); ②

Task task = context.GetObject<Task>(taskId); ③

ObjectModel.Services.IStateService stateService = context.GetService<ObjectModel.
Services.IStateService>(); ④

ObjectModel.Services.IAccessCheckingService accessService = context.GetService
<ObjectModel.Services.IAccessCheckingService>(); ⑤

bool operationAllowed = stateService.IsOperationAllowed(Task.TakeToWorkOperation, task)
&& accessService.IsOperationAllowed(task, Task.TakeToWorkOperation); ⑥
```

① Получаем контекст.

② Получаем идентификатор задания.

③ Получаем задание.

④ Получаем сервис состояний.

⑤ Получаем сервис проверки доступа.

⑥ Используем сервис для проверки разрешённости операции по состояниям и по правам доступа.

Проверка подлинности ЭП в документе

Ниже приведён пример проверки действительности подписи, установленной на поля документа. В этом случае проверка подписи на файлах не проводится. Для проверки вызывается метод `CheckActualSignature` сервиса документов `IDocumentService`.

```
IDocumentService documentService = objectContext.GetService<IDocumentService>();
IUserProfileCardService iUserProfileCardService = objectContext.GetService
<IUserProfileCardService>();

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ①

var fields = documentService.GetKindSettings(document.SystemInfo.CardKind
).DocumentSignature.Fields; ②

foreach (var item in document.MainInfo.SignatureList.Signatures)
{
 X509Certificate2 certificate = iUserProfileCardService.GetCertificate(item.Signer
.AccountName); ③
```

```
 bool isCheckOk = documentService.CheckActualSignature(document, fields, item); ④
 Console.WriteLine("{0} -- {1}",item.Signer.DisplayName, isCheckOk);
}
```

① Получение документа с идентификатором `00000000-0000-0000-0000-000000000000`.

② Получение списка подписываемых полей в данном виде документа.

③ Получение сертификата из профиля пользователя подписавшего документ (`item.Signer`).

④ Проверка подписи, при `true` — подпись действительна.

Синхронизация Справочника сотрудников с Active Directory

Для синхронизации *Справочника сотрудников* с Active Directory достаточно вызвать специальный метод сервиса для работы со Справочником сотрудников

```
IStaffService staffService = objectContext.GetService<IStaffService>(); ①

SynchronizationDepartmentsSettings syncSettings = new
SynchronizationDepartmentsSettings(); ②

LongProcessManager longProcessManager = new LongProcessManager(); ③

staffService.SynchronizeWithActiveDirectory(syncSettings, longProcessManager); ④
```

① Инициализация контекста объектов и получение сервиса.

② Инициализация параметров синхронизации.

③ Инициализация менеджера длительных процессов.

④ Выполнение синхронизации.

В приведенном примере класс `SynchronizationDepartmentsSettings` должен реализовывать интерфейс `IADSynchronizationSettings`, например, следующим образом:

```
class SynchronizationDepartmentsSettings : IADSynchronizationSettings
{
 public bool CreateFolders { get { return false; } }
 public bool DeleteItems { get { return false; } }
 public bool LoadCertificates { get { return false; } }
 public bool LoadDisabled { get { return false; } }
 public bool NotCheckDomain { get { return false; } }
```

```
 public ADSynchronizationMode SynchronizationMode
 {
  get { return ADSynchronizationMode.FromADToDictionary; } ①
 }
 public bool SynchronizeChildren { get { return false; } }

 public bool SynchronizeDepartments
 {
  get { return true; } ②
 }
 public bool SynchronizeGroups { get { return false; } }
 public bool SynchronizeNewMoved { get { return false; } }
 public bool SynchronizeRoles { get { return false; } }
}
```

① Направление синхронизации в сторону Справочника сотрудников.

② Разрешить синхронизацию подразделений.

Здесь класс `SynchronizationDepartmentsSettings` устанавливает требование по синхронизации подразделений в *Справочнике сотрудников* информацией, полученной из Active Directory.

Полное описание свойств класса `SynchronizationDepartmentsSettings` смотрите в описании интерфейса `IADSynchronizationSettings`.

Менеджер длительных процессов `longProcessManager` должен реализовывать интерфейс `ILongProcessManager`, и для простоты определим его следующим образом:

```
class LongProcessManager : ILongProcessManager
{
 public void Cancel() { }
 public bool CancellationPending { get {return false;} }
 public bool PausePending { get { return false; } }
 public void LogMessage(string message) { }
}
```

Чтобы синхронизация могла быть выполнена необходимо связать элемент *Справочника сотрудников* и объект Active Directory. Подробнее об этом читайте в разделе /dv6/backoffice/6.1/desdirs/staff/active-directory-sychronization/[по ссылке].

Согласование документа

Для отправки документа на согласование с помощью API необходимо, чтобы в шаблоне, из которого создаётся согласование, был установлен флаг **Запускать согласование без показа карточки**. Также должен существовать режим создания согласования (в *Справочнике видов карточек*), который задействует данный шаблон.

1. Подключите к проекту или скрипту карточки дополнительные сборки: `DocsVision.ApprovalDesigner.ObjectModel.dll` и `DocsVision.DocumentsManagement.ObjectModel.dll`, в которых определены преобразователь данных и сервис для отправки согласования.
2. Подключите дополнительные пространства имён:
   - `DocsVision.ApprovalDesigner.ObjectModel.Mapping`
   - `DocsVision.ApprovalDesigner.ObjectModel.Services`
   - `DocsVision.BackOffice.ObjectModel.Services`
   - `DocsVision.DocumentsManagement.ObjectModel.Services`
   - `DocsVision.Platform.ObjectModel.Mapping`
   - `System.Linq`
3. Подключите в коде сервисы и преобразователи данных для работы с согласованием:

```
IServiceFactoryRegistry serviceFactoryRegistry = context.GetService
<IServiceFactoryRegistry>(); ①
serviceFactoryRegistry.RegisterFactory(typeof(ApprovalDesignerServiceFactory));

var mapperFactoryRegistry = context.GetService<IObjectMapperFactoryRegistry>(); ②
mapperFactoryRegistry.RegisterFactory(typeof(ApprovalDesignerMapperFactory));
```

① Добавляем в контекст фабрику `ApprovalDesignerServiceFactory`.

② Добавляем в контекст преобразовать данных `ApprovalDesignerMapperFactory`.

4. Получите идентификатор согласуемого документа, например, так:

```
Guid initialCardId = context.GetObjectRef(this.BaseObject).Id; ①
```

① Получаем идентификатор документа, при в скрипте карточки.

5. Получите настройки режима создания согласования:

```
KindsCardKind cardKind = context.GetObject<KindsCardKind>(new Guid("9C0E5586-41B8-411E-B5CD-C94B605CB7A1")); ①

KindsCardCreationSetting cardCreationSetting = cardKind.CreationSettings
.FirstOrDefault(t => t.ModeName.Equals("СогласованиеПример")); ②
```

① Получаем настройки вида карточки "Согласование КС".

② Получаем настройки режима создания "СогласованиеПример". Режим использует шаблон согласования с флагом **Запускать согласование без показа карточки**.

6. Создайте новое согласование для документа:

```
Guid reconciliationId = context.GetService<IReconcileService
>().CreateReconciliationCard(initialCardId, cardCreationSetting);
```

7. Вызовите метод HandleDocumentAfterReconcileCreated, который завершит подготовку согласования и запустит его:

```
CardData reconciliationCardData = Session.CardManager.GetCardData(reconciliationId);
context.GetService<IReconcileService>().HandleDocumentAfterReconcileCreated((Document)
this.BaseObject, reconciliationCardData);
```

Приведенным способом можно отправлять документы на согласование из скрипта карточки или из внешнего приложения.

Использование базового API

Руководство разработчика Docsvision

Использование базового API

В разделе представлены примеры работы с базовым API Docsvision.

- Создание нового представления
- Пример использования режима отложенных изменений

- Позиционирование курсора на папке
- Больше подробностей в навигационном меню слева...

Создание нового представления

```
DocsVision.Platform.ObjectManager.ViewModel.View view = session.CreateView(); ①

view.Columns.AddNew("Name").Caption = "Тема документа";
view.Columns.AddNew("Number").Caption = "Номер документа";
view.Columns.AddNew("ApproverPos").Caption = "Должность согласующего лица";
view.Columns.AddNew("FilesCount").Caption = "Количество файлов";
view.Columns.AddNew("PropertiesCount").Caption = "Количество свойств"; ②

view.Sorting.AddNew("Number").Ascending = true; ③

ViewElement element = view.Elements.AddNew(new Guid("8C77892A-21CC-4972-AD71-
A9919BCA8187")); ④

element.SectionFields.AddNew("Name").Name = "Name";
element.SectionFields.AddNew("Number").Name = "FullNumber"; ⑤

JoinDefinition join1 = element.JoinDefinitions.AddNew("Employees"); ⑥
join1.SourceField = "InstanceID";
join1.DestinationField  = "InstanceID";
join1.SectionId = new Guid("47C41171-9C64-450A-A3A6-102B3156AD79");

DataCondition cond = join1.ConditionGroup.Conditions.AddNew();
cond.Operation = ConditionOperation.Equals;
DataConditionItem item1 = cond.ConditionItems.AddNew();
item1.SectionAlias = "Employees";
item1.Value = "Type";
item1.DataType = DataType.Integer;
DataConditionItem item2 = cond.ConditionItems.AddNew();
item2.DataType = DataType.Integer;
item2.Value = 3; ⑦

JoinDefinition join2 = element.JoinDefinitions.AddNew("RefStaffEmployees");
join2.SourceField = "EmployeeID";
join2.DestinationField = "RowID";
join2.SectionId = new Guid("DBC8AE9D-C1D2-4D5E-978B-339D22B32482"); ⑧

JoinDefinition join3=element.JoinDefinitions.AddNew("RefStaffPositions");
join3.SourceField = "Position";
join3.DestinationField = "RowID";
join3.SectionId = new Guid("CFDFE60A-21A8-4010-84E9-9D2DF348508C"); ⑨
```

```
SectionField field = element.SectionFields.AddNew("ApproverPos");
field.SectionAlias = "RefStaffPositions";
field.Name = "Name"; ⑩

CardType type =  session.CardManager.CardTypes[new Guid("C1FED883-08DE-420F-8FB4-
C16CEFFC1630")]; ⑪

CardVirtualField virtfield = type.VirtualFields["FilesCount"];
element.MergeVirtualField(virtfield.Export(), "Main", "FilesCount");

JoinDefinition join4 = element.JoinDefinitions.AddNew("Properties");
join4.SourceField = "InstanceID";
join4.DestinationField = "InstanceID";
join4.SectionId = new Guid("B822D7D1-2280-4B51-AE58-A1CF757C5672"); ⑫

ComputedField compfield = element.ComputedFields.AddNew("PropertiesCount"); ⑬
ComputationPart comppart = compfield.ComputationGroup.ComputationParts.AddNew();
comppart.DataItem.SectionAlias = "Properties";
comppart.DataItem.Value = "RowID";
```

① Создание представления по документам с пятью колонками:

- Тема документа.
- Номер документа.
- Должность согласующего лица.
- Количество файлов.
- Количество свойств.

② Добавление колонок.

③ Сортировка — по полю *Номер*.

④ В представление будут выводиться данные только по Входящим документам.

Поэтому будет только один элемент данных (`ViewElement`). Его базовая секция — *Основная информация* Входящего документа.

⑤ Данные двух колонок — *Тема* и *Номер* выводятся непосредственно из физических полей секции.

⑥ Формирование колонки *Должность согласующего лица*.

Базовой секцией представления является "Основная информация". Согласующее лицо хранится в секции `Сотрудники`. Поэтому сначала нужно присоединить секцию `Сотрудники` к представлению.

⑦ Условие присоединения: в секции `Сотрудники` поле `Type` должно иметь значение `3` (согласующее лицо).

⑧ Далее нужно присоединить секцию справочника сотрудников.

⑨ Наконец, для получения названия должности, нужно присоединить секцию должностей.

⑩ Добавляем к элементу представления колонку — название должности.

⑪ Четвертая колонка — количество файлов — уже определена в описании карточки в виде виртуального поля. Нужно только добавить её к представлению.

⑫ Последняя колонка — *Количество свойств* — это вычисляемое поле. Представляет собой агрегацию (количество строк) по секции `Свойства`.

⑬ Присоединяем секцию `Свойства`.

Пример использования режима отложенных изменений

В следующем примере в текущей сессии `userSession` мы получаем данные карточки. Активировав режим отложенных изменений для карточки, мы проводим изменение секции `MainInfo`, после чего отключаем режим отложенных изменений.

```
CardData cardData = userSession.CardManager.GetCardData(new System.Guid
("идентификатор_карточки")); ①

if (!cardData.InUpdate) cardData.BeginUpdate(); ②

try
{
    SectionData sectionData = cardData.Sections[card.Type.Sections["MainInfo"].Id]; ③

    RowData rowData = sectionData.FirstRow; ④

    rowData["Author"] = "Иванов";
    rowData["Number"] = 11;
    rowData["CreationDate"] = DateTime.Now; ⑤

    cardData.EndUpdate(); ⑥
}
catch(Exception ex)
{
   if (cardData.InUpdate) cardData.CancelUpdate(); ⑦
```

```
}
```

① Получение с сервера данных карточки с известным идентификатором.

② Включение режима отложенных изменений.

③ Получение данных секции с именем `MainInfo`.

④ Получение первой строки секции, если строки нет — она будет создана.

⑤ Запись значений в поля.

⑥ Выключение режима отложенных изменений, передача всех изменений на сервер.

⑦ При возникновении ошибки — отмена всех изменений.

Позиционирование курсора на папке

Ниже приведён пример кода расширения Windows-клиента, которое регистрирует кнопку ленты инструментов, позволяющей осуществить позиционирование курсора на нужной папке.

```
using DocsVision.Platform.Extensibility;
using DocsVision.Platform.WinForms;
using System;
using System.Collections.Generic;
using System.Runtime.InteropServices;

namespace DocsVision.Test.NavPluging
{
  [ComVisible(true)]
  [Guid("6A0676EE-1DAF-4A59-B5EB-E0B5B4C175E1")]
  [ClassInterface(ClassInterfaceType.None)]
  public partial class Plugin : NavExtension
  {
   public Plugin()
   {
    InitializeComponent();
   }

   protected override NavExtensionTypes SupportedTypes ①
   {
    get
    {
     return NavExtensionTypes.Command;
    }
   }
```

```
    protected override string GetExtensionName(NavExtensionTypes extensionType) ②
    {
     return "Быстрый выбор";
    }

    protected override IEnumerable<NavCommand> CreateCommands() ③
    {
     NavCommand myCommand = new NavCommand();
     myCommand.CommandType = NavCommandTypes.ToolBar;
     myCommand.Name = "Выбрать";
     myCommand.Description = "Выбирает домашнюю папку";

     return new List<NavCommand> { myCommand };
    }

    protected override NavCommandStatus QueryCommandStatus(NavCommand command,
NavCommandContext context)
    {
     return NavCommandStatus.Enabled;
    }

    protected override void InvokeCommand(NavCommand command, NavCommandContext context)
    {

     CardFrame.CardHost.ActivateFolder(new Guid("3C2A8C6F-B992-4C68-813E-0EE36A205746"),
false); ④
    }
   }
}
```

① Возвращает типы расширений, реализуемых данным классом.

② Возвращаемое имя будет использовано Windows-клиентом при именовании группы кнопок в ленте инструментов.

③ Возвращает Windows-клиенту список доступных команд.

④ Выполнение позиционирования на папке с идентификатором `3C2A8C6F-B992-4C68-813E-0EE36A205746`.

В примере, мы регистрируем только одну команду, которая вызывает метод `ActivateFolder(Guid, Boolean)`, выполняющий позиционирование курсора на определённой папке Windows-клиента. Папка может быть обычной, либо виртуальной.

Правило создание расширение Windows-клиента приведено в разделе

Расширение Windows-клиента.

Пример использования серверного поиска

Ниже приведён пример серверного поиска, выполняемого из скрипта карточки, по следующим данным карточки: части названия, автору и дате создания (должна быть больше указанной). Необходимые данные для серверного поиска получаем из соответствующих элементов управления.

```
private void Поиск_Click(System.Object sender, System.EventArgs e)
{
 ICustomizableControl control = CardControl;

 ILayoutPropertyItem cardName = control.FindPropertyItem<ILayoutPropertyItem>(
"Название");
 ILayoutPropertyItem cardAuthorId = control.FindPropertyItem<ILayoutPropertyItem>(
"Автор");
 ILayoutPropertyItem cardCreateBy = control.FindPropertyItem<ILayoutPropertyItem
>("ДатаСозданияОт"); ①

 if(cardName = null || cardAuthorId = null || cardCreateBy = null) return;

 SearchQuery searchQuery = this.Session.CreateSearchQuery(); ②
 CardTypeQuery cardTypeQuery = searchQuery.AttributiveSearch.CardTypeQueries.AddNew
(CardDocument.ID);
 SectionQuery sectionQuery = cardTypeQuery.SectionQueries.AddNew(CardDocument.MainInfo.
ID);
 sectionQuery.ConditionGroup.Operation = ConditionGroupOperation.And; ③

 string cardNameValue = cardName.ControlValue as string;
 if(!string.IsNullOrEmpty(cardNameValue)) ④
 {
  sectionQuery.ConditionGroup.Conditions.AddNew(CardDocument.MainInfo.Name, FieldType
.Unistring, ConditionOperation.Contains, cardNameValue);
 }

 Guid cardAuthorIdValue = (Guid)cardAuthorId.ControlValue;
 if(cardAuthorIdValue != Guid.Empty) ⑤
 {
  sectionQuery.ConditionGroup.Conditions.AddNew(CardDocument.MainInfo.Author, FieldType
.RefId, ConditionOperation.Equals, cardAuthorIdValue);
 }

 DateTime? cardCreateByValue = (DateTime?)cardCreateBy.ControlValue;
 if(cardCreateByValue.HasValue) ⑥
 {
```

```
  sectionQuery.ConditionGroup.Conditions.AddNew(CardDocument.MainInfo.RegDate, FieldType
.Unistring, ConditionOperation.GreaterEqual, cardCreateByValue.Value.Date);
 }

 CardDataCollection result = Session.CardManager.FindCards(searchQuery.GetXml()); ⑦
}
```

① Получение элементов управления, в которых указаны необходимые данные.

② Организация атрибутивного поиска по секции `MainInfo` карточек типа Документ.

③ Искомые карточки должны удовлетворять всем условиям.

④ Добавление условия поиска по названию документа.

⑤ Добавление условия поиска по автору документа.

⑥ Добавление условия поиска по дате создания документа.

⑦ Результат поиска, который может быть выведен, к примеру, в список на карточке.

Примеры реализации компонента карточек, библиотек карточек и расширений

Руководство разработчика Docsvision

Примеры реализации компонента карточек, библиотек карточек и расширений

В разделе представлены примеры реализации компонента карточек, библиотек карточек и расширений.

- Реализация компонента карточки с обработчиками событий
- Пример реализации командного расширения Windows-клиента
- Создание карточки учета перемещения техники
- Больше подробностей в навигационном меню слева…

Реализация компонента карточки с обработчиками событий

В данном примере, предполагается реализовать для компонента карточки обработчики событий инициализации и закрытия карточки. Предполагается, что интерфейс пользователя был или будет реализован.

```
using System;
using System.Windows.Forms;
using DocsVision.Platform.ObjectManager;
using DocsVision.Platform.WinForms;
using DocsVision.Platform.WinForms.DataSource;
using DocsVision.Platform.CardHost;

namespace DocsVision.Test
{
    public sealed partial class TestCard : CardControl
    {
        private bool isReadOnly;
        private bool isChanged;

        public TestCard() { }

        protected override void OnCardInitialized(EventArgs e) ①
        {
            base.OnCardInitialized(e);
            InitializeComponent();
        }

        protected override void OnCardActivated(CardActivatedEventArgs e) ②
        {
            base.OnCardActivated(e);

            isReadOnly = (e.ActivateMode != ActivateMode.Edit);
            isChanged = ((e.ActivateFlags & ActivateFlags.New) = ActivateFlags.New); ③
        }

        protected override void OnCardClosing(CardClosingEventArgs e) ④
        {
            base.OnCardClosing(e);

            if (!isReadOnly && isChanged) ⑤
            {
                e.ActionFlags = ActionFlags.None;
                MessageResult result = ShowMessage("Сохранить изменения?", "Карточка",
null,
                    MessageType.Question, MessageButtons.YesNoCancel);

                switch (result)
                {
                    case MessageResult.Yes:
                        e.ActionFlags = ActionFlags.ContinueAction | ActionFlags
.CommittedData;
                        break;
```

```
                    case MessageResult.No:
                        e.ActionFlags = ActionFlags.ContinueAction;
                        break;

                    default:
                        break;
                }
            }
        }
    }
}
```

① Инициируется после создания компонента карточки, и передачи ей актуальных данных..

② Инициируется после активации компонента карточки Windows-клиентом.

③ Проверка режим редактирования, либо чтения.

④ Возникает до закрытия пользовательского интерфейса карточки.

⑤ Карточка будет сохранять данные только при `ActivateMode.Edit` и наличии изменений.

Пример реализации командного расширения Windows-клиента

В проекте создаётся расширения Windows-клиента, которое добавляет на ленту инструментов новую команду, организующую перевод курсора на личную папку текущего сотрудника. Личная папка должна быть задана в *Справочнике сотрудников*.

Для реализации расширения Windows-клиента требуется:

1. Создать проект (Microsoft Visual Studio) типа "Class Library".
2. Создайте элемент управления "User Control" (Windows Forms).
3. Измените базовый класс элемента управления с UserControl на `NavExtension`.
4. Переопределить необходимые методы и свойства базового класса (см. пример).
5. Создать библиотеку карточек (доступна в /dv6/programmer/dv6/_attachments/winclientExtension.zip[архиве] проекта), включив в неё карточку, использующую созданный компонент.
6. Зарегистрировать реализованный компонент в GAC (предпочтительно) на всех клиентских машинах.

7. Зарегистрировать с использованием программы CardManager, входящей в Resource Kit, созданную библиотеку карточек на сервере Docsvision.

```
using DocsVision.Platform.Extensibility;
using DocsVision.Platform.ObjectManager;
using DocsVision.Platform.ObjectManager.Metadata;
using DocsVision.Platform.ObjectManager.SearchModel;
using DocsVision.Platform.WinForms;
using System;
using System.Collections.Generic;
using System.Runtime.InteropServices;

namespace DocsVision.Test.NavPluging
{
 [ComVisible(true)]
 [Guid("6A0676EE-1DAF-4A59-B5EB-E0B5B4C175E1")]
 [ClassInterface(ClassInterfaceType.None)]
 public partial class Plugin : NavExtension
 {
  public Plugin() { }

  protected override NavExtensionTypes SupportedTypes ①
  {
   get
   {
    return NavExtensionTypes.Command;
   }
  }

  protected override string GetExtensionName(NavExtensionTypes extensionType) ②
  {
   return "Быстрый выбор";
  }

  protected override IEnumerable<NavCommand> CreateCommands() ③
  {
   NavCommand myCommand = new NavCommand();
   myCommand.CommandType = NavCommandTypes.ToolBar; <.> // В данном случае -- команда
размещается на ленте инструментов
   myCommand.Name = "Выбрать папку";
   myCommand.Description = "Команда выбирает заданную папку";
   myCommand.Location = new NavCommandLocation()
   {
    RibbonTabLocations = new RibbonTabLocation[] ④
    {
     new RibbonTabLocation
     {
```

```
      Tab = "Доп. команды",
      Groups = new string[] { "Группы" }
     }
    }
   };
   return new List<NavCommand> { myCommand };
  }

  protected override NavCommandStatus QueryCommandStatus(NavCommand command,
NavCommandContext context) ⑤
  {
   return NavCommandStatus.Enabled | NavCommandStatus.Supported; ⑥
  }

  protected override void InvokeCommand(NavCommand command, NavCommandContext context) ⑦
  {
   string accountName = base.Session.Properties["AccountName"].Value.ToString();

   Guid Employees = new Guid("DBC8AE9D-C1D2-4D5E-978B-339D22B32482"); ⑧

   SectionQuery sectionQuery = base.Session.CreateSectionQuery();
   sectionQuery.ConditionGroup.Conditions.AddNew("AccountName", FieldType.String,
ConditionOperation.Equals, accountName); ⑨

   CardData staffData = base.Session.CardManager.GetDictionaryData(RefStaff, false);
   RowDataCollection users = staffData.Sections[Employees].FindRows(sectionQuery.
GetXml());

   if (users.Count > 0)
   {

    Guid? personalFolderID = users[0].GetGuid("PersonalFolder"); ⑩
    if (personalFolderID.HasValue)
    {

     CardFrame.CardHost.ActivateFolder(personalFolderID.Value, true); ⑪
    }
   }
  }
 }
}
```

① Тип расширения.

② Название расширения.

③ Инициализация списка команд.

④ Создание вкладки на панели инструментов, для помещения в ней новой команды.

⑤ Проверка команды на доступность.

⑥ Команда всегда доступна.

⑦ Непосредственное исполнение команды расширения.

⑧ Идентификаторы Справочника сотрудников и секции Сотрудники `Guid RefStaff = new Guid("6710B92A-E148-4363-8A6F-1AA0EB18936C")`.

⑨ Поиск в секции Сотрудники сотрудника с учетной записью `accountName` (учетная запись текущего сотрудника).

⑩ Получение личной папки.

⑪ Установка курсора на папку.

В реализуемых методах, проверка параметра `command` не выполняется, однако в реальных проектах это может понадобиться.

Готовая сборка должна иметь строгое имя, см. Создание и использование сборок со строгими именами.

Полностью проект, содержащий командное расширение и библиотеку карточек, доступен по /dv6/programmer/dv6/_attachments/winclientPlugin.zip[ссылке].

Создание карточки учета перемещения техники

В данном примере будет рассмотрен практический пример создания карточки для учета перемещений техники. В основе карточки будет использоваться стандартный компонент карточки библиотеки *Базовые объекты*.

Архив готовой библиотеки карточек, приведенной в данном примере, доступен по /dv6/programmer/dv6/_attachments/cardLibrary.zip[ссылке].

Разработка схемы данных

Прежде всего необходимо определить архитектуру будущей карточки — это будет отдельная карточка для каждой единицы техники, содержащая название и список перемещений: дата перемещения, ответственный сотрудник, подразделение-получатель и примечание.

Данный пример рассчитан на специалиста, знакомого с методикой разработки схемы карточки и библиотеки карточек. Для получения данной информации обратитесь к разделу Разработка схемы данных карточки.

1. Создайте новую библиотеку карточек с использованием утилиты *Docsvision CardManager*.

   В данном случае в параметрах библиотеки достаточно указать *Alias* и *Названия*.

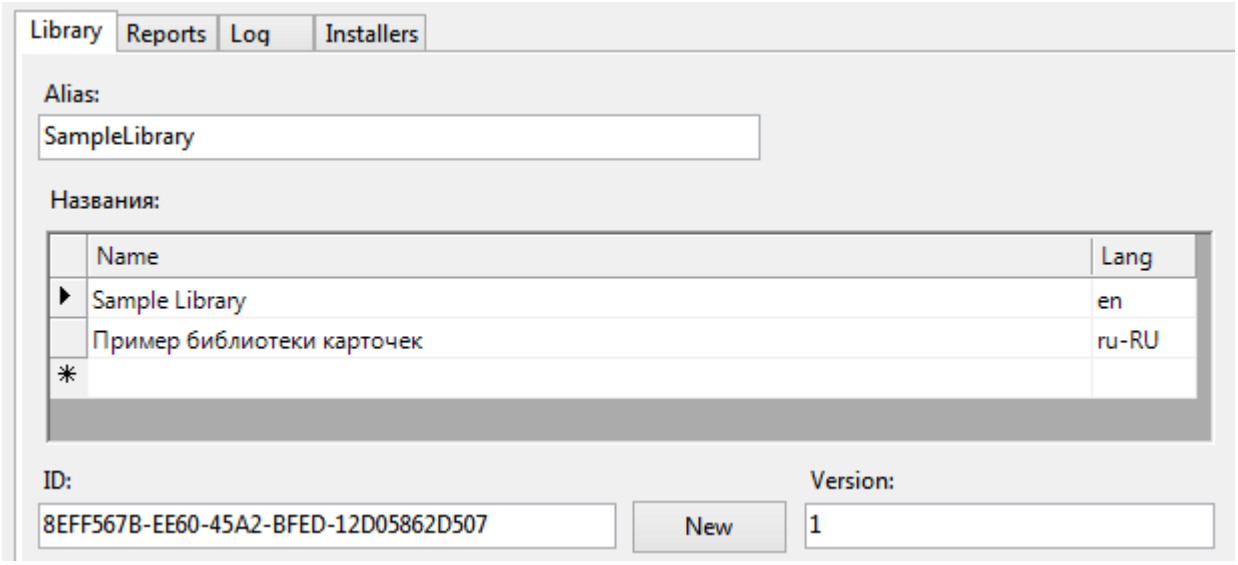


Рисунок 78. Создание библиотеку "Simple Library"

2. Создайте в данной библиотеке новую карточку.

   Для карточки требуются следующие изменения:

   - Укажите *Псевдоним* и *Названия*.
   - В поле *Строка активации* введите значение `clsid:{16f0201f-589f-465c-9721-f6a571eb9512}` — идентификатор компонента `BaseCardControl`.
   - В поле *Имя типа* укажите значение `DocsVision.BackOffice.WinForms.BaseCardControl, DocsVision.BackOffice.WinForms, Version=5.0.0.0, Culture=neutral, PublicKeyToken=7148afe997f90519` — полное название класса, являющегося основной для создаваемой карточки и карточек библиотеки *Базовые объекты*.
   - В списке *Режим чтения данных* выберите значение ***FETCH_CARD*** — данные карточки будут получены клиентом полностью.
   - Установите флаги `Card can be marked as template` и `Instances can be copied`.

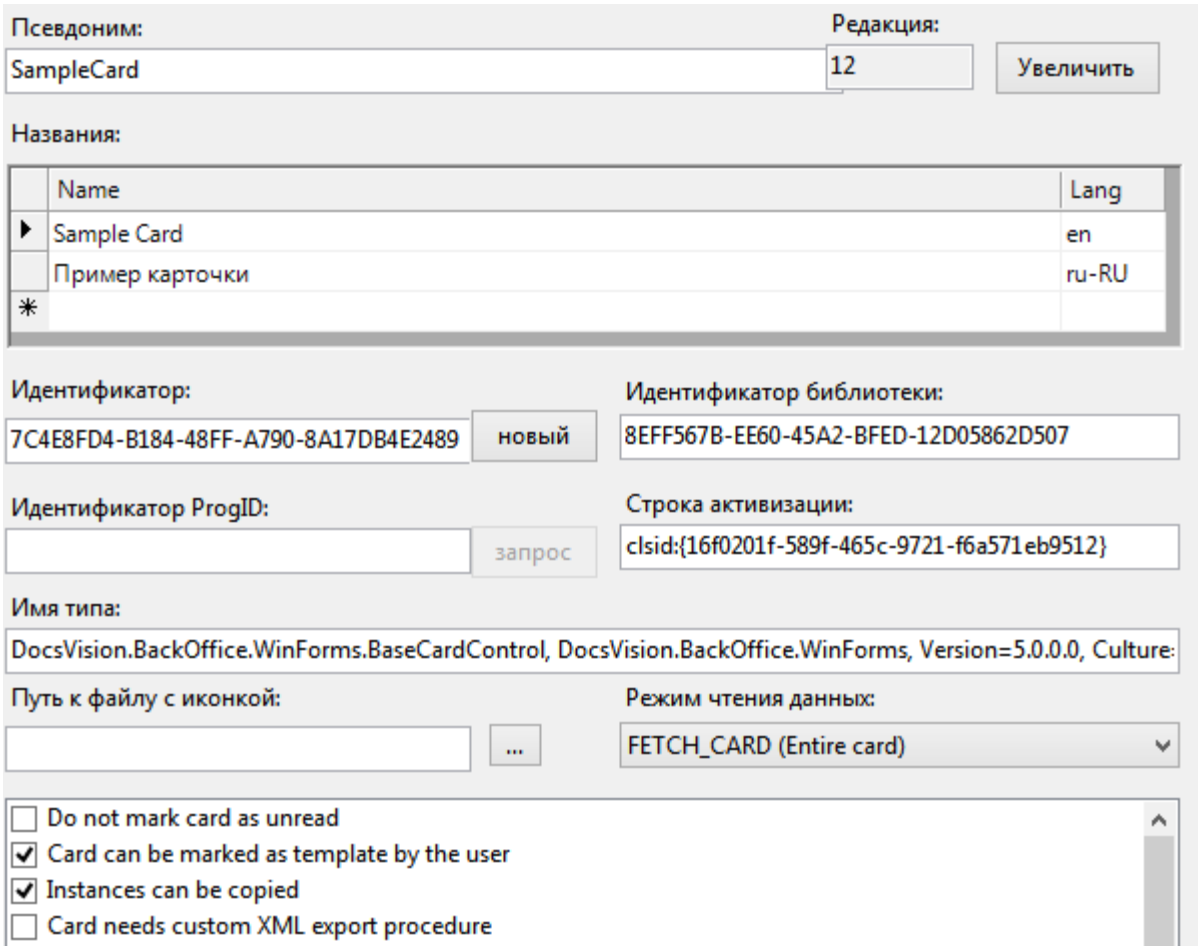


Рисунок 79. Создание карточки в новой библиотеке

3. Добавьте в карточку секцию `MainInfo` — базовая информация о единице техники.

   Для данной секции произведите следующие изменения:

   - Укажите *Названия*.
   - Выберите тип организации данных (список *Тип*) в секции `struct`.

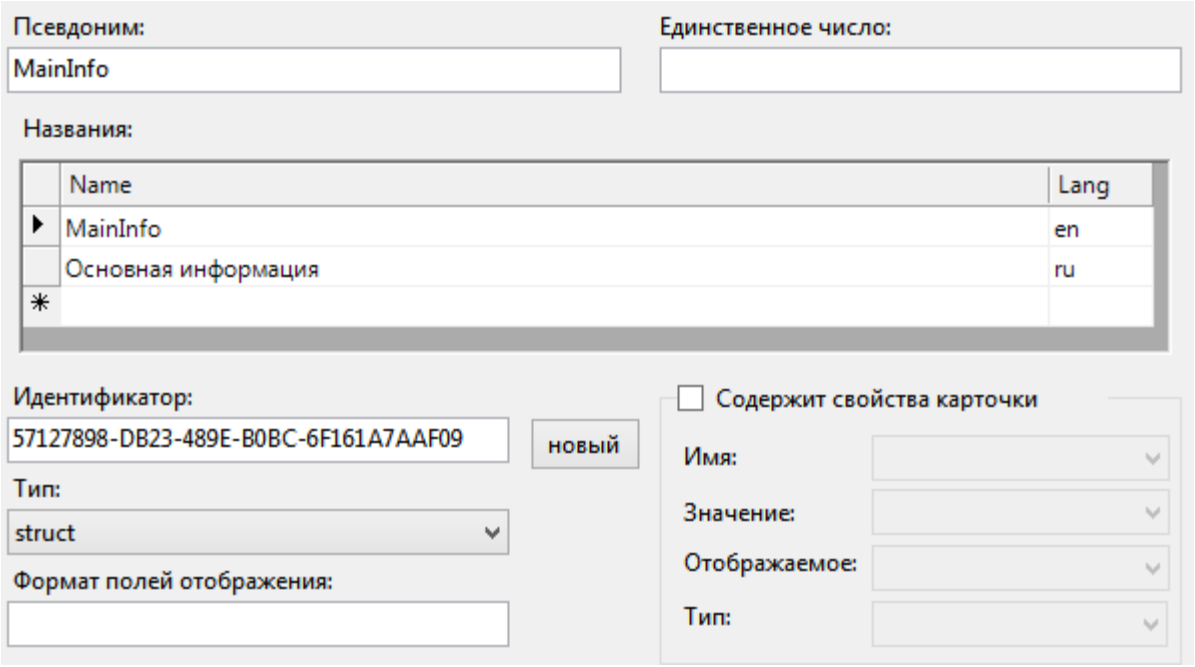


Рисунок 80. Добавление секции "MainInfo" в карточку

В данную секцию добавьте поле `Name` с типом `unitext`. Данное поле будет представлять название отдельной единицы оборудования.

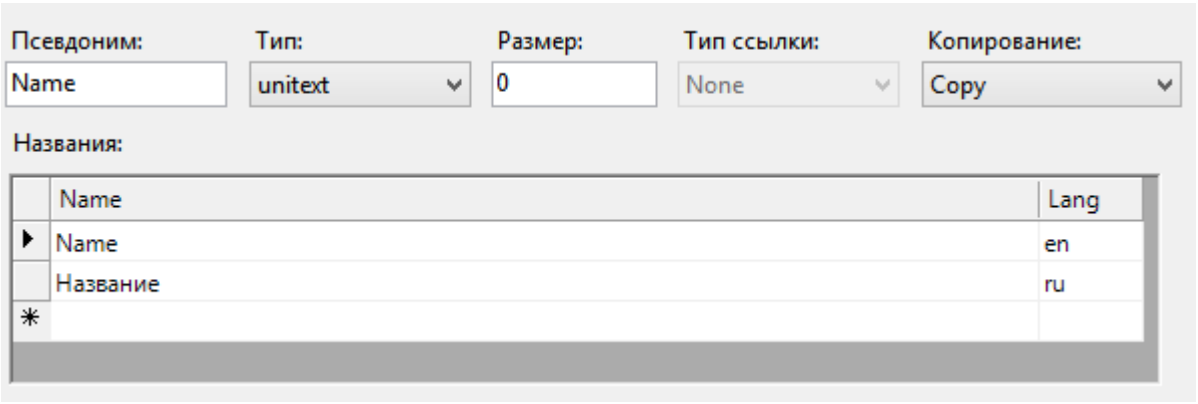


Рисунок 81. Добавление поля "Name" в секцию

4. Добавьте в карточку секцию `History` — содержит историю перемещения техники.

   Для данной секции произведите следующие изменения:

   - Укажите *Названия*.
   - Выберите тип организации данных (список *Тип*) в секции `table` (для возможности хранения нескольких элементов).

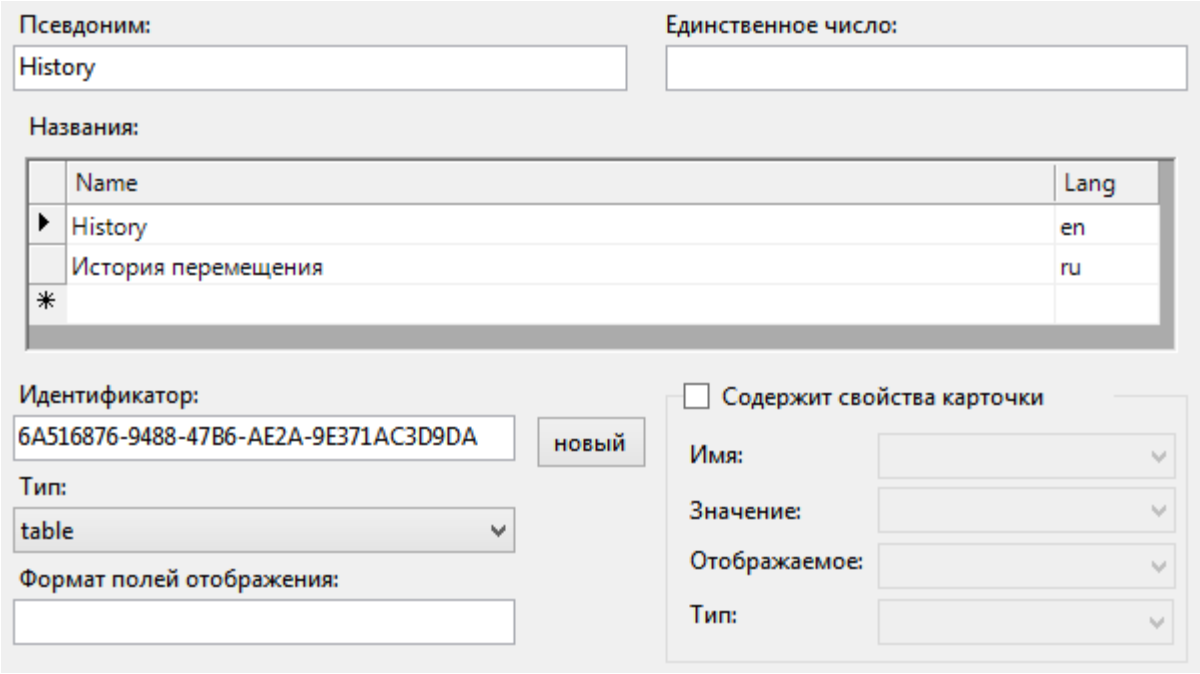


Рисунок 82. Добавление секции "History"

- `Employee` с типом `refid`, в котором содержится ссылка на сотрудника, выполнившего перемещение техники. В свойствах поля необходимо указать идентификаторы в полях *Тип, куда ссылаемся (ID)* (идентификатор *Справочника сотрудников*) и *Раздел, куда ссылаемся (ID)* (идентификатор секции *Сотрудники*). Дополнительно можно установить флаг `Обязательное`.

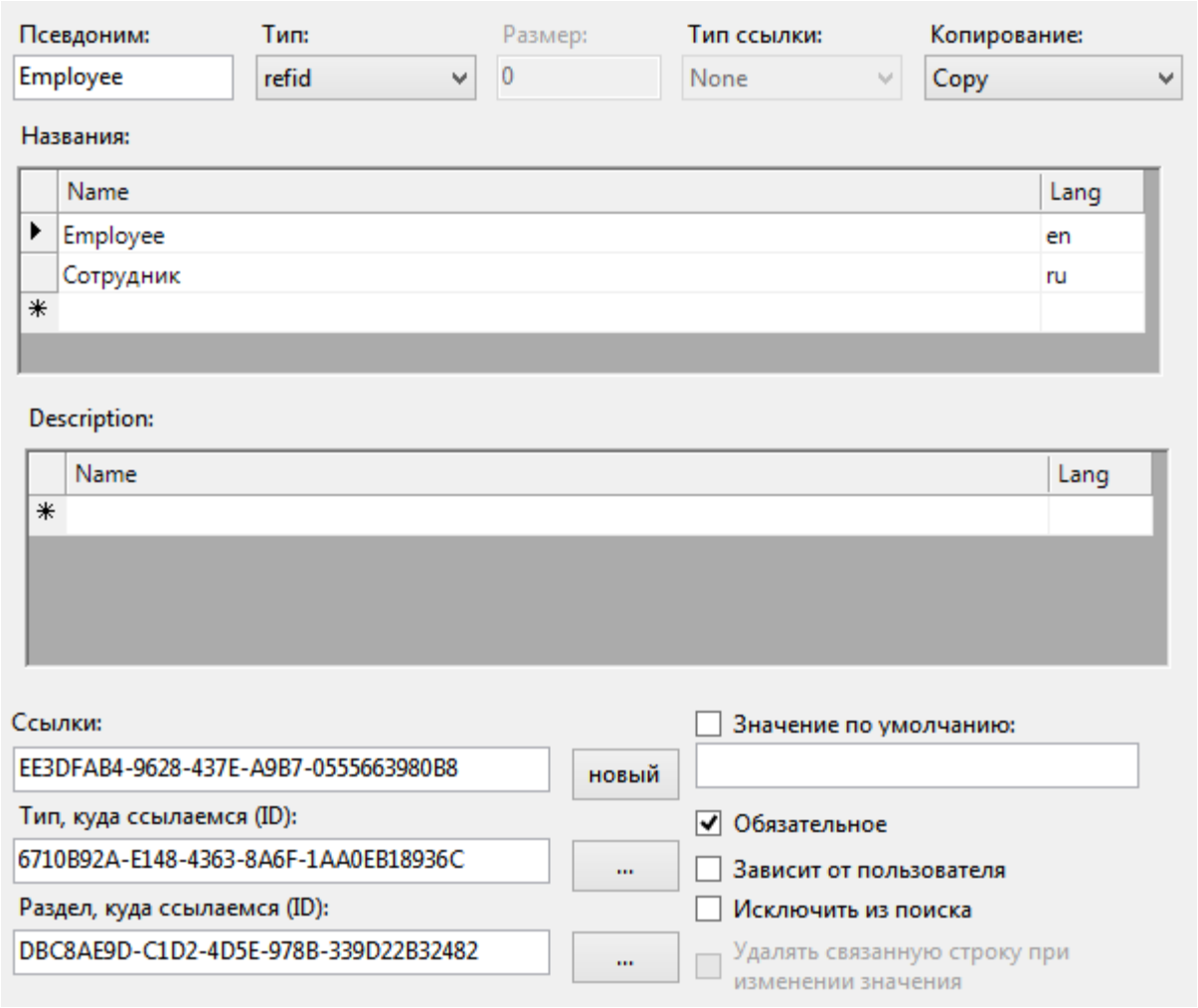


Рисунок 83. Добавление поля "Employee"

- `Date` с типом **`date`**. Данное поле содержит дату перемещения оборудования.

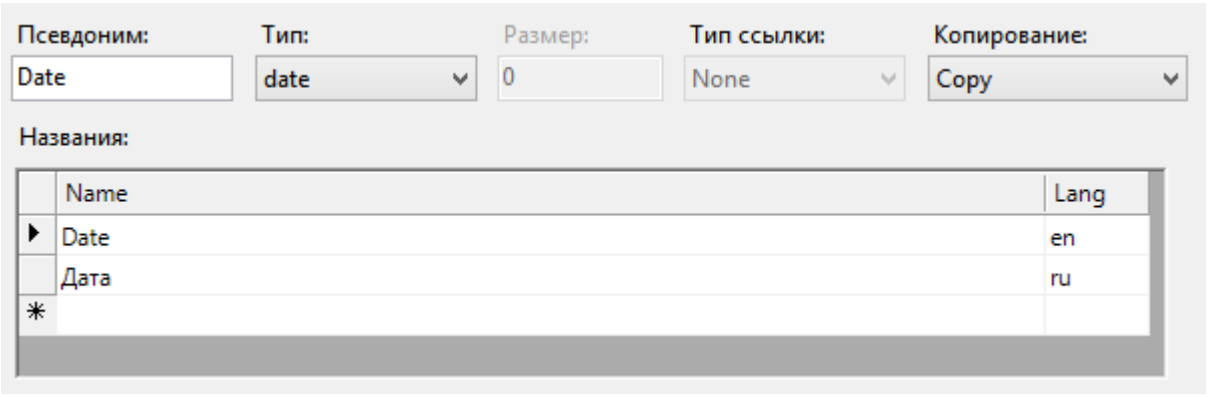


Рисунок 84. Добавление поля "Date"

- `Recipient` с типом **`refid`**, в котором содержится ссылка на организацию или подразделение. В свойствах поля необходимо указать идентификаторы в полях `Тип, куда ссылаемся (ID)` (идентификатор *Справочника сотрудников*) и `Раздел, куда ссылаемся (ID)` (идентификатор секции *Подразделения*). Дополнительно можно установить флаг **`Обязательное`**.

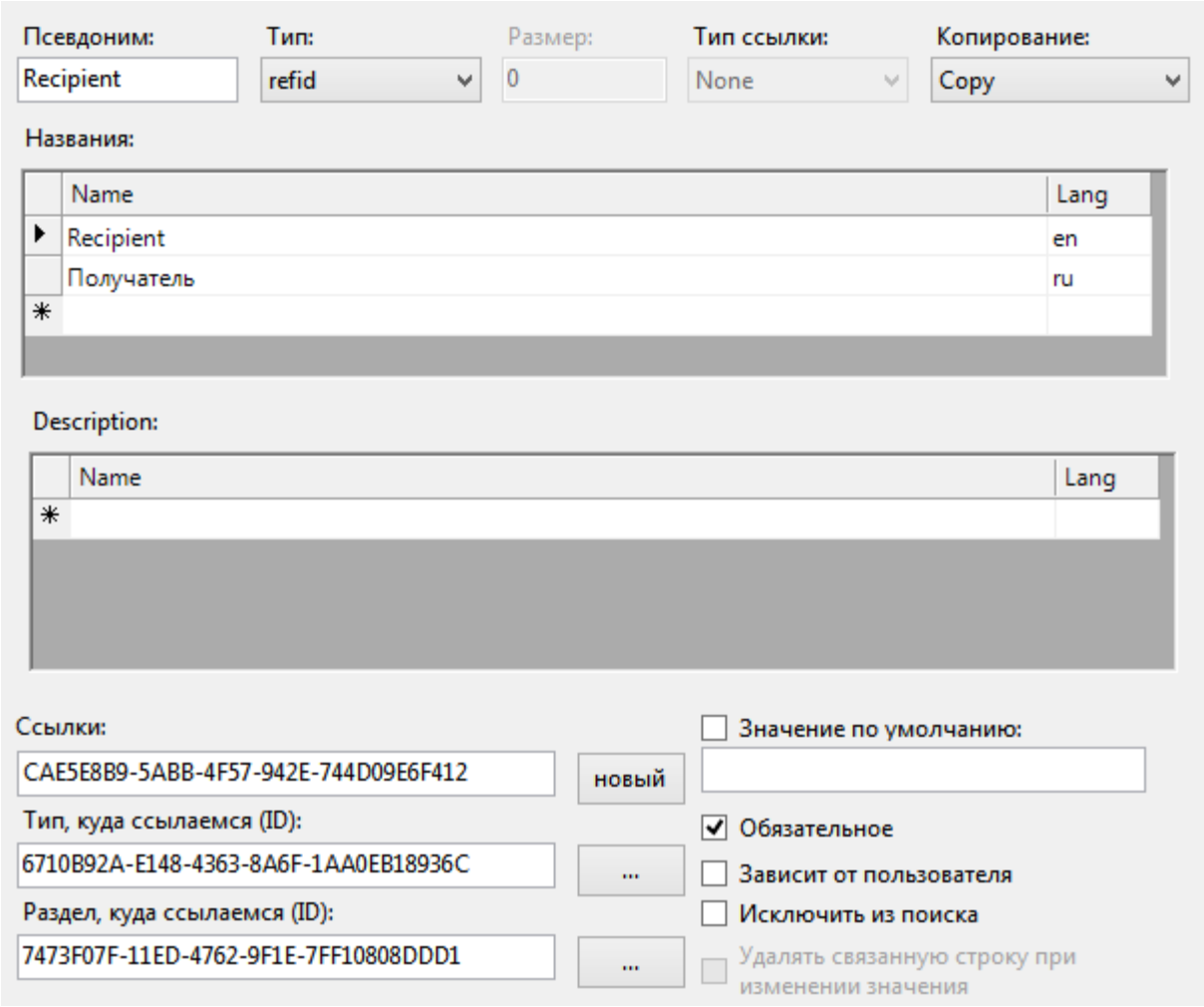


Рисунок 85. Добавление поля "Recipient"

- `Description` с типом `string`. Тестовый комментарий к операции перемещения техники.

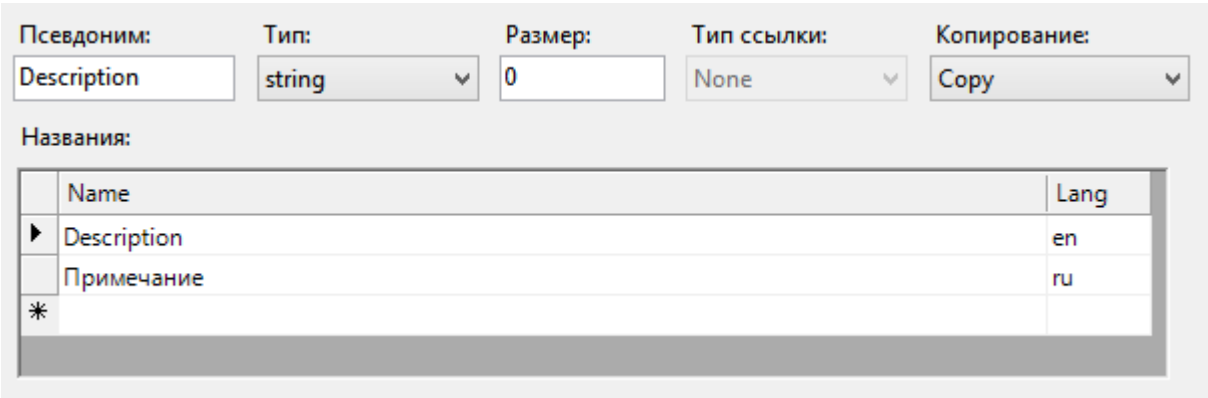


Рисунок 86. Добавление поля "Description"

5. Добавить в карточку секции `System`, `Numbers` и `Processes`. Данные секции являются типовыми и могут быть скопированы (операции *Copy* и *Paste* из контекстного меню) из схемы карточки типа *Документ* библиотеки *Базовые объекты*. Если секции создаются методом копирования, необходимо изменить

псевдоним секции на начальное значение.

После создания схемы, библиотека карточек должна быть загружена на сервер Docsvision (см. раздел Загрузка схем карточек в базу данных).

После добавления карточки потребуется перезапуск сервиса Docsvision или перезагрузка сервера

Установка параметров карточки в конструкторах

После добавления карточки в базу Docsvision необходимо произвести дополнительную настройку карточки при помощи конструкторов Docsvision.

1. В *Справочнике видов карточек* откройте созданную библиотеку и карточку и проверьте настройки на соответствие представленному ниже изображению (название карточки и библиотеки может отличаться). Сохраните изменения.

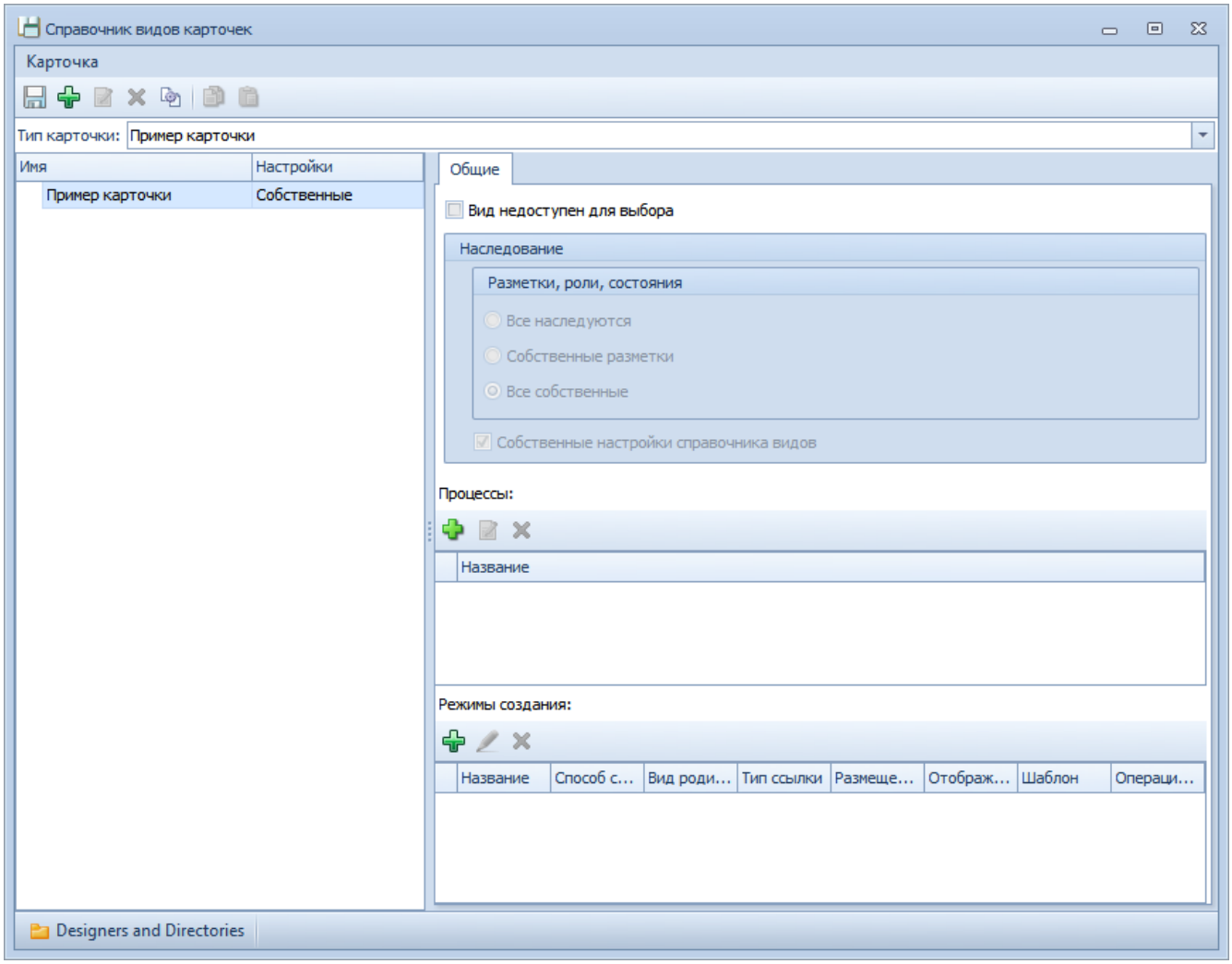


Рисунок 87. Карточка в справочнике видов карточек

2. В *Конструкторе состояний* определите минимум одно состояние, которое будет являться начальным, а также установите доступность операций редактирования.

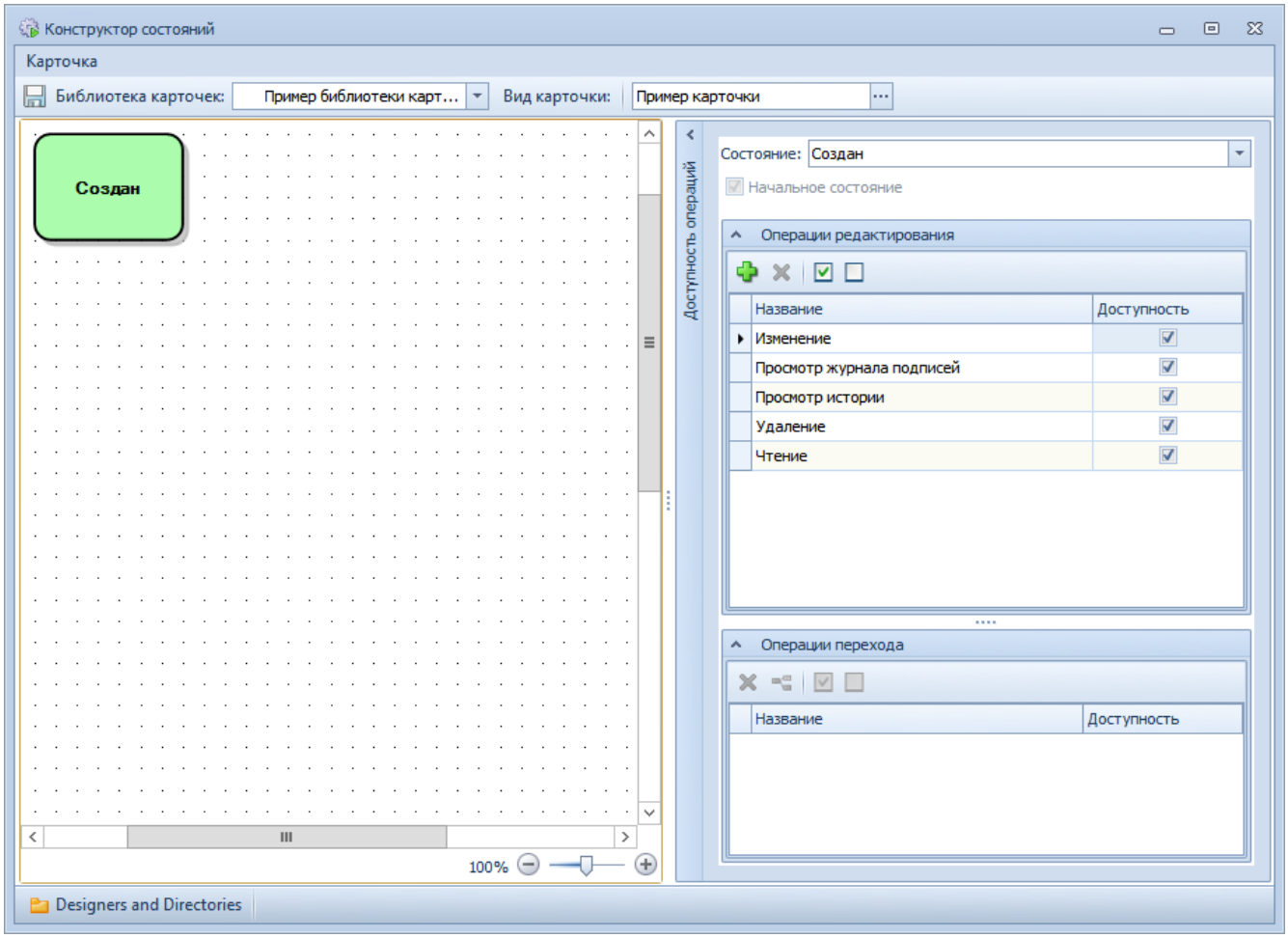


Рисунок 88. Карточка в конструкторе состояний

3. В *Конструкторе ролей* определите роль для карточки (например, Все).

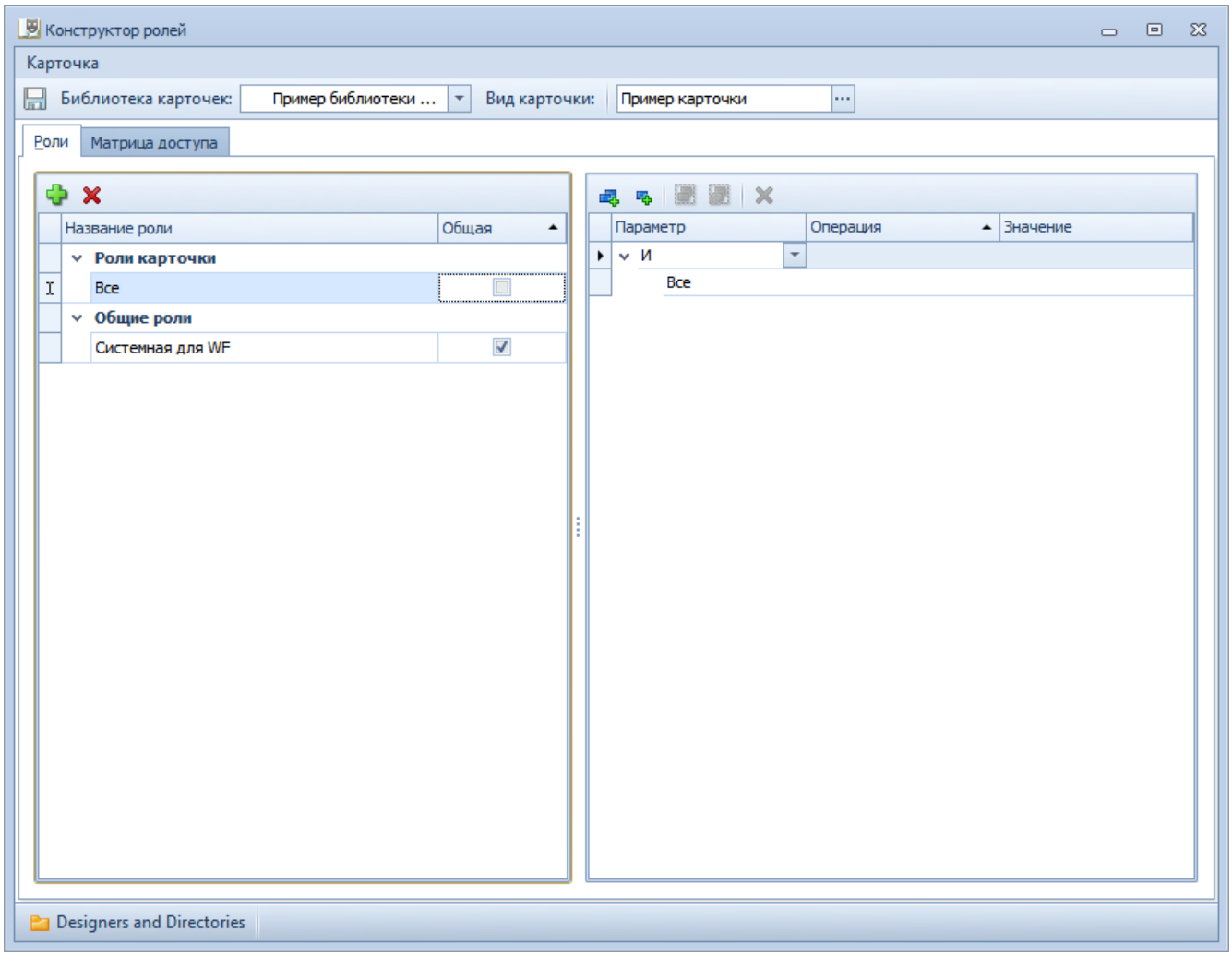


Рисунок 89. Карточка в конструкторе ролей

Установите необходимый уровень доступа для роли к операциям с карточкой в *Матрице доступа*.

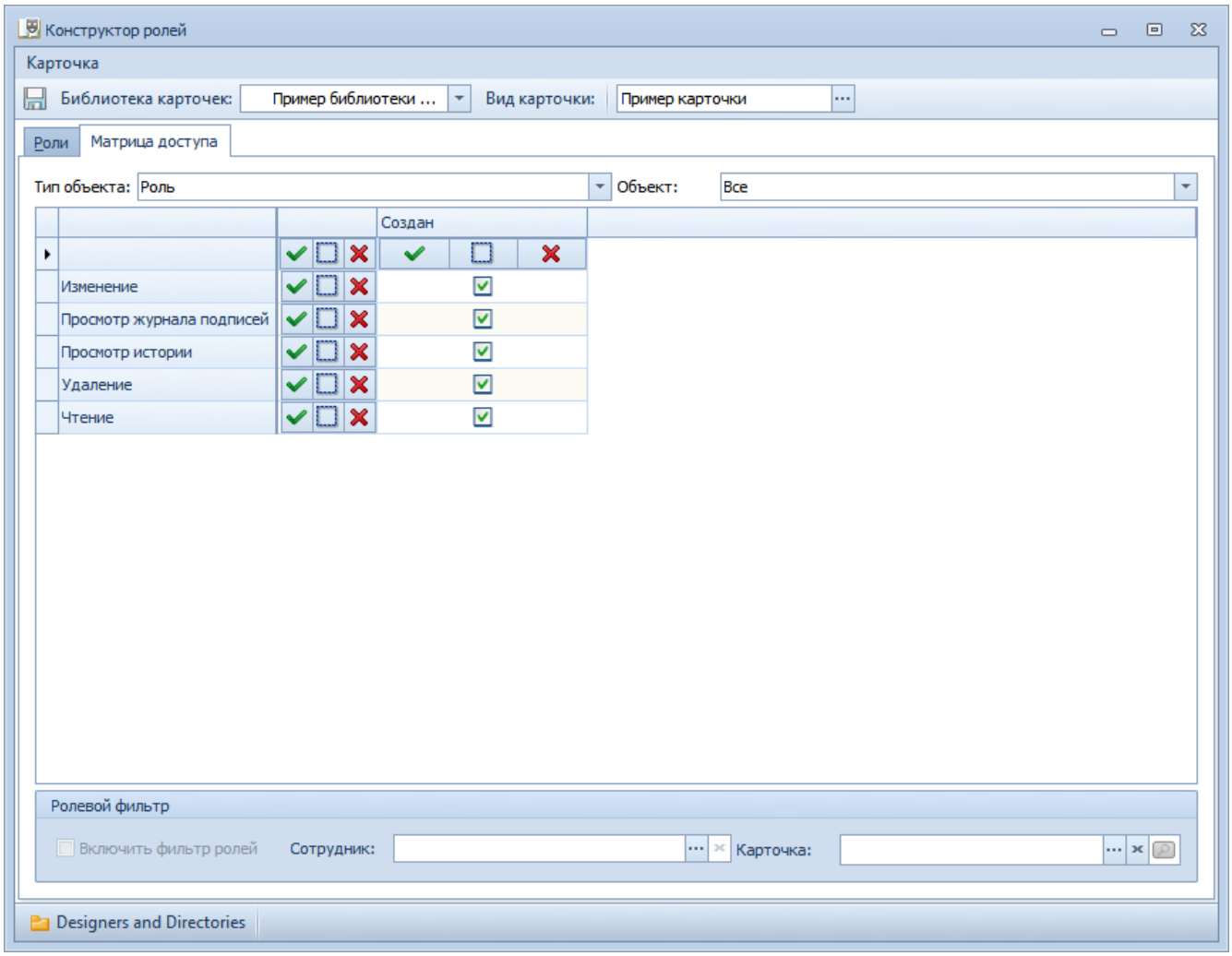


Рисунок 90. Настройки в Матрице доступа

4. В *Конструкторе разметок* требуется создать разметку, соответствующую содержимому карточки.

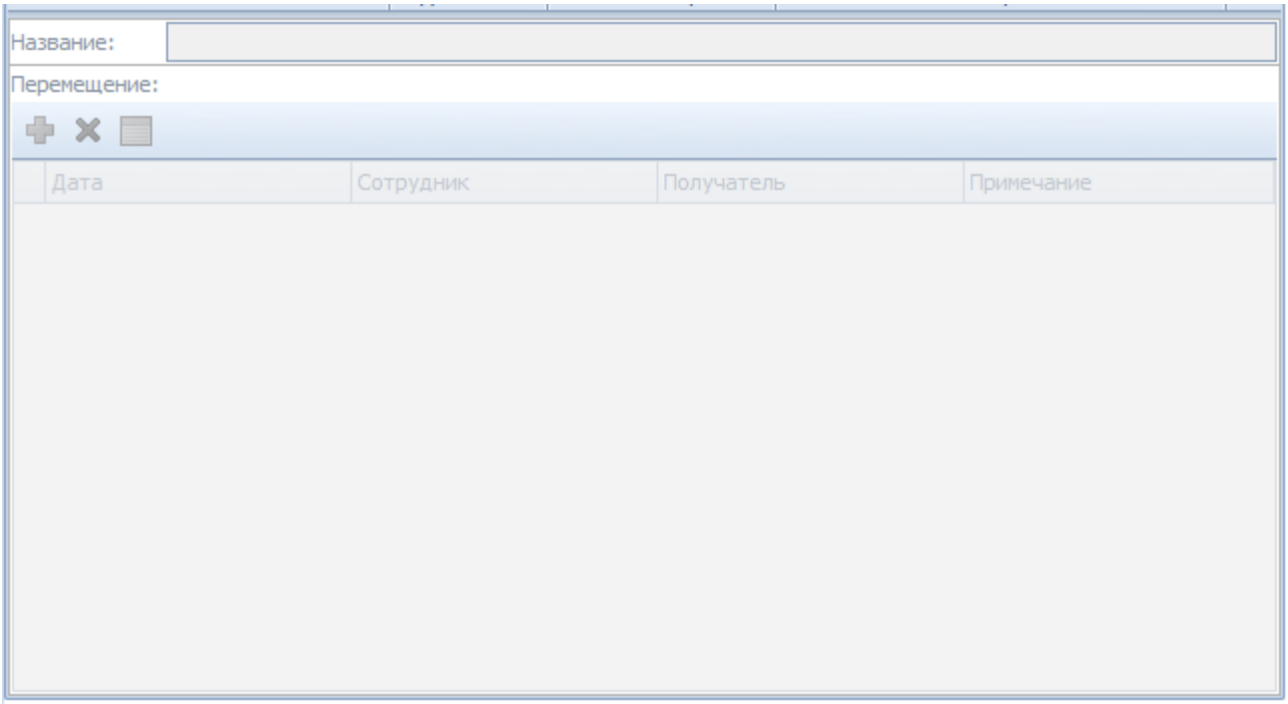


Рисунок 91. Создание разметки для карточки

Название — элемент управления типа *Строка*, *Источником данных* для которого установлена секция *Основная информация*, а *Поле данных — Название*.

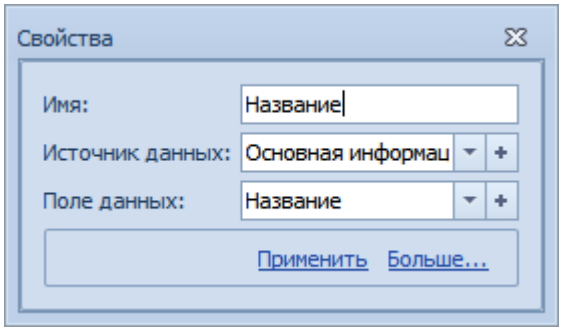


Рисунок 92. Свойства элемента управления "Название"

В качестве элемента управления Перемещение используется *Таблица*, у которой *Источником данных* установлена секция *История перемещения*.

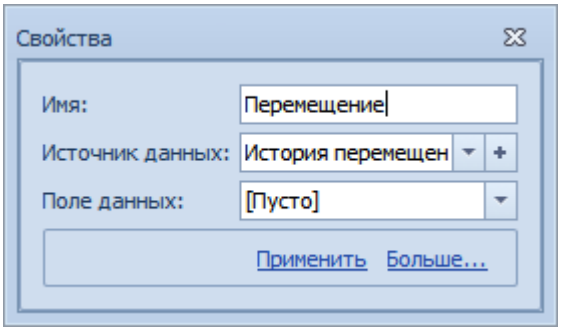


Рисунок 93. Свойства элемента управления "Перемещение"

Для данного элемента управления необходимо добавить столбцы: *Дата*, *Сотрудник*, *Получатель* и *Примечание*. В каждом столбце выбирается соответствующий элемент данных.

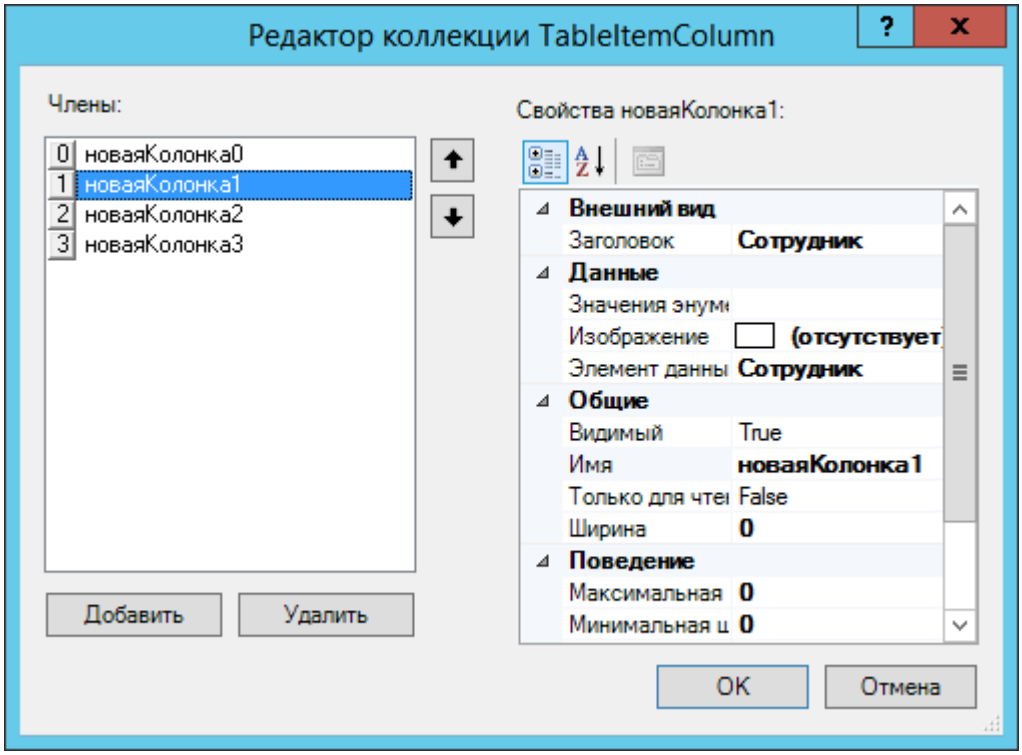


Рисунок 94. Создание столбцов для элемента управления "Перемещение"

После сохранения всех настроек появится возможность создать экземпляр новой карточки и заполнить её данными.

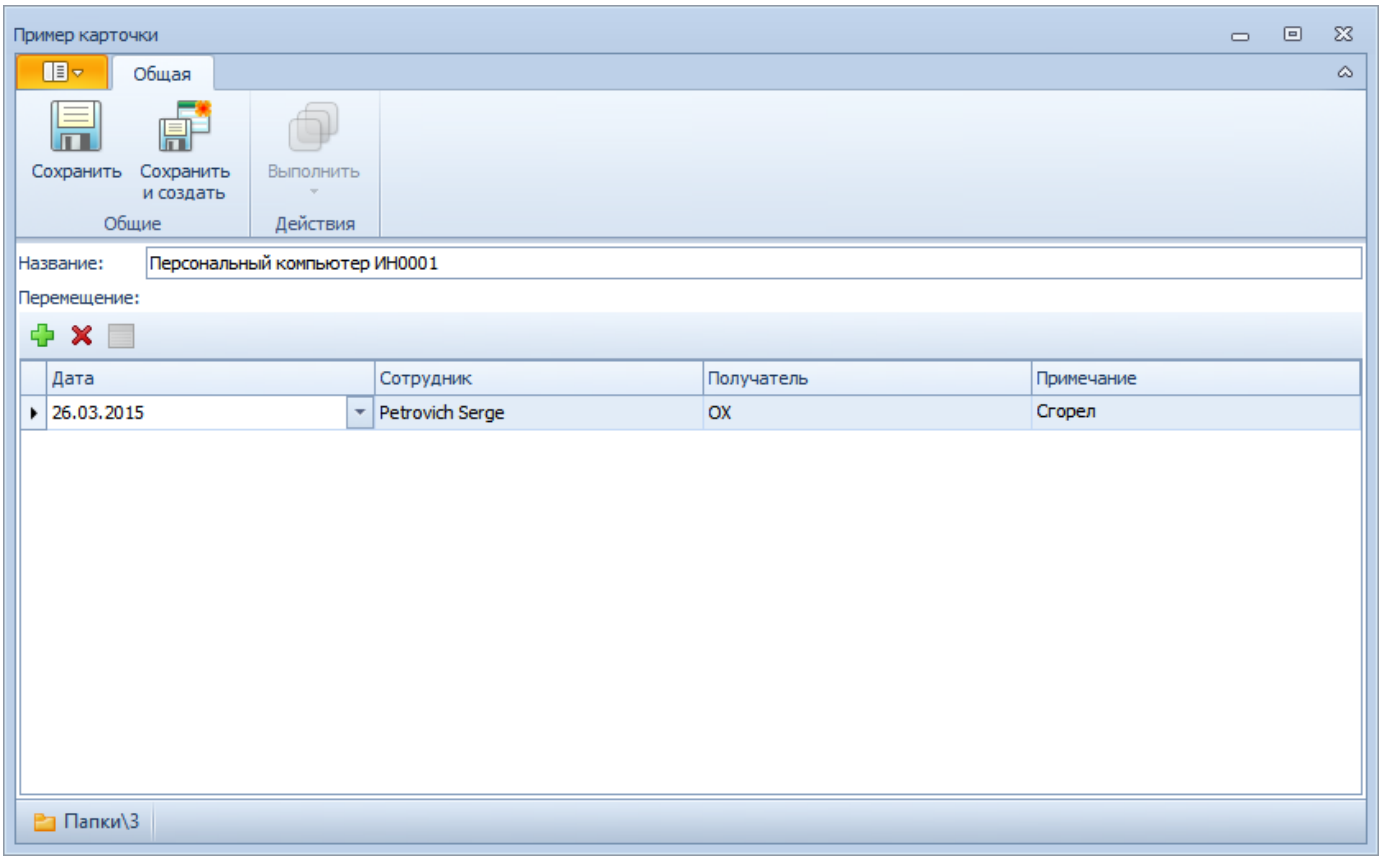


Рисунок 95. Создание карточки из Windows-клиента

Пример расширения страницы свойств Windows-клиента

В данном примере создаётся расширение Windows-клиента, добавляющее собственную вкладку на форму свойств карточки *Задание*. Новая вкладка должна предоставлять возможность получать основную информацию о задании без открытия карточки.

В процессе разработки должны быть получены два компонента — собственно компонент расширения, а также компонент, предоставляющий страницу (графический интерфейс и логику) свойств карточки, реализованные в одной сборке.

Архив с исходным кодом решения доступен по /dv6/programmer/dv6/_attachments/winclientExtension.zip[ссылке].

1. Создаем (в Visual Studio) новый проект типа `Class Library`.

   Первоначально реализуем компонент страницы свойств карточки.

2. Добавляем в проект компонент типа `User Control` (WPF), в котором будет реализован графический интерфейс страницы. Базовым классом страницы

(для WPF) является тип `DocsVision.Platform.Wpf.NavPropertyPageControl` (сборка `DocsVision.Platform.Wpf.dll`) — переопределяем `UserControl` на него. Класс должен быть помечен атрибутами COM-видимости.

```
[ComVisible(true)]
[Guid("95821EC8-34C2-4914-A4C2-125D1BF6FFDE")]
[ClassInterface(ClassInterfaceType.None)]
public partial class PropertyPageControl : NavPropertyPageControl
```

Если для реализации пользовательского интерфейса используются компоненты WinForms, а не WPF, то нужно наследовать свой класс от типа `DocsVision.Platform.WinForms`, реализованного в сборке `DocsVision.Platform.WinForms.dll`. Название типов в WPF и WinForms-версиях сборок совпадают.

3. Реализуем графический интерфейс страницы, на котором будет отображаться информация о задании, а также логику получения соответствующих данных.

   Чтобы получить информацию о карточке, можно использовать данные, переданные с параметрами активации страницы, для чего достаточно получить элемент с названием `CardData` из массива параметров `this.ActivateParams`. Также можно самостоятельно получить данные из пользовательской сессии — `this.Session`. В данном примере, реализован иной способ, в котором запрашивается объектная модель карточки. Это позволяет, в случае необходимости, задействовать сервисы объектной модели для выполнения бизнес-функций.

   Пример получения контекста объектов приведён в разделе Инициализация контекста объектов, а также в исходных кодах данного проекта.

   В данном проекте на страницу свойств задания выводится лишь часть информации о нем, но в собственном решении можно вывести больше информации, а также реализовать дополнительные методы. Можно, например, делегировать или завершать по кнопке, или открывать вложенные файлы. В качестве примера, отображаем название, приоритет, состояние, автора, дату исполнения и содержимое задания:

```
private void PostInit()
{
```

```
 TaskName.Content = BaseObject.MainInfo.Name;

 switch (BaseObject.MainInfo.Priority)
 {
  case DocsVision.BackOffice.ObjectModel.TaskPriority.High:
   TaskPriority.Content = "Высокий";
   break;
  case DocsVision.BackOffice.ObjectModel.TaskPriority.Low:
   TaskPriority.Content = "Низкий";
   break;
  default:
   TaskPriority.Content = "Нормальный";
   break;
 }

 TaskState.Content = BaseObject.SystemInfo.State.LocalizedName;
 TaskAuthor.Content = BaseObject.MainInfo.Author.DisplayName;
 TaskEndDateActual.Content = BaseObject.MainInfo.EndDate.HasValue ? BaseObject
.MainInfo.EndDate.Value.ToString("dd.MM.yyyy") : string.Empty;
 TaskContent.Content = BaseObject.MainInfo.Content;
}
```

Собственно графический интерфейс реализуется обычным образом, но т.к. базовый класс компонента был изменён, аналогичное изменение должно быть сделано в XAML:

```
<dv:NavPropertyPageControl
 ...
 xmlns:dv="http://schemas.docsvision.com/winfx/2007/xaml/presentation"
 ...
```

4. Добавляем в проект компонент типа User Control (WPF) для реализации основного класса расширения. Базовый класс необходимо изменить на тип DocsVision.Platform.Wpf.NavExtension, а также пометить его атрибутами доступности для COM:

```
[ComVisible(true)]
[Guid("4A8331EC-9E4E-4E26-91A4-8E621D39870D")]
[ClassInterface(ClassInterfaceType.None)]
public partial class Extension : NavExtension
```

При реализации на компонентах WinForms нужно наследовать (см. комментарий выше) от класса

> `DocsVision.Platform.WinForms.NavExtension`, реализованного в сборке `DocsVision.Platform.WinForms.dll`.

5. Реализуем в классе `Extension` функциональность расширения, добавляющего страницы в свойства карточки.

   Прежде всего необходимо определить функциональное назначение расширения — его тип. Расширение может быть нескольких типов (см. раздел Расширение Windows-клиента), причем возможна реализация нескольких типов в одном компоненте расширения.

   Как было сказано в описании данного расширения, оно добавляет страницу в свойства карточки, т.е. является типом `NavExtensionTypes.PropertyPages`.

   Чтобы указать тип расширения, нужно переопределить свойство `SupportedTypes`*:*

```
protected override NavExtensionTypes SupportedTypes
{
 get
 {
  return NavExtensionTypes.PropertyPages; ①
 }
}
```

   ① Если расширение реализует несколько типов, они перечисляются через `|`.

6. Реализуем специфическую функциональность типа расширения. Каждый тип расширения добавляет в базовый класс собственную функциональность (методы или свойства), список элементов которой приведён в разделе Расширение Windows-клиента. Для расширения типа `NavExtensionTypes.PropertyPages` нужно переопределить метод `CreatePropertyPages`, чтобы он возвращал описание страниц, добавляемых расширением:

```
protected override IEnumerable<NavPropertyPage> CreatePropertyPages()
{
 return new NavPropertyPage[] {  ①
  new NavPropertyPage() {
   PageType = NavPropertyPageTypes.All,
   Name = "Информация о задании",
   Clsid = typeof(PropertyPageControl).GUID
  }
 };
```

```
}
```

① Для каждой страницы определяется тип объекта, в свойствах которого она будет отображаться, заголовок страницы, а также идентификатор класса.

Существует существенное ограничение, которое не позволяет ограничить отображение страницы одним типом карточки. То есть если страница отображается в свойствах задания, она будет отображаться также в свойствах документа и любого другого типа карточки.

В данном примере это ограничение обходится за счет формирования исключения при инициализации страницы `OnPageInitialized` в её компоненте (класс `PropertyPageControl`). Если в процессе исключения происходит ошибка, то страница в свойствах не отображается.

7. Собираем проект и распространяем сборку на клиентские компьютеры. Сборка должна быть зарегистрирована как COM-компонент утилитой *regasm*.
8. Реализуем библиотеку карточек с карточкой, компонентом который является сборка, содержащая разработанный ранее код (сборка может быть подписанной и размещена в GAC, либо в каталоге приложения).

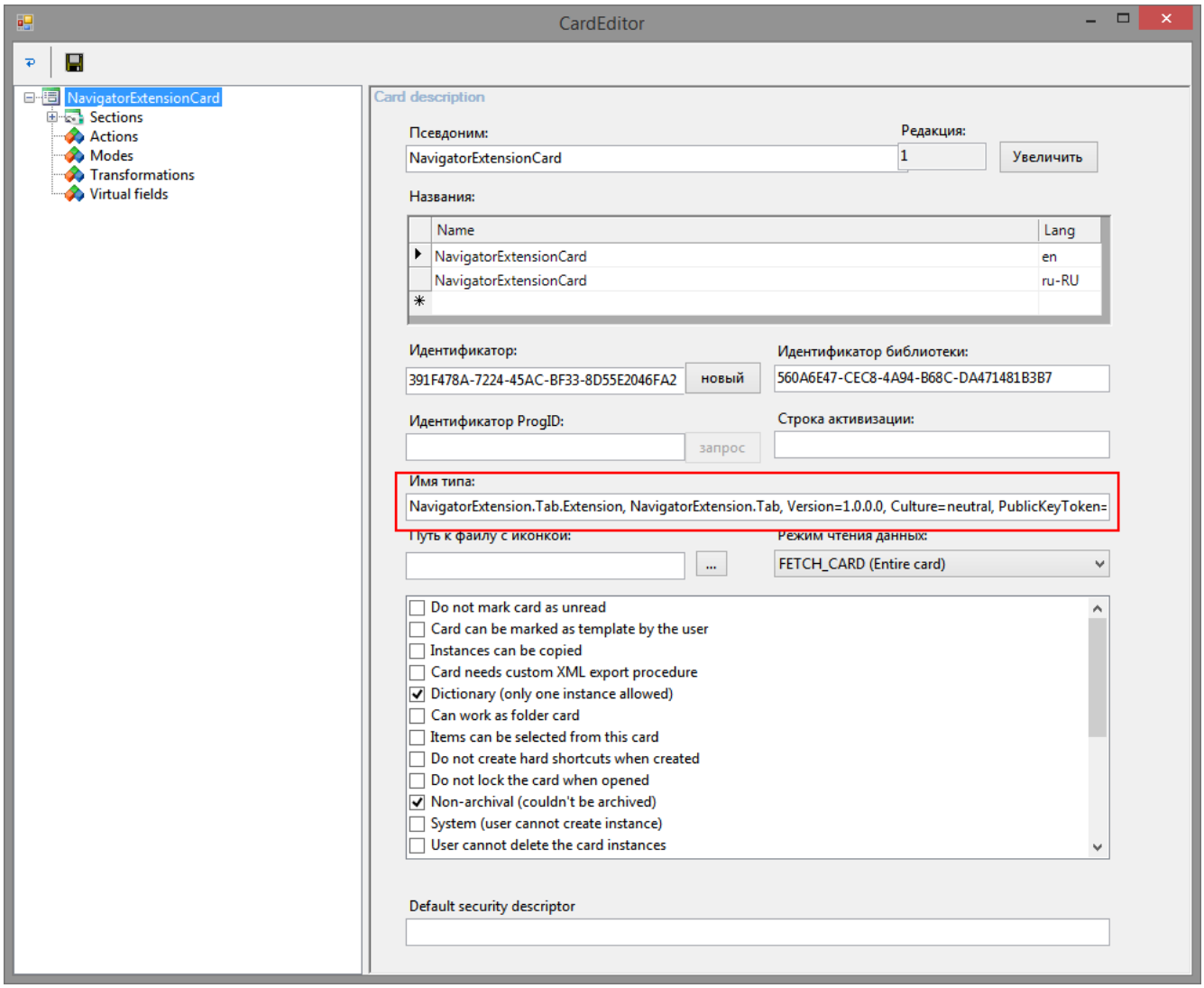


Рисунок 96. Добавление кода сборки

Условия разработки библиотеки карточек для расширения см. в разделе Расширение Windows-клиента.

9. Загружаем схему карточки в базу данных целевого севера Docsvision. После перезапуска Docsvision в свойствах карточки *Задание* появится реализованная вкладка:

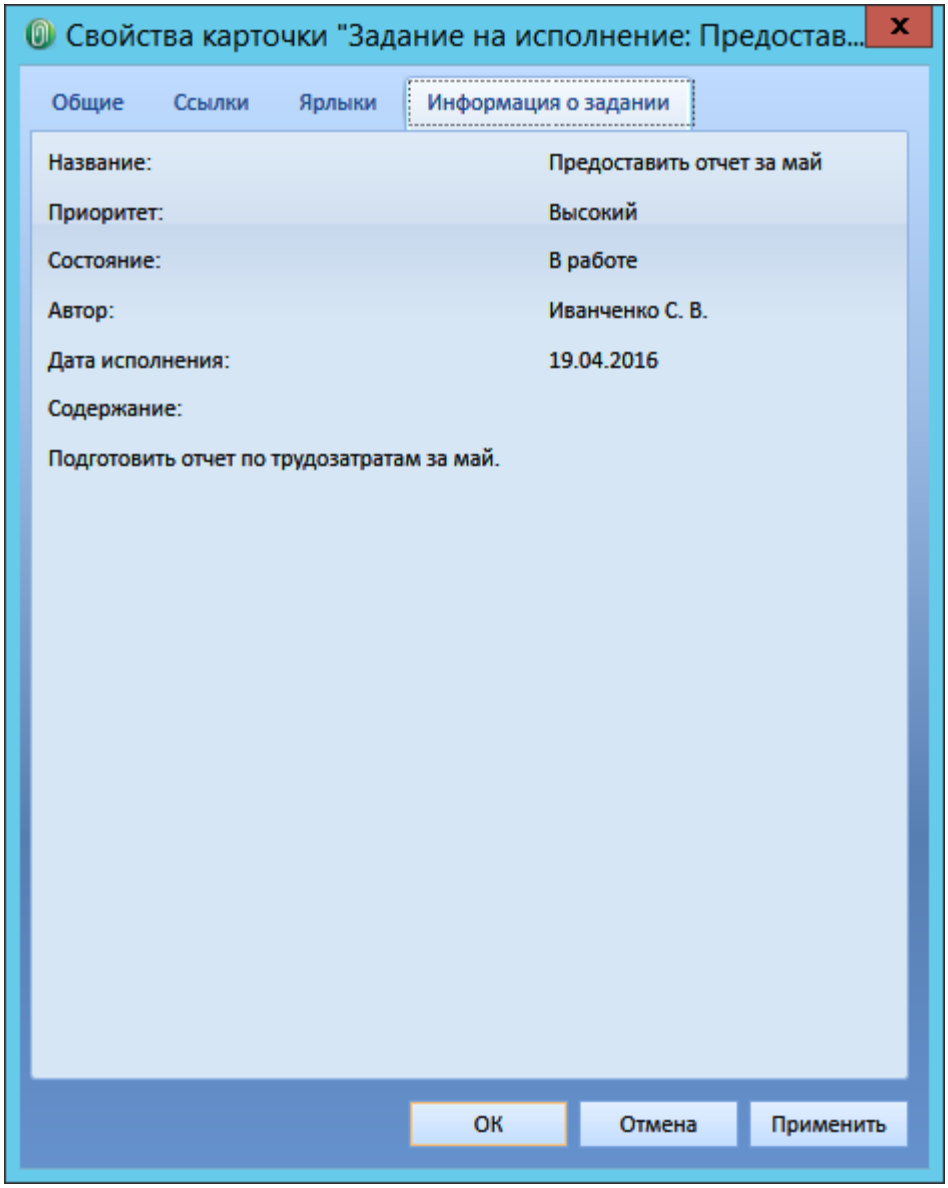


Рисунок 97. Вкладка "Информация о задании"

Примеры использования API Workflow

Руководство разработчика Docsvision

Примеры использования API Workflow

В разделе представлены примеры использования API Workflow.

- Получение ФИО пользователя
- Получение заместителя сотрудника
- Получение URL-ссылки на карточку
- Больше подробностей в навигационном меню слева…

Получение ФИО пользователя

В приведенном далее примере *сценарий бизнес-процесса* формирует строку, содержащую ФИО сотрудника, из объекта `Сотрудник DV`.

Сценарий работает со следующими переменными процесса:

- `UserDV` — сотрудник, для которого будет получена ФИО. Тип "Сотрудник DV".
- `User` — переменная будет использована для сохранения строки с ФИО. Тип "Строка".

```
using System;
using System.Xml; ①

using DocsVision.Workflow.Objects;
using DocsVision.Workflow.Runtime;
using DocsVision.Workflow.Gates;
using DocsVision.Platform.HelperAPI; ②

namespace DVScriptHost
{
 class DVScript
 {
  public void Execute(DocsVision.Workflow.Runtime.ProcessInfo process, PassState
passInfo) ③
  {
   try
   {

    ProcessVariable oEmployeeID = process.GetVariableByName("UserDV");
    ProcessVariable oUserFIO = process.GetVariableByName("User"); ④

    DVPrincipal oEmployee = oEmployeeID.Value as DVPrincipal; ⑤

    string sTempFIO = "";
    if (oEmployee.LastName.Length > 0) ⑥
    {
     sTempFIO = oEmployee.LastName;
    }
    if (oEmployee.FirstName.Length > 0)
    {
     sTempFIO += " " + oEmployee.FirstName.Substring(0, 1) + ".";
    }
    if (oEmployee.MiddleName.Length > 0)
    {
     sTempFIO += oEmployee.MiddleName.Substring(0, 1) + ".";
    }
```

```
    oUserFIO.Value = sTempFIO; ⑦
   } catch (Exception ex)
   {

    process.LogMessage("Ошибка выполнения скрипта:" + ex.Message); ⑧
   }
  }
 }
}
```

① подключение системных библиотек.

② подключение библиотек СУБП.

③ Получение ФИО пользователя DV.

④ Получение переменных процесса.

⑤ Приведение к типу Сотрудник DV.

⑥ Получение строки вида Фамилия И.О..

⑦ Сохранение найденного значения в переменную.

⑧ Добавление сообщения об ошибке в журнал процесса.

Получение заместителя сотрудника

Приведенный далее скрипт используется для получения первого активного постоянного заместителя с правом исполнения.

Сценарий работает со следующими переменными процесса:

- `UserDV` — сотрудник, для которого определяется заместитель.
- `Deputy` — переменная будет использована для сохранения заместителя.

```
using System;
using System.Xml; ①

using DocsVision.Workflow.Objects;
using DocsVision.Workflow.Runtime;
using DocsVision.Workflow.Gates;
using DocsVision.Platform.HelperAPI; ②

namespace DVScriptHost
{
 class DVScript
 {
```

```
  public void Execute(DocsVision.Workflow.Runtime.ProcessInfo process, PassState
passInfo)
  {
   try
   {

    ProcessVariable oUserDV = process.GetVariableByName("UserDV");
    ProcessVariable oDeputy = process.GetVariableByName("Deputy"); ③

    DVPrincipal dvUser = oUserDV.Value as DVPrincipal; ④

    DVPrincipal dvDeputy = dvUser.ActiveDeputy; ⑤

    oDeputy.Value = dvDeputy; ⑥
   } catch (Exception ex)
   {

    process.LogMessage("Ошибка выполнения скрипта:" + ex.Message); ⑦
   }
  }
 }
}
```

① Подключение системных библиотек.

② Подключение библиотек СУБП.

③ Получение переменных процесса.

④ Приведение к типу "Сотрудник DV".

⑤ Получение первого (по списку) активного постоянного заместителя с правом исполнения.

⑥ Сохранение найденного значения в переменную.

⑦ Добавление сообщения об ошибке в журнал процесса.

Свойство `ActiveDeputy` сущности типа `DVPrincipal` возвращает первого активного постоянного заместителя с правом исполнения, которого также можно получить вызвав метод `GetActiveDeputy(true, false,false,true)` объекта типа `DVPrincipal`. Установив значение соответствующего параметра метода `GetActiveDeputy(bool Performing, bool Control, bool Signature, bool PermanentDeputy)` в `true` можно получить заместителя, обладающего необходимым набором прав:

- `Performing` — с правом исполнения.
- `Control` — с правом ответственного исполнения.

- `Signature` — с правом подписи.
- `PermanentDeputy` — постоянный заместитель.

Если все параметры в значении `false`, будет возвращен первый активный заместитель с любыми правами.

Получение URL-ссылки на карточку

Приведенный далее скрипт позволяет получить ссылку на заданную карточку в виде URL-строки.

Сценарий работает со следующими переменными процесса:

- `Card` — карточка, для которой формируется ссылка. Тип "Карточка DV".
- `URL` — переменная будет использована для сохранения ссылки. Тип "Строка".

```
using System;
using System.Xml; ①

using DocsVision.Workflow.Objects;
using DocsVision.Workflow.Runtime;
using DocsVision.Workflow.Gates;
using DocsVision.Platform.HelperAPI; ②

namespace DVScriptHost
{
 class DVScript
 {
  public void Execute(DocsVision.Workflow.Runtime.ProcessInfo process, PassState
passInfo)
  {
   try
   {

    ProcessVariable oURL = process.GetVariableByName("URL");
    ProcessVariable oCard = process.GetVariableByName("Card"); ③

    DVCard dvCard = oCard.Value as DVCard; ④

    string strUrl = oURL.Value.ToString(); ⑤

    DVGate dvGate = process.Gates[DVGate.GateID] as DVGate;

    if (dvGate.BaseURL != null)  ⑥
    {
```

```
     strUrl += Environment.NewLine + "<" + dvGate.BaseURL + ((dvGate.BaseURL.IndexOf('?')
> 0) ? "&" : "?") + "CardID=" + dvCard.ID + ">";
    }

    oURL.Value = strUrl; ⑦
   } catch (Exception ex)
   {

    process.LogMessage("Ошибка выполнения скрипта:" + ex.Message); ⑧
   }
  }
 }
}
```

① Подключение системных библиотек.

② Подключение библиотек СУБП.

③ Получение переменных процесса.

④ Приведение переменной к типу "Карточка DV".

⑤ Приведение переменной к строковому типу.

⑥ Формирование строки URL.

⑦ Сохранение значения URL в переменную.

⑧ Добавление сообщения об ошибке в журнал процесса.

Перемещение ярлыка карточки

Приведенный далее скрипт осуществляет перемещение сильной ссылки на карточку в заданную папку.

Сценарий работает со следующими переменными процесса:

- `TargetFolder` — папка Docsvision, в которую будет перемещена ссылка. Тип "Папка DV".
- `Card` — карточка, на которую указывает сильная ссылка. Тип "Карточка DV".

```
using System;
using System.Xml; ①

using DocsVision.Workflow.Objects;
using DocsVision.Workflow.Runtime;
using DocsVision.Workflow.Gates;
using DocsVision.Platform.HelperAPI;
using DocsVision.Platform.ObjectManager.SystemCards;
```

```
using DocsVision.Platform.ObjectManager; ②

namespace DVScriptHost
{
 class DVScript
 {
  public void Execute(DocsVision.Workflow.Runtime.ProcessInfo process, PassState
passInfo)
  {
   try
   {

    ProcessVariable oTargetFolder = process.GetVariableByName("TargetFolder");
    ProcessVariable oCard = process.GetVariableByName("Card"); ③

    DVFolder dvTargetFolder = oTargetFolder.Value as DVFolder; ④

    DVCard dvCard = oCard.Value as DVCard; ⑤

    DVGate dvGate = process.Gates[DVGate.GateID] as DVGate;

    FolderCard folder = dvGate.UserSession.CardManager.GetCard(new Guid("DA86FABF-4DD7-
4A86-B6FF-C58C24D12DE2")) as FolderCard; ⑥

    Shortcut shortcut = folder.FindHardLink(new Guid(dvCard.ID)); ⑦

    if(shortcut != null)
    {

     shortcut.Move(new Guid(dvTargetFolder.ID)); ⑧
    }
   } catch (Exception ex)
   {

    process.LogMessage("Ошибка выполнения скрипта:" + ex.Message); ⑨
   }
  }
 }
}
```

① Подключение системных библиотек.

② Подключение библиотек СУБП.

③ Получение переменных процесса.

④ Приведение переменной к типу "Папка DV".

⑤ Приведение переменной к типу "Карточка DV".

⑥ Получение карточки папок.

⑦ Получение сильной ссылки.

⑧ Перемещение ярлыка.

⑨ Добавление сообщения об ошибке в журнал процесса.

Удаление унаследованных прав на карточку

Сценарий, приведенный в примере, предназначен для удаления унаследованных прав на карточку

Сценарий работает со следующими переменными процесса:

- `Card` — карточка, для которой осуществляется удаление унаследованных прав. Тип "Карточка DV".

```
using System;
using System.Security.AccessControl;
using System.Xml; ①

using DocsVision.Workflow.Objects;
using DocsVision.Workflow.Runtime;
using DocsVision.Workflow.Gates;
using DocsVision.Platform.HelperAPI;
using DocsVision.Platform.ObjectManager;
using DocsVision.Platform.Security.AccessControl; ②

namespace DVScriptHost
{
 class DVScript
 {
  public void Execute(DocsVision.Workflow.Runtime.ProcessInfo process, PassState
passInfo)
  {
   try
   {

    ProcessVariable varCard = process.GetVariableByName("Card"); ③

    DVCard oCard = varCard.Value as DVCard; ④

    DVGate dvGate = process.Gates[DVGate.GateID] as DVGate;
    CardData card = dvGate.UserSession.CardManager.GetCardData(new Guid(oCard.ID));

    CardDataSecurity cardDataSecurity = card.GetAccessControl(); ⑤
```

```
    byte[] descBytes = cardDataSecurity.GetSecurityDescriptorBinaryForm();
    CommonSecurityDescriptor desc = new CommonSecurityDescriptor(true, false, descBytes,
0); ⑥

    desc.SetDiscretionaryAclProtection(true, false);

    descBytes = new byte[desc.BinaryLength];
    desc.GetBinaryForm(descBytes, 0);
    cardDataSecurity.SetSecurityDescriptorBinaryForm(descBytes); ⑦

    card.SetAccessControl(cardDataSecurity); ⑧

   } catch (Exception ex)
   {

    process.LogMessage("Ошибка выполнения скрипта:" + ex.Message); ⑨
   }
  }
 }
}
```

① Подключение системных библиотек.

② Подключение библиотек СУБП.

③ Получение переменных процесса.

④ Приведение переменной к типу "Карточка DV".

⑤ Получение ACL для карточки.

⑥ Получение дескриптора безопасности.

⑦ Отключение наследования прав.

⑧ Сохранение дескриптора.

⑨ Обновление ACL для карточки.

Изменения состояния бизнес-процесса из сценария

Скрипт, приведенный в примере, осуществляет запуск существующего экземпляра *бизнес-процесса*.

Сценарий работает со следующими переменными процесса:

- `ProcessID` — идентификатор экземпляра *бизнес-процесса*. Тип "Строка".

```
using System;
using System.Xml; ①
```

```
using DocsVision.Workflow.Objects;
using DocsVision.Workflow.Runtime;
using DocsVision.Workflow.Gates;
using DocsVision.Platform.HelperAPI; ②

namespace DVScriptHost
{
 class DVScript
 {
  public void Execute(DocsVision.Workflow.Runtime.ProcessInfo process, PassState
passInfo)
  {
   try
   {

    ProcessVariable varProcessID = process.GetVariableByName("ProcessID"); ③

    string processID = (string) varProcessID.Value; ④

    Process processObject = process.Library.GetProcess(new Guid(processID)); ⑤

    processObject.Start(string.Empty, process.Library.Dictionary, ExecutionModeEnum
.Automatic); ⑥
   } catch (Exception ex)
   {

    process.LogMessage("Ошибка выполнения скрипта:" + ex.Message); ⑦
   }
  }
 }
}
```

① Подключение системных библиотек.

② Подключение библиотек СУБП.

③ Получение переменных процесса.

④ Приведение к строковому типу.

⑤ Получение бизнес-процесса.

⑥ Запуск процесса.

⑦ Добавление сообщения об ошибке в журнал процесса.

Использование COM API ObjectManager

Ниже приведён пример запуска экземпляра бизнес процесса из кода Visual Basic с использованием используя COM API ObjectManager.

```
Public Sub StartProcess(ByVal oUserSession As DVObjectManager.UserSession, ByVal
sProcessID As String)
 Dim oWorkflowManager As DVObjectManager.IWorkflowManager2
 Set oWorkflowManager = oUserSession.WorkflowManager ①

 Dim oProcessInfo As DVObjectManager.ProcessInfo
 Set oProcessInfo = oWorkflowManager.GetProcess(sProcessID) ②

 oProcessInfo.Start ③
End Sub
```

① Получаем менеджер бизнес-процессов.

② Получаем объект для управления бизнес-процессом.

③ Запускаем бизнес-процесс.

Подписка на разблокировку карточки

В процессе исполнения бизнес-процесса возникают ситуации, когда процессу необходимо получить доступ к карточке, у которой установлена блокировка, но принудительная разблокировка является нежелательной. В таком случае можно воспользоваться механизмом подписки, который предлагает шлюз к Docsvision. Далее приведён пример оформления такой подписки.

```
using System;
using System.Xml;
using DocsVision.Workflow.Objects;
using DocsVision.Workflow.Runtime;
using DocsVision.Workflow.Gates;
using DocsVision.Platform.HelperAPI;
using DocsVision.Workflow.Functions;

namespace DVScriptHost
{
 class DVScript
 {
  public ExecResultEnum Execute (ProcessInfo process, PassState passInfo)
  {
   try
   {
```

```
    DVGate dvGate = (DVGate)process.Gates[DVGate.GateID]; ①

    ProcessVariable cardVar = process.GetVariableByName("Карточка"); ②

    Guid cardId = new Guid(((DVCard)cardVar.Value).ID); ③

    var cardData = dvGate.UserSession.CardManager.GetCardData(cardId); ④

    if ((int)cardData.LockStatus != 0) ⑤
    {
     process.LogMessage("Ожидание разблокировки карточки");

     dvGate.SubscriptionChannel.Subscribe((int)DocsVisionSubscriptionType.ObjectUnlock,
process.ID, passInfo.FunctionID, cardId); ⑥

     return ExecResultEnum.WaitForMessage; ⑦
    } else
    {
     process.LogMessage("Карточка разблокирована.");

     return ExecResultEnum.Done; ⑧
    }
   } catch (Exception ex)
   {
    process.LogMessage("Ошибка выполнения скрипта:" + ex.Message);

    return ExecResultEnum.Error; ⑨
   }
  }
 }
}
```

① Получение шлюза к Docsvision.

② Получение карточки из переменной процесса.

③ Получение идентификатора карточки.

④ Получение данных карточки.

⑤ Выполнение проверки наличия блокировки карточки. Если карточка является заблокированной, то будет выполнена подписка на событие разблокировки шлюза.

⑥ Выполнение подписки.

⑦ Ожидание наступления события.

⑧ Возвращает признак успешного выполнения функции.

⑨ Возвращение ошибки.

Подписка осуществляется путем передачи в метод `Subscribe`:

- Типа подписки — `DocsVisionSubscriptionType` (`DocsVisionSubscriptionType.ObjectUnlock`).
- Идентификатора бизнес-процесса (`process.ID`).
- Идентификатора текущей функции (`passInfo.FunctionID`).
- Идентификатора карточки (`cardId`), на события которой осуществляется подписка.

Пример разработки решения

В данном практическом примере содержится типовой сценарий разработки решения.

Архив с исходным кодом решения доступен по /dv6/programmer/dv6/_attachments/netstatSolution.zip[ссылке].

Для работы с проектом "Решения" необходимы:

1. Visual Studio 2013 или выше. **Для сборки проекта библиотеки карточек требуется Visual Studio 2013**.
2. WiX Toolset — для сборки пакетов установки. Последняя версия доступна на странице проекта.
3. Установленные клиентские компоненты базовых модулей Docsvision (обязательно Платформа, Windows-клиент), совместимые с платформой Docsvision версии 5.5.2.

Основные сведения

Пример, приведенный в данном разделе, содержит полный сценарий создания собственного решения (на платформе Docsvision) для учета сетевого оборудования организации и сбора статистики его отказов, а также исходные коды всего проекта.

В процессе его создания будут рассмотрены вопросы, связанные с разработкой библиотеки карточек, пакетов установки, а также необходимых модулей Docsvision. Вопросы, связанные с локализацией решений, а также созданием пакетов установки для различных платформ, здесь не рассматриваются.

Для данного решения будут созданы:

- Библиотека карточек и её компонент.
- Карточка **учета сетевого оборудования** и её компонент.
- Модуль расширения *Консоли настройки* (далее — `SnapIn`).
- Два пакета установки: клиентская и серверная части.

Помимо этого разработаем объектную модель карточки. Также введём дополнительное требование на совместимость карточки с конструкторами Docsvision, чтобы иметь возможность настраивать разметку карточки при помощи *конструктора разметок*.

Чтобы решение было законченным, также потребуются:

- Бизнес-процесс *Проверка доступности сетевых узлов*, который должен фиксировать факт изменения доступности сетевого устройства.
  Непосредственная реализация логики проверки не предусмотрена.
- Выгруженные настройки *справочника видов карточек*, *конструктора разметок*, *конструктора состояний* и другие.
- В качестве примера, экземпляр карточки *учета сетевого оборудования*, который предполагается загружаться в Docsvision при установке решения. В собственном решении подобным образом может быть загружен шаблон карточки.

Данное руководство сопровождается исходными кодами всего решения, в том числе пакетов установки, а также схемами библиотеки карточек и карточки.

Архив решения имеет следующую структуру:

- `bin` — каталог с основными сборками Docsvision, использованными в проекте компонента карточки.

Все сборки Docsvision, использованные в проектах решения, устанавливаются вместе с Docsvision Windows-клиент и платформой Docsvision.

- `Binaries` — каталог для размещения готовых сборок.
- `CardDefs` — каталог со схемами карточки *учета сетевого оборудования* и библиотеки *Модуля учета сетевых устройств*, а также их ресурсами.
- `CardLib` — каталог с проектом (C++) компонента библиотеки карточек.

- `CardPackage` — каталог с данными справочников и конструкторов, выгруженных из базы данных Docsvision. Используются на этапе Формирование "CardPackage" и "SqlPackage".
- `Database` — каталог с файлом, определяющим путь к файлу с метаданными библиотеки карточек. Используются на этапе Формирование "CardPackage" и "SqlPackage".
- `Installers` — каталог сборки пакетов установки.
- `NetstatSolution.Snapin` — каталог с проектом модуля расширения *Консоли настройки* Docsvision.
- `NetstatSolutionCard` — каталог с проектом компонента карточки, а также его объектной моделью.
- `NetstatSolutionClientInstaller` — каталог с проектом пакета установки клиентской части решения.
- `NetstatSolutionInstaller` — каталог с проектом пакета установки серверной части решения.

Идентификаторы некоторых сущностей (к примеру, идентификатор библиотеки карточек) используются в нескольких компонентах, что будет обозначено дополнительно в тексте. Таким идентификаторам будет присвоено уникальное название.

Разработка библиотеки карточек

Информация, приведенная в данном и последующих разделах, рассчитана на то, что разработчик знаком с принципами разработки новых библиотек карточек (см. раздел Разработка карточек и библиотек карточек).

Разработка библиотеки карточек включает следующие шаги:

1. Получение файла с метаданными библиотеки карточек `NetstatSolution`.
2. Получение файла с метаданными карточки *учета сетевого оборудования*.
3. Разработка компонента карточки *учета сетевого оборудования*.
4. Разработка компонента библиотеки карточек.

Помимо этого разработаем объектную модель карточки и сервис для работы с ней.

В данном разделе

- Создание схемы метаданных библиотеки карточки
- Создание схемы метаданных карточки
- Разработка объектной модели карточки
- Разработка сервиса для работы с карточкой
- Разработка компонента карточки
- Разработка компонента библиотеки карточек

Создание схемы метаданных библиотеки карточки

Готовая схема библиотеки карточек (файл `NetstatSolutionCardLib.xml`), а также используемые ресурсы расположены в подкаталоге /dv6/programmer/dv6/_attachments/netstatSolution.zip[архива] `CardDefs`.

Подробный сценарий создания схемы библиотеки карточек приведён в разделе Подготовка библиотеки карточек. Здесь отмечены только основные моменты.

1. Кодировка SQL-файла разрабатываемой карточки должна быть совместима с Unicode. В качестве эталона рекомендуется использовать `UTF-8 BOM`.
2. Если в SQL-файлах предназначенных для MsSql используются разделители как в примере ниже, в соответствующем файле должны быть переносы строк вида `CR LF`:

```
--Delimiter
GO
```

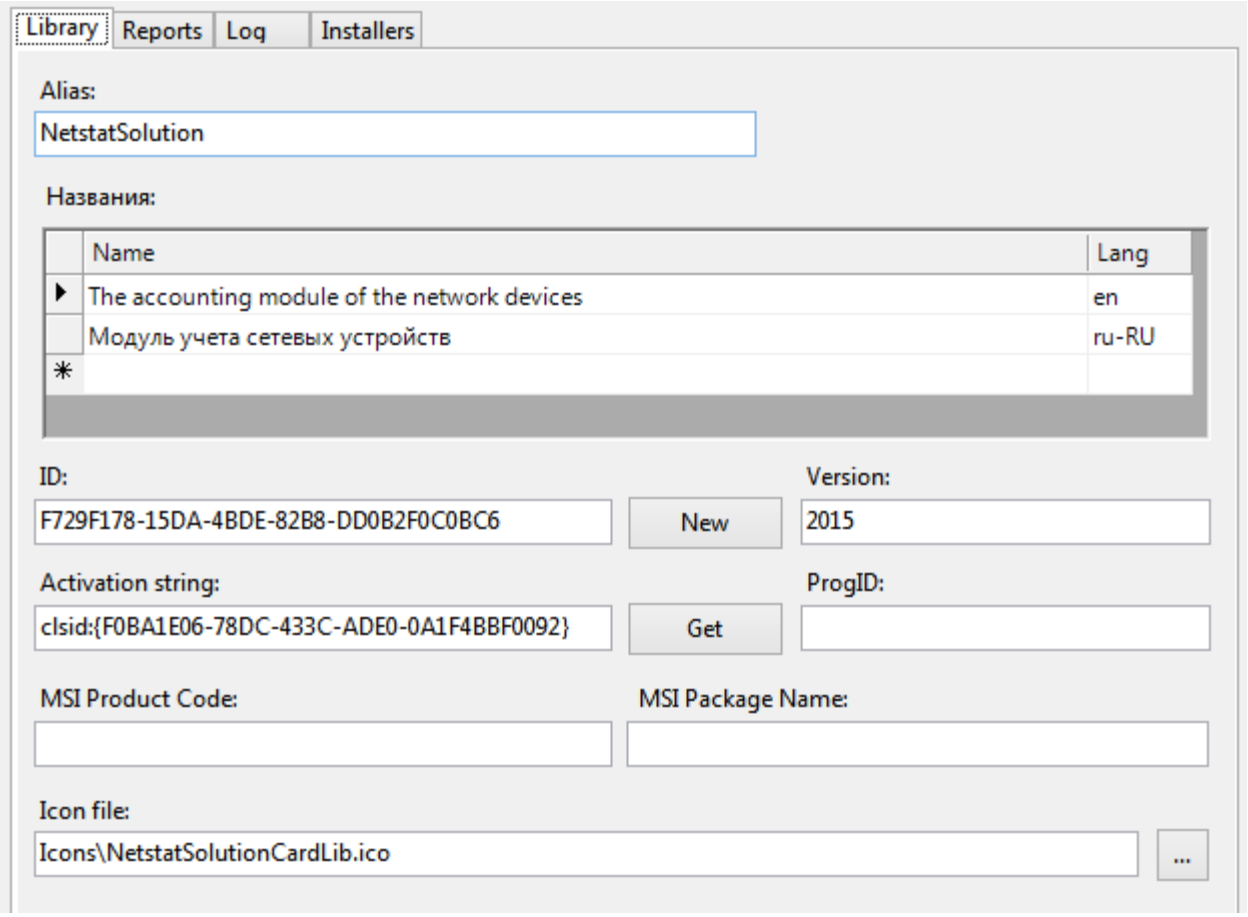


Рисунок 98. Заполнение вкладки "Library"

На вкладке "Library" заполняем:

- *ID* — идентификатор библиотеки карточек. Помимо схемы метаданных, задействован при разработке компонента `SnapIn`.
- *Activation string* — строка активации, которая включает идентификатор компонента библиотеки карточек. Данный идентификатор также задействован при разработке компонента библиотеки карточек.
- *Version* — версия библиотеки карточек. Также используется в компоненте библиотеки.

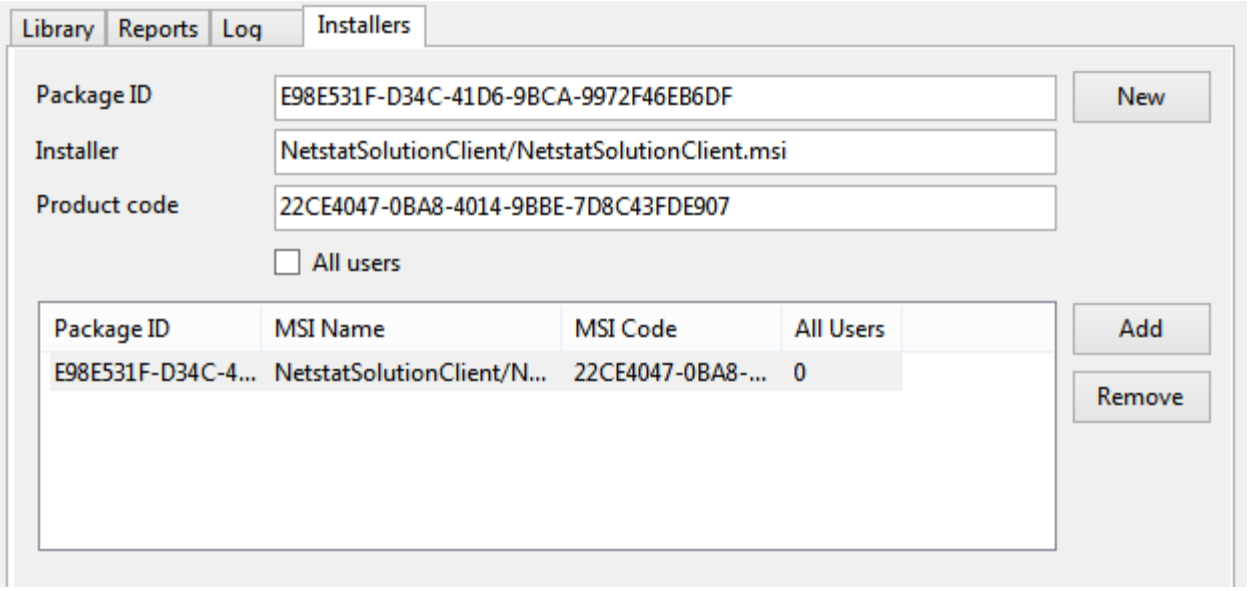


Рисунок 99. Заполнение вкладки "Library"

На вкладке "Installers" указываем:

- *Package ID* — идентификатор пакета установки, используемый при регистрации пакета установки на сервере Docsvision (рассмотрено в разделе Создание расширения для Консоли настройки).

  При необходимости (например, если клиентская часть не установлена), Windows-клиент запрашивает пакет установки с заданным идентификатором из общего списка пакетов, зарегистрированных на сервере Docsvision. Идентификатор пакета является постоянным для пакета установки.

- *Installer* — относительный путь (относительно каталога `C:\Program Files (x86)\Docsvision\Platform\Site\Setup`) к пакету установки клиентской части.

  Пакет установки клиентской части будет разработан позже, но можно указать расположение заранее, либо вернуться к настройке данного параметра позже. Название подкаталога (здесь `NetstatSolutionClient`) определяется исходя из названия установочного файла.

- *Product Code* — код продукта, который соответствует коду продукта в самом пакете установки. Данный идентификатор используется при регистрации клиентской части установки в *SnapIn*, а также в проекте пакета установки клиентской части.

  Идентификатор должен быть обновлен в случае обновления версии библиотеки.

- `All users` — признак установки приложения для текущего или для всех пользователей. Пакет установки должен уметь обрабатывать данный флаг.

Для одной библиотеки карточек можно определить несколько пакетов установки, но, как правило, в этом нет необходимости.

Создание схемы метаданных карточки

Модуль учета сетевых устройств содержит единственную карточку (*Карточка учета сетевого оборудования*), схема и ресурсы которой расположены в подкаталоге /dv6/programmer/dv6/_attachments/netstatSolution.zip[архива] `CardDefs`.

Сценарий создания схемы метаданных карточки достаточно подробно рассмотрен в Создание и редактирование схемы данных, поэтому здесь отмечены только ключевые моменты.

Модуль учета сетевых устройств использует один тип карточки для хранения информации по сетевым устройствам и для хранения журнала доступности. Если создаётся рабочее решение подобного назначения, логичнее будет использовать имеющийся в Docsvision *справочник серверов*, а для хранения журнала отдельные карточки (например, расширить карточку *сервера*).

В метаданных карточки указываем, помимо прочего:

- *Идентификатор* — идентификатор карточки, который также задействован при создании компонента библиотеки карточек.
- *Строка активизации* — содержит идентификатор COM-компонента карточки. Данный идентификатор будет задействован при разработке компонента карточки.

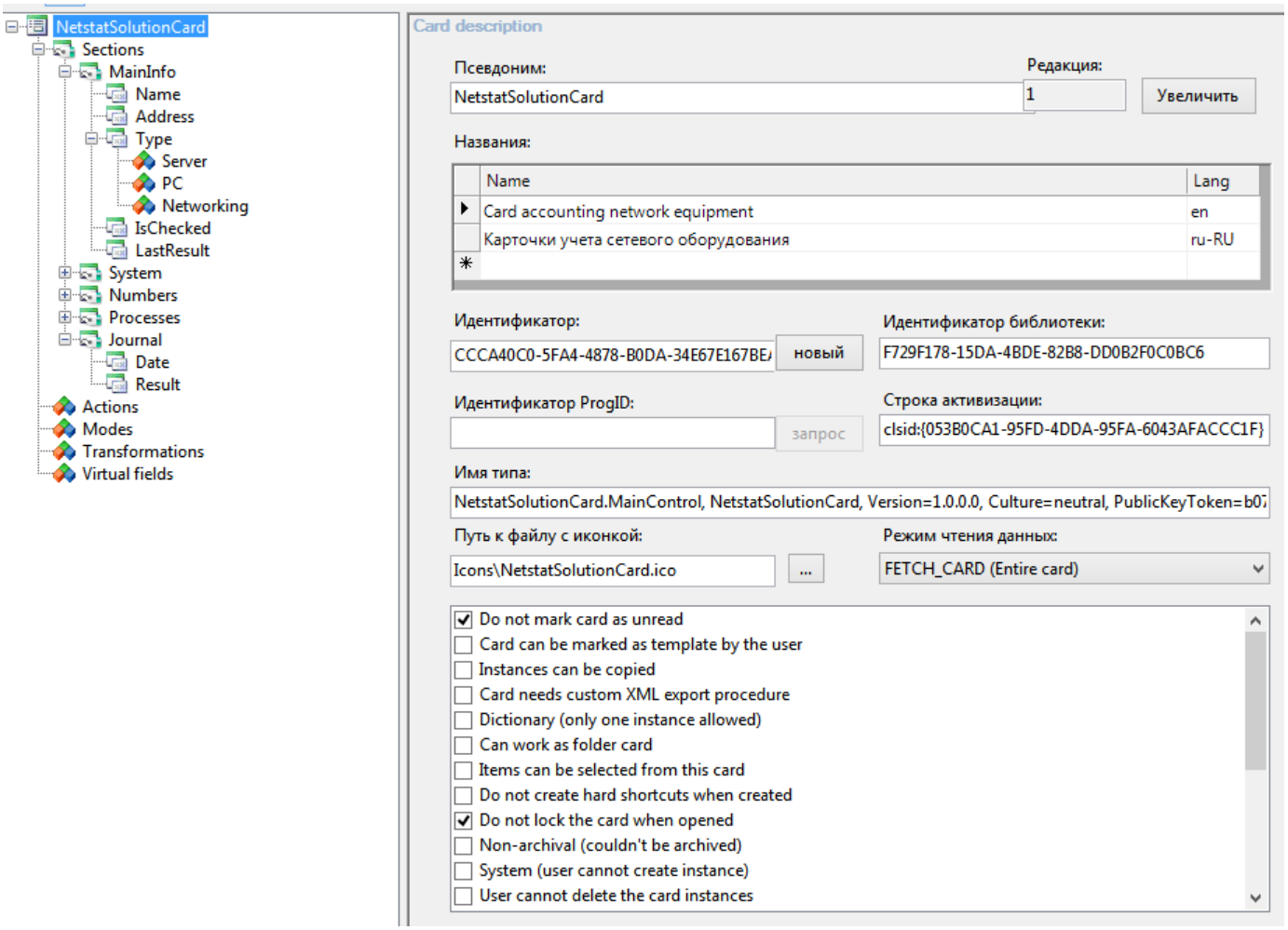


Рисунок 100. Добавление секций в карточку

В схему карточки были добавлены две секции для хранения данных:

- `MainInfo` — содержит базовую информацию о сетевом узле.
- `Journal` — коллекционная секция, в которой хранится журнал проверки доступности узла.

Также добавлены секции `System`, `Numbers` и `Processes`, наличие которых является обязательным для возможности настройки карточки при помощи конструкторов Docsvision.

Данные секции должны иметь определённые идентификаторы, названия и поля, поэтому проще просто скопировать их из существующей карточки (к примеру, "Документ") библиотеки карточек *Базовые объекты*. При копировании необходимо обратить внимание на то, что названия секций меняются — необходимо их переименовать, чтобы соответствовали названиям источников.

Разработка объектной модели карточки

После формирования схемы метаданных карточки, может быть разработана её объектная модель, для чего должны быть созданы:

- Объектная модель строк секций карточки.
- Объектная модель самой карточки.
- Преобразователи данных для строк секций и карточки.

Общая информация о получении объектной модели карточки приведена в разделе Создание объектной модели карточки.

Все типы объектной модели карточки собраны в папке `ObjectModel` проекта `NetstatSolutionCard`.

Разработка объектной модели строк секций

Единственная карточка библиотеки содержит секции: `MainInfo`, `Journal`, `System`, `Numbers`, `Processes`, причем три последних были скопированы из карточки типа "Документ". Базовый класс карточек библиотеки *Базовые объекты* уже содержит описание данных секций, поэтому достаточно создать объектную модель для оставшихся двух: `MainInfo` и `Journal`.

Первоначально разработаем объектную модель строчки секции `MainInfo`:

1. Создаем класс строки секции, используя в качестве базового класс `BaseCardSectionRow` — базовый класс строчки секции карточки библиотеки *Базовые объекты*:

```
public class NetstatSolutionCardMainInfo : BaseCardSectionRow
```

2. Добавляем в класс статическое поле для каждого поля секции карточки:

```
public static readonly ObjectProperty NameProperty; ①
public static readonly ObjectProperty AddressProperty; ②
③
```

① Название узла.

② IP-адрес узла.

③ И так далее.

3. Добавляем в класс свойства, по которым будут доступны значения соответствующих полей:

```
public string Name
 {
  get { return (string)base.GetValue(NetstatSolutionCardMainInfo.NameProperty); }
  set { base.SetValue(NetstatSolutionCardMainInfo.NameProperty, value); }
 }
 public string Address
 {
  get { return (string)base.GetValue(NetstatSolutionCardMainInfo.AddressProperty); }
  set { base.SetValue(NetstatSolutionCardMainInfo.AddressProperty, value); }
 }
①
```

① И так далее.

4. В статическом конструкторе класса регистрируем связи между полями карточки и объектной моделью:

```
static NetstatSolutionCardMainInfo()
{
 NetstatSolutionCardMainInfo.NameProperty = ObjectProperty.Register("Name", typeof
(string), typeof(NetstatSolutionCardMainInfo)); ①
 NetstatSolutionCardMainInfo.AddressProperty = ObjectProperty.Register("Address",
typeof(string), typeof(NetstatSolutionCardMainInfo));
②
```

① В метод `Register` передаем название поля (как в схеме метаданных), его тип (как определён выше) и тип самой секции (текущий класс)

② И так далее.

Аналогичным образом разрабатывается объектная модель строчек секции `Journal`.

Разработка объектной модели карточки

Поскольку ранее было решено поддержать возможность настройки карточки при помощи конструкторов, необходимо сохранить в её объектной модели секции `System`, `Numbers`, `Processes`. Для этого достаточно унаследовать свой класс объекта карточки от базового класса `DocsVision.BackOffice.ObjectModel.BaseCard`.

1. Создаем класс объекта карточки:

```
public class NetstatSolutionCard : BaseCard
```

2. Для каждой секции добавляем статическое поле класса:

```
public static readonly ObjectProperty MainInfoProperty;
public static readonly ObjectProperty JournalProperty;
```

3. Добавляем в класс свойства для доступа к данным секции:

```
public NetstatSolutionCardMainInfo MainInfo
{
 get
 {
  if (((ObjectCollection<NetstatSolutionCardMainInfo>)base.GetValue
(NetstatSolutionCard.MainInfoProperty)).Count = 0) ①
  {
   ((ObjectCollection<NetstatSolutionCardMainInfo>)base.GetValue(NetstatSolutionCard
.MainInfoProperty)).Add(new NetstatSolutionCardMainInfo());
  }
  return ((ObjectCollection<NetstatSolutionCardMainInfo>)base.GetValue
(NetstatSolutionCard.MainInfoProperty)).First<NetstatSolutionCardMainInfo>();
 }
}
public ObjectCollection<NetstatSolutionCardJournal> Journal
{
 get
 {
  return (ObjectCollection<NetstatSolutionCardJournal>)base.GetValue
(NetstatSolutionCard.JournalProperty);
 }
}
```

① Считаем, что данная секция должна быть, а если её нет, то создаем пустую.

4. Регистрируем связи между секциями карточки и их объектной моделью в статическом конструкторе класса:

```
static NetstatSolutionCard()
{
 NetstatSolutionCard.MainInfoProperty = ObjectProperty.Register("MainInfo", typeof
(ObjectCollection<NetstatSolutionCardMainInfo>), typeof(NetstatSolutionCard));
 NetstatSolutionCard.JournalProperty = ObjectProperty.Register("Journal", typeof
```

```
(ObjectCollection<NetstatSolutionCardJournal>), typeof(NetstatSolutionCard));
}
```

В платформе Docsvision принято выносить бизнес-логику карточки из её объектной модели в сервисы. Последуем данному правилу и вынесем основные функции карточки в отдельный сервис (будет разработан позднее).

Разработка преобразователей данных

Для строчек секций и карточки должны быть разработаны преобразователи данных.

Разработка преобразователя данных для строк секций

Поскольку базовым классом строк секций был выбран класс `BaseCardSectionRow`, преобразователь данных должен наследоваться от класса `BaseCardSectionRowMapper<T>`.

1. Добавляем класс преобразователя данных:

```
public class NetstatSolutionCardMainInfoMapper : BaseCardSectionRowMapper
<NetstatSolutionCardMainInfo>
```

2. Регистрируем поля секции в карте преобразователя данных:

```
private static void InitializeObjectMap()
{
 NetstatSolutionCardMainInfoMapper.map = new ObjectMap();
 NetstatSolutionCardMainInfoMapper.map.ObjectTypeId = NetstatSolutionCardDefs.
MainInfo.ID;
 NetstatSolutionCardMainInfoMapper.map.Field(NetstatSolutionCardMainInfo.AddressProper
ty, "Address");
 NetstatSolutionCardMainInfoMapper.map.Field(NetstatSolutionCardMainInfo.IsCheckedProp
erty, "IsChecked");
①
```

① И так далее.

Для строчек секции `Journal` преобразователь данных создаётся аналогичным способом.

Разработка преобразователя данных для карточки

Преобразователь данных для карточки оформляется аналогично

преобразователю данных для строчек секции, за исключением базового класса.

1. Добавляем класс преобразователя данных:

```
public class NetstatSolutionCardMapper : BaseCardMapper<NetstatSolutionCard>
```

2. Регистрируем секции карточки в карте преобразователя данных:

```
private static void InitializeObjectMap()
{
 NetstatSolutionCardMapper.map = new ObjectMap();
 NetstatSolutionCardMapper.map.ObjectTypeId = NetstatSolutionCardDefs.ID;
 NetstatSolutionCardMapper.map.Collection(NetstatSolutionCard.MainInfoProperty,
NetstatSolutionCardDefs.MainInfo.ID);
 NetstatSolutionCardMapper.map.Collection(NetstatSolutionCard.JournalProperty,
NetstatSolutionCardDefs.Journal.ID);
}
```

Создание фабрики преобразователей данных

Для добавления функциональности преобразователей данных в контекст объектов необходимо разработать фабрику преобразователей данных.

1. Создаем класс фабрики преобразователей данных с базовым классом `ObjectMapperFactory`:

```
public class NetstatSolutionMapperFactory : ObjectMapperFactory
```

2. Добавляем конструктор, принимающий контекст объектов. В конструкторе регистрируются все разработанные ранее преобразователи данных.

```
public NetstatSolutionMapperFactory(ObjectContext context) : base(context)
{
 base.RegisterObjectMapper(typeof(NetstatSolutionCard), typeof
(NetstatSolutionCardMapper)); ①
 base.RegisterObjectMapper(typeof(NetstatSolutionCardMainInfo), typeof
(NetstatSolutionCardMainInfoMapper));
 base.RegisterObjectMapper(typeof(NetstatSolutionCardJournal), typeof
(NetstatSolutionCardJournalMapper));
}
```

① Регистрируем в фабрике созданные ранее преобразователи данных.

В `RegisterObjectMapper` передается тип объекта и тип преобразователя, соответствующего объекту.

После разработки объектной модели карточки (и секций), а также фабрики преобразователей данных, собрав и подключив библиотеку к проекту, уже можно получить объект карточки из контекста объектов, если фабрика преобразователей данных была зарегистрирована в контексте объектов (см. пример инициализации контекста).

Разработка сервиса для работы с карточкой

Поскольку ранее было решено вынести бизнес-логику карточки за пределы её класса в сервис, разработаем его.

1. Создаем интерфейс, описывающий функциональность будущего сервиса:

```
public interface INetstatSolutionService
{
 NetstatSolutionCard.ObjectModel.NetstatSolutionCard CreateCard(string name, string
address, int type); ①

 ObjectModel.NetstatSolutionCardJournal AddJournalEntry(NetstatSolutionCard
.ObjectModel.NetstatSolutionCard card, DateTime date, bool result); ②
}
```

① Создание новой карточки.

② Добавление результата проверки в журнал.

2. Добавляем класс, реализующий интерфейс `INetstatSolutionService` и наследующий от `ContextService` — базовый класс сервисов объектной модели:

```
internal class NetstatSolutionService : ContextService, INetstatSolutionService
```

   Специально ограничиваем область видимости класса, т.к. контекст объектов предоставляет сервис по его интерфейсу.

3. Реализуем функциональность, описанную интерфейсом `INetstatSolutionService`.

   Реализацию смотрите в проекте решения.

4. Создаем фабрику сервисов в виде класса, наследующего от `ServiceFactory`.

Фабрика предоставляет экземпляр сервиса по его типу.

```
public class NetstatSolutionServiceFactory : ServiceFactory
{
 protected override object GetService(Type serviceType)
 {
  if (serviceType = typeof(INetstatSolutionService)) ①
  {
   return new NetstatSolutionService();
  }
  return null;
 }
}
```

① Метод должен вернуть экземпляр класса, реализующего `INetstatSolutionService`, по интерфейсу.

После реализации фабрики сервисов, можно собрать проект и подключить библиотеку, содержащую сервис (интерфейс, реализацию) и фабрику сервисов, к проекту и запросить данный сервис из контекста объектов, если фабрика сервисов зарегистрирована в контексте объектов (см. пример инициализации контекста объектов).

Типы и интерфейс, разработанные для сервиса, собраны в папке `Services` проекта `NetstatSolutionCard`.

Разработка компонента карточки

Исходный код компонента карточки расположен в подкаталоге `NetstatSolutionCard` приложенного /dv6/programmer/dv6/_attachments/netstatSolution.zip[архива].

Поскольку пример разработки компонента карточки приведён в разделе Разработка компонента карточки, здесь отмечены только ключевые моменты его реализации для *Модуля учета сетевых устройств*.

Реализация интерфейса карточки в компоненте не планируется, вместо этого в *Конструкторе разметок* будет настроена её разметка, с последующей выгрузкой этих настроек (рассмотрено в разделе Формирование "CardPackage" и "SqlPackage") для размещения в пакете установки. Чтобы разметка карточки могла быть настроена в конструктора, класс компонента карточки отмечен специальным атрибутом `Customizable`.

В конструктор класса компонента карточки добавим простой алгоритм проверки лицензионного ключа, значение которого считывается из реестра Windows. Для ввода лицензионного ключа предполагается использовать *SnapIn*, который будет разработан позднее.

Проверку лицензии при необходимости можно реализовать несколькими методами (см. Лицензирование).

Класс компонента должен быть отмечен атрибутом `Guid`, значение которого соответствует *Идентификатору COM-компонента карточки* (см. Создание схемы метаданных карточки).

```
using DocsVision.BackOffice.WinForms;
using DocsVision.Platform.CardHost;
using System;
using System.Runtime.InteropServices;

namespace NetstatSolutionCard
{
 [ComVisible(true)]
 [Customizable(true)]
 [Guid("053B0CA1-95FD-4DDA-95FA-6043AFACCC1F")]
 [ClassInterface(ClassInterfaceType.None)]
 public partial class MainControl : DocsVision.BackOffice.WinForms.BaseCardControl
 {
  public MainControl()
  {
   if (!CheckLicense())
   {
    throw new Exception("Отсутствует лицензия для Модуля учета сетевых устройств"); ①
   }

   InitializeComponent();
  }

  private bool CheckLicense() ②
  {
   bool result = false;

   return result; ③
  }
 }
}
```

① В случае отсутствия лицензии, при открытии карточки будет выдано сообщение об ошибке.

② Алгоритм проверки лицензии.

③ Логику проверки смотрите в исходном коде проекта.

Если вы работаете с исходными кодами проекта, то компонент карточки собирается первым.

Разработка компонента библиотеки карточек

Исходный код компонента библиотеки карточек расположен в подкаталоге `CardLib` приложенного /dv6/programmer/dv6/_attachments/netstatSolution.zip[архива].

Компонент библиотеки карточек является обязательной частью решения, он определяет корректность установки решения в целом, а его отсутствие является сигналом Windows-клиенту о необходимости установки клиентской части.

При большом размере и количестве схем карточек, зарегистрированных в библиотеке, а также большом числе клиентов Docsvision, компонент библиотеки карточек выполняет роль статического кэша на стороне клиента со схемами и ресурсами карточек, что предполагает выигрыш в части снижения нагрузки на сервер Docsvision.

Пример разработки компонента библиотеки карточек на C# приведён в Руководстве разработчика в разделе Разработка компонента библиотеки карточек.

В версии модуля Платформа 5.5.2 требования к компоненту библиотеки карточек были изменены.

Компонент библиотеки карточек (C++) проще всего создать на основе готового шаблона с заменой отдельных частей кода на собственные значения. Например, использовать в основе код компонента данной библиотеки и заменить идентификаторы библиотеки карточек и карточек.

При разработке компонента без использования шаблона, требования к реализации остаются стандартными:

- Должны быть реализованы интерфейсы `ICardLibraryInfo` и

ICardLibraryInfo2.

- Должны быть загружены ресурсы: схемы карточек и библиотеки карточек, а также их иконки.
- Компонент библиотеки должен быть реализован как COM-компонент.

Полный алгоритм создания готового компонента на основе шаблона можно разбить на следующие шаги:

1. Укажите действительную *Версию библиотеки карточек* в файле dvver.h, в определении идентификатора сборки VERSION_BUILD. В этом же файле можно заменить авторство сборки.
2. Добавьте в файл ресурсов (CardLib.rc) следующие ресурсы:
   - В XMLDEF добавляются все схемы карточек, включенных в данную библиотеку, а также схема самой библиотеки карточек.
   - В Icon добавляются иконки карточек и библиотеки карточек.

   При необходимости добавьте в файл с идентификаторами ресурсов (Resource.h) идентификаторы новых ресурсов.
3. Измените скрипт регистрации CardLib.rgs:
   - В CLSID и ForceRemove нужно указать *идентификатор компонента библиотеки карточек*, который был указан при разработке схемы библиотеки карточек в параметре Activation string.
   - В TypeLib нужно указать произвольный (сгенерировать самостоятельно) *идентификатор библиотеки типов* — заменить NetstatSolution на собственное название для регистрации COM-компонента.
4. В файле CardLib.def укажите название файла сборки.
5. В файле CardLib.idl необходимо указать *идентификатор компонента библиотеки карточек* и *идентификатор библиотеки типов*, которые были использованы на шаге 3.
6. В файле CardLibInfo.cpp:
   - В массиве GetCardType.Cards перечислите идентификаторы всех карточек библиотеки (соблюдайте формат), их схемы (название ресурса) и иконки.
   - В методе get_Version в sCardLibDef нужно указать собственное название ресурса со схемой библиотеки карточек.

7. В свойствах проекта укажите необходимое название для сборки.

Если вы работаете с исходными кодами проекта, то компонент библиотеки карточки собирается вторым.

Формирование "CardPackage" и "SqlPackage"

На этапе установки решения, в Docsvision могут быть импортированы определённые данные, например: данные справочников, карточки и бизнес-процессы, для чего должен быть подготовлен специальный пакет — `CardPackage`, содержащий файлы с данными карточек и т.д., а также файл со ссылками на эти файлы.

Один из сценариев использования пакета `CardPackage` — настроить разметку своего типа карточки в *Конструкторе разметок*, после чего выгрузить эту разметку и распространить её с пакетом установки на целевые сервера Docsvision. Аналогично можно поступить с *Конструктором состояний* и *Конструктором ролей*.

Файлы пакета `CardPackage` расположены в подкаталоге `CardPackage` приложенного /dv6/programmer/dv6/_attachments/netstatSolution.zip[архива].

Все действия, связанные с формированием пакетов `CardPackage` и `SqlPackage`, должны выполняться на отдельном сервере Docsvision.

Предварительная подготовка

Прежде всего необходимо загрузить библиотеку карточек в базу данных Docsvision (см. раздел Загрузка схем карточек в базу данных), а также разместить на клиенте компоненты карточки и библиотеки карточек.

Компонент карточки на данном этапе достаточно разместить в каталоге Docsvision Windows-клиент.

Компонент библиотеки карточек должен быть зарегистрирован командой: `regsvr32.exe Полный-путь-к-сборке`, к примеру, `regsvr32.exe NetstatSolutionCardLib.dll`.

Для регистрации библиотеки с использованием утилиты `regsvr32` необходимы права администратора.

Формирование данных

Когда в Docsvision был загружен новый тип карточки, настраиваем его при помощи конструкторов, заполняем справочник, если решение его содержит, первоначальными данными, создаем необходимые бизнес-процессы и карточки.

Тип "Карточка учета сетевого оборудования" была настроен с использованием:

- *Конструктора разметок* — создана разметка с привязкой к данным карточки.
- *Конструктора состояний* — добавлено начальное состояние.
- *Конструктора ролей* — добавлены типовые права на работу с карточкой.

Также были внесены изменения (создан корневой вид карточки) в *Справочник видов карточек*.

Дополнительно был сформирован пустой (реализация бизнес-процесса выходит за рамки примера) бизнес-процесс "Проверка доступности сетевых узлов", а также экземпляр карточки *учета сетевого оборудования*.

Все эти данные предполагается автоматически загружать в Docsvision при установке решения.

Выгрузка данных

Сформированные данные необходимо выгрузить из Docsvision в формате XML, т.к. при установке решения они просто импортируются средствами платформы.

Карточки (включая бизнес-процессы) экспортируются стандартными средствами Docsvision Windows-клиент.

Также возможно выгрузить данные из справочников и конструкторов.

Создание "CardPackage"

Чтобы закончить формирование пакета `CardPackage`, требуется создать файл `CardPackage.xml` следующего формата:

```
<?xml version="1.0" encoding="UTF-8"?>
<CardPackage>
 <Card Path="относительный_путь_к_файлу_XML" ID="ID_карточки" Replace=""/> ①
 ...
</CardPackage>
```

Replace="1" если карточка должна заменить существующую, `0` — если дополнить.

Создание "SqlPackage"

Готовый файл (`NetstatSolutionInstall.xml`) пакета "SqlPackage" содержится в каталоге `Database` приложенного /dv6/programmer/dv6/_attachments/netstatSolution.zip[архива].

Помимо `CardPackage`, для пакета установки решения требуется создать пакет `SqlPackage`, состоящий из единственного файла, в котором указано расположение файлов со схемами метаданных библиотек карточек. Файл должен быть загружен в базу данных при установке решения.

"CardPackage" должен соответствовать формату:

```
<?xml version="1.0" encoding="UTF-8"?>
<SqlPackage>
 <SqlFile>"относительный_путь_к_файлу_XML"</SqlFile>
 ...
</SqlPackage>
```

Как правило `SqlPackage` содержит одну единственную ссылку на файл со схемой библиотеки карточек.

Создание расширения для Консоли настройки

Исходный код *SnapIn* расположен в подкаталоге `NetstatSolution.Snapin` приложенного /dv6/programmer/dv6/_attachments/netstatSolution.zip[архива].

На данном этапе необходимо создать расширение для Консоли настройки (*SnapIn*), которое отвечает за загрузку `CardPackage` и `SqlPackage` в Docsvision, а также регистрирует пакет установки клиентской части.

SnapIn является обязательным компонентом, при условии, что решение использует собственную библиотеку карточек.

Как часть "Модуля учета сетевых устройств", данный компонент выполняет следующие задачи:

- Устанавливает и удаляет библиотеку карточек *Модуля учета сетевых устройств*.
- Импортирует в Docsvision бизнес-процесс *Проверка доступности сетевых узлов*, а также пример карточки *Карточка учета сетевого оборудования*, а также данные *справочника видов карточек*, *конструктора разметок*,

конструктора состояний и *конструктора ролей*.

- Регистрирует клиентский пакет установки.
- Добавляет в *Консоль настройки Docsvision* форму для ввода конфигурационных параметров *Модуля учета сетевых устройств*.

Пример реализации простого *SnapIn* приведён в разделе Модуль расширения Консоли настройки

Компонент *SnapIn* представляет собой библиотеку DLL, в которой присутствует класс, реализующий интерфейс `DocsVision.Tools.ServerConsole.ISnapIn` (сборка `DocsVision.Tools.ServerConsole.Interfaces.dll`).

Также данный класс может реализовывать несколько дополнительных интерфейсов:

- `IDBInformation` — является обязательным при создании решения, т.к. именно реализация данного интерфейса позволяет *SnapIn* предоставлять пути к `CardPackage` и `SqlPackage`.
- `IConfigurator` — добавляет в *SnapIn* метод Execute, который запускается в режиме конфигурации модуля — вызывается командой `ServerConsole.exe /c /n {ИМЯ_РЕШЕНИЯ}`. Обычно данный метод отвечает за регистрацию пакетов установки клиентской части решения.
- `IUninstallSnapIn` — добавляет метод `Uninstall`, который вызывается при удалении решения — вызывается командой `ServerConsole.exe /rs /n ИМЯ_РЕШЕНИЯ`.

Помимо выполнения сервисных функций, *SnapIn* может предоставлять пользовательский интерфейс для различных режимов работы *Консоли настройки*.

Если вы работаете с исходными кодами проекта, то *SnapIn* собирается третьим.

Реализация стандартных интерфейсов

Расширение Консоли настройки, разрабатываемое для *Модуля учета сетевых устройств*, должно регистрировать клинский пакет установки, устанавливать новую библиотеку карточек, а также загружать в базу данных Docsvision заранее подготовленные данные карточек и справочников. Для этого в классе *SnapIn* должно быть реализовано несколько интерфейсов.

Реализация интерфейса ISnapIn

Основной интерфейс расширения, который добавляет:

- `LibraryID` — *идентификатор библиотеки карточек*, с которой данное расширение ассоциировано. `LibraryID` требуется для регистрации пакетов установки, добавления библиотеки карточек в базу данных, удаления решения, а также для некоторых других сценариев работы *Консоли настройки*.

```
public string LibraryID
{
 get { return "{F729F178-15DA-4BDE-82B8-DD0B2F0C0BC6}"; }
}
```

- `Name` — название модуля. Используется в некоторых сервисных функциях, например: при отображении названия расширения в списке зарегистрированных расширений *Консоли настройки*.

```
public string Name
{
 get { return "Консоль управления модулем Проверка сети"; }
}
```

- `Метод Initialize(IEnvironment)` — позволяет получить переменные контекста работы Консоли настройки, которые предоставляют доступ к сервису регистрации пакетов установки клиентской части.

```
public void Initialize(IEnvironment environment)
{
 _environment = environment;
}
```

- Метод `QueryInterface(SnapInInterfacesEnum)` — возвращает пользовательский интерфейс, если он предусмотрен, для различных режимов работы *Консоли настройки*. Подробнее см. в разделе Модуль расширения Консоли настройки.

```
public object QueryInterface(SnapInInterfacesEnum itf)
{
 object result = null;
 switch (itf)
 {
```

```
   case SnapInInterfacesEnum.CONFIGURATOR:
   case SnapInInterfacesEnum.UNINSTALL_SNAP_IN:
   case SnapInInterfacesEnum.DB_INFORMATION:
    result = this;
    break;

   case SnapInInterfacesEnum.CONSOLE_CONTROL:
    result = new SnapInForm(); ①
    break;
   }
  return result;
 }
```

① Экземпляр типа, реализующего пользовательский интерфейс.

Реализация интерфейса "IConfigurator"

Данный интерфейс должен быть реализован, если расширение может выполнять первоначальную настройку решения.

Как правило, метод `Execute`, при установке расширения вызываемый последним, регистрирует пакеты установки клиентской части, для чего указываются: путь к пакету установки, *Идентификатор пакета установки* и *Код продукта*, которые были указаны при разработке библиотеки карточек:

```
public bool Execute()
{
 ILog log = (ILog)_environment.QueryService(EnvironmentServiceEnum.LOG);

 ICardLibConfigurator2 cardLibConfig = (ICardLibConfigurator2)_environment.QueryService
(EnvironmentServiceEnum.CARD_LIB_CONFIGURATOR); ①

②
 cardLibConfig.RegisterPackage(
  "E98E531F-D34C-41D6-9BCA-9972F46EB6DF", ③
  "22CE4047-0BA8-4014-9BBE-7D8C43FDE907", ④
  Path.Combine(AssemblyFolder, "NetstatSolutionClient.msi")); ⑤

 log.WriteMessage("Конфигурирование завершено");

 return true; ⑥
}
```

① Регистрация клиентского пакета установки.

② Регистрация пакета установки.

③ Идентификатор пакета установки.

④ Код продукта.

⑤ Абсолютный путь к пакету установки клиентской части.

⑥ Если метод должен сообщить об ошибке, то необходимо вызвать исключение. В обычном случает, метод возвращает `true`.

Реализация интерфейса "IDBInformation"

Данный интерфейс добавляет методы, предоставляющие *Консоли настройки* пути к *CardPackage* и *SqlPackage*.

Метод `GetCardPackage` должен вернуть абсолютный путь к пакету *CardPackage*:

```
public string GetCardPackage()
{
 return Path.Combine(AssemblyFolder, "CardPackage\\CardPackage.xml");
}
```

Метод `GetScript(ScriptTypeEnum)` должен вернуть абсолютный путь к пакету *SqlPackage*, если запрашивается тип `ScriptTypeEnum.INSTALL_TABLES`:

```
public string GetScript(ScriptTypeEnum type)
{
 string result = string.Empty;

 switch (type)
 {
  case ScriptTypeEnum.INSTALL_TABLES:
   result = Path.Combine(AssemblyFolder, "Database\\NetstatSolutionInstall.xml");
   break;
 }
 return result;
}
```

Также добавляется переменная `ContainsCardLib`, которая определяет наличие у решения собственной библиотеки карточек.

```
public bool ContainsCardLib
{
 get { return true; }
}
```

Реализация интерфейса IUninstallSnapIn

Данный интерфейс добавляет единственный метод `Uninstall(Boolean)`, вызываемый при удалении расширения (и решения в целом).

Метод `Uninstall` может, к примеру, удалять библиотеку карточек, а также настройки решения, как в данном случае:

```
public void Uninstall(bool removeSettings)
{
 ILog log = (ILog)_environment.QueryService(EnvironmentServiceEnum.LOG);
 ICardLibConfigurator cardLibConfig = (ICardLibConfigurator)_environment.QueryService
(EnvironmentServiceEnum.CARD_LIB_CONFIGURATOR);
 log.WriteMessage("Запущено удаление");

 cardLibConfig.RemoveCardLib(LibraryID); ①

 if (removeSettings) ②
 {
  using (RegistryKey key = Common.GetSubKey(Registry.LocalMachine, Common
.NetstatSolutionKey))
  {
   key.DeleteValue(Common.EmailAdminRegName, false);
   key.DeleteValue(Common.CheckIsEnabledRegName, false);
   key.DeleteValue(Common.LicenseRegName, false);
  }
 }
 log.WriteMessage("Записи реестра удалены");
}
```

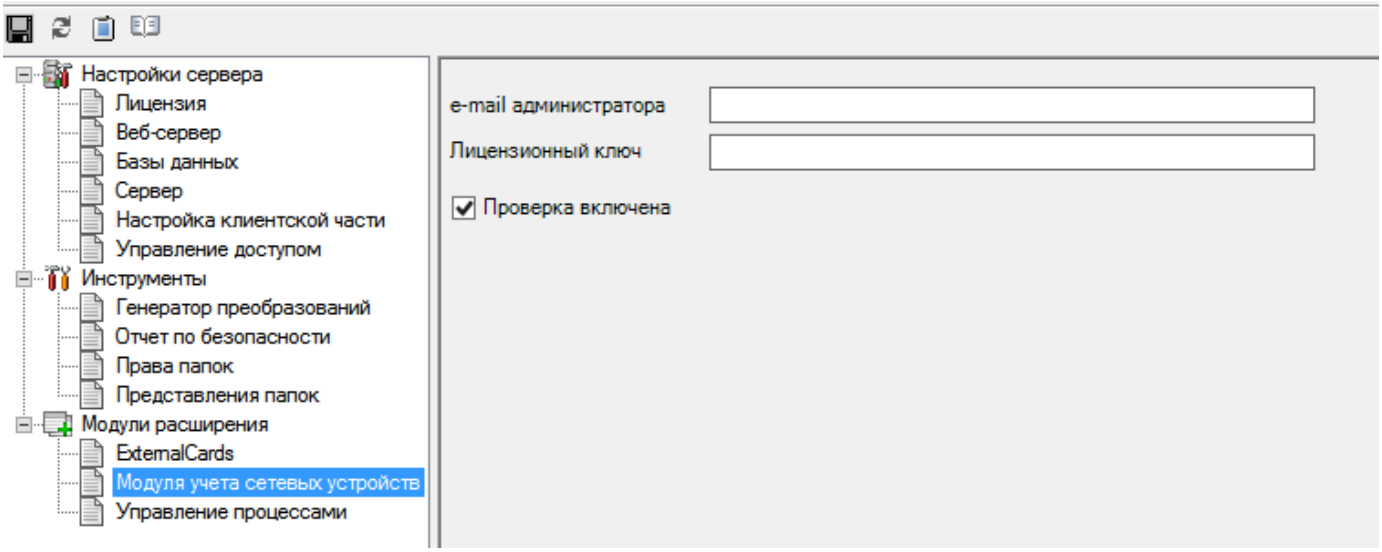
① Удаление библиотеки карточек.

② Удаление настроек модуля.

Пользовательский интерфейс расширения

Если для решения необходима административная настройка, к примеру: выбор почтового сервера, то для расширения **Консоли настройки Docsvision** может быть реализован пользовательский интерфейс.

Для *Модуля учета сетевых устройств* предусмотрен ввод некоторых данных: e-mail администратора, лицензионный ключ, признак глобального отключения проверки.



Для работы с карточками *учета сетевого оборудования* требуется лицензионный ключ: введите в поле *Лицензионный ключ* значение "ЛИЦЕНЗИЯ".

Интерфейс управления для расширения консоли настройки — это обычный WinForms элемент управления (типа `System.Windows.Forms.UserControl`), в котором реализован интерфейс `IConsoleControl` (сборка `DocsVision.Tools.ServerConsole.Interfaces.dll`).

Интерфейс `IConsoleControl` *добавляет:*

- Свойство `Changed`, которое возвращает признак наличие изменений в данных. При закрытии формы, при наличии изменений, будет выдано предупреждение о необходимости сохранения данных.

```
public bool Changed
{
 set
 {
  isChanged = value;
  ControlChanged();
 }
 get
 {
  return isChanged;
 }
}
```

- Свойство *Valid*, которое возвращает результат проверки данных на наличии ошибок. Проверка выполняется при закрытии формы. Алгоритм верификации

данных разрабатывается самостоятельно.

```
public bool Valid
{
 get
 {
  bool result = true;
  if (Admin_eMail.Text.Contains("@") = false)
  {
   result = false; ①
  }
  return result;
 }
}
```

① Возвращаем ошибку, если e-mail некорректен.

- Свойство `Caption` — название модуля, которое отображается в узле *Модули расширения Консоли настройки*.

```
public string Caption
{
 get { return "Модуль учета сетевых устройств"; }
}
```

- Свойство `Instance` — должно возвращать экземпляр класса, реализующего элемент управления.

```
public UserControl Instance
{
 get { return this; }
}
```

- Метод `Execute` — вызывается при сохранении данных. Данный метод будет вызван, если на запрос сохранения данных, был получен утвердительный ответ.

```
public bool Execute()
{
 try
 {
  using (RegistryKey key = Common.GetSubKey(Registry.LocalMachine, Common
```

```
.NetstatSolutionKey))
  {
   key.SetValue(Common.EmailAdminRegName, Admin_eMail.Text);
   key.SetValue(Common.CheckIsEnabledRegName, Convert.ToInt32(CheckIsOn.Checked));
   key.SetValue(Common.LicenseRegName, License.Text);
  }
  isChanged = false;
 } catch
 {
  return false;
 }
 return true;
}
```

- Метод `Initialize` — отвечает за предварительную загрузку данных и другие подготовительные действия. Вызывается перед открытием формы.

```
public void Initialize()
{
 using (RegistryKey key = Common.GetSubKey(Registry.LocalMachine, Common
.NetstatSolutionKey))
 {
  Admin_eMail.Text = key.GetValue(Common.EmailAdminRegName, string.Empty).ToString();
  CheckIsOn.Checked = Convert.ToBoolean((int)key.GetValue(Common.
CheckIsEnabledRegName, 1));
  License.Text = key.GetValue(Common.LicenseRegName, string.Empty).ToString();
 }
 isChanged = false;
}
```

- Генерация событие `ControlChanged` позволяет уведомить *Консоль настройки* об изменении данных, что активирует кнопку сохранения данных. Если самостоятельно не вызвать данный метод, сохранить данные можно будет только при закрытии формы, при появлении соответствующего запроса.

Создание пакета установки

В данном разделе приведена краткая последовательность создания пакетов установки клиентской и серверной части решения с использованием WiX Toolset. Пакеты установки могут быть получены с использованием других средств при условии, что будет получен пакет Windows Installer.

Пакет установки клиентской части решения

Исходный код *SnapIn* расположен в подкаталоге `NetstatSolutionClientInstaller` /dv6/programmer/dv6/_attachments/netstatSolution.zip[архива] проекта.

Пакет установки клиентской части решения устанавливается на рабочем месте сотрудника, и отвечает за установку и регистрацию компонентов карточек и библиотеки карточек. Клиентская часть устанавливается автоматически при открытии *Windows-клиента*, если компонент библиотеки карточек не зарегистрирован.

Типовой сценарий установки клиентской части:

1. Создать подкаталог решения в каталоге Windows-клиента.

Крайне желательно соблюсти правило размещения компонентов в каталоге Windows-клиента: `Каталог-Windows-клиента\Название-решения`.

2. Распаковать компоненты карточки и библиотеки карточек из пакета установки в созданный подкаталог.
3. Зарегистрировать распакованные компоненты.
   - Компонент карточки не регистрируется.
   - Компонент библиотеки карточек регистрируется командой: `regsvr32.exe Путь-к-файлу-компонента` (файл должен быть доступен всем пользователям) — данная команда пишет в общую ветку реестра, для чего требуются права администратора. Можно зарегистрировать компонент для текущего пользователя, но для этого потребуется самостоятельно реализовать скрипт для регистрации COM-компонента в ветке реестра `HKEY_CURRENT_USER`.
4. Создать в ветке реестра `HKEY_CURRENT_USER\SOFTWARE\DocsVision\Platform\Client\Applications` строковый параметр с названием решения. В значении должно быть указано название созданного подкаталога:

```
<Component Id="RegistryEntriesComponents" Guid="76563983-1339-469A-9144-2C7C0D6C7F45">
 <RegistryKey Root="HKCU" Key="SOFTWARE\DocsVision\Platform\Client\Applications">
  <RegistryValue Type="string" Name="NetstatSolution" Value="NetstatSolution" />
 </RegistryKey>
```

```
</Component>
```

При запуске Windows-клиента автоматически подключаются библиотеки, расположенные в подкаталогах, указанных в узле `Applications` реестра.

При создании пакета установки в качестве идентификатора продукта необходимо использовать *Код продукта*, который был использован в схеме библиотеки карточки на этапе разработки библиотеки карточки:

```
<Product Id="22CE4047-0BA8-4014-9BBE-7D8C43FDE907"
```

Чтобы клиентская часть поддерживала обновление, добавляем:

```
<MajorUpgrade DowngradeErrorMessage="Более новая версия продукта [ProductName] уже
установлена." />

<Upgrade Id="$(var.UpgradeCode)">
 <UpgradeVersion Minimum="$(var.ProductVersion)"
  IncludeMinimum="no"
  OnlyDetect="yes"
  Language="1049"
  Property="NEWPRODUCTFOUND" />

 <UpgradeVersion Minimum="$(var.RTMProductVersion)"
  IncludeMinimum="yes"
  Maximum="$(var.ProductVersion)"
  IncludeMaximum="no"
  Language="1049"
  Property="UPGRADEFOUND" />
</Upgrade>
```

Если вы работаете с исходными кодами проекта, то пакет установки клиентской части собирается четвертым.

Пакет установки серверной части решения

Исходный код *SnapIn* расположен в подкаталоге `NetstatSolutionInstaller` /dv6/programmer/dv6/_attachments/netstatSolution.zip[архива] проекта.

Пакет установки серверной части включает в себя все разработанные ранее компоненты, в том числе пакет установки клиентской части. Данный установщик отвечает за весь цикл регистрации и удаления решения.

Далее перечислены основные шаги, которые должен выполнить пакет установки серверной части решения:

1. Создать подкаталог решения в папке Docsvision.
2. Выгрузить в созданный подкаталог компонент *SnapIn* и пакет установки клиентской части решения.
3. Создать в каталоге решения подкаталог `CardPackage` и выгрузить в него `CardPackage`, а также связанные с ним XML-файлы.
4. Создать в каталоге решения подкаталог `Database` и выгрузить в него `SqlPackage`, а также схемы карточек и библиотека карточек, а также их ресурсы.
5. В ветке реестра `HKEY_LOCAL_MACHINE\SOFTWARE\DocsVision\Platform\Console\SnapIns` создать раздел с названием решения.
6. В созданный раздел добавить два строковых параметра:
   - `Path` — путь к сборке *SnapIn*.
   - `TypeName` — название класса *SnapIn*, который реализует интерфейс `ISnapIn`.

   ```
   <Component Id="RegistryEntries" Guid="75B4E976-5771-423C-AB7D-DC583114F893"
   Win64="yes">
    <RegistryKey Root="HKLM"
   Key="SOFTWARE\DocsVision\Platform\Console\SnapIns\NetstatSolution"
   ForceDeleteOnUninstall="yes" ForceCreateOnInstall="yes">
     <RegistryValue Type="string" Name="Path" Value="C:\Program Files
   (x86)\Docsvision\NetstatSolution\NetstatSolution.Snapin.dll"/>
     <RegistryValue Type="string" Name="TypeName"
   Value="NetstatSolution.Snapin.SnapIn"/>
    </RegistryKey>
   </Component>
   ```

7. Вызвать Консоль настройки Docsvision с параметрами: `"C:\Program Files (x86)\Docsvision\Platform\Tools\ServerConsole.exe" /c /n \{Название решения}`. В проекте пакета установки серверной части за это отвечает сценарий:

   ```
   <CustomAction Id="RegisterSnapIn" Directory='INSTALLFOLDER' Return="check" Execute=
   'deferred'
    ExeCommand='"C:\Program Files (x86)\Docsvision\Platform\Tools\ServerConsole.exe" /c
   /n NetstatSolution' />
   ```

```
<InstallExecuteSequence>
 <Custom Action='RegisterSnapIn' Before='InstallFinalize'>NOT Installed</Custom>
</InstallExecuteSequence>
```

При удалении решения, необходимо вызвать *Консоль настройки* Docsvision с параметрами: `"C:\Program Files (x86)\Docsvision\Platform\Tools\ServerConsole.exe" /rs /n Название-решения`.

В проекте пакета установки серверной части за это отвечает сценарий:

```
<CustomAction Id="UnRegisterSnapIn" Directory='INSTALLFOLDER' Return="ignore" Execute=
'deferred'
 ExeCommand='"C:\Program Files (x86)\Docsvision\Platform\Tools\ServerConsole.exe" /rs /n
NetstatSolution' />
<InstallExecuteSequence>
 <Custom Action='UnRegisterSnapIn' After="InstallInitialize">REMOVE~="All" AND (NOT
UPGRADINGPRODUCTCODE)</Custom>
</InstallExecuteSequence>
```

Чтобы серверная часть поддерживала обновление, добавляем:

```
<MajorUpgrade DowngradeErrorMessage="Более новая версия продукта [ProductName] уже
установлена." />
 <Upgrade Id="$(var.UpgradeCode)">
  <UpgradeVersion Minimum="$(var.ProductVersion)"
   IncludeMinimum="no"
   OnlyDetect="yes"
   Language="1049"
   Property="NEWPRODUCTFOUND" />

  <UpgradeVersion Minimum="$(var.RTMProductVersion)"
   IncludeMinimum="yes"
   Maximum="$(var.ProductVersion)"
   IncludeMaximum="no"
   Language="1049"
   Property="UPGRADEFOUND" />
 </Upgrade>
```

Если вы работаете с исходными кодами проекта, то пакет установки серверной части собирается последним.

Фактически данный шаг является заключительным: после сборки, пакет может быть установлен на сервер для всестороннего тестирования. Клиентская часть устанавливается автоматически при запуске Windows-клиента.

Обновление решения

Обновление решения, созданного на платформе Docsvision включает два этапа:

1. Обновление серверной части.
2. Обновление клиентской части.

Обновление серверной части

Чтобы серверная часть могла обновиться, что включает удаление старой версии и установку новой, достаточно изменить идентификатор продукта (атрибут "ID") в проекте пакета установки серверной части, либо использовать значение * в атрибута "ID" этого проекта.

Естественно в проект должны быть загружены новые компоненты, а также новый пакет установки клиентской части.

Обновление клиентской части

Обновление клиентской части выполняется автоматически при следующих условиях:

- Библиотека карточек имеет новую версию.
- Пакет установки клиентской части имеет новый *код продукта* и поддерживает обновление.

Чтобы условия выполнялись:

1. Увеличиваем номер версии библиотеки карточек в схеме метаданных библиотеки:

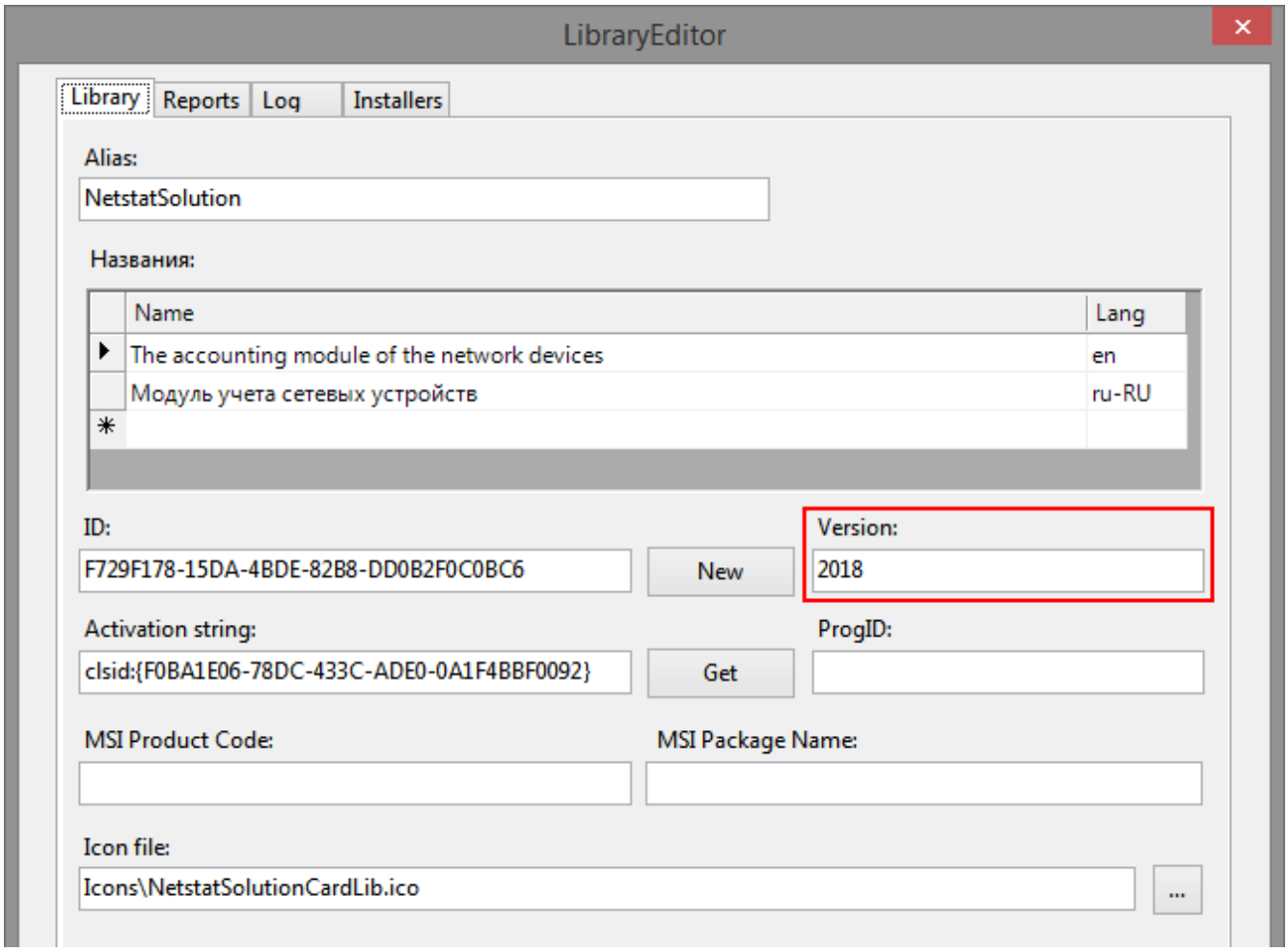


Рисунок 101. Назначение новой версии

2. Изменяем номер версии библиотеки в компоненте библиотеки (файл `dvver.h`) на номер, указанный в схеме метаданных:

```
#define VERSION_MAJOR                5
#define VERSION_MINOR                0
#define VERSION_BUILD             2018
#define VERSION_REVISION             0
#define VERSION_DISPLAY_MINOR        2
```

3. Устанавливаем в схеме метаданных библиотеки карточек новый `Код продукта`:

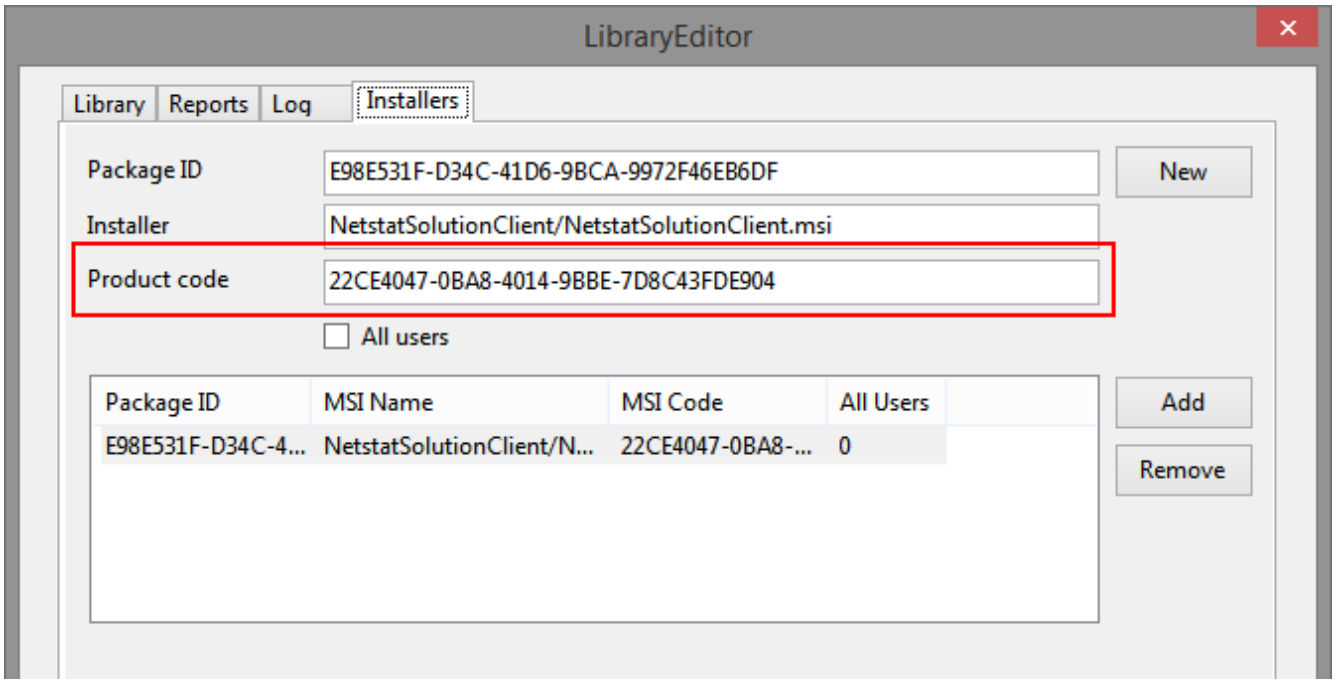


Рисунок 102. Установка нового кода продукта

4. Изменяем *код продукта* в проекте расширения *Консоли настройки*, в методе Execute (см. раздел Реализация стандартных интерфейсов) на значение, использованное в схеме метаданных:

```
public bool Execute()
{
 /* часть кода пропущена */

 cardLibConfig.RegisterPackage(
  "E98E531F-D34C-41D6-9BCA-9972F46EB6DF", ①
  "22CE4047-0BA8-4014-9BBE-7D8C43FDE904", ②
   Path.Combine(AssemblyFolder, "NetstatSolutionClient.msi"));

 return true;
}
```

① Идентификатор пакета установки.

② Код продукта.

5. Изменяем значение `Product Id` в проекте пакета установки клиентской части (файл `Product.wxs`), используя *код продукта* из схемы метаданных:

```
<Product Id="22CE4047-0BA8-4014-9BBE-7D8C43FDE904"
 Name="NetstatSolutionClient"
 Language="1049"
```

```
  Version="$(var.ProductVersion)"
  Manufacturer="Docsvision"
  UpgradeCode="$(var.UpgradeCode)">
```

В итоге, при запуске пакета установки, серверная часть будет обновлена до новой версии, а клиентская часть обновиться автоматически при запуске *Docsvision Windows-клиент*.

Лицензирование

Для решения может быть задействована одна из моделей лицензирования:

- Для партнеров Docsvision, которые осуществляют продажу своих решений через канал продаж Docsvision, доступно лицензирование средствами платформы с возможностью проверки наличия лицензии и ограничения по количеству подключений.

  При использовании такого варианта лицензия добавляется как дополнительная опция в лицензию сервера Docsvision. Проверка осуществляется, например, при обращении к карточке и запуске бизнес-процесса.

- Для разработчиков, создающих свои решения и распространяющих их самостоятельно, методика проверки лицензии выбирается самостоятельно.

В *Модуле учета сетевых устройств* используется несколько упрощенный второй вариант: лицензионный ключ вводится администратором в *Консоль настройки Docsvision*, а проверка лицензии выполняется при открытии карточки в Windows-клиенте.

Пример реализации расширения дерева папок Windows-клиента

```
using DocsVision.Platform.Extensibility;
using DocsVision.Platform.WinForms;
using System;
using System.Runtime.InteropServices;

namespace DocsVision.Test.FolderTreeExtension
{
 [ComVisible(true)]
 [Guid("6A0676EE-1DAF-4A59-B5EB-E0B5B4C175E1")]
 [ClassInterface(ClassInterfaceType.None)]
 public partial class Plugin : NavExtension, INavFolderTreeExtension
 {
```

```
  public Plugin()
  {
  }

  protected override NavExtensionTypes SupportedTypes
  {
   get
   {
    return NavExtensionTypes.FolderTree;
   }
  }

  protected override string GetExtensionName(NavExtensionTypes extensionType) ①
  {
   return "Расширение дерева папок {wincl}а";
   }

  public NavFolderInfo[] GetTreeExtensionFolders()
  {
   NavFolderInfo folder = new NavFolderInfo();
   folder.Flags = NavFolderTreeFlags.ShowInPersonalFolder;
   folder.FolderId = new Guid("FC849072-406E-4BB2-AA25-18F33E6AB252");
   folder.Location = NavFolderLocationFlags.AboveRegular;
   return new NavFolderInfo[] { folder };
  }
 }
}
```

① Название расширения.

Часто задаваемые вопросы

Каким образом получить родительский объект имея подчиненный?

Для этого предназначен метод `ObjectHelper.GetParent`.

Метод, к примеру, можно использовать для получения документа, которому принадлежит файл:

```
Document document = ObjectHelper.GetParent<Document>(documentFile); ①
```

① `documentFile` — файл (тип `DocumentFile`) искомого документа.

Как изменить значение в элементе управления в карточке

Для доступа к элементу управления из скрипта карточки, достаточно обратиться к методу `FindPropertyItem` базового класса карточки `BaseCardControl`, передав название элемента управления:

```
ICustomizableControl control = CardControl;

ILayoutPropertyItem layoutItem = control.FindPropertyItem<ILayoutPropertyItem>("Theme");
①

layoutItem.ControlValue = "Новое значение"; ②

layoutItem.Commit(); ③
```

① Получение элемента управления.

② Установка нового значения.

③ Сохранение.

Название элемента можно посмотреть в *Конструкторе разметок*.

Пример работы с Docsvision из Powershell

В приведенном ниже примере, создаётся скрипт, выполняющий подключение к серверу Docsvision, и получение номера из карточки Нумератора.

```
Add-Type -Path c:\DocsVision.Platform.dll
Add-Type -Path c:\DocsVision.Platform.ObjectManager.dll ①

$connectionString =
"ConnectAddress=http://docsvision.domain.com/DocsVision/StorageServer/StorageServerServic
e.asmx" ②
$sessionManager = [DocsVision.Platform.ObjectManager.SessionManager]::CreateInstance
($connectionString)
$userSession = $sessionManager.CreateSession()

[GUID]$numeratorCardID = "00000000-0000-0000-0000-000000000000" ③
[DocsVision.Platform.ObjectManager.SystemCards.NumeratorCard]$numeratorCard =
$userSession.CardManager.GetCard($numeratorCardID)
[GUID]$outId = [GUID]::Empty

$number = $numeratorCard.Zones[0].GetNumber([GUID]::Empty, 1, [ref] $outId); ④

write $number ⑤
```

① Требуется указать действительный путь к используемым библиотекам Docsvision.

② Требуется указать действительный адрес подключения к Docsvision.

③ Требуется передать действительный адрес Нумератора.

④ Получение номера из первой зоны нумерации.

⑤ Вывод полученного номера.

Аналогичная задача может быть решена без объектной модели — прямым обращением к сервису Docsvision

```
$serverUri =
"http://docsvision.domain.com/DocsVision/StorageServer/StorageServerService.asmx" ①

[GUID]$numeratorCardID = "00000000-0000-0000-0000-000000000000"
$zoneName = "Zone" ②

$result = 0
[GUID]$numberId = [GUID]::Empty
[GUID]$userID = [GUID]::Empty

$client = New-WebServiceProxy -Uri $serverUri"?WSDL" -UseDefaultCredential

[GUID]$sessionId = $client.SessionLogin("","", [ref]$result) ③

$number = $client.NumGetFirstFree($sessionId, $numeratorCardID, $zoneName, $userID, [ref
]$numberId) ④
```

① Адрес сервиса Docsvision.

② Идентификатор карточки нумератора и название зоны, в которых будет резервироваться номер.

③ Открытие сессии.

④ Получение номера.

Полное описание методов сервиса приведено в разделе Методы веб-сервиса Docsvision.

Как узнать идентификатор карточки

Узнать идентификатор существующий карточки можно несколькими способами:

- Посмотреть в Windows-клиенте — идентификатор карточки отображается в её свойствах.
- Используя утилиту Docsvision Explorer, входящей в Resource Kit — идентификатор карточки отображается в поле `Selected card ID`.

Получить идентификатор созданной карточки, используя объектную модель

Для получения идентификатора объекта (например, объект document типа `DocsVision.BackOffice.ObjectModel.Document`) можно воспользоваться методом `ObjectContext.GetObjectRef(ObjectBase)`:

```
Guid id_document = objectContext.GetObjectRef(document).Id;
или так (начиная с HO9):
Guid id_document = document.GetObjectId();
```

Корректный идентификатор нового объекта может быть получен только после сохранения объекта в *контексте объектов*.

Получить отображаемое имя текущего сотрудника

```
StaffEmployee currentEmployee = objectContext.GetService<IStaffService
>().GetCurrentEmployee();
string displayName = currentEmployee.DisplayName;
```

Получить карточку сотрудника по его идентификатору (id_empl)

```
StaffEmployee staffEmployee = staffService.Get(new Guid("id_empl"));
```

Считывание данных с элемента управления типа коллекция, например "Сотрудники"

Для сотрудников, в `ControlValue` соответствующего элемента управления содержится коллекция идентификатором выбранных сотрудников — `IEnumerable<Guid>`. Для записи, в `ControlValue` передается массив таких идентификаторов (`Guid[]`).

Доступ к значению поля

```
int? result = userSession.CardManager.GetCardData(cardID).Sections[sectionID].FirstRow
.GetInt32(fieldAlias); ①
```

① `cardID` — идентификатор карточки; `sectionID` — идентификатор секции; `fieldAlias` — псевдоним поля.

С точки зрения доступа, динамические поля не отличаются от статических,

поэтому получение значения такого поля может выглядеть следующим образом

```
Guid cardID = new Guid("00000000-0000-0000-0000-000000000000"); ①

string sectionAlias = "DynSection"; ②

string fieldAlias = "DynField"; ③

CardType type = userSession.CardManager.GetCardData(cardID).Type; ④

Guid sectionID = type.Sections[sectionAlias].Id; ⑤

int? result = userSession.CardManager.GetCardData(cardID).Sections[sectionID].FirstRow
.GetInt32(fieldAlias); ⑥

Document document = objectContext.GetObject<Document>(cardID); ⑦
result = (((BaseCardSectionRow)document.GetSection(sectionID)[0])[fieldAlias] as int?);
```

① Идентификатор карточки.

② Название динамической секции.

③ Название динамического поля.

④ Получение типа карточки.

⑤ Получение идентификатора.

⑥ Получение значения поля через старую объектную модель.

⑦ Или даже так, если через новую объектную модель.

Как узнать идентификатор определённого вида карточки

Воспользуйтесь утилитой Docsvision Explorer, входящей в Resource Kit.

1. Нажать **Card types**. В открывшемся окне определить идентификатор типа карточки (идентификатор в "Selected type ID"), также идентификатор можно найти в разделе /dv6/schemas/dv6/[Описание полей стандартной карточки] (идентификатор указан вначале описания конкретной карточки).
2. Нажать **Cards**. В списке типов карточек выбрать "Справочник видов карточек", после чего нажать **Search**.
3. Выбрать найденный справочник (будет в единственном экземпляре). В справочнике нужно найти строку с найденным ранее идентификатором типа карточки (по содержимому поля "CardTypeId").

4. Нажать **Enter section**.
5. Найти в дереве видов карточек нужный вид по названию. В значении поля RowID будет искомый идентификатор вида карточки.

Получение сертификата сотрудника

Сертификаты используются при подписании, шифровании и расшифровке данных карточек, а также при верификации установленных подписей. Сертификат сотрудника может быть получен несколькими способами:

1. Вне контекста Windows-клиента:

```
IUserProfileCardService iUserProfileCardService = objectContext.GetService
<IUserProfileCardService>();
X509Certificate2 certificate = iUserProfileCardService.GetCertificate
(@"DOMAIN\IvanovII");
```

Для получения сертификата сотрудника с использованием метода `GetCertificate`, сертификат должен быть указан в соответствующем поле (см. Руководство по настройке), в /dv6/backoffice/6.1/desdirs/staff/employees/main-tab/#certificate[Справочнике сотрудников].

Другой вариант — получения сертификата из хранилища сертификатов ОС Windows. В примере получаем первый сертификат с закрытым ключом.

```
X509Certificate2 GetCertificate()
{
 X509Store store = new X509Store(StoreName.My, StoreLocation.CurrentUser);
 store.Open(OpenFlags.OpenExistingOnly);
 foreach (var item in store.Certificates)
 {
  if (item.HasPrivateKey) return item;
 }
 return null;
}
```

2. В контексте Windows-клиента (например, в скрипте карточки)

```
bool cancel = false;
X509Certificate2 certificate = DocsVision.BackOffice.WinForms.Controls
.SelectCertificateForm.SelectCertificate(ref cancel, objectContext);
```

```
if(cancel) return;
```

В данном случае, сотруднику будет выведено окно выбора сертификата.

Библиотека классов

Библиотека классов Docsvision содержит описание объектов и методов предоставляемых API Docsvision.

Руководство по разработке находится в процессе наполнения, т.е. библиотека классов не содержит описание всех сущностей, возможны некоторые неточности, которые будут исправлены в будущих версиях.

API Docsvision (версии 5.4) представлен объектами, которые были унаследованы от предыдущих версий платформы (4.X и раньше) и объектами созданными для 5 версии, которые в основном определены в пространстве имён `DocsVision.BackOffice.ObjectModel.ObjectModel`, на описание сущностей которого был сделан основной упор.

Объектная модель 5.X работает поверх 4.X, поэтому если Вам не достает методов и свойств новой версии, попробуйте обратиться к старой. Старая объектная модель — это в основном `DocsVision.Platform.ObjectManager`, при том что многие классы, например отвечающие за работу с объектами безопасности, могут быть реализованы в других сборках и пространствах имён.

Пространства имён

Пространство имён	Описание
`DocsVision`	Основное пространство имён, содержащее все сущности объектной модели Docsvision. Включает подпространства имён в которых реализованы методы доступа к карточкам и бизнес-процессам, а также представлены функции управления доступом в рамках трёх моделей безопасности.

Docsvision — пространство имён

Основное пространство имён, содержащее все сущности объектной модели Docsvision. Включает подпространства имён в которых реализованы методы

доступа к карточкам и бизнес-процессам, а также представлены функции управления доступом в рамках трёх моделей безопасности.

Пространства имён

Пространство имён	Описание
DocsVision.ApprovalDesigner	Пространство имён DocsVision.ApprovalDesigner содержит типы и методы приложения *Конструктор согласований*.
DocsVision.BackOffice	Пространство имён DocsVision.BackOffice содержит типы представляющие объектную модель библиотеки Базовые объекты.
DocsVision.DocumentsManagement	Пространство имён DocsVision.DocumentsManagement содержит описание сервиса для отправки согласования.
DocsVision.Platform	В пространстве имён DocsVision.Platform содержатся классы реализующие API уровня ядра Docsvision.
DocsVision.Workflow	Пространство имён DocsVision.Workflow реализует доступ к подсистеме управления бизнес-процессами, а также содержит средства организации взаимодействия платформы и различных внешних систем.

DocsVision.ApprovalDesigner — пространство имён

Пространство имён DocsVision.ApprovalDesigner содержит типы и методы приложения *Конструктор согласований*.

Пространства имён

Пространство имён	Описание
DocsVision.ApprovalDesigner.ObjectModel	Содержит типы объектной модели модуля Конструктор согласований.

Заметки

Для возможности получения или создания сущностей пространства имён DocsVision.ApprovalDesigner в контексте объектов помимо стандартных должны быть зарегистрированы дополнительные фабрики из сборки DocsVision.ApprovalDesigner.ObjectModel.dll:

- Фабрика преобразователей данных ApprovalDesignerMapperFactory.
- Фабрика сервисов ApprovalDesignerServiceFactory.

```
using DocsVision.ApprovalDesigner.ObjectModel.Mapping;
// ...

var serviceFactoryRegistry = objectContext.GetService<IServiceFactoryRegistry>();
serviceFactoryRegistry.RegisterFactory(typeof(ApprovalDesignerServiceFactory));

var mapperFactoryRegistry = objectContext.GetService<IObjectMapperFactoryRegistry>();
mapperFactoryRegistry.RegisterFactory(typeof(ApprovalDesignerMapperFactory));
```

DocsVision.ApprovalDesigner.ObjectModel — пространство имён

Содержит типы объектной модели приложения Конструктор согласований.

Пространства имён

Пространство имён	Описание
DocsVision.ApprovalDesigner.ObjectModel.Services	Пространство имён DocsVision.ApprovalDesigner.ObjectModel.Services содержит методы публичной функциональности Конструктора согласований.

Классы

Класс	Описание
ApprovalFilesCreator	Класс ApprovalFilesCreator содержит статический метод для добавления файлов в карточку Согласование.
ApprovalPath	Объектная модель карточки Маршрут согласования

Класс	Описание
ApprovalPathMainInfo	Содержит основную информацию карточки Маршрут согласования
ApprovalPathRoadMap	Представляет карту этапов Маршрута согласования.
ApprovalPathState	Представляет состояние документа в Маршруте согласования.
ApprovalResult	Представляет результат согласования.
ApprovalResultMainInfo	Содержит основную информацию карточки *Результат согласования*.
ApprovalStage	Этап согласования в карте этапов Маршрута согласования.
ApprovalStageAdditionalSettings	Содержит дополнительные настройки этапа согласования.
ApprovalStageApprover	Представляет согласующего этапа согласования.
ApprovalStageDecision	Этап согласования в карте этапов Маршрута согласования.
ApprovalStageMainInfo	Содержит основную информацию карточки Этап согласования.
ApprovalStageState	Представляет состояние документа в матрице состояний этапа согласования.
ApprovalStageTaskSettings	Предоставляет настройки задания участников согласования.

Интерфейсы

Интерфейс	Описание
ISearchWordsManager	Содержит метод, возвращающий описание поискового слова по его идентификатору.

Перечисления

Перечисление	Описание
AdditionSemantics	Семантики завершения задания при добавлении согласующих.
ApprovalRejectionCase	Действия в случае отказа.
ApprovalStageMode	Режимы маршрутизации согласования.
ApprovalType	Тип маршрутизации согласования.
DecisionSemantics	Семантики завершения этапа согласования.
FileVersionType	Возможные типы версии файла.
TaskDecisionSemantics	Семантики завершения задания согласования.
VersionTreeLevel	Уровни дерева версий.

DocsVision.ApprovalDesigner.ObjectModel.Services — пространство имён

Пространство имён DocsVision.ApprovalDesigner.ObjectModel.Services содержит методы публичной функциональности Конструктора согласований.

Классы

Класс	Описание
ApprovalStageEventHandlerService	Обрабатывает события этапа согласования.
ApprovalStageService	Класс содержит методы для работы с этапом согласования.

Интерфейсы

Интерфейс	Описание
IApprovalPathService	Сервис IApprovalPathService предоставляет методы для работы с маршрутами согласования.
IApprovalResultService	Сервис IApprovalResultService предоставляет метод для добавления результата согласования.

Интерфейс	Описание
IApprovalService	Сервис IApprovalService предоставляет методы для добавления согласующих в этап согласования.
IApprovalStageService	Сервис IApprovalStageService предоставляет методы для работы с этапами согласования.
ISearchWordResolverService	Сервис ISearchWordResolverService предоставляет метод получения согласующих по поисковому слову.

ApprovalStageEventHandlerService — класс

Обрабатывает события этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public class ApprovalStageEventHandlerService : ApprovalDesignerEventHandlerService,
IApprovalStageEventHandlerService
```

Методы

Имя	Описание
GetApprovalStageService(ApprovalStage)	Получает специальный сервис логики этапа согласования.
GetId() ⇒ ServiceId	Получает идентификатор специального сервиса логики этапа.
GetHandlersInfo()	Получает информацию об обработчиках событий.
GetTargetObjectType()	Получает тип этапа согласования.
OnStartRequested(ApprovalStage, ApprovalStageStartRequestedEventArgs)	Вызывается перед запуском согласования.

Имя	Описание
`OnNextTaskRequested(ApprovalStage, ApprovalStageNextTaskRequestedEventArgs)`	Вызывается перед запуском следующего задания.
`OnCompletionRequested(ApprovalStage, ApprovalStageCompletionRequestedEventArgs)`	Вызывается при запросе на завершение этапа.
`OnCancelRequested(ApprovalStage, ApprovalStageCancelRequestedEventArgs)`	Вызывается при запросе на отмену этапа.
`OnReturnRequested(ApprovalStage, ApprovalStageReturnRequestedEventArgs)`	Вызывается при запросе на возврат этапа.
`OnCardStateChanged(ApprovalStage, OnStateChangedEventArgs)`	Вызывается при изменении состояния карточки.

Примечание

Если у передаваемой на обработку карточки этапа `ApprovalStage` в `MainInfo.ServiceTypeName` прописан собственный тип сервиса-обработчика, создаётся его экземпляр и используется его бизнес-логика для работы внутри `ApprovalStageEventHandlerService`.

ApprovalStageService — класс

Класс содержит методы для работы с этапом согласования.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel.Services`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public class ApprovalStageService : ContextService, IApprovalStageService
```

Методы

Имя	Описание
`AddSelectedApprover(ApprovalStage, Guid)`	Добавляет выбранных согласующих из поискового слова.
`AddSelectedApprover(ApprovalStage, StaffEmployee)`	Добавляет выбранного согласующего из справочника сотрудников.
`ApprovalStageApprover AddSelectedApprover(ApprovalStage, StaffUnit)`	Добавляет выбранных согласующих из подразделения.
`AddSelectedApprover(ApprovalStage, StaffGroup)`	Добавляет выбранных согласующих из группы.
`AddSelectedApprover(ApprovalStage, StaffRole)`	Добавляет выбранных согласующих из роли.
`AddCurrentApprover(ApprovalStage, StaffEmployee)`	Добавляет текущего согласующего.
`InsertCurrentApproverAfter(ApprovalStage, ApprovalStageCurrentApprover, StaffEmployee)`	Добавляет текущего согласующего после другого согласующего.
`AddAdditionalApprover(ApprovalStage, StaffEmployee, StaffEmployee)`	Добавляет дополнительного согласующего.
`ReorderSelectedApprovers(ObjectCollection<ApprovalStageApprover>)`	Изменяет порядок выбранных согласующих.
`AddDecision(ApprovalStage)`	Добавляет новое решение.
`ReorderDecisions(ObjectCollection<ApprovalStageDecision>)`	Изменяет порядок решений.
`AddDecisionLocalization(ApprovalStageDecision, int, string)`	Локализация решений этапа согласования.

Имя	Описание
`AddMainInfoNameLocalization(ApprovalStage, int, string)`	Локализация решений этапа согласования.
`AddState(ApprovalStage)`	Добавляет новое состояние.
`CreateApprover(StaffEmployee)`	Создаёт согласующего по указанному сотруднику.
`CopyApproversFromDocument(ApprovalStage, Document)`	Копирует согласующих из документа в этап.
`ComputeApproversForStage(ApprovalStage, Reconcile, ApprovalPath)`	Вычисляет согласующих при запуске этапа.
`GetStageApprovers(ApprovalStage, Document)`	Получает согласующих этапа.
`ProcessExcludedApprovers(ApprovalStage, Document, IEnumerable<ApprovalStageApprover>)`	Позволяет исключать сотрудников из этапа. Метод исключает сотрудников, соответствующих указанным ролям, группам и поисковым словам.
`GetPossibleSemantics(ApprovalType)`	Получает набор возможных семантик решений в зависимости от типа согласования.
`StartStage(ApprovalStage, Reconcile)`	Начало работы этапа согласования.
`OnStageStarted(ApprovalStage, ApprovalPath, Reconcile)`	Вызывается после запуска этапа.
`CreateNextTask(ApprovalStage, Reconcile)`	Создание задания согласования.
`OnTaskCreated(ApprovalStage, ApprovalPath, Reconcile, Task)`	Вызывается после создания задания, но до его запуска.
`OnTaskStarted(ApprovalStage, ApprovalPath, Reconcile, Task)`	Вызывается после запуска задания.

Имя	Описание
`ChangeDocumentState(ApprovalStage, Document, DecisionSemantics)`	Изменяет состояние документа.
`OnDocumentStateChanging(ApprovalStage, Document, DecisionSemantics)`	Вызывается в `ChangeDocumentState` перед сменой состояния документа и передаёт вычисленное новое состояние документа, ожидая, что метод может вернуть другое состояние документа, которое и нужно установить.
`OnDocumentStateChanged(ApprovalStage, Document, DecisionSemantics)`	Вызывается в `ChangeDocumentState` после смены состояния документа.
`CompleteTask(ApprovalStage, ApprovalStageCurrentApprover)`	Завершает задание.
`OnTaskCompleting(ApprovalStage, Reconcile, ApprovalStageCurrentApprover)`	Вызывается в `CompleteTask` до отработки основной логики метода.
`OnTaskCompleted(ApprovalStage, Reconcile, ApprovalStageCurrentApprover)`	Вызывается в `CompleteTask` после отработки основной логики метода.
`RecallTask(ApprovalStage, ApprovalStageCurrentApprover)`	Отзывает задачу.
`CompleteStage(ApprovalStage, ApprovalPath, Reconcile)`	Завершает этап.
`SkipStage(ApprovalStage, ApprovalPath, Reconcile, DecisionSemantics)`	Вызывается после пропуска этапа.
`OnStageCompleting(ApprovalStage, ApprovalPath, Reconcile)`	Вызывается в `CompleteStage` до отработки основной логики метода.

Имя	Описание
`OnStageCompleted(ApprovalStage, ApprovalPath, Reconcile)`	Вызывается в `CompleteStage` после отработки основной логики метода.
`CancelStage(ApprovalStage, ApprovalPath, Reconcile)`	Отменяет этап.
`OnStageCanceling(ApprovalStage, ApprovalPath, Reconcile)`	Вызывается в `CancelStage` до отработки основной логики метода.
`OnStageCanceled(ApprovalStage, ApprovalPath, Reconcile)`	Вызывается в `CancelStage` после отработки основной логики метода.
`ReturnStage(ApprovalStage, ApprovalPath, Reconcile)`	Возвращает этап.
`OnStageReturning(ApprovalStage, ApprovalPath, Reconcile)`	Вызывается в `ReturnStage` до отработки основной логики метода.
`OnStageReturned(ApprovalStage, ApprovalPath, Reconcile)`	Вызывается в `ReturnStage` после отработки основной логики метода.
`CompleteInitiatorTask(ApprovalStage, ApprovalStageCurrentApprover)`	Завершает задание инициатора.
`OnCompletingInitiatorTask(ApprovalStage, Reconcile, ApprovalStageCurrentApprover)`	Вызывается в `CompleteInitiatorTask` до отработки основной логики метода.
`OnCompletedInitiatorTask(ApprovalStage, Reconcile, ApprovalStageCurrentApprover)`	Вызывается в `CompleteInitiatorTask` после отработки основной логики метода.

IApprovalPathService — интерфейс

Сервис `IApprovalPathService` предоставляет методы для работы с маршрутами согласования.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel.Services`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public interface IApprovalPathService
```

Методы

Имя	Описание
AddDocumentState(ApprovalPath, KindsCardKind, StatesState)	Добавляет состояние в маршрут согласования
AddRoadMap(ApprovalPath)	Добавляет этап в маршрут согласования.

IApprovalPathService.AddDocumentState — метод (ApprovalPath, KindsCardKind, StatesState)

Добавляет итоговое состояние документа в Маршрут согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalPathState AddDocumentState(ApprovalPath approvalPath, KindsCardKind kind,
StatesState state)
```

Параметры

approvalPath

Тип: /dv6/programmer/dv6/ApprovalDesigner/ObjectModel/ApprovalPath_CL/[ApprovalPath]

Маршрут согласования

kind

Тип: KindsCardKind

Вид карточки Документ

state

Тип: StatesState

Состояние документа

Возвращаемое значение

Тип: ApprovalPathState

Добавленное состояние документа

IApprovalPathService.AddRoadMap — метод (ApprovalPath)

Добавляет этап в маршрут согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalPathRoadMap AddRoadMap(ApprovalPath approvalPath)
```

Параметры

approvalPath

Тип: ApprovalPath

Маршрут согласования

Возвращаемое значение

Тип: ApprovalPathRoadMap

Добавленный этап

IApprovalResultService — интерфейс

Сервис IApprovalResultService предоставляет метод для добавления результата согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public interface IApprovalResultService
```

Методы

Имя	Описание
CreateApprovalResult	Создаёт новый результат согласования.

IApprovalResultService.CreateApprovalResult — метод

Создаёт новый результат согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalResult CreateApprovalResult()
```

Возвращаемое значение

Тип: ApprovalResult

Результат согласования

IApprovalService — интерфейс

Сервис IApprovalService предоставляет методы для добавления согласующих в этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public interface IApprovalService
```

Методы

Имя	Описание
AddApprover(ApprovalStage, StaffEmployee)	Добавляет согласующего в этап согласования.
AddApprover(ApprovalStage, IEnumerable<StaffEmployee>)	Добавляет несколько согласующих в этап согласования.

IApprovalService.AddApprover — метод (ApprovalStage, StaffEmployee)

Добавляет согласующего в этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void AddApprover(ApprovalStage stage, StaffEmployee employee)
```

Параметры

stage

Тип: ApprovalStage

Этап согласования

employee

Тип: StaffEmployee

Добавляемый согласующий сотрудник

IApprovalService.AddApprovers — метод (ApprovalStage, IEnumerable<StaffEmployee>)

Добавляет несколько согласующих в этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void AddApprovers(ApprovalStage stage, IEnumerable<StaffEmployee> employees)
```

Параметры

stage

Тип: ApprovalStage

Этап согласования

employees

Тип: System.Collections.Generic.IEnumerable<StaffEmployee>

Добавляемые согласующие сотрудники. Тип элемента — `StaffEmployee`

IApprovalStageService — интерфейс

Сервис `IApprovalStageService` предоставляет методы для работы с этапами согласования.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel.Services`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public interface IApprovalStageService
```

Методы

Имя	Описание
`AddSelectedApprover(ApprovalStage, Guid)`	Добавляет согласующего по поисковому слову в этап согласования.
`AddSelectedApprover(ApprovalStage, StaffEmployee)`	Добавляет согласующего сотрудника в этап согласования.
`AddSelectedApprover(ApprovalStage, StaffUnit)`	Добавляет согласующих из подразделения в этап согласования.
`AddSelectedApprover(ApprovalStage, StaffGroup)`	Добавляет согласующих из группы в этап согласования.
`AddSelectedApprover(ApprovalStage, StaffRole)`	Добавляет согласующих из роли в этап согласования.
`AddCurrentApprover(ApprovalStage, StaffEmployee)`	Добавляет текущего сотрудника в этап согласования.
`InsertCurrentApproverAfter(ApprovalStage, ApprovalStageCurrentApprover, StaffEmployee)`	Вставляет текущего сотрудника после другого сотрудника.
`AddAdditionalApprover(ApprovalStage approvalStage, StaffEmployee employee, StaffEmployee addedBy)`	Добавляет дополнительного согласующего.

Имя	Описание
`ReorderSelectedApprovers(ObjectCollection<ApprovalStageApprover>)`	Сортирует список, содержащий согласующих, по порядковому номеру согласующего.
`AddDecision(ApprovalStage)`	Добавляет решение для этапа маршрута согласования.
`ReorderDecisions(ObjectCollection<ApprovalStageDecision>)`	Сортирует список, содержащий решения по согласованию, по порядковому номеру решения.
`AddDecisionLocalization(ApprovalStageDecision, int, string)`	Добавляет новую локализацию решения.
`AddMainInfoNameLocalization(ApprovalStage, int, string)`	Добавляет новую локализацию названия основной информации.
`AddState(ApprovalStage)`	Добавляет состояние для этапа согласования.
`CreateApprover(StaffEmployee)`	Создаёт согласующего без добавления в этап согласования.
`CopyApproversFromDocument(ApprovalStage, Document)`	Получает согласующих из согласуемого документа.
`GetStageApprovers(ApprovalStage approvalStage, Document document)`	Возвращает согласующих этапа в результате вызова `CopyApproversFromDocument`.
`ProcessExcludedApprovers(ApprovalStage approvalStage, Document document, IEnumerable<ApprovalStageApprover> approvers)`	Исключает согласующих.
`GetPossibleSemantics(ApprovalType approvalStageType)`	Получает список возможных семантик решения в зависимости от типа согласования.
`StartStage(ApprovalStage approvalStage, Reconcile reconcileCard)`	Начальный этап согласования.

Имя	Описание
`CreateNextTask(ApprovalStage approvalStage, Reconcile reconcileCard)`	Создаёт следующее задание.
`ChangeDocumentState(ApprovalStage approvalStage, Document document, DecisionSemantics decisionSemantics)`	Изменяет состояние документа.
`CompleteTask(ApprovalStage approvalStage, ApprovalStageCurrentApprover approverWithTask)`	Завершает задание.
`RecallTask(ApprovalStage approvalStage, ApprovalStageCurrentApprover approverWithTask)`	Отзывает задание.
`CompleteStage(ApprovalStage approvalStage, ApprovalPath approvalPath, Reconcile reconcileCard)`	Завершает этап.
`SkipStage(ApprovalStage approvalStage, ApprovalPath approvalPath, Reconcile reconcileCard, DecisionSemantics stageDecision)`	Пропускает этап.
`CancelStage(ApprovalStage approvalStage, ApprovalPath approvalPath, Reconcile reconcileCard)`	Отменяет этап.
`ReturnStage(ApprovalStage approvalStage, ApprovalPath approvalPath, Reconcile reconcileCard)`	Повторяет этап.

Имя	Описание
CompleteInitiatorTask(ApprovalStage approvalStage, ApprovalStageCurrentApprover approverWithInitiatorTask)	Завершает задание инициатора.

IApprovalStageService.AddAdditionalApprover — метод (ApprovalStage, StaffEmployee, StaffEmployee)

Добавляет дополнительного согласующего.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageAdditionalApprover AddAdditionalApprover(ApprovalStage approvalStage,
StaffEmployee employee, StaffEmployee addedBy)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования

employee

Тип: StaffEmployee

Согласующий сотрудник

addedBy

Тип: StaffEmployee

Сотрудник, добавивший дополнительного согласующего.

Возвращаемое значение

Тип: ApprovalStageAdditionalApprover

Дополнительный согласующий

IApprovalStageService.AddCurrentApprover — метод (ApprovalStage, StaffEmployee)

Добавляет текущего сотрудника в этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageApprover AddSelectedApprover(ApprovalStage approvalStage, StaffRole role)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования

employee

Тип: StaffRole

Сотрудник

Возвращаемое значение

Тип: ApprovalStageCurrentApprover

Текущий сотрудник на этапе согласования

IApprovalStageService.AddDecision — метод (ApprovalStage)

Добавляет решение для этапа маршрута согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageDecision AddDecision(ApprovalStage approvalStage)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап маршрута согласования

Возвращаемое значение

Тип: ApprovalStageDecision

Решение

AddDecisionLocalization — метод (ApprovalStageDecision, int, string)

Добавляет новую локализацию решения.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageDecisionLocalization AddDecisionLocalization(ApprovalStageDecision decision,
int localeId, string name)
```

Параметры

decision

Тип: ApprovalStageDecision

Вариант решения для завершения этапа согласования

localeId

Тип: System.Int32

Номер локали

name

Тип: System.String

Точное и полное название локали

Возвращаемое значение

Тип: ApprovalStageDecisionLocalization

Локализацию решения

AddMainInfoNameLocalization — метод (ApprovalStage, int, string)

Добавляет новую локализацию названия основной информации.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageMainInfoLocalization AddMainInfoNameLocalization(ApprovalStage
approvalStage, int localeId, string name)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

int

Тип: System.Int32

Номер локали

string

Тип: System.String

Точное и полное название локали

Возвращаемое значение

Тип: ApprovalStageDecisionLocalization

Локализация названия основной информации

IApprovalStageService.AddSelectedApprover — метод (ApprovalStage, Guid)

Добавляет согласующего по поисковому слову в этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageApprover AddSelectedApprover(ApprovalStage approvalStage, Guid searchWord)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования

searchWord

Тип: System.Guid

Идентификатор поискового слова, по которому будут выбираться согласующие

Возвращаемое значение

Тип: ApprovalStageApprover

Согласующий этапа

IApprovalStageService.AddSelectedApprover — метод (ApprovalStage, StaffEmployee)

Добавляет согласующего сотрудника в этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageApprover AddSelectedApprover(ApprovalStage approvalStage, StaffEmployee
employee)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования

employee

Тип: StaffEmployee

Согласующий сотрудник

Возвращаемое значение

Тип: ApprovalStageApprover

Согласующий этапа

IApprovalStageService.AddSelectedApprover — метод (ApprovalStage, StaffGroup)

Добавляет согласующих из группы в этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageApprover AddSelectedApprover(ApprovalStage approvalStage, StaffGroup group)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования

group

Тип: StaffGroup

Группа согласующих

Возвращаемое значение

Тип: ApprovalStageApprover

Согласующий этапа

IApprovalStageService.AddSelectedApprover — метод (ApprovalStage, StaffRole)

Добавляет согласующих из роли в этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageApprover AddSelectedApprover(ApprovalStage approvalStage, StaffRole role)
```

Параметры

approvalStage

Тип: `ApprovalStage`

Этап согласования

role

Тип: `StaffRole`

Роль согласующих

Возвращаемое значение

Тип: `ApprovalStageApprover`

Согласующий этапа

IApprovalStageService.AddSelectedApprover — метод (ApprovalStage, StaffUnit)

Добавляет согласующих из подразделения в этап согласования.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel.Services`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
ApprovalStageApprover AddSelectedApprover(ApprovalStage approvalStage, StaffUnit unit)
```

Параметры

approvalStage

Тип: `ApprovalStage`

Этап согласования

unit

Тип: `StaffUnit`

Подразделение согласующих

Возвращаемое значение

Тип: `ApprovalStageApprover`

Согласующий этапа

IApprovalStageService.AddState — метод (ApprovalStage)

Добавляет состояние для этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageState AddState(ApprovalStage approvalStage)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования

Возвращаемое значение

Тип: ApprovalStageState

Состояние этапа согласования

SkipStage — метод (ApprovalStage, ApprovalPath, Reconcile)

Отменяет этап.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void CancelStage(ApprovalStage approvalStage, ApprovalPath approvalPath, Reconcile
reconcileCard)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

approvalPath

Тип:

/dv6/programmer/dv6/ApprovalDesigner/ObjectModel/ApprovalPath_CL/[ApprovalPath]

Маршрут согласования

reconcileCard

Тип: Reconcile

Карточка согласования

CreateNextTask — метод (ApprovalStage, Document, DecisionSemantics)

Изменяет состояние документа.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void ChangeDocumentState(ApprovalStage approvalStage, Document document,
DecisionSemantics decisionSemantics)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

document

Тип: Document

Документ

decisionSemantics

Тип: DecisionSemantics

Семантика решения

SkipStage — метод (ApprovalStage, ApprovalStageCurrentApprover)

Завершает задание инициатора.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services

- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void CompleteInitiatorTask(ApprovalStage approvalStage, ApprovalStageCurrentApprover approverWithInitiatorTask)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

approvalPath

Тип: /dv6/programmer/dv6/ApprovalDesigner/ObjectModel/ApprovalPath_CL/[ApprovalPath]

Маршрут согласования

approverWithInitiatorTask

Тип: ApprovalStageCurrentApprover

Текущий сотрудник на этапе согласования

CompleteStage — метод (ApprovalStage, ApprovalPath, Reconcile)

Завершает этап.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void CompleteStage(ApprovalStage approvalStage, ApprovalPath approvalPath, Reconcile reconcileCard)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

approvalPath

Тип: /dv6/programmer/dv6/ApprovalDesigner/ObjectModel/ApprovalPath_CL/[ApprovalPath]

Маршрут согласования

reconcileCard

Тип: Reconcile

Карточка согласования

CompleteTask — метод (ApprovalStage, ApprovalStageCurrentApprover)

Завершает задание.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void CompleteTask(ApprovalStage approvalStage, ApprovalStageCurrentApprover
approverWithTask)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

approverWithTask

Тип: ApprovalStageCurrentApprover

Текущий сотрудник на этапе согласования

IApprovalStageService.CopyApproversFromDocument — метод (ApprovalStage, Document)

Получает согласующих из согласуемого документа.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
IEnumerable<ApprovalStageApprover> CopyApproversFromDocument(ApprovalStage approvalStage,
Document document)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования, в который копируются согласующие

document

Тип: Document

Документ

Возвращаемое значение

Тип: ApprovalStageApprover

Добавленные согласующие

Заметки

Метод CopyApproversFromDocument возвращает согласующих, полученных из документа document. Согласующие выбираются из поля секции, определённого в настройках этапа согласования approvalStage, в MainInfo.ApproversField.

IApprovalStageService.CreateApprover — метод (StaffEmployee)

Создаёт согласующего без добавления в этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageApprover CreateApprover(StaffEmployee employee)
```

Параметры

employee

Тип: StaffEmployee

Согласующий сотрудник

Возвращаемое значение

Тип: ApprovalStageApprover

Согласующий

CreateNextTask — метод (ApprovalStage, Reconcile)

Создаёт следующее задание.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void CreateNextTask(ApprovalStage approvalStage, Reconcile reconcileCard)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

reconcileCard

Тип: Reconcile

Карточка согласования

GetPossibleSemantics — метод (ApprovalType)

Получает список возможных семантик решения в зависимости от типа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
IEnumerable<DecisionSemantics> GetPossibleSemantics(ApprovalType approvalStageType)
```

Параметры

approvalStageType

Тип: ApprovalType

Тип этапа согласования

Возвращаемое значение

Тип: DecisionSemantics

Семантики завершения этапа согласования

IApprovalStageService.InsertCurrentApproverAfter — метод (ApprovalStage,ApprovalStageCurrentApprover, StaffEmployee)

Вставляет текущего сотрудника после другого сотрудника.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
ApprovalStageCurrentApprover InsertCurrentApproverAfter(ApprovalStage approvalStage,
ApprovalStageCurrentApprover insertAfter, StaffEmployee employee)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования

insertAfter

Тип: ApprovalStageCurrentApprover

Текущий сотрудник на этапе согласования

employee

Тип: StaffEmployee

Согласующий сотрудник

Возвращаемое значение

Тип: ApprovalStageCurrentApprover

Текущий сотрудник на этапе согласования

ProcessExcludedApprovers — метод (ApprovalStage, Document, IEnumerable<ApprovalStageApprover>)

Исключает согласующих.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
IEnumerable<ApprovalStageApprover> ProcessExcludedApprovers(ApprovalStage approvalStage,
Document document, IEnumerable<ApprovalStageApprover> approvers)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

document

Тип: Document

Документ

approvers

Тип: ApprovalStageApprover

Согласующие

Возвращаемое значение

Тип: ApprovalStageApprover

Согласующий

RecallTask — метод (ApprovalStage, ApprovalStageCurrentApprover)

Отзывает задание.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void RecallTask(ApprovalStage approvalStage, ApprovalStageCurrentApprover
approverWithTask)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

approverWithTask

Тип: ApprovalStageCurrentApprover

Текущий сотрудник на этапе согласования

ApprovalDesigner:ObjectModel/Services/GetStageApprovers_MT.adoc

GetStageApprovers — метод (ApprovalStage, Document)

Возвращает согласующих этапа в результате вызова CopyApproversFromDocument.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
IEnumerable<ApprovalStageApprover> GetStageApprovers(ApprovalStage approvalStage,
Document document)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования

document

Тип: Document

Документ

Возвращаемое значение

Тип: ApprovalStageApprover

IApprovalStageService.ReorderDecisions — метод (ObjectCollection<ApprovalStageDecision>)

Сортирует список, содержащий решения по согласованию, по порядковому номеру решения.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void ReorderDecisions(ObjectCollection<ApprovalStageDecision> decisions)
```

Параметры

decisions

Тип: ObjectCollection<ApprovalStageDecision>

Решения по согласованию. Тип элемента — ApprovalStageDecision

Заметки

Метод ReorderDecisions просто сортирует коллекцию decisions по значению Order элементов коллекции.

IApprovalStageService.ReorderSelectedApprovers — метод (ObjectCollection<ApprovalStageApprover>)

Сортирует список, содержащий согласующих, по порядковому номеру согласующего.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void ReorderSelectedApprovers(ObjectCollection<ApprovalStageApprover> selectedApprovers)
```

Параметры

selectedApprovers

Тип: ObjectCollection<ApprovalStageApprover>

Согласующие. Тип элемента — ApprovalStageApprover

Заметки

Метод ReorderSelectedApprovers просто сортирует коллекцию selectedApprovers по значению Order элементов коллекции.

SkipStage — метод (ApprovalStage, ApprovalPath, Reconcile)

Повторяет этап.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void ReturnStage(ApprovalStage approvalStage, ApprovalPath approvalPath, Reconcile
reconcileCard)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

approvalPath

Тип: /dv6/programmer/dv6/ApprovalDesigner/ObjectModel/ApprovalPath_CL/[ApprovalPath]

Маршрут согласования

reconcileCard

Тип: Reconcile

Карточка согласования

SkipStage — метод (ApprovalStage, ApprovalPath, Reconcile)

Пропускает этап.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void SkipStage(ApprovalStage approvalStage, ApprovalPath approvalPath, Reconcile
reconcileCard, DecisionSemantics stageDecision)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

approvalPath

Тип:
/dv6/programmer/dv6/ApprovalDesigner/ObjectModel/ApprovalPath_CL/[ApprovalPath]

Маршрут согласования

reconcileCard

Тип: Reconcile

Карточка согласования

StartStage — метод (ApprovalStage, Reconcile)

Начальный этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
void StartStage(ApprovalStage approvalStage, Reconcile reconcileCard)
```

Параметры

approvalStage

Тип: ApprovalStage

Этап согласования в карте этапов Маршрута согласования

reconcileCard

Тип: Reconcile

Карточка согласования

ISearchWordResolverService — интерфейс

Сервис ISearchWordResolverService предоставляет метод получения согласующих по поисковому слову.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public interface ISearchWordResolverService
```

Методы

Имя	Описание
ExtractSearchWord(ObjectContext, Guid, Guid, Guid)	Возвращает сотрудников, найденных по поисковому слову.
ExtractSearchWord(ObjectContext, Guid, Guid, Guid, Int32)	Возвращает сотрудников, найденных по поисковому слову.

ISearchWordResolverService.ExtractSearchWord — метод (ObjectContext, Guid, Guid, Guid)

Возвращает сотрудников, найденных по поисковому слову.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
IEnumerable<StaffEmployee> ExtractSearchWord(ObjectContext context, Guid searchWordId,
Guid initiatorId, Guid registratorId)
```

Параметры

context

Тип: ObjectContext

Текущий контекст объектов

searchWordId

Тип: System.Guid

Идентификатор поискового слова, по которому ищутся сотрудники

initiatorId

Тип: System.Guid

Идентификатор инициатора согласования

registratorId

Тип: System.Guid

Идентификатор согласующего

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<StaffEmployee>

Найденные сотрудники. Тип элемента — StaffEmployee

ISearchWordResolverService.ExtractSearchWord — метод (ObjectContext, Guid, Guid, Guid, Int32)

Возвращает сотрудников, найденных по поисковому слову.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel.Services
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
IEnumerable<DocsVision.BackOffice.ObjectModel.StaffEmployee> ExtractSearchWord
(ObjectContext context, Guid searchWordId, Guid initiatorId, Guid registratorId, int
hierarchyLevel)
```

Параметры

context

Тип: ObjectContext

Текущий контекст объектов

searchWordId

Тип: System.Guid

Идентификатор поискового слова, по которому ищутся сотрудники

initiatorId

Тип: `System.Guid`

Идентификатор инициатора согласования

registratorId

Тип: `System.Guid`

Идентификатор согласующего

hierarchyLevel

Тип: `System.Int32`

Уровень иерархии при получении руководителей

Возвращаемое значение

Тип: `System.Collections.Generic.IEnumerable<StaffEmployee>`

Найденные сотрудники. Тип элемента — `StaffEmployee`

ApprovalFilesCreator — класс

Класс ApprovalFilesCreator содержит статический метод для добавления файлов в карточку Согласование.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public static class ApprovalFilesCreator
```

Методы

Имя	Описание
`CreateFiles(ObjectContext, Document , SectionData, IEnumerable<DocumentFile>)`	Добавляет файлы документа в карточку Согласование.

ApprovalFilesCreator.CreateFiles — метод (ObjectContext, Document , SectionData, IEnumerable<DocumentFile>)

Добавляет файлы документа в карточку Согласование.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public static void CreateFiles(ObjectContext context, Document initialDocument,
SectionData approvalFilesSectionData, IEnumerable<DocumentFile> files)
```

Параметры

context

Тип: ObjectContext

Текущий контекст объектов

initialDocument

Тип: Document

Документ, файлы которого добавляются в Согласование; инициирующий документ

approvalFilesSectionData

Тип: SectionData

Секция карточки Согласование, в которую будут добавлены ссылки на файлы

files

Тип: System.Collections.Generic.IEnumerable<DocumentFile>

Добавляемые в Согласование файлы документа; тип — DocumentFile

Заметки

Для добавления файлов документа в Согласование используйте вместо CreateFiles метод FillFiles сервиса IReconcileService (реализованного в сборке DocsVision.ApprovalDesigner.ObjectModel.dll).

ApprovalPath — класс

Объектная модель карточки Маршрут согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public sealed class ApprovalPath : BaseCard
```

Свойства

Имя	Описание
MainInfo	Предоставляет доступ к основной информации маршрута согласования.
RoadMaps	Возвращает карты этапов маршрута согласования.
States	Возвращает состояния документов в маршруте согласования.

Поля

Имя	Описание
MainInfoProperty	Определяет свойство "Основная информация".
RoadMapProperty	Определяет свойство "Карта этапов".
StatesProperty	Определяет свойство "Состояния".

ApprovalPath.MainInfo — свойство

Предоставляет доступ к основной информации маршрута согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ApprovalPathMainInfo MainInfo { get; }
```

Значение свойства

Тип: ApprovalPathMainInfo

Данные из секции "Основная информация" карточки

ApprovalPath.RoadMaps — свойство

Возвращает карты этапов маршрута согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ObjectCollection<ApprovalPathRoadMap> RoadMaps { get; }
```

Значение свойства

Тип: ObjectCollection<ApprovalPathRoadMap>

Этапы согласования. Тип элемента — ApprovalPathRoadMap

ApprovalPath.States — свойство

Возвращает состояния документов в маршруте согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ObjectCollection<ApprovalPathState> States { get; }
```

Значение свойства

Тип: ObjectCollection<ApprovalPathState>

Состояния. Тип элементов — ApprovalPathState

ApprovalPathMainInfo — класс

Содержит основную информацию карточки Маршрут согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel

- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public sealed class ApprovalPathMainInfo : BaseCardSectionRow
```

Свойства

Имя	Описание
AuthorField	Задаёт или возвращает значение поля Автор.
BusinessProcessFolder	Задаёт или возвращает папку бизнес-процесса.
Name	Задаёт или возвращает название маршрута согласования.
RegistratorField	Задаёт или возвращает значение поля Регистратор.

Поля

Имя	Описание
AuthorFieldProperty	Определяет свойство "Поле автора".
BusinessProcessFolderProperty	Определяет свойство "Папка бизнес процесса".
NameProperty	Определяет свойство "Название".
RegistratorFieldProperty	Определяет свойство "Поле регистратора".

ApprovalPathMainInfo.AuthorField — свойство

Задаёт или возвращает значение поля Автор.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public string AuthorField { get; set; }
```

Значение свойства

Тип: System.String

Автор

Заметки

Значение задаётся и возвращается в виде: {CardSectionGuid}\{FieldName}, где: FieldName — название поля секции CardSectionGuid, из которого будет получен автор маршрута согласования. К примеру: 30eb9b87-822b-4753-9a50-a1825dca1b74\Author — из поля "Автор" секции "Основная информация" карточки "Документ".

ApprovalPathMainInfo.Name — свойство

Задаёт или возвращает название маршрута согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public string Name { get; set; }
```

Значение свойства

Тип: System.String

Название маршрута

ApprovalPathMainInfo.RegistratorField — свойство

Задаёт или возвращает значение поля Регистратор.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public string RegistratorField { get; set; }
```

Значение свойства

Тип: `System.String`

Регистратор

Заметки

Значение задаётся и возвращается в виде: `{CardSectionGuid}\{FieldName}`, где: `FieldName` — название поля секции `CardSectionGuid`, из которого будет получен регистратор маршрута согласования. К примеру: `30eb9b87-822b-4753-9a50-a1825dca1b74\Registrar` — из поля "Регистратор" секции "Основная информация" карточки "Документ".

ApprovalPathRoadMap — класс

Представляет карту этапов Маршрута согласования.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public sealed class ApprovalPathRoadMap : BaseCardSectionRow
```

Свойства

Имя	Описание
`Condition`	Задаёт или возвращает поисковый запрос условия старта этапа.
`Condition2`	Задаёт или возвращает дополнительное условие запуска этапа.
`Excluded`	Задаёт или возвращает признак исключения этапа из маршрута согласования.
`Order`	Задаёт или возвращает приоритет этапа согласования.
`Stage`	Задаёт или возвращает этап согласования.

Поля

Имя	Описание
Condition2Property	Определяет свойство "Условие 2".
ConditionProperty	Определяет свойство "Условие".
ExcludedProperty	Определяет свойство "Этап исключён".
OrderProperty	Определяет свойство "Приоритет".
StageProperty	Определяет свойство "Этап".

ApprovalPathRoadMap.Condition — свойство

Задаёт или возвращает поисковый запрос условия старта этапа.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public string Condition { get; set; }
```

Значение свойства

Тип: System.String

Поисковый запрос

Заметки

ApprovalPathRoadMap.Condition2 — свойство

Задаёт или возвращает дополнительное условие запуска этапа.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public string Condition2 { get; set; }
```

Значение свойства

Тип: System.String

Дополнительное условие

Заметки

ApprovalPathRoadMap.Excluded — свойство

Задаёт или возвращает признак исключения этапа из маршрута согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public bool Excluded { get; set; }
```

Значение свойства

Тип: System.Boolean

true — исключён, иначе — false

ApprovalPathRoadMap.Order — свойство

Задаёт или возвращает приоритет этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public int Order { get; set; }
```

Значение свойства

Тип: System.Int32

Приоритет (порядок)

ApprovalPathRoadMap.Stage — свойство

Задаёт или возвращает этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ApprovalStage Stage { get; set; }
```

Значение свойства

Тип: ApprovalStage

Этап согласования

ApprovalPathState — класс

Представляет состояние документа в Маршруте согласования..

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public sealed class ApprovalPathState : BaseCardSectionRow
```

Свойства

Имя	Описание
DocumentKind	Задаёт или возвращает вид документа.
DocumentState	Задаёт или возвращает состояние документа.

Поля

Имя	Описание
DocumentKindProperty	Определяет свойство "Вид документа".
DocumentStateProperty	Определяет свойство "Состояние документа".

ApprovalResult — класс

Представляет результат согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public sealed class ApprovalResult : BaseCard
```

Свойства

Имя	Описание
MainInfo	Возвращает основную информацию карточки Результат согласования.

Поля

Имя	Описание
MainInfoProperty	Определяет свойство "Основная информация".

ApprovalResult.MainInfo — свойство

Возвращает основную информацию карточки Результат согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ApprovalResultMainInfo MainInfo { get; }
```

Значение свойства

Тип: ApprovalResultMainInfo

Основная информация из результата согласования

ApprovalResultMainInfo — класс

Содержит основную информацию карточки Результат согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public sealed class ApprovalResultMainInfo : BaseCardSectionRow
```

Свойства

Имя	Описание
ApprovalId	Задаёт или возвращает идентификатор карточки согласования.
ApprovingDocumentId	Задаёт или возвращает идентификатор согласуемого документа.
Cycle	Задаёт или возвращает номер цикла согласования.
Decision	Задаёт или возвращает семантику решения по согласованию.
Name	Задаёт или возвращает название результата согласования.
StageId	Задаёт или возвращает идентификатор этапа согласования.

Поля

Имя	Описание
ApprovalIdProperty	Определяет свойство "Карточка согласования".
ApprovingDocumentIdProperty	Определяет свойство "Идентификатор согласуемого документа".
CycleProperty	Определяет свойство "Номер цикла".
DecisionProperty	Определяет свойство "Решение".
NameProperty	Определяет свойство "Название".
StageIdProperty	Определяет свойство "Идентификатор этапа".

ApprovalResultMainInfo.Decision — свойство

Задаёт или возвращает семантику решения по согласованию.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public DecisionSemantics Decision { get; set; }
```

Значение свойства

Тип: DecisionSemantics

Семантика решения

ApprovalStage — класс

Этап согласования в карте этапов Маршрута согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public class ApprovalStage : BaseCard
```

Свойства

Имя	Описание
AdditionalSettings	Возвращает дополнительные настройки этапа согласования.
Decisions	Возвращает варианты решений для завершения этапа согласования.
MainInfo	Возвращает основную информацию этапа согласования.
SelectedApprovers	Возвращает согласующих этапа согласования.
States	Возвращает возможные состояния этапа согласования.
TaskSettings	Возвращает настройки заданий участников этапа согласования.

Поля

Имя	Описание
AdditionalSettingsProperty	Определяет свойство "Дополнительные настройки".
DecisionsProperty	Определяет свойство "Решения".
MainInfoProperty	Определяет свойство "Основная информация".
SelectedApproversProperty	Определяет свойство "Согласующие".
StatesProperty	Определяет свойство "Состояния".
TaskSettingsProperty	Определяет свойство "Настройки задания".

ApprovalStage.AdditionalSettings — свойство

Возвращает дополнительные настройки этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ApprovalStageAdditionalSettings AdditionalSettings { get; }
```

Значение свойства

Тип: ApprovalStageAdditionalSettings

Дополнительные настройки

ApprovalStage.Decisions — свойство

Возвращает варианты решений для завершения этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ObjectCollection<ApprovalStageDecision> Decisions { get; }
```

Значение свойства

Тип: ObjectCollection<ApprovalStageDecision>

Варианты решений. Тип элемента — ApprovalStageDecision

ApprovalStage.MainInfo — свойство

Возвращает основную информацию этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ApprovalStageMainInfo MainInfo { get; }
```

Значение свойства

Тип: ApprovalStageMainInfo

Данные секции "Основная информация" карточки Этап согласования

ApprovalStage.SelectedApprovers — свойство

Возвращает согласующих этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ObjectCollection<ApprovalStageApprover> SelectedApprovers { get; }
```

Значение свойства

Тип: ObjectCollection<ApprovalStageApprover>

Согласующие. Тип элемента — ApprovalStageApprover

ApprovalStage.States — свойство

Возвращает возможные состояния этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel

- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ObjectCollection<ApprovalStageState> States { get; }
```

Значение свойства

Тип: ObjectCollection<ApprovalStageState>

Состояния. Тип элемента — ApprovalStageState

ApprovalStage.TaskSettings — свойство

Возвращает настройки заданий участников этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ApprovalStageTaskSettings TaskSettings { get; }
```

Значение свойства

Тип: ApprovalStageTaskSettings

Настройки задания участников согласования

ApprovalStageAdditionalSettings — класс

Содержит дополнительные настройки этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public class ApprovalStageAdditionalSettings : BaseCardSectionRow
```

Свойства

Имя	Описание
AdditionSemantics	Задаёт или возвращает семантику при добавлении согласующих.
AllowAlternatePerforming	Задаёт или возвращает признак разрешения альтернативного исполнения.
BusinessProcessOnStageCompletion	Задаёт или возвращает идентификатор бизнес-процесса, запускаемого при завершении этапа.
CanAddFiles	Задаёт или возвращает признак разрешения добавления файлов участниками согласования.
CanEditMainFilesInDocument	Задаёт или возвращает признак разрешения редактирования основных файлов участниками согласования.
CompleteAfterFirstRejection	Задаёт или возвращает признак завершения после первого отказа.
InitiatorTaskKind	Задаёт или возвращает вид заданий согласующих.
MoveAdditionalFilesToDocument	Задаёт или возвращает признак переноса дополнительных файлов в карточку документа.
MoveVersionsFromPreviousApprovers	Задаёт или возвращает признак переноса версий предыдущих согласующих.
NotAddToReconcilationList	Задаёт или возвращает признак запрета добавления решения в лист согласования.
RejectionCase	Задаёт или возвращает действие в случае отказа в согласовании.
RepeatApproving	Задаёт или возвращает признак отправки на повторное согласование по дополнительных согласующих.

Имя	Описание
RequiereInitiatorConfirmation	Задаёт или возвращает признак запроса подтверждения на добавление согласующих у инициатора согласования.
ShowReconcilationListInDocument	Задаёт или возвращает признак отображения листа согласования в документе.
VersionTreeLevel	Задаёт или возвращает уровень дерева версий этапа согласований.

Поля

Имя	Описание
AdditionSemanticsProperty	Определяет свойство "Семантика при добавлении".
AllowAlternatePerformingProperty	Определяет свойство "Разрешить альтернативное исполнение".
BusinessProcessOnStageCompletionProperty	Определяет свойство "Бизнес-процесс при завершении этапа".
CanAddFilesProperty	Определяет свойство "Добавление собственных файлов участниками согласования".
CanEditMainFilesInDocumentProperty	Определяет свойство "Редактирование основных файлов".
CompleteAfterFirstRejectionProperty	Определяет свойство "Завершить после первого отказа".
InitiatorTaskKindProperty	Определяет свойство "Вид задания инициатора".
MoveAdditionalFilesToDocumentProperty	Определяет свойство "Переносить дополнительные файлы в карточку документа".
MoveVersionsFromPreviousApproversProperty	Определяет свойство "Переносить версии от предыдущих согласующих".

Имя	Описание
NotAddToReconcilationListProperty	Определяет свойство "Не добавлять в лист согласования".
RejectionCaseProperty	Определяет свойство "Действие в случае отказа".
RepeatApprovingProperty	Определяет свойство "Отправлять на пересогласование".
RequiereInitiatorConfirmationProperty	Определяет свойство "Запрашивать подтверждение инициатора".
ShowReconcilationListInDocumentProperty	Определяет свойство "Отображать лист согласования в документе".
VersionTreeLevelProperty	Определяет свойство "Уровень дерева версий".

ApprovalStageAdditionalSettings.AdditionSemantics — свойство

Задаёт или возвращает семантику при добавлении согласующих.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public AdditionSemantics AdditionSemantics { get; set; }
```

Значение свойства

Тип: AdditionSemantics

Семантика при добавлении согласующих

ApprovalStageAdditionalSettings.RejectionCase — свойство

Задаёт или возвращает действие в случае отказа в согласовании.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ApprovalRejectionCase RejectionCase { get; set; }
```

Значение свойства

Тип: ApprovalRejectionCase

Вариант действия при завершении согласования с отрицательной семантикой

ApprovalStageAdditionalSettings.VersionTreeLevel — свойство

Задаёт или возвращает уровень дерева версий этапа согласований.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public VersionTreeLevel VersionTreeLevel { get; set; }
```

Значение свойства

Тип: VersionTreeLevel

Уровень дерева

ApprovalStageApprover — класс

Представляет согласующего этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public class ApprovalStageApprover : BaseCardSectionRow
```

Свойства

Имя	Описание
Employee	Задаёт или возвращает данные сотрудника.

Имя	Описание
`Excluded`	Задаёт или возвращает признак исключения из этапа.
`Group`	Задаёт или возвращает группу, из которой выбираются согласующие.
`Order`	Задаёт или возвращает порядковый номер согласующего в этапе.
`Role`	Задаёт или возвращает роль, из которой выбираются согласующие.
`SearchWord`	Задаёт или возвращает идентификатор поискового слова, по которому определяется согласующих.
`Unit`	Задаёт или возвращает подразделение, из которого выбираются согласующие.

Поля

Имя	Описание
`EmployeeProperty`	Определяет свойство "Сотрудник".
`ExcludedProperty`	Определяет свойство "Временно исключён".
`GroupProperty`	Определяет свойство "Группа".
`OrderProperty`	Определяет свойство "Номер".
`RoleProperty`	Определяет свойство "Роль".
`SearchWordProperty`	Определяет свойство "Поисковое слово".
`UnitProperty`	Определяет свойство "Подразделение".

ApprovalStageDecision — класс

Вариант решения для завершения этапа согласования.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public class ApprovalStageDecision : BaseCardSectionRow
```

Свойства

Имя	Описание
Image	Задаёт или возвращает идентификатор файла иконки варианта решения.
Name	Задаёт или возвращает название варианта решения.
Order	Задаёт или возвращает номер варианта решения.
RequestDigitalSignature	Задаёт или возвращает признак требования электронной подписи при завершении этапа.
Semantics	Задаёт или возвращает семантику завершения этапа согласования.
SignatureLabel	Задаёт или возвращает метку подписи для этапа согласования.

Методы

Имя	Описание
LoadImage(ObjectContext)	Выгружает иконку варианта завершения этапа согласования.
SaveImage(ObjectContext, ApprovalStage, Image)	Загружает иконку для варианта завершения этапа согласования.

Поля

Имя	Описание
ImageProperty	Определяет свойство "Иконка".
NameProperty	Определяет свойство "Название".
OrderProperty	Определяет свойство "Номер".

Имя	Описание
RequestDigitalSignatureProperty	Определяет свойство "Запрашивать электронную подпись".
SemanticsProperty	Определяет свойство "Семантика".
SignatureLabelProperty	Определяет свойство "Метка подписи".

ApprovalStageDecision.Image — свойство

Задаёт или возвращает идентификатор файла иконки варианта решения.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public Guid Image { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор файла изображения

ApprovalStageDecision.Name — свойство

Задаёт или возвращает название варианта решения.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public string Name { get; set; }
```

Значение свойства

Тип: System.String

Название варианта завершения

ApprovalStageDecision.Order — свойство

Задаёт или возвращает номер варианта решения.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public int Order { get; set; }
```

Значение свойства

Тип: System.Int32

Порядковый номер

ApprovalStageDecision.RequestDigitalSignature — свойство

Задаёт или возвращает признак требования электронной подписи при завершении этапа.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public bool RequestDigitalSignature { get; set; }
```

Значение свойства

Тип: System.Boolean

true — для завершения требуется подписание ЭП; иначе — false

ApprovalStageDecision.Semantics — свойство

Задаёт или возвращает семантику завершения этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public DecisionSemantics Semantics { get; set; }
```

Значение свойства

Тип: DecisionSemantics

Семантика завершения

ApprovalStageDecision.SignatureLabel — свойство

Задаёт или возвращает метку подписи для этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public SignatureLabel SignatureLabel { get; set; }
```

Значение свойства

Тип: SignatureLabel

Метка подписи

ApprovalStageDecision.LoadImage — метод (ObjectContext)

Выгружает иконку варианта завершения этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public Image LoadImage(ObjectContext objectContext)
```

Параметры

objectContext

Тип: ObjectContext

Текущий контекст объектов

Возвращаемое значение

Тип: System.Drawing.Image

Изображение

ApprovalStageDecision.SaveImage — метод (ObjectContext, ApprovalStage, Image)

Загружает иконку для варианта завершения этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public void SaveImage(ObjectContext objectContext, ApprovalStage stage, Image image)
```

Параметры

objectContext

Тип: ObjectContext

Текущий контекст объектов

stage

Тип: ApprovalStage

Этап согласования, содержащий данный вариант завершения.

image

Тип: System.Drawing.Image

Загружаемое изображение

ApprovalStageMainInfo — класс

Содержит основную информацию карточки Этап согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public class ApprovalStageMainInfo : BaseCardSectionRow
```

Свойства

Имя	Описание
AllowEditBeforeReconcilation	Задаёт или возвращает признак возможности изменения этапа перед запуском согласования.
ApproversBusinessProcess	Задаёт или возвращает идентификатор бизнес-процесса для вычисления согласующих.
ApproversBusinessProcessSpecified	Задаёт или возвращает признак того, что указан бизнес-процесс для вычисления согласующих.
ApproversField	Задаёт или возвращает поле карточки, из которого будут выбираться согласующие.
ApproversFieldSpecified	Задаёт или возвращает признак того, что указано поле карточки, из которого будут получены согласующие.
ApproversSpecified	Задаёт или возвращает признак того, что согласующие заданы.
AutoCompleteTaskAfterDeadline	Задаёт или возвращает признак того, что задание КС должно автоматически завершаться по истечению срока исполнения.
DefaultDecision	Задаёт или возвращает для этапа согласования семантику завершения по умолчанию при автосогласовании.
Duration	Задаёт или возвращает длительность выполнения задания КС.
Hidden	Задаёт или возвращает признак скрытия этапа из маршрута согласования.
HierarchyLevel	Задаёт или возвращает значение уровня иерархии руководителей.

Имя	Описание
Mode	Задаёт или возвращает режим маршрутизации этапа согласования.
Name	Задаёт или возвращает название этапа согласования.
NextDuration	Задаёт или возвращает длительность завершения задания/этапа на следующем цикле.
NextDurationSpecified	Задаёт или возвращает признак того, что для следующего цикла согласования установлено значение длительности завершения задания/этапа.
ReconcileType	Задаёт или возвращает тип маршрутизации этапа согласования.
SkipRepeated	Задаёт или возвращает признак того, что этап должен пропускаться на следующем цикле согласования.
SpecificDuration	Задаёт или возвращает признак того, что длительность завершения задания/этапа установлена.

Поля

Имя	Описание
AllowEditBeforeReconcilationProperty	Определяет свойство "Разрешить изменение этапа перед стартом согласования".
ApprovalStageModeProperty	Определяет свойство "Режим".
ApproversBusinessProcessProperty	Определяет свойство "Бизнес-процесс для выбора согласующих".
ApproversBusinessProcessSpecifiedProperty	Определяет свойство "Бизнес-процесс для выбора согласующих задан".

Имя	Описание
ApproversFieldProperty	Определяет свойство "Поле согласующих".
ApproversFieldSpecifiedProperty	Определяет свойство "Поле согласующих задано".
ApproversSpecifiedProperty	Определяет свойство "Согласующие заданы".
AutoCompleteTaskAfterDeadlineProperty	Определяет свойство "Автоматически завершать задание по истечению срока исполнения".
DefaultDecisionProperty	Определяет свойство "Решение по умолчанию".
DurationProperty	Определяет свойство "Длительность".
HiddenProperty	Определяет свойство "Скрыть этап".
HierarchyLevelProperty	Определяет свойство "Уровень иерархии руководителей".
NameProperty	Определяет свойство "Название этапа".
NextDurationProperty	Определяет свойство "Длительность на последующих циклах".
NextDurationSpecifiedProperty	Определяет свойство "Флаг, показывающий задана ли длительность на последующих циклах".
ReconcileTypeProperty	Определяет свойство "Тип согласования".
SkipRepeatedProperty	Определяет свойство "Пропускать при повторе".
SpecificDurationProperty	Определяет свойство "Флаг, показывающий, задана ли длительность для каждого согласующего или на всем этапе".

ApprovalStageMainInfo.AllowEditBeforeReconcilation — свойство

Задаёт или возвращает признак возможности изменения этапа перед запуском

согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public bool AllowEditBeforeReconcilation { get; set; }
```

Значение свойства

Тип: System.Boolean

true — этап может быть изменён, иначе — false

ApprovalStageMainInfo.ApproversBusinessProcess — свойство

Задаёт или возвращает идентификатор бизнес-процесса для вычисления согласующих.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public Guid ApproversBusinessProcess { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор бизнес-процесса

ApprovalStageMainInfo.ApproversBusinessProcessSpecified — свойство

Задаёт или возвращает признак того, что указан бизнес-процесс для вычисления согласующих.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public bool ApproversBusinessProcessSpecified { get; set; }
```

Значение свойства

Тип: System.Boolean

true — бизнес-процесс выбран, иначе — false

ApprovalStageMainInfo.ApproversField — свойство

Задаёт или возвращает поле карточки, из которого будут выбираться согласующие.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public string ApproversField { get; set; }
```

Значение свойства

Тип: System.String

Название поля

ApprovalStageMainInfo.ApproversFieldSpecified — свойство

Задаёт или возвращает признак того, что указано поле карточки, из которого будут получены согласующие.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public bool ApproversFieldSpecified { get; set; }
```

Значение свойства

Тип: System.Boolean

true — поле указано, иначе — false

ApprovalStageMainInfo.ApproversSpecified — свойство

Задаёт или возвращает признак того, что согласующие заданы.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public bool ApproversSpecified { get; set; }
```

Значение свойства

Тип: System.Boolean

true — согласующие заданы, иначе — false

ApprovalStageMainInfo.AutoCompleteTaskAfterDeadline — свойство

Задаёт или возвращает признак того, что задание КС должно автоматически завершаться по истечению срока исполнения.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public bool AutoCompleteTaskAfterDeadline { get; set; }
```

Значение свойства

Тип: System.Boolean

true — задание завершается автоматически, иначе — false

ApprovalStageMainInfo.DefaultDecision — свойство

Задаёт или возвращает для этапа согласования семантику завершения по умолчанию при автосогласовании.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public DecisionSemantics DefaultDecision { get; set; }
```

Значение свойства

Тип: DecisionSemantics

Семантика завершения задания

ApprovalStageMainInfo.Duration — свойство

Задаёт или возвращает длительность выполнения задания КС.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public int Duration { get; set; }
```

Значение свойства

Тип: System.Int32

Длительность выполнения задания/этапа согласования

ApprovalStageMainInfo.Hidden — свойство

Задаёт или возвращает признак скрытия этапа из маршрута согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public bool Hidden { get; set; }
```

Значение свойства

Тип: System.Boolean

true — этап скрыт, иначе — false

ApprovalStageMainInfo.HierarchyLevel — свойство

Задаёт или возвращает значение уровня иерархии руководителей, если для выбора согласующих используется поисковое слово "Все руководители инициатора".

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public int HierarchyLevel { get; set; }
```

Значение свойства

Тип: System.Int32

Уровень иерархии

Заметки

Значение устанавливается, если для выбора согласующих используется поисковое слово "Все руководители инициатора"

ApprovalStageMainInfo.Mode — свойство

Задаёт или возвращает режим маршрутизации этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ApprovalStageMode Mode { get; set; }
```

Значение свойства

Тип: ApprovalStageMode

Режим маршрутизации

ApprovalStageMainInfo.Name — свойство

Задаёт или возвращает название этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public string Name { get; set; }
```

Значение свойства

Тип: System.String

Название

ApprovalStageMainInfo.NextDuration — свойство

Задаёт или возвращает длительность завершения задания/этапа на следующем цикле.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public int NextDuration { get; set; }
```

Значение свойства

Тип: System.String

Длительность

ApprovalStageMainInfo.NextDurationSpecified — свойство

Задаёт или возвращает признак того, что для следующего цикла согласования установлено значение длительности завершения задания/этапа.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public bool NextDurationSpecified { get; set; }
```

Значение свойства

Тип: System.Boolean

true — значение длительности установлено, иначе — false

ApprovalStageMainInfo.ReconcileType — свойство

Задаёт или возвращает тип маршрутизации этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public ApprovalType ReconcileType { get; set; }
```

Значение свойства

Тип: ApprovalType

Тип маршрутизации

ApprovalStageMainInfo.SkipRepeated — свойство

Задаёт или возвращает признак того, что этап должен пропускаться на следующем цикле согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public bool SkipRepeated { get; set; }
```

Значение свойства

Тип: System.Boolean

true — этап будет пропущен при повторе, иначе — false

Заметки

Признак пропуска этапа значим только для пройденных этапов.

ApprovalStageMainInfo.SpecificDuration — свойство

Задаёт или возвращает признак того, что длительность завершения задания/этапа установлена.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public bool SpecificDuration { get; set; }
```

Значение свойства

Тип: System.Boolean

true — значение длительности установлено, иначе — false

ApprovalStageState — класс

Представляет состояние документа в матрице состояний этапа согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public class ApprovalStageState : BaseCardSectionRow
```

Свойства

Имя	Описание
DocumentKind	Задаёт или возвращает вид документ.
NegativeState	Задаёт или возвращает состояние при отрицательном состоянии.
PositiveState	Задаёт или возвращает состояние при положительном состоянии.
StageState	Задаёт или возвращает состояние этапа.

Поля

Имя	Описание
`DocumentKindProperty`	Определяет свойство "Вид документа".
`NegativeStateProperty`	Определяет свойство "Состояние при отрицательном результате".
`PositiveStateProperty`	Определяет свойство "Состояние при положительном результате".
`StageStateProperty`	Определяет свойство "Состояние на этапе".

ApprovalStageTaskSettings — класс

Предоставляет настройки задания участников согласования.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public class ApprovalStageTaskSettings : BaseCardSectionRow
```

Свойства

Имя	Описание
`Calendar`	Задаёт или возвращает идентификатор бизнес-календаря, используемого при расчете дат в задании.
`Content`	Задаёт или возвращает идентификатор содержание задания.
`Kind`	Задаёт или возвращает вид карточки Задание.
`Name`	Задаёт или возвращает название задания.
`` `TaskDecision` ``	Задаёт или возвращает семантику для автоматически завершаемого задания.

Поля

Имя	Описание
CalendarProperty	Определяет свойство "Бизнес-календарь".
ContentProperty	Определяет свойство "Содержание".
KindProperty	Определяет свойство "Вид".
NameProperty	Определяет свойство "Название".
TaskDecisionProperty	Определяет свойство "Семантика завершения задания".

ApprovalStageTaskSettings.TaskDecision — свойство

Задаёт или возвращает семантику для автоматически завершаемого задания.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public TaskDecisionSemantics TaskDecision { get; set; }
```

Значение свойства

Тип: TaskDecisionSemantics

Семантика завершения задания

AdditionSemantics — перечисление

Семантики завершения задания при добавлении согласующих.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public enum AdditionSemantics
```

Члены

Имя члена	Описание
`Positive`	Положительное
`Negative`	Отрицательное
`Neutral`	Условно-положительное

ApprovalRejectionCase — перечисление

Действия в случае отказа.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public enum ApprovalRejectionCase
```

Члены

Имя члена	Описание
`ContinueReconciliation`	Продолжать согласование
`CompleteStage`	Завершать этап
`CompleteReconciliation`	Завершать согласование

ApprovalStageMode — перечисление

Режимы маршрутизации согласования.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public enum ApprovalStageMode
```

Члены

Имя члена	Описание
`Approval`	Согласование
`Signing`	Подписание
`Consolidation`	Консолидация

ApprovalType — перечисление

Тип маршрутизации согласования.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public enum ApprovalType
```

Члены

Имя члена	Описание
`Sequential`	Последовательное
`Parallel`	Параллельное
`Consolidation`	Консолидация
`Alternative`	Альтернативное

DecisionSemantics — перечисление

Семантики завершения этапа согласования.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public enum DecisionSemantics
```

Члены

Имя члена	Описание
`Positive`	Положительное
`Negative`	Отрицательное
`Neutral`	Условно-положительное
`Cancellation`	Отмена
`NewCycle`	Новый цикл
`ApproversAddition`	Добавление согласующих
`Completion`	Завершение

FileVersionType — перечисление

Возможные типы версии файла.

- **Пространство имён:** `DocsVision.ApprovalDesigner.ObjectModel`
- **Сборка:** `DocsVision.ApprovalDesigner.ObjectModel.dll`

Синтаксис

```
public enum FileVersionType
```

Члены

Имя члена	Описание
`Root`	Корневая версия
`Cycle`	Версия цикла
`Stage`	Версия этапа
`Approver`	Версия согласующего
`ForApprover`	*Не используется*
`CurrentStageVersion`	Текущая версия этапа
`MergedVersion`	Объединенная версия
`ConsolidatedVersion`	Консолидированная версия
`VersionOfConsolidator`	Версия консолидатора

TaskDecisionSemantics — перечисление

Семантики завершения задания согласования.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public enum TaskDecisionSemantics
```

Члены

Имя члена	Описание
Positive	Положительное
Negative	Отрицательное
Neutral	Условно-положительное
Cancellation	Отмена
NewCycle	Новый цикл
Completion	Завершение

VersionTreeLevel — перечисление

Уровни дерева версий.

- **Пространство имён:** DocsVision.ApprovalDesigner.ObjectModel
- **Сборка:** DocsVision.ApprovalDesigner.ObjectModel.dll

Синтаксис

```
public enum VersionTreeLevel
```

Члены

Имя члена	Описание
Process	Процесс
Cycle	Цикл
Stage	Этап

DocsVision.BackOffice — пространство имён

Пространство имён DocsVision.BackOffice содержит типы представляющие объектную модель библиотеки Базовые объекты.

Пространства имён

Пространство имён	Описание
DocsVision.BackOffice.CardLib	Пространство имён DocsVision.BackOffice.CardLib содержит типы, возвращающие псевдонимы и идентификаторы карточек библиотеки *Базовые объекты*.
DocsVision.BackOffice.Cards	Пространство имён DocsVision.BackOffice.Cards предоставляет элементы управления и компоненты карточек BackOffice.
DocsVision.BackOffice.DigitalSignature	Пространство имён DocsVision.BackOffice.DigitalSignature содержит классы и методы для получения расширенной информации о штампе цифровой подписи.
DocsVision.BackOffice.Helpers	Пространство имён DocsVision.BackOffice.Helpers содержит статический класс с кодами ошибок для различных методов.
DocsVision.BackOffice.ObjectModel	В пространстве имён DocsVision.BackOffice.ObjectModel реализованы классы объектной модель уровня бизнес-логики для карточек библиотеки *Базовые объекты*.
DocsVision.BackOffice.ServerExtension	Типы данного пространства имён содержат реализацию серверной логики BackOffice, в т.ч. реализацию ролевой модели BackOffice.

Пространство имён	Описание
DocsVision.BackOffice.SnapIn	В пространстве имён DocsVision.BackOffice.SnapIn содержится базовый класс для наследования расширением Консоли настройки.
DocsVision.BackOffice.WinForms	Данное пространство имён предоставляет коллекцию стандартных элементов управления и набор классов для работы с разметками DevXpress.
DocsVision.BackOffice.Xml	Данное пространство содержит подпространство имён с классами обеспечивающими сериализацию и десериализацию XML-объектов.

DocsVision.BackOffice.CardLib — пространство имён

Пространство имён DocsVision.BackOffice.CardLib содержит типы, возвращающие псевдонимы и идентификаторы карточек библиотеки *Базовые объекты*.

Пространства имён

Пространство имён	Описание
CardDefs	Пространство имён DocsVision.BackOffice.CardLib.CardDefs содержит статические классы с псевдонимами и идентификаторами карточек библиотеки *Базовые объекты*.

DocsVision.BackOffice.CardLib.CardDefs — пространство имён

Пространство имён DocsVision.BackOffice.CardLib.CardDefs содержит статические классы с псевдонимами и идентификаторами карточек библиотеки *Базовые объекты*.

Классы

Класс	Описание
CardBaseUniversalItem	Карточка строки справочника.
CardCalendar	Бизнес-календарь.

Класс	Описание
CardCategoryList	Список категорий.
CardDepartment	Карточка подразделения.
CardDocument	Карточка Документ.
CardEmployee	Карточка сотрудника.
CardFileList	Список ссылок на карточки версионного файла.
CardPartnersDepartment	Карточка подразделения контрагента.
CardPartnersEmployee	Карточка сотрудника контрагента.
CardReconcile	Согласование.
CardReferenceList	Ход согласования.
CardServer	Карточка сервера.
CardSignatureList	Список подписей.
CardSurveyList	Список опросов.
CardTask	Задание.
CardTaskGroup	Группа заданий.
CardTaskList	Список ссылок на карточки заданий.
RefBaseUniversal	Конструктор справочников.
RefCategories	Справочник категорий.
RefKinds	Справочник категорий.
RefLayouts	Конструктор разметок.
RefLinks	Справочник ссылок.
RefNavExtension	Расширение Windows-клиента.
RefNumerationRules	Конструктор правил нумерации.
RefPartners	Справочник контрагентов.
RefRoleModel	Конструктор ролей.
RefScripting	Конструктор скриптов.
RefServers	Справочник серверов.

Класс	Описание
RefSignatureLabels	Справочник меток подписей.
RefStaff	Справочник сотрудников.
RefStates	Конструктор состояний.

DocsVision.BackOffice.Cards — пространство имён

Пространство имён DocsVision.BackOffice.Cards предоставляет элементы управления и компоненты карточек BackOffice.

Пространства имён

Пространство имён	Описание
DocsVision.BackOffice.Cards.CardDocument	Содержит стандартные компоненты карточек Документ.

DocsVision.BackOffice.Cards.CardDocument — пространство имён

Пространство имён DocsVision.BackOffice.Cards стандартные компоненты карточек Документ.

Интерфейсы

Интерфейс	Описание
IFilesView	Определяет события и свойства элемента для работы с файлами карточки.
IFilesViewWithPreview	Определяет события и свойства элемента для предпросмотра содержимого файла.

IFilesView — интерфейс

Данный интерфейс определяет методы элемента управления для работы с файлами карточки.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public interface IFilesView
```

Методы

Имя	Описание
AddAdditionalScanFile	Добавляет дополнительный отсканированный файл.
AddAdditionalScanFileAsync	Добавляет дополнительный отсканированный файл асинхронно.
AddMainFSFile	Добавляет основной файл из файловой системы.
AddMainFSFileAsync	Добавляет основной файл из файловой системы асинхронно.
AddMainScanFile	Добавляет основной отсканированный файл.
AddMainScanFileAsync	Добавляет основной отсканированный файл асинхронно.
AddMainRecognizedFile	Добавляет основной файл как распознанный поток.
AddMainRecognizedFileAsync	Добавляет основной файл как распознанный поток асинхронно.
CancelLockMainFile	Отменяет блокировку основного файла.
CopyAdditionalFileToClipboard	Копирует дополнительный файл в буфер обмена.
CopyMainFileToClipboard	Копирует основной файл в буфер обмена.
CopyMainFileToClipboardAsync	Копирует основной файл в буфер обмена. асинхронно.
GetSelectedAdditionalFiles	Получает выбранные дополнительные файлы.
GetSelectedMainFiles	Получает выбранные основные файлы.

Имя	Описание
LockMainFile	Блокирует основной файл.
OpenAdditionalFile	Открывает дополнительный файл.
OpenAdditionalFileAsync	Открывает дополнительный файл асинхронно.
OpenMainFile	Открывает основной файл.
OpenMainFileAsync	Открывает основной файл асинхронно.
RefreshFiles	Обновляет файлы.
RemoveAdditionalFiles	Удаляет дополнительные файлы.
RemoveMainFiles	Удаляет основные файлы.
SaveAdditionalFileToFS	Сохраняет дополнительный файл в файловую систему.
SaveAdditionalFileToFSAsync`	Сохраняет дополнительный файл в файловую систему асинхронно.
SaveMainFileToFS	Сохраняет основной файл в файловую систему
SaveMainFileToFSAsync	Сохраняет основной файл в файловую систему асинхронно.
UnlockMainFile	Разблокирует основной файл.

AddAdditionalScanFile — метод

Добавляет дополнительный отсканированный файл.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void AddAdditionalScanFile();
```

AddAdditionalScanFileAsync — метод

Добавляет дополнительный отсканированный файл асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
Task AddAdditionalScanFileAsync();
```

AddMainFSFile — метод

Добавляет основной файл из файловой системы.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void AddMainFSFile();
```

AddMainFSFileAsync — метод

Добавляет основной файл из файловой системы асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
Task AddMainFSFileAsync();
```

AddMainScanFile — метод

Добавляет основной файл со сканера асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void AddMainScanFile();
```

AddMainScanFileAsync — метод

Добавляет основной файл со сканера асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
Task AddMainScanFileAsync();
```

AddMainRecognizedFile — метод

Добавляет основной файл как распознанный поток.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void AddMainRecognizedFile();
```

AddMainRecognizedFileAsync — метод

Добавляет основной файл как распознанный поток асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
Task AddMainRecognizedFileAsync();
```

CancelLockMainFile — метод

Отменяет блокировку основного файла.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void CancelLockMainFile();
```

CopyAdditionalFileToClipboard — метод

Копирует дополнительный файл в буфер обмена.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
void CopyAdditionalFileToClipboard();
```

CopyMainFileToClipboard — метод

Копирует основной файл в буфер обмена.

- Пространство имён: DocsVision.BackOffice.Cards.CardDocument
- Сборка: DocsVision.BackOffice.WinForms.dll

Синтаксис

```
void CopyMainFileToClipboard();
```

CopyMainFileToClipboardAsync — метод

Копирует основной файл в буфер обмена асинхронно.

- Пространство имён: DocsVision.BackOffice.Cards.CardDocument
- Сборка: DocsVision.BackOffice.WinForms.dll

Синтаксис

```
Task CopyMainFileToClipboardAsync();
```

GetSelectedAdditionalFiles — перечисление

Получает выбранные дополнительные файлы.

- Пространство имён: DocsVision.BackOffice.Cards.CardDocument

- Сборка: DocsVision.BackOffice.WinForms.dll

Синтаксис

```
IEnumerable<DocumentFile> GetSelectedAdditionalFiles();
```

GetSelectedMainFiles — перечисление

Получает выбранные основные файлы.

- Пространство имён: DocsVision.BackOffice.Cards.CardDocument
- Сборка: DocsVision.BackOffice.WinForms.dll

Синтаксис

```
IEnumerable<DocumentFile> GetSelectedMainFiles();
```

LockMainFile — метод

Блокирует основной файл.

- Пространство имён: DocsVision.BackOffice.Cards.CardDocument
- Сборка: DocsVision.BackOffice.WinForms.dll

Синтаксис

```
void LockMainFile();
```

OpenAdditionalFile — метод

Открывает дополнительный файл.

- Пространство имён: DocsVision.BackOffice.Cards.CardDocument
- Сборка: DocsVision.BackOffice.WinForms.dll

Синтаксис

```
void OpenAdditionalFile();
```

OpenAdditionalFileAsync — метод

Открывает дополнительный файл асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
Task OpenAdditionalFileAsync();
```

OpenMainFile — метод

Открывает основной файл.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
FileOpenInfo OpenMainFile();
```

OpenMainFileAsync — метод

Открывает основной файл асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
Task<FileOpenInfo> OpenMainFileAsync();
```

RefreshFiles — метод

Обновляет файлы.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void RefreshFiles();
```

RemoveAdditionalFiles — метод

Удаляет дополнительные файлы.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void RemoveAdditionalFiles();
```

RemoveMainFiles — метод

Удаляет основные файлы.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void RemoveMainFiles();
```

SaveAdditionalFileToFS — метод

Сохраняет дополнительный файл в файловую систему.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void SaveAdditionalFileToFS();
```

SaveMainFileToFS — метод

Сохраняет основной файл в файловую систему

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void SaveMainFileToFS();
```

SaveMainFileToFSAsync — метод

Сохраняет основной файл в файловую систему асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
Task SaveMainFileToFSAsync();
```

UnlockMainFile — метод

Разблокирует основной файл.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void UnlockMainFile();
```

IFilesViewWithPreview — интерфейс

Интерфейс IFilesViewWithPreview определяет события и свойства элемента для предпросмотра содержимого файла.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
public interface IFilesViewWithPreview
```

Методы

Имя	Описание
AddFile(DocumentLoadSource)	Добавляет файл из указанного источника.
AddFileAsync(DocumentLoadSource)	Добавляет файл из указанного источника асинхронно.
AddMainFSFile	Добавляет основной файл из файловой системы.
AddMainFSFileAsync	Добавляет основной файл из файловой системы асинхронно.
AddMainScanFile	Добавляет основной файл со сканера.
AddMainScanFileAsync	Добавляет основной файл со сканера.
OpenMainFile	Открывает основной файл.
OpenMainFileAsync	Открывает основной файл асинхронно.

Свойства

Имя	Описание
FileOperationInProgress	Возвращает признак выполнения длительной операции в списке файлов. Если true — в данный момент выполняется длительная операция.

События

Имя	Описание
AdditionalFileAdded	Событие возникает после открытия дополнительного файла.
AdditionalFileRemoved	Событие возникает после удаления дополнительного файла.
AdditionalFileSelected	Событие возникает при выборе дополнительного файла.
AfterFileOpened	Событие возникает после открытия файла (основного или дополнительного).

Имя	Описание
BeforeFileOpened	Событие возникает перед открытием файла (основного или дополнительного).
BeforeMainFileAdded	Событие возникает перед открытием основного файла.
BeforeAdditionalFileAdded	Событие возникает перед открытием дополнительного файла.
BeforeMainFileRemoved	Событие возникает перед удалением основного файла.
BeforeAdditionalFileRemoved	Событие возникает перед удалением дополнительного файла.
DocumentUnlocking	Событие возникает перед снятием блокировки с файла (основного или дополнительного).
DocumentUnlocked	Событие возникает после снятия блокировки с файла (основного или дополнительного).
MainFileSelected	Событие возникает при выборе основного файла.
MainFileRemoved	Событие возникает после удаления основного файла.
MainFileAdded	Событие возникает после открытия основного файла.

AddFile — метод (DocumentLoadSource)

Добавляет файл из указанного источника.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void AddFile(DocumentLoadSource documentLoadSource);
```

Параметры

documentLoadSource

Тип: DocumentLoadSource

Источник загрузки файла

AddFileAsync — метод (DocumentLoadSource)

Добавляет файл из указанного источника асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
Task AddFileAsync(DocumentLoadSource documentLoadSource);
```

Параметры

documentLoadSource

Тип: DocumentLoadSource

Источник загрузки файла

AddMainFSFile — метод

Добавляет основной файл из файловой системы.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void AddMainFSFile();
```

AddMainFSFileAsync — метод

Добавляет основной файл из файловой системы асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
Task AddMainFSFileAsync();
```

AddMainScanFile — метод

Добавляет основной файл со сканера асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void AddMainScanFile();
```

AddMainScanFileAsync — метод

Добавляет основной файл со сканера асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
Task AddMainScanFileAsync();
```

OpenMainFile — метод

Открывает основной файл.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
void OpenMainFile();
```

OpenMainFileAsync — метод

Открывает основной файл асинхронно.

- **Пространство имён:** DocsVision.BackOffice.Cards.CardDocument
- **Сборка:** DocsVision.BackOffice.Cards.dll

Синтаксис

```
Task<FileOpenInfo> OpenMainFileAsync();
```

DocsVision.BackOffice.DigitalSignature — пространство имён

Пространство имён DocsVision.BackOffice.DigitalSignature содержит классы для получения расширенной информации о цифровой подписи.

Классы

Класс	Описание
CertificateInfo	Представляет информацию о сертификате подписи
SignatureInfo	Представляет информацию о подписи
TimestampInfo	Представляет информацию о штампе времени подписи

Конструкторы

Конструктор	Описание
CertificateInfo	Инициализирует новый экземпляр класса CertificateInfo
SignatureInfo	Инициализирует новый экземпляр класса SignatureInfo
TimestampInfo	Инициализирует новый экземпляр класса TimestampInfo

Интерфейсы

Перечисления	Описание
IComplexSignatureService	Содержит методы для обращения к Java-сервису улучшения подписи

Перечисления

Перечисления	Описание
`TimestampType`	Представляет информацию о сертификате подписи

CertificateInfo — класс

Представляет информацию о сертификате подписи

- **Пространство имён:** `DocsVision.BackOffice.DigitalSignature`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class CertificateInfo
```

Конструкторы

Имя	Описание
`CertificateInfo(string serialNumber, string subjectName, DateTime notBefore, DateTime notAfter, string issuerName)`	Инициализирует новый экземпляр класса CertificateInfo.

Свойства

Имя	Описание
`SerialNumber`	Серийный номер сертификата
`SubjectName`	Имя того, кому выдан сертификат
`NotBefore`	Дата и время, с которых сертификат считается действительным
`NotAfter`	Дата и время, с которых сертификат считается недействительным
`IssuerName`	Имя компании, выдавшей сертификат
`CommonName`	`CN` из поля `Subject` в `Distinguished Name` Active Directory

Имя	Описание
OrganizationUnit	OU из поля Subject в Distinguished Name Active Directory
OrganizationName	O из поля Subject в Distinguished Name Active Directory

CertificateInfo — конструктор (string serialNumber, string subjectName, DateTime notBefore, DateTime notAfter, string issuerName)

Инициализирует новый экземпляр класса CertificateInfo.

- **Пространство имён:** DocsVision.BackOffice.DigitalSignature
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public CertificateInfo(string serialNumber, string subjectName, DateTime notBefore,
DateTime notAfter, string issuerName)
```

Параметры

serialNumber

Тип: String

Серийный номер сертификата

subjectName

Тип: String

Имя того, кому выдан сертификат

notBefore

Тип: System.DateTime

Дата и время, с которых сертификат считается действительным

notAfter

Тип: System.DateTime

Дата и время, с которых сертификат считается недействительным

issuerName

Тип: `String`

Имя компании, выдавшей сертификат

SignatureInfo — класс

Представляет информацию о подписи

- **Пространство имён:** `DocsVision.BackOffice.DigitalSignature`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class SignatureInfo
```

Конструкторы

Имя	Описание
`SignatureInfo(SignatureType signatureType, DateTime expireDate, CertificateInfo certificate, IEnumerable<TimestampInfo> timestamps)`	Инициализирует новый экземпляр класса `SignatureInfo`.

Свойства

Имя	Описание
`Type`	Получает тип подписи.
`ExpireDate`	Получает дату и время истечения срока действия подписи.
`Certificate`	Получает информацию о сертификате подписи.
`Timestamps`	Получает набор штампов времени в подписи.

SignatureInfo — конструктор (SignatureType signatureType, DateTime expireDate, CertificateInfo certificate, IEnumerable<TimestampInfo> timestamps)

Инициализирует новый экземпляр класса `SignatureInfo`.

- **Пространство имён:** DocsVision.BackOffice.DigitalSignature
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public SignatureInfo(SignatureType signatureType, DateTime expireDate, CertificateInfo
certificate, IEnumerable<TimestampInfo> timestamps)
```

Параметры

signatureType

Тип: SignatureType

Получает тип подписи

expireDate

Тип: System.DateTime

Получает дату и время истечения срока действия подписи

certificate

Тип: CertificateInfo

Получает информацию о сертификате подписи

timestamps

Тип: IEnumerable<TimestampInfo>

Получает набор штампов времени в подписи

TimestampInfo — класс

Представляет информацию о штампе времени подписи

- **Пространство имён:** DocsVision.BackOffice.DigitalSignature
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TimestampInfo
```

Конструкторы

Имя	Описание
`TimestampInfo (DateTime creationDate, DateTime expireDate, CertificateInfo certificate, TimestampType timestampType)`	Инициализирует новый экземпляр класса `TimestampInfo`.

Свойства

Имя	Описание
`CreationDate`	Дата и время формирования штампа времени.
`ExpireDate`	Дата и время истечения срока действия штампа времени.
`Certificate`	Сертификат подписи штампа.
`Type`	Тип штампа

TimestampInfo — конструктор (DateTime creationDate, DateTime expireDate, CertificateInfo certificate, TimestampType timestampType)

Инициализирует новый экземпляр класса `TimestampInfo`.

- **Пространство имён:** `DocsVision.BackOffice.DigitalSignature`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public TimestampInfo(DateTime creationDate, DateTime expireDate, CertificateInfo
certificate, TimestampType timestampType)
```

Параметры

creationDate

Тип: `System.DateTime`

Дата и время формирования штампа времени

expireDate

Тип: `System.DateTime`

Дата и время истечения срока действия штампа времени

timestampType

Тип: TimestampType

Тип штампа

TimestampType — перечисление

Представляет тип штампа времени

- **Пространство имён:** DocsVision.BackOffice.DigitalSignature
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum TimestampType
```

Члены

Имя члена	Описание
Unknown = 0	Неизвестный тип штампа
Cades_T = 1	Cades_T
Cades_XLT1 = 2	Cades_XLT1
Cades_A = 3	Cades_A

IComplexSignatureService — интерфейс

Содержит методы для обращения к Java-сервису улучшения подписи.

- **Пространство имён:** DocsVision.BackOffice.DigitalSignature
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IComplexSignatureService : IServiceWithWorkingState
```

Методы

Имя	Описание
`GetExpirationDate(SignatureActionRequest request)`	Получает дату и время окончания срока действия сертификата подписи
`GetSignatureType(byte[] signatureBytes)`	Получает тип подписи
`VerifySignature(SignatureActionRequest request)`	Возвращает результат проверки действительности подписи
`SignatureActionResponse EnhanceSignature(SignatureActionRequest request)`	Улучшает подпись
`SignatureActionResponse VerifyAndEnhanceSignature(SignatureActionRequest request)`	Проверяет и при необходимости улучшает подпись
`GetSignatureInfo(SignatureActionRequest request)`	Возвращает информацию об улучшении подписи

DocsVision.BackOffice.ObjectModel — пространство имён

В пространстве имён `DocsVision.BackOffice.ObjectModel` реализованы классы объектной модель уровня бизнес-логики для карточек библиотеки *Базовые объекты*.

Пространства имён

Пространство имён	Описание
`DocsVision.BackOffice.ObjectModel.Mapping`	Пространство имён `DocsVision.BackOffice.ObjectModel.Mapping` содержит фабрику преобразователей данных и сами преобразователи для базовых объектов.
`DocsVision.BackOffice.ObjectModel.Services`	Пространство имён `DocsVision.BackOffice.ObjectModel.Services` содержит фабрику сервисов и сервисы для работы с базовыми объектами.

Классы

Класс	Описание
`AccessInfo`	Результат проверки доступности операции для вида карточки и набора исполняемых ролей.
`AccessInfoOperation`	Сопоставляет операции доступность или недоступность её выполнения.
`AccessInfoRole`	Предоставляет специфичную для ролей информацию.
`BackOffice`	Предоставляет статические методы для получения ссылок на основные справочники BackOffice.
`BaseCard`	Базовый класс для объектно-ориентированных карточек библиотеки *Базовые объекты*.
`BaseCardNumber`	Объектная модель секции "Номера" базовой карточки.
`BaseCardExternalPowerOfAttorney`	Раздел списка подписей для хранения информации о машиночитаемых доверенностях контрагентов.
`BaseCardProcess`	Предоставляет ссылку и настройки бизнес-процесса.
`BaseCardProperty`	Предоставляет доступ к коллекции `BaseCardPropertyField`.
`BaseCardPropertyField`	Представляет описание ссылочного поля.
`BaseCardSectionRow`	Базовый класс для представления строки секции карточки.
`BaseCardSignature`	Содержит данные ЭП установленной на карточку.
`BaseCardSignaturePart`	Представляет часть ЭП.
`BaseCardSystemInfo`	Содержит системную информацию карточки (секция "Системные свойства").

Класс	Описание
BaseDictionaryCard	Базовый класс для справочников библиотеки *Базовые объекты*.
BaseUniversal	Объектная модель *Конструктора справочников*.
BaseUniversalItem	Представляет строку в узле *Конструктора справочников*.
BaseUniversalItemCard	Объектная модель *Карточки строки справочника*.
BaseUniversalItemCardMainInfo	Основная информация *Карточки строки справочника*.
BaseUniversalItemType	Представляет узел *Конструктора справочников*.
BaseUniversalItemTypeViewField	Представляет отображаемое поле узла *Конструктора справочников*.
BackOffice-ObjectModel-BuiltIn:BuiltInBranch_CL.adoc[BuiltInBranch]	Представляет встроенный переход в автомате состояний.
BackOffice-ObjectModel-BuiltIn:BuiltInOperation_CL.adoc[BuiltInOperation]	Предоставляет методы регистрации встроенных операций и получения их свойств.
BackOffice-ObjectModel-BuiltIn:BuiltInState_CL.adoc[BuiltInState]	Представляет встроенное состояние конструктора состояний.
BackOffice-ObjectModel-Calendar:Calendar_CL.adoc[Calendar]	Объектная модель карточки "Бизнес-календарь".
BackOffice-ObjectModel-Calendar:CalendarDay_CL.adoc[CalendarDay]	Определяет свойства рабочего дня в карточке *Бизнес-календарь*.
BackOffice-ObjectModel-Calendar:CalendarDefaultWorkTime_CL.adoc[CalendarDefaultWorkTime]	Данные стандартного рабочего времени для карточки *Бизнес-календарь*.
BackOffice-ObjectModel-Calendar:CalendarMainInfo_CL.adoc[CalendarMainInfo]	Основная информация карточки *Бизнес-календарь*.

Класс	Описание
BackOffice-ObjectModel-Calendar:CalendarWorkTime_CL.adoc[CalendarWorkTime]	Определяет отрезок рабочего времени в карточке *Бизнес-календарь*.
BackOffice-ObjectModel-Calendar:CalendarYear_CL.adoc[CalendarYear]	Календарный год карточки *Бизнес-календарь*.
Categories	Объектная модель *Справочника категорий*.
CategoriesCategory	Представляет категорию *Справочника категорий*.
CategoriesLinkedCategory	Представляет связанную категорию *Справочника категорий*.
CategoriesMain	Основная информация *Справочника категорий*.
CategoryList	Объектная модель карточки *Список категорий*.
CategoryListCategory	Представляет категорию в *Списке категорий*.
Document	Объектная модель карточки "Документ".
DocumentFile	Объектная модель секции "Файлы" карточки "Документ".
DocumentMainInfo	Объектная модель секции "Основная информация" карточки "Документ".
EmployeeCard	Класс EmployeeCard представляет объектную модель карточки сотрудника.
EmployeeCardMainInfo	Класс EmployeeCardMainInfo представляет объектную модель секции "Основная информация" *Карточки сотрудника*.
Kinds	Объектная модель карточки "Справочник видов карточек".

Класс	Описание
KindsCardCreationSetting	Представляет в *Справочнике видов карточек* настройки режима создания вида карточки.
KindsCardExtendedSetting	Параметры расширения *Справочника видов карточек*.
KindsCardExtendedSettingCollection	Представляет коллекцию настроек расширений видов карточек. Предоставляет метод получения настроек для расширения.
KindsCardExtendedSettingGroup	Группа настроек расширения *Справочника видов карточек*.
KindsCardExtendedSettingGroupCollection	Предоставляет коллекцию сгруппированных настроек расширения *Справочника видов карточек*.
KindsCardExtendedSettingGroupSetting	Настройки из группы настроек расширения *Справочника видов карточек*.
KindsCardExtendedSettingGroupSettingCollection	Коллекция настроек для группы расширения *Справочника видов карточек*.
KindsCardExtension	Объектная модель расширения *Справочника видов карточек*.
KindsCardKind	Представляет вид карточки объектной модели уровня бизнес-логики.
KindsCardProcess	Определяет параметры бизнес-процесса, запускаемого из карточки определённого вида
KindsCardProcessBranch	Переход состояний бизнес-процесса карточки "Справочник видов карточек" (секция "Переходы состояний").

Класс	Описание
KindsCardProcessOperation	Определяет операцию, указанную в качестве инициирующей запуск бизнес-процесса из определённого вида карточки.
KindsCardProcessVariable	Параметры бизнес-процесса карточки "Справочник видов карточек".
KindsCardType	Представляет объектную модель типа карточки, представленного в *Справочник видов карточек*.
Layouts	Объектная модель *Конструктора разметок*.
LayoutsCardKindSettings	Предоставляет настройки вида карточки для карточки "Конструктор разметок".
LayoutsColumnAttribute	Определяет атрибуты столбца для разметки в карточке "Конструктор разметок".
LayoutsDesignTree	Дерево параметров дизайна представления в карточке "Конструктор разметок" (секция "Дерево дизайнов")
LayoutsLayout	Определяет параметры разметки карточки "Конструктор разметок".
LayoutsLayoutAttribute	Определяет атрибуты разметки карточки "Конструктор разметок".
LayoutsNodeLayout	Разметка узла дерева дизайнов карточки "Конструктор разметок".
LayoutsProperty	Объектная модель секции "Свойство карточки" карточки "Конструктор разметок".
LinksDictionary	Объектная модель карточки "Справочник ссылок".

Класс	Описание
LinksLinkType	Тип ссылки, зарегистрированной в *Справочнике ссылок*.
LinksLinkTypeMapper	Преобразователь данных для секция "Типы ссылок" карточки "Справочник ссылок".
LinksMapper	Преобразователь данных для карточки "Справочник ссылок".
NumerationRules	Представляет *конструктор правил нумерации*.
NumerationRulesNumerator	Представляет нумератор в *конструкторе правил нумерации*.
NumerationRulesRule	Представляет правило нумерации *конструктора правил нумерации*.
NumerationRulesRuleItem	Элемент правила нумерации в *конструкторе правил нумерации*.
NumerationRulesRuleItemCondition	Условие в элементе правила нумерации *конструктора правил нумерации*.
Partners	Объектная модель *Справочника контрагентов*.
PartnersAddresse	Представляет адресные данные контрагента карточки "Справочник контрагентов".
PartnersAllDepViewField	Отображаемые поля подразделений карточки "Справочник контрагентов".
PartnersAllEmplViewField	Отображаемые поля сотрудников карточки "Справочник контрагентов".
PartnersAllGrpViewField	Отображаемые поля группы карточки "Справочник контрагентов".
PartnersBankAccount	Банковские реквизиты контрагента карточки "Справочник контрагентов".

Класс	Описание
PartnersChEnumValue	Значения перечисления для сотрудников контрагента карточки "Справочник контрагентов".
PartnersChProperty	Свойства для сотрудников контрагента карточки "Справочник контрагентов".
PartnersChSelectedValue	Выбранные значения сотрудников контрагента карточки "Справочник контрагентов".
PartnersCode	Коды контрагента карточки "Справочник контрагентов".
PartnersCompany	Организация контрагента в *Справочнике контрагентов*.
PartnersContact	Контакты контрагента карточки "Справочник контрагентов".
PartnersDepartmentCard	Объектная модель карточки "Карточка подразделения контрагента".
PartnersDepartmentCardMainInfo	Объектная модель секции "Основная информация" карточки "Карточка подразделения контрагента" (не используется).
PartnersDepViewField	Отображаемые поля подчиненных подразделений контрагента карточки "Справочник контрагентов".
PartnersEmployee	Представляет сотрудника контрагента.
PartnersEmployeeCard	Объектная модель карточки "Карточка сотрудника контрагента".
PartnersEmployeeCardMainInfo	Объектная модель секции "Основная информация" карточки "Карточка сотрудника контрагента".
PartnersEmplViewField	Отображаемые поля сотрудников подразделения карточки "Справочник контрагентов".

Класс	Описание
PartnersEnumValue	Значения перечисления в свойстве контрагента карточки "Справочник контрагентов".
PartnersGroup	Группа подразделений контрагента в *Справочнике контрагентов*.
PartnersGroupGroup	Представляет подразделение контрагента в группе *Справочника контрагентов*.
PartnersGrpViewField	Отображаемые поля группы карточки "Справочник контрагентов".
PartnersNameCase	Определяет падеж имени сотрудника контрагента карточки "Справочник контрагентов".
PartnersOrgType	Тип юридического лица в *Справочнике контрагентов*.
PartnersPosition	Представляет должность сотрудника контрагента в *Справочнике контрагентов*.
PartnersProperty	Свойства контрагента карточки "Справочник контрагентов".
PartnersSelectedValue	Выбранные значения в свойстве контрагента карточки "Справочник контрагентов".
PartnersTabSection	Раздел свойств контрагента карточки "Справочник контрагентов".
PartnersTitle	Обращения карточки "Справочник контрагентов".
PartnersUserSetting	Пользовательские настройки карточки "Справочник контрагентов".
ReferenceList	Объектная модель карточки *Список ссылок на карточки*.
ReferenceListReference	Ссылка в *Списке ссылок на карточки*.

Класс	Описание
RoleModel	Объектная модель карточки "Конструктор ролей".
RoleModelCardField	Представляет поле для карточки "Конструктор ролей".
RoleModelCardKindRoleSetting	Настройка прав в карточке "Конструктор ролей".
RoleModelConditionDayWorkStatus	Передает статус дня согласно данным бизнес-календаря.
RoleModelConditionTimeWorkStatus	Передает статус времени согласно данным бизнес-календаря.
RoleModelConditionWorkStatus	Содержит идентификатор бизнес-календаря.
RoleModelCustomOperation	Представляет пользовательскую операцию в ролевой модели.
RoleModelCustomParameter	Представляет пользовательский параметр в ролевой модели.
RoleModelLink	Связывает статус операции в ролевой модели.
RoleModelRole	Представляет роль в ролевой модели.
RoleModelRoleCondition	Условие в ролевой модели.
RoleModelRoleConditionGroup	Предоставляет группу условий для ролевой модели.
Scripting	Объектная модель карточки "Конструктор скриптов".
ScriptingAssembly	Сборка для скрипта в карточке "Конструктор скриптов".
ScriptingScript	Класс ScriptingScript представляется скрипт из *Конструктора скриптов*.
ScriptingScriptCode	Текст скрипта в карточке "Конструктор скриптов".

Класс	Описание
`ServerCard`	Объектная модель карточки "Карточка сервера".
`ServersDictionary`	Объектная модель карточки "Справочник серверов".
`ServersMainInfo`	Объектная модель секции "Основная информация" карточки "Справочник серверов".
`ServersServer`	Сервер карточки "Справочник серверов".
`SessionProvider`	Реализация интерфейса ISessionProvider.
`SignatureLabel`	Метка подписи из Справочника меток подписей.
`SignatureLabelName`	Локализованное имея метки для карточки "Справочник меток подписей".
`SignatureLabelsDictionary`	Объектная модель карточки "Справочник меток подписей".
`SignatureList`	Объектная модель карточки "Список подписей".
`Staff`	Класс `Staff` представляет объектную модель *Справочника сотрудников*.
`StaffAddresse`	Предоставляет адресные данные организации в *Справочнике сотрудников*.
`StaffADsMapping`	Определяет соответствие между атрибутом Active Directory и названием поля в справочнике сотрудников.
`StaffAllDepViewField`	Отображаемые поля подразделений карточки "Справочник сотрудников".
`StaffAllEmplViewField`	Отображаемые поля сотрудников карточки "Справочник сотрудников".

Класс	Описание
StaffAllGrpViewField	Отображаемые поля группы карточки "Справочник сотрудников".
StaffChEnumValue	Значения перечисления для сотрудников карточки "Справочник сотрудников".
StaffChProperty	Свойства для сотрудников карточки "Справочник сотрудников".
StaffChSelectedValue	Выбранные значения сотрудников карточки "Справочник сотрудников".
StaffContain	Представляет контейнер роли в *справочнике сотрудников*.
StaffDeputy	Объектная модель заместителя сотрудника в *справочнике сотрудников*.
StaffDepViewField	Отображаемые поля подчиненных подразделений карточки "Справочник сотрудников".
StaffEmployee	Представляет сотрудника подразделения из справочника сотрудников.
StaffEmployeesFormat	Формат отображения данных сотрудников подразделения.
StaffEmplViewField	Отображаемые поля сотрудников подразделения карточки "Справочник сотрудников".
StaffEnumValue	Значения перечисления в свойстве карточки "Справочник сотрудников".
StaffGroup	Группа сотрудников *Справочника сотрудников*.
StaffGroupFolder	Представляет папку определённую в параметрах группы пользователей в *Справочнике сотрудников*.

Класс	Описание
StaffGroupItem	Представляет сотрудника в группе *Справочника сотрудников*.
StaffGrpViewField	Отображаемые поля группы в карточке "Справочник сотрудников".
StaffNameCase	Определяет падеж имени сотрудника карточки "Справочник сотрудников".
StaffPicture	Фотография сотрудника в *справочнике сотрудников*.
StaffPosition	Объектная модель должности сотрудника в *справочнике сотрудников*.
StaffProperty	Свойства карточки "Справочник сотрудников".
StaffRole	Представляет роль в *справочнике сотрудников*.
StaffRoleFolder	Представляет папку роли в *справочнике сотрудников*.
StaffSelectedValue	Выбранные значения в карточке "Справочник сотрудников".
StaffTabSection	Раздел свойств карточки "Справочник сотрудников".
StaffUnit	Объектная модель подразделения из *Справочника сотрудников*.
StaffUserSetting	Пользовательская настройка в карточке "Справочник сотрудников".
StatesCardKindStateSetting	Представляет настройки вида карточки, заданные в *Конструкторе состояний*.
StatesDictionary	Объектная модель карточки "Конструктор состояний".

Класс	Описание
StatesOperation	Представляет операцию зарегистрированную в конструкторе состояний.
StatesOperationCollection	Представляет коллекцию объектов типа StatesOperation.
StatesOperationDescription	Описание операции в *конструкторе состояний*.
StatesOperationName	Локализованное название операции в *конструкторе состояний*.
StatesState	Представляет состояние из конструктора состояний.
StatesStateCollection	Представляет коллекцию объектов типа StatesState.
StatesStateMachineBranch	Представляет переход автомата состояний.
StatesStateMachineBranchCollection	Представляет коллекцию объектов типа StatesStateMachineBranch.
StatesStateMachineLayout	Класс StatesStateMachineLayout представляет разметку автомата состояний в *Конструкторе состояний*.
StatesStateName	Локализованное имя состояния в карточке "Конструктор состояний".
SurveyList	Объектная модель карточки "Список опросов".
SurveyListSurvey	Опрос в карточке "Список опросов".
SurveyListSurveyAnswer	Ответы на вопросы в карточке "Список опросов".
SurveyListSurveyAnswerIssue	Набор вопросов в карточке "Список опросов".
SurveyListSurveyAnswerIssueValue	Значение в наборе вопросов в карточке "Список опросов".

Класс	Описание
SurveyListSurveyQuestion	Вопроса в карточке "Список опросов".
SurveyListSurveyQuestionEnumValue	Значения перечисления в карточке "Список опросов".
Task	Объектная модель карточки *Задание*.
TaskActualDelegate	Предоставляет данные актуального делегата карточки *Задание*.
TaskComment	Комментарий к заданию в карточке "Задание".
TaskCompletionAdditionalOption	Дополнительные опции завершения задания в карточке "Задание".
TaskCompletionOption	Варианты завершения задания в карточке "Задание".
TaskCompletionOptionAdditionalField	Дополнительные атрибуты варианта завершения задания.
TaskCompletionParameter	Параметры завершения задания.
TaskCurrentPerformer	Класс TaskCurrentPerformer представляет текущего исполнителя задания
TaskDelegate	Список делегирования задания в карточке "Задание".
TaskDelegatedPerformer	Исполнитель задания в карточке "Задание".
TaskDelegatedTo	Предоставляет информацию о том, кому было делегировано задание.
TaskGroup	Объектная модель карточки "Группа заданий".
TaskGroupMainInfo	Объектная модель секции "Основная информация" карточки "Группа заданий".
TaskGroupPresets	Представляет индивидуальные настройки исполнителя группы заданий.

Класс	Описание
TaskGroupPresetsDelegate	Представляет исполнителя в индивидуальных настройках исполнителя группы заданий.
TaskGroupSelectedPerformer	Выбранный исполнитель группы заданий.
TaskList	Объектная модель карточки "Список ссылок на карточки заданий".
TaskListTask	Класс TaskListTask представляется объектную модель задания, определённую в списке заданий.
TaskListTaskGroup	Класс TaskListTaskGroup представляет объектную модель группы заданий, определённую в списке заданий.
TaskMainInfo	Объектная модель секции "Основная информация" карточки "Задание".
TaskPerformer	Класс TaskPerformer представляет назначенного исполнителя здания.
TaskPreset	Настройки задания в карточки "Задание".
TaskPresetAttachmentLinkType	Настройка дополнительных типов ссылок в карточки "Задание".
TaskPresetChildCopyField	Настройка копирования карточки "Задание".
TaskPresetChildKind	Класс TaskPresetChildKind предоставляет настройки вида подчинённого задания.
TaskPresetChildKindSetting	Представляет вид задания, доступный для создания подчиненного задания.
TaskPresetCompletion	Настройки завершения задания карточки "Задание".
TaskPresetDelegate	Предоставляет параметры выбора делегата для задания.

Класс	Описание
TaskPresetGroupChildKind	Настройки вида подчиненной группы заданий карточки "Задание".
TaskPresetGroupChildKindSetting	Вид, доступный для создания подчиненной группы заданий карточки "Задание".
TaskPresetMainLinkType	Настройка основных типов ссылок в задание карточки "Задание".
TaskPresetReportLinkType	Настройка типов ссылок отчётов карточки "Задание".
TaskPresetRouting	Настройка маршрутизации карточки "Задание".
TaskSelectedPerformer	Класс TaskSelectedPerformer представляет выбранного исполнителя задания.
TasksTreeInfo	Инициализирует и представляет методы загрузки дерева задания.
TaskTreeInfo	Класс TaskTreeInfo возвращает информацию из узлу дерева заданий, полученного из списка заданий.
TaskTreeInfoDelegate	Содержит методы управления делегированием для дерева заданий.
TaskTreeInfoPerformer	Исполнитель задания в узле дерева заданий.

Интерфейсы

Интерфейс	Описание
IGridViewField	Добавляет к таблице представления возможность управления сортировкой.
IScriptable	Добавляет возможность хранения скрипта *Конструктора скриптов*.

Перечисления

Перечисление	Описание
AccessInfoOperationResult	Определяет режим доступа к операции.
BackOffice-ObjectModel-Calendar:CalendarDayType_EN.adoc[CalendarDayType]	Определяет тип календарного дня в карточке *Бизнес-календарь*.
DeputyAccessRights	Определяет права заместителя сотрудника в *Справочнике сотрудников*.
DocumentFileType	Определяет тип файла документа.
DocumentVersioningType	Определяет тип версий файла для карточки "Документ".
KindsCardCreationSettingLocation	Определяет способ размещения карточки определённого вида.
KindsCardProcessPolicy	Определяет политику запуска бизнес-процесса для вида карточки.
KindsCardProcessVariableSync	Тип синхронизации параметров.
LayoutsDesignTreeType	Определяет тип узла дерева дизайнов.
LayoutsLayoutAttributeVisibility	Определяет режим отображения атрибута разметки.
LayoutsPropertyItem	Определяет тип элемента управления в разметке.
LayoutsPropertyType	Определяет тип свойства в конструкторе разметок.
NumerationRulesNumeratorZoneType	Определяет режим обновления зоны нумератора.
NumerationRulesRuleItemLexeme	Тип префикса при создании номеров нумератора.
PartnersAddresseAddressType	Определяет тип адреса контрагента.
PartnersChPropertyParamType	Определяет тип параметра в свойствах сотрудника контрагента.

Перечисление	Описание
PartnersCompanyType	Определяет тип подразделения контрагента в *Справочнике контрагентов*.
PartnersEmployeeGender	Пол сотрудника контрагента.
PartnersNameCaseNameCase	Определяет падежи имён для сотрудников контрагента.
PartnersPropertyParamType	Определяет тип параметра для свойства контрагента карточки *Справочник контрагентов*.
PartnersPropertyShowType	Определяет режим отображения свойства *Справочника контрагентов*.
PartnersUserSettingOpenMode	Определяет пользовательские настройки режима открытия карточки контрагента .
PartnersUserSettingSearchFor	Определяет пользовательские настройки области поиска контрагента.
RoleModelConditionValueDayOfWeek	Определяет дни недели для условий, создаваемых для ролевой модели.
RoleModelConditionValueDayWorkStatus	Определяет статусы дня недели для условий, создаваемых для ролевой модели.
RoleModelConditionValueTimeWorkStatus	Определяет статус времени для условий, создаваемых для ролевой модели.
RoleModelCustomOperationParameter	Определяет тип параметра для пользовательской операции. Используется ролевой моделью
RoleModelCustomOperationValueType	Определяет тип значения для пользовательской операции. Используется ролевой моделью
RoleModelCustomParameterType	Определяет тип пользовательского параметра. Используется ролевой моделью

Перечисление	Описание
RoleModelOperationStatus	Определяет статус операции. Используется ролевой моделью
RoleModelRoleConditionGroupOperation	Определяет операцию группы условий для роли. Используется ролевой моделью
RoleModelRoleConditionOperation	Определяет операцию условия для роли. Используется ролевой моделью
RoleModelRoleConditionParameter	Определяет предустановленный параметр для условий. Используется ролевой моделью
ScriptingLanguage	Определяет языки программирования, используемые *Конструктором скриптов*.
SignatureType	Определяет типы подписи.
StaffAddresseAddressType	Определяет тип адреса организации в *Справочнике сотрудников*.
StaffChPropertyParamType	Определяет тип параметра в свойствах сотрудника в *Справочнике сотрудников*.
StaffContainRefType	Тип ссылки в *Справочнике сотрудников*.
StaffGroupRole	Определяет роли сотрудника в рабочей группе.
StaffEmployeeGender	Пол сотрудника в справочнике сотрудников.
StaffEmployeeInactiveStatus	Определяет возможные состояния сотрудника в период его неактивности.
StaffEmployeeRoutingType	Определяет тип маршрутизации для сотрудника в справочнике сотрудников.
StaffEmployeeStatus	Определяет возможные состояния сотрудника организации.

Перечисление	Описание
StaffNameCaseNameCase	Определяет падежи имён для сотрудников.
StaffPictureImageFormat	Определяет тип сжатия хранимой фотографии сотрудника в *справочнике сотрудников*.
StaffPropertyParamType	Определяет тип параметра для свойства подразделения в справочнике сотрудников.
StaffPropertyShowType	Определяет режим вывода параметра для свойства подразделения в справочнике сотрудников.
StaffUnitType	Определяет тип подразделения в справочнике сотрудников.
StaffUserSettingOpenMode	Определяет, для пользовательских настроек, режим открытия карточки справочника сотрудников.
StaffUserSettingSearchFor	Определяет область поиска в справочнике сотрудников.
StatesStateMachineBranchBranchType	Определяет тип перехода состояния в автомате состояний.
SurveyListSurveyQuestionDataType	Определяет типы значений в вопросе карточки "Список опросов".
TaskCompletionOptionAdditionalFieldShowDialog	Определяет необходимость отображения поля в диалоге завершения.
TaskDelegateReason	Определяет возможные причины делегирования задания.
TaskGroupExecutionType	Типы выполнения этапов группы заданий.
TaskGroupPerformanceControl	Определяет наличие контроля производительности выполнения группы заданий.

Перечисление	Описание
`TaskGroupUrgency`	Определяет срочность задания в группе заданий.
`TaskPriority`	Определяет важность задания.

DocsVision.BackOffice.ObjectModel.Mapping — пространство имён

Содержит фабрику преобразования данных и сами преобразователи для базовых объектов.

Классы

Класс	Описание
`BackOfficeMapperFactory`	Фабрика преобразователей данных для библиотеки *Базовые объекты*.
`BaseCardMapper`	Базовый класс для преобразователей данных карточек библиотеки карточек *Базовые объекты*.

BackOfficeMapperFactory — класс

Фабрика преобразователей данных для библиотеки *Базовые объекты*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Mapping`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class BackOfficeMapperFactory : ObjectMapperFactory
```

Конструкторы

Имя	Описание
`BackOfficeMapperFactory(ObjectContext)`	Инициализирует новый экземпляр класса `BackOfficeMapperFactory`.

Заметки

При инициализации регистрирует преобразователи данных все объектной модели библиотеки *Базовые объекты*.

BaseCardMapper<T> — класс

Базовый класс для преобразователей данных карточек библиотеки карточек *Базовые объекты*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Mapping
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public abstract class BaseCardMapper<T> : CardMapper<T> where T : BaseCard
```

Свойства

Имя	Описание
Mapping	Возвращает карту преобразователя данных.

Методы

Имя	Описание
CreateObject(T)	При переопределении в производном классе должен создавать экземпляр объекта типа T.
LoadObject(ObjectRef<T>, IDictionary<String, Object>)`	Получает данные карточки из хранилища и загружает их в новый экземпляр карточки.

BaseCardMapper<T>.Mapping — свойство

Возвращает карту преобразователя данных.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Mapping
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public override ObjectMap Mapping { get; }
```

Значение свойства

Тип: `ObjectMap`

Карта преобразователя

DocsVision.BackOffice.ObjectModel.Services — пространство имён

Пространство имён `DocsVision.BackOffice.ObjectModel.Services` содержит фабрику сервисов для библиотеки *Базовые объекты*.

Пространства имён

Пространство имён	Описание
`DocsVision.BackOffice.ObjectModel.Services.Entities`	Пространство имён `DocsVision.BackOffice.ObjectModel.Services` содержит фабрику сервисов и сервисы для работы с базовыми объектами.

Классы

Класс	Описание
`BackOfficeServiceFactory`	Фабрика сервисов для библиотеки *Базовые объекты*
`ServiceError`	Содержит статические методы формирования исключений, возникающих при работе сервисов.

Интерфейсы

Интерфейс	Описание
`IAccessCheckingService`	Предоставляет методы получения списка ролей и доступных операций для сотрудника, а также методы сброса кэша ролевой модели.
`IBarcodeService`	Определяет методы генерации и назначения штрих-кодов карточки.
`IBaseCardService`	Данный сервис определяет общие для всех базовых объектов методы.

Интерфейс	Описание
IBaseUniversalService	Описывает сервис для работы с *Конструктором справочников*.
ICalendarService	Сервис для работы с бизнес-календарем.
ICategoriesService	Сервис для работы со *Справочником категорий*
ICategoryListService	Сервис для работы со списком категорий.
ICryptService	Сервис для работы с шифрованием базовых объектов.
IDocumentService	Представляет сервис для работы с документами.
IKindService	Сервис для работы со справочником видов карточек.
ILayoutService	Описывает сервис для работы с *Конструктором разметок*.
ILinkService	Представляет сервис для работы со *Справочником ссылок*.
ILockService	Сервис управления блокировкой объектов.
ILogService	Сервис для работы с историей карточки.
INumerationRulesService	Сервис для работы с конструктором правил нумерации.
INumeratorCardService	Сервис для работы с карточкой нумератора.
IOcspService	Описывает функциональность сервиса для работы с центром установления статуса сертификатов по протоколу OCSP.
IPartnersService	Описывает сервис для работы со справочником контрагентов.

Интерфейс	Описание
IPowerOfAttorneyMachineReadableProvider	Определяет методы формирования и чтения машиночитаемой доверенности.
IPowerOfAttorneyService_IN	Предоставляет методы для работы с системной карточкой доверенности.
IReferenceListService	Сервис для работы со списками ссылок на карточки.
IRoleModelService	Сервис для работы с конструктором ролей.
IScriptingService	Сервис для работы с конструктором скриптов.
IServerExtensionProxyService	Предоставляет доступ к некоторым методам серверных расширений, специфичным для карточек BackOffice.
IServersService	Описывает сервис для работы со *Справочником серверов*.
ISettingsCardService	Сервис для работы с системными настройками.
ISettingsService	Определяет вспомогательные методы создания и получения объектов содержащихся в значениях настроек расширений справочника видов.
ISignatureLabelService	Описывает сервис для работы со справочником меток подписей.
ISignatureService	Предоставляет сервис подписи.
IStaffService	Сервис для работы со *Справочником сотрудников*.
IStateService	Сервис для взаимодействия с *конструктором состояний*.
ISurveyService	Сервис для работы со списком опросов.
ITaskGroupService	Сервис для работы со группами заданий.

Интерфейс	Описание
ITaskListService	Сервис ITaskListService предоставляет методы для работы со списками заданий.
ITaskService	Сервис ITaskService предоставляет методы создания и управления заданиями.
ITspService	Описывает функциональность сервиса для работы со службой штампов времени по протоколу TSP.
IUserProfileCardService	Описывает сервис для работы с карточкой настроек пользователя.
IVersionedFileCardService	Сервис для работы с карточкой версионного файла.

DocsVision.BackOffice.ObjectModel.Services.Entities — пространство имён

Пространство имён DocsVision.BackOffice.ObjectModel.Services.Entities содержит общие типы для различных видов карточек.

Пространства имён

Пространство имён	Описание
ActiveDirectory	В пространстве имён DocsVision.BackOffice.ObjectModel.Services.Entities.ActiveDirectory представлены типы доступные для синхронизации с доменом ActiveDirectory.
KindSetting	Пространство имён DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting содержит типы, определяющие параметры вида карточки, определённые в справочнике видов карточек.

Классы

Класс	Описание
BaseCardSignatureVerification	Возвращает информацию по проверке подписи на карточке.
BusinessProcessCancelEventArgs	Поставляет данные для событий BeforeBusinessProcessStart.
BusinessProcessVariableEventArgs	Поставляет данные для событий BusinessProcessVariableInitialized.
CheckSignatureResult	Представляет результат проверки подписи.
CompletedTaskInfo	Представляет идентификатор процесса.
CustomOperation	Предоставляет для производного класса текст базового скрипта унаследованный от CustomOperation.
CustomParameter	Предоставляет для производного класса текст базового скрипта унаследованный от CustomParameter.
DocumentAdditionalFileEventArgs	Поставляет данные для событий AdditionalFileAdded и AdditionalFileRemoved.
DocumentCancelLockedEventArgs	Поставляет данные для событий MainFileCancelLocked.
DocumentMainFileEventArgs	Поставляет данные для событий MainFileAdded, MainFileCurrentVersionChanged, MainFileLocked и MainFileRemoved.
DocumentMainFileEventArgs	Поставляет данные для событий MainFileAdded, MainFileCurrentVersionChanged, MainFileLocked и MainFileRemoved.
DocumentSignatureEventArgs	Поставляет данные для событий SignatureAdded.
DocumentUnlockedEventArgs	Поставляет данные для событий MainFileUnlocked.

Класс	Описание
DocumentUnlockingEventArgs	Поставляет данные для событий MainFileUnlocking.
ErrorInfo	Содержит информацию по созданному исключению.
LockedTaskInfo	Предоставляет хранилище свойств для процесса блокировки объекта.
LogMessage	Сообщение для журнала.
LogMessageEventArgs	Поставляет данные для событий LogMessage.
MailNotificationInfo	Предоставляет данные электронного сообщения для сервиса отправки электронной почты.
PerformerFavoritesManager	Создаёт и сохраняет в профиль пользователя список избранных исполнителей.
PerformerSearchWord	Представляет данные поискового слова.
PerformerSearchWordsManager	Представляет статические методы работы с поисковыми словами.
PowerOfAttorneyData	Базовый класс сведений о доверенности.
PowerOfAttorneyDigest	Представляет собой сведения о доверенности
PowerOfAttorneyEMCHDData	Сведения о доверенности в формате единой формы доверенности
AddressInfo	Сведения об адресе (АдрТип)
BranchOfRussianEntityInfo	Сведения о филиале (обособленном подразделении) юридического лица (ФилПерПолн)
ConfirmationOfAuthorityDocument	Документ, подтверждающий полномочия лица, действующего без доверенности (ДокПдтвТип)

Класс	Описание
DelegatedAuthorityPrincipalsInfo	Сведения о лице (лицах), передавшем (передавших) полномочия (СвПередПолн)
DelegatedPowerOfAttorneyPrincipal	Сведения о лице, передавшем полномочия (ПередПолн)
FIO	Фамилия, имя, отчество (при наличии) (ФИОТип)
ForeignEntityInfo	Сведения об иностранном юридическом лице (СвИнОргТип)
ForeignLegalEntityPrincipalInfo	Сведения об иностранном юридическом лице (или филиале (аккредитованном представительстве) иностранного юридического лица) (ИнОргДоверТип)
HandwrittenSignature	Рукописная подпись (ПодпРукопис)
IndividualInfo	Сведения о физическом лице (СведФЛТип)
IndividualPrincipalInfo1	Сведения о физическом лице (ФЛПерПолн)
IndividualPrincipalInfo	Сведения о физическом лице (доверителе) (ФЛДоверТип)
IrrevocablePowerOfAttorneyInfo	Сведения о безотзывной доверенности (Безотзыв)
LegalEntityInfo	Сведения о юридическом лице (или филиале (обособленном подразделении) юридического лица) (СвОргТип)
MachineReadablePowerRestrictions	Сведения об ограничении (ограничениях) к полномочию (ОгрСвПолн)
MachineReadablePowersInfo	Сведения о машиночитаемом полномочии (машиночитаемых полномочиях) (МашПолн)

Класс	Описание
NotaryCertificateInfo	Сведения о нотариальном удостоверении (СвНотУдТип)
NotaryDeputyInfo	ВРИО нотариуса (ВрионНотТип)
NotaryInfo	Сведения о нотариусе, совершившем нотариальное действие (СвНотДействТип)
PowerOfAttorneyDocument	Состав и структура документа (Документ)
PowerOfAttorneyDocumentData	Доверенность (Довер)
PowerOfAttorneyInfo	Сведения о доверенности (СвДовТип)
PrimaryPowerOfAttorneyInfo	Сведения о первоначальной доверенности (или доверенности, на основании которой осуществляется передоверие) (СвПервДоверТип)
PrimaryPowerOfAttorneyPrincipal	Сведения о доверителе (доверителях) первоначальной доверенности (СвДоверПерв)
PrimaryPowerOfAttorneyPrincipalInfo	Сведения о доверителе (доверителях) первоначальной доверенности (СвДоверПерв)
PrincipalInfo	Сведения о доверителе (Доверит)
PrincipalWithoutPowerOfAttorneyInfo	Сведения о лице (лицах), в том числе законном представителе, действующем (действующих) без доверенности (ЛицоБезДовТип)
RetrustPowerOfAttorneyDocumentData	Передоверие (Передов)
RussianLegalEntityPrincipalInfo	Сведения о юридическом лице (РосОргДоверТип)

Класс	Описание
SoleExecutiveIndividualInfo	Сведения о физическом лице / руководителе юридического лица (иностранного юридического лица (или филиала (аккредитованного представительства) иностранного юридического лица)) (СвФЛТип)
SoleExecutiveManagementCompanyInfo	Сведения о единоличном исполнительном органе – управляющей компании (СВЮЛ)
SoleProprietorInfo0	Сведения об индивидуальном предпринимателе (СведИПТип)
PrincipalsInfo	Сведения о доверителе (доверителях) (СвДоверит)
RepresentativesInfo	Сведения о представителе (представителях) (СвУпПредТип)
RepresentativePowersInfo	Сведения о полномочиях представителя (представителей) (СвПолнТип)
RepresentativesInfo	Сведения о представителе (представителях) (СвУпПредТип)
PowerOfAttorneyFNSData	Базовый класс сведений о доверенности в формате ФНС.
PowerOfAttorneyFNSDOVBBData	Сведения о доверенности ФНС в формате DOVBB.
AddressInfo	Сведения об адресе (АдрТип)
BasicPowerOfAttorneyInfo	Сведения об Основной доверенности (СвОснДовер)
BasicPowerOfAttorneyPrincipalInfo	Сведения о доверителе Основной доверенности (СвДовер0)
BranchManagerInfo	Сведения о руководителе обособленного подразделения (СвРукОП)

Класс	Описание
ConfirmationOfAuthorityDocument	Реквизиты документа, подтверждающего полномочия (РеквДокПдтвТип)
DelegatedAuthorityPrincipalInfo	Сведения о лице, передавшем полномочия (СвЛицПередПолн)
ElectronicDocumentTransferMethod	Способ передачи электронного нотариального документа (СпПрдЭНотДок)
FIO	Фамилия, имя, отчество (при наличии) (ФИОТип)
ForeignEntityInfo	Сведения об иностранном юридическом лице (СвИнОргТип)
ForeignLegalEntityPrincipalInfo	Сведения о доверителе – иностранном юридическом лице (ИнОргДовер)
HandwrittenSignature	Рукописная подпись (ПодпРукопис)
IdentityCardOfIndividual	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип)
IndividualDelegatedAuthorityInfo	Сведения о лице, передавшем полномочия – физическом лице (ФЛПрдПолн)
IndividualInfo0	Сведения по физическому лицу (СвФЛ)
IndividualInfo1	Сведения по физическому лицу (СвПоФЛ)
IndividualInfo2	Сведения о физическом лице (СведФизЛТип)
IndividualInfo	Сведения о физическом лице (СведФЛТип)
IndividualInfoBase	Управляет получением сведений о физическом лице.
IndividualPrincipalInfo	Сведения о доверителе – физическом лице (ФЛДоверТип)

Класс	Описание
IrrevocablePowerOfAttorneyInfo	Сведения о безотзывной доверенности (БезотзывТип)
LegalEntityInfo	Сведения об организации (СвОргТип)
LegalRepresentativeInfo	Сведения о законном представителе физического лица (СвЗакПредТип)
NotaryCertificateInfo	Сведения о нотариальном удостоверении (СвНотУдТип)
NotaryDeputyInfo	ВРИО нотариуса (ВрионНот)
NotaryInfo	Сведения о нотариусе, совершившем нотариальное действие (СвНотДейств)
NotaryPaymentInfo	Сведения об оплате за совершение нотариального действия (ОплатНотДейст)
OrganizationInfo	Сведения об организации (СвОрг)
PowerOfAttorneyDocument	Состав и структура документа (Документ)
PowerOfAttorneyDocumentData	Доверенность (Довер)
PowerOfAttorneyInfo	Сведения доверенности (СвДовТип)
PrincipalInfo	Сведения о доверителе (СвДоверит)
PrincipalWithoutPowerOfAttorneyInfo	Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов)
RepresentativeInfo	Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип)
RepresentativePowerInfo	Сведения о полномочиях представителя (представителей) (СвПолнТип)
RetrustPowerOfAttorneyInfo	Сведения доверенности, выданной в порядке передоверия (СвДовПер)
RetrustPowerOfAttorneyInfoData	Передоверие (Передов)

Класс	Описание
RussianEntityInfo	Сведения о российском юридическом лице (СвРосОргТип)
RussianLegalEntityPrincipalInfo	Сведения о доверителе – российском юридическом лице (РосОргДовер)
SoleProprietorInfo0	Сведения об индивидуальном предпринимателе (СведИПТип)
SoleProprietorInfo1	Сведения об индивидуальном предпринимателе (СвИПТип)
PowerOfAttorneyFNSDOVEL502Data_CL	Данные доверенности в формате, подтверждающем полномочия представителя налогоплательщика, версия 5.02.
PowerOfAttorneyDocument	Состав и структура документа (Документ). Код формы по КНД принимает значение 1110310 в соответствии с форматом и не может быть изменён.
PowerOfAttorneyDocumentData	Доверенность (Довер) — Таблица 4.3
PowerOfAttorneyInfo	Сведения доверенности (СвДов) — Таблица 4.4
PrincipalInfo	Сведения о доверителе (СвДоверит) — Таблица 4.5
RussianCompanyInfo	Сведения о российской организации (НПЮЛ) — Таблица 4.6
PrincipalWithoutPowerOfAttorneyInfo	Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов) — Таблица 4.7
IndividualInfo	Сведения по физическому лицу (СвФЛ) — Таблица 4.8
ForeignCompanyInfo	Сведения об иностранной организации (ИО) — Таблица 4.9

Класс	Описание
SeparateSubdivisionManagerInfo	Сведения о руководителе обособленного подразделения (СвРукОП) — Таблица 4.10
RetrustPowerOfAttorneyDocumentData	Передоверие (Передов) — Таблица 4.11
RetrustPowerOfAttorneyInfo	Сведения доверенности, совершённой (выданной) в рамках передоверия (СвДовП) — Таблица 4.12
PrimaryPowerOfAttorneyInfo	Сведения об основной доверенности (СвОснДов) — Таблица 4.13
PrimaryPrincipalInfo	Сведения о доверителе основной доверенности (СвДоверитОсн) — Таблица 4.14
ForeignCompanyInfo1	Сведения о доверителе – иностранном юридическом лице (ДоверитИО) — Таблица 4.15
ParentPrincipalInfo	Сведения о доверителе из доверенности, на основании которой осуществляется передоверие (СвДоверщП) — Таблица 4.16
RetrustPrincipalInfo	Сведения о доверителе в порядке передоверия (СвДоверщ) - 4.17
OrganizationInfo	Сведения об организации (СвОргТип) — Таблица 4.18
RussianCompanyInfo1	Сведения о российском юридическом лице (СведЮЛТип) — Таблица 4.19
RepresentativeInfo	Сведения об уполномоченном представителе (уполномоченных представителях) (СвПредТип) — Таблица 4.20
IndividualInfo2	Сведения о физическом лице, в том числе индивидуальном предпринимателе (СвФизЛицТип) — Таблица 4.21

Класс	Описание
IndividualInfo3	Сведения по физическому лицу (СведФЛТип) — Таблица 4.22
AddressInfo	Адрес доверителя в доверенности (АдрДовТип) — Таблица 4.23
IdentityCardInfo	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип) — Таблица 4.24
FIO	Фамилия, имя, отчество (ФИОТип) — Таблица 4.25
PowerOfAttorneyFNSDOVEL503Data_CL	Данные доверенности в формате, подтверждающем полномочия представителя налогоплательщика, версия 5.02.
PowerOfAttorneyDocument	Состав и структура документа (Документ). Код формы по КНД принимает значение 1110310 в соответствии с форматом и не может быть изменён.
PowerOfAttorneyDocumentData	Доверенность (Довер) — Таблица 4.3
PowerOfAttorneyInfo	Сведения доверенности (СвДов) — Таблица 4.4
PrincipalInfo	Сведения о доверителе (СвДоверит) — Таблица 4.5
RussianCompanyInfo	Сведения о российской организации (НПЮЛ) — Таблица 4.6
PrincipalWithoutPowerOfAttorneyInfo	Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов) — Таблица 4.7
IndividualInfo	Сведения по физическому лицу (СвФЛ) — Таблица 4.8
ForeignCompanyInfo	Сведения об иностранной организации (ИО) — Таблица 4.9

Класс	Описание
SeparateSubdivisionManagerInfo	Сведения о руководителе обособленного подразделения (СвРукОП) — Таблица 4.10
RetrustPowerOfAttorneyDocumentData	Передоверие (Передов) — Таблица 4.11
RetrustPowerOfAttorneyInfo	Сведения доверенности, совершённой (выданной) в рамках передоверия (СвДовП) — Таблица 4.12
PrimaryPowerOfAttorneyInfo	Сведения об основной доверенности (СвОснДов) — Таблица 4.13
PrimaryPrincipalInfo	Сведения о доверителе основной доверенности (СвДоверитОсн) — Таблица 4.14
ForeignCompanyInfo1	Сведения о доверителе – иностранном юридическом лице (ДоверитИО) — Таблица 4.15
ParentPrincipalInfo	Сведения о доверителе из доверенности, на основании которой осуществляется передоверие (СвДоверщП) — Таблица 4.16
RetrustPrincipalInfo	Сведения о доверителе в порядке передоверия (СвДоверщ) - 4.17
OrganizationInfo	Сведения об организации (СвОргТип) — Таблица 4.18
RussianCompanyInfo1	Сведения о российском юридическом лице (СведЮЛТип) — Таблица 4.19
RepresentativeInfo	Сведения об уполномоченном представителе (уполномоченных представителях) (СвПредТип) — Таблица 4.20
IndividualInfo2	Сведения о физическом лице, в том числе индивидуальном предпринимателе (СвФизЛицТип) — Таблица 4.21

Класс	Описание
IndividualInfo3	Сведения по физическому лицу (СведФЛТип) — Таблица 4.22
AddressInfo	Адрес доверителя в доверенности (АдрДовТип) — Таблица 4.23
IdentityCardInfo	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип) — Таблица 4.24
FIO	Фамилия, имя, отчество (ФИОТип) — Таблица 4.25
PowerOfAttorneyFNSDOVELDat	Сведения о доверенности ФНС в формате DOVEL.
PowerOfAttorneyInfo	Сведения доверенности (СвДовТип)
PrincipalInfo	Сведения о доверителе (СвДоверит)
RussianEntityPrincipalInfo	Сведения о российской организации (НПЮЛ)
PrincipalWithoutPowerOfAttorneyInfo	Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов)
IndividualPrincipalInfo	Сведения о доверителе — физическом лице (ФЛДоверТип)
ForeignLegalEntityPrincipalInfo	Сведения о доверителе — иностранном юридическом лице (ИнОргДовер)
HeadOfSeparateDivisionInfo	Сведения о руководителе обособленного подразделения (СвРукОП)
IdentityDocumentInfo	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип)
IndividualPrincipalInfo	Сведения о доверителе — физическом лице (ФЛДоверТип)
IndividualPrincipalInfo1	Сведения по физическому лицу (СвФЛ)

Класс	Описание
RepresentativeAndAuthorityInfo	Сведения о доверителе — физическом лице (ФЛДоверТип)
RepresentativeInfo	Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип)
IndividualInfo	Сведения о физическом лице (СведФЛТип)
OrganizationInfo	Сведения об организации (СвОрг)
PowerOfAttorneyMachineReadableInfo	Содержит информацию о МЧД.
RoleModelAccessChecker	Предоставляет методы проверки доступа в ролевой модели.
PowerOfAttorneyVerification	Содержит результат проверки действительности доверенности.
ImportESNSIResults	Результат импорта полномочий из ЕСНСИ.
IPowersService	Сервис справочника полномочий.
FindPowerOfAttorneyCardsResults	Содержит результаты поиска использования кодов полномочий в СКД.
FindPowerOfAttorneyCardsResultsItem	Результат поиска использования кода полномочий в СКД.
ServerConnectionInfo	Параметры соединения с сервером.
StartBusinessProcessErrorInfo	Представляет содержимое ошибки запуска бизнес-процесса.
TaskStopExecutionInfo	Представляет содержимое ошибки остановки исполнения задания.
TaskCopyResultsOptions	Предоставляет параметры переноса отчёта об исполнении подчиненного задания в родительское.
TaskTreeNodeInfo	Содержит параметры узла дерева заданий.

Класс	Описание
UserProfileCardSettings	Возвращает параметры цветовой схемы по умолчанию для Windows-клиента.
VersionedFileEventArgs	Поставляет данные для событий CheckedInFile.
ViewCardField	Определяет свойства поля для представления.
ViewCardFieldsGroup	Определяет свойства группы полей представления.

Интерфейсы

Интерфейс	Описание
IDocumentPropertySetting	Добавляет к производному классу возможность связывать XML-поле со свойством карточки.
IDocumentPropertySyncronizer	Добавляет к производному классу возможность загрузки и сохранение свойств в XML-документе.
ILongProcessManager	Добавляется в производный класс возможности менеджера длительных процессов.
IMessageLogger	Добавляет к производному классу возможность сохранения произвольной строки в журнал.

Перечисления

Перечисление	Описание
PowerOfAttorneyHandlingFlags	Флаги обработки доверенностей.
BusinessProcessErrorType	Определяет типы ошибок формируемых при проверке бизнес-процесса.
DocumentPropertyDirection	Определяет направление синхронизации свойств документа и полей карточки.

Перечисление	Описание
GridViewFieldCollectionType	Определяет тип отображаемого поля справочника сотрудников или контрагентов.
PerformerType	Определяет тип исполнителя задания.
TaskTreeNodeType	Определяет тип узла в дереве заданий.
AuthorityType	Тип полномочия
CitizenshipType	Признак наличия гражданства
EntityType	Тип лица, передавшего полномочия / тип представителя
Gender	Пол
JointRepresentationType	Признак совместных полномочий
NotarialActionParticipantStatus	Статус участника нотариального действия
PowerOfAttorneyForm	Форма доверенности
PowerOfAttorneyKind	Вид доверенности
PowerOfAttorneyLossOfAuthorityType	Признак утраты полномочий при передоверии
PowerOfAttorneyOption	Признак доверенности
PrincipalType	Тип доверителя
RevocationCondition	Условие отзыва доверенности
RevocationPossibleType	Признак безотзывной доверенности
SoleExecutiveAuthorityType	Вид полномочий единоличного исполнительного органа
RetrustType	Признак возможности оформления передоверия.
RetrustType	Признак возможности оформления передоверия.

DocsVision.BackOffice.ObjectModel.Services.Entities.ActiveDirectory — пространство имён

В пространстве имён

DocsVision.BackOffice.ObjectModel.Services.Entities.ActiveDirectory представлены типы доступные для синхронизации с доменом ActiveDirectory.

Пространства имён

Пространство имён	Описание
ADSync	Содержит интерфейс и перечисление характеризующие возможность синхронизации базы Docsvision и домена основанного на ActiveDirectory.

Классы

Класс	Описание
ADDomain	Представляет параметры для синхронизации домена.
ADEntryBase	Базовый класс для описания синхронизируемой сущности.
ADGroup	Представляет параметры для синхронизации группы.
ADUnit	Представляет параметры для синхронизации подразделения.
ADUser	Представляет параметры для синхронизации пользователя.
StaffADSyncException	Предоставляет данные для ошибки синхронизации.

Перечисления

Перечисление	Описание
UserAccountControl	Определяет характеристики учетной записи пользователя.

DocsVision.BackOffice.ObjectModel.Services.Entities.ActiveDirectory.ADSync — пространство имён

Содержит интерфейс и перечисление характеризующие возможность синхронизации базы Docsvision и домена основанного на ActiveDirectory.

Интерфейсы

Интерфейс	Описание
IADSynchronizationSettings	Определяет в производном классе параметры синхронизации с доменом.

Перечисления

Перечисление	Описание
ADSynchronizationMode	Определяет направление синхронизации.

IADSynchronizationSettings — интерфейс

Определяет в производном классе параметры синхронизации с доменом.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities.ActiveDirectory.ADSync
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IADSynchronizationSettings
```

Свойства

Имя	Описание
CreateFolders	Возвращает признак создания папок.
DeleteItems	Возвращает признак возможности удаления объектов.
LoadCertificates	Возвращает признак необходимости загрузки сертификатов.
LoadDisabled	Возвращает признак загрузки заблокированных учетных записей.
NotCheckDomain	Возвращает признак отмены проверки домена сотрудников.
SynchronizationMode	Возвращает направление синхронизации.

Имя	Описание
SynchronizeChildren	Возвращает признак необходимости синхронизации дочерних объектов.
SynchronizeDepartments	Возвращает признак необходимости синхронизации подразделений.
SynchronizeGroups	Возвращает признак необходимости синхронизации групп.
SynchronizeNewMoved	Возвращает признак добавления и переноса объектов.
SynchronizeRoles	Возвращает признак необходимости синхронизации ролей.

Примеры

Ниже приведён пример реализованного класса, которым предусмотрена синхронизация исключительно подразделений в справочнике Docsvision информацией, полученной из Active Directory

```
class SynchronizationDepartmentsSettings : IADSynchronizationSettings
{
 public bool CreateFolders { get { return false; } }
 public bool DeleteItems { get { return false; } }
 public bool LoadCertificates { get { return false; } }
 public bool LoadDisabled { get { return false; } }
 public bool NotCheckDomain { get { return false; } }

 public ADSynchronizationMode SynchronizationMode
 {
  get { return ADSynchronizationMode.FromADToDictionary; } //Задаем направление в
Справочник сотрудников
 }
 public bool SynchronizeChildren { get { return false; } }

 public bool SynchronizeDepartments
 {
  get { return true; } // Разрешаем синхронизировать подразделения
 }
 public bool SynchronizeGroups { get { return false; } }

 public bool SynchronizeNewMoved
 {
  get { return true; } // Разрешаем создание и перемещение объектов
```

```
  }
  public bool SynchronizeRoles { get { return false; } }
}
```

DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting — пространство имён

Пространство имён DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting содержит типы, определяющие параметры вида карточки, определённые в справочнике видов карточек.

Классы

Класс	Описание
`CardFieldSetting`	Предоставляет параметры атрибута, включаемого в подпись и определённого в Справочника видов карточек.
`ContextualPropertyObject`	Разрешает объектам, содержащим настройки вида карточки, доступ к контексту объектов.
`ContextualPropertyObjectCollection<T>`	Содержит методы для работы с коллекцией `ContextualPropertyObject`.
`DocumentAttachedFileSetting`	Предоставляет свойства дополнительного файла, который автоматически прикладывается к карточке Документа.
`DocumentExportSetting`	Настройки экспорта документа.
`DocumentExportTransformSetting`	Предоставляет стандартный набор свойств, описывающих настройки трансформации применяемой при экспорте документа.
`DocumentMainFileSetting`	Предоставляет свойства основного файла, который автоматически прикладывается к карточке Документа.

Класс	Описание
`DocumentMap`	Статический класс, представляющий дерево свойств вида карточки, описанных в справочнике видов карточек.
`DocumentOperationSignatureSetting`	Содержит параметры подписания операций карточки документа.
`DocumentPropertySetting`	Предоставляет доступ к свойствам (определённым в справочнике видов) карточки документа.
`DocumentSetting`	Предоставляет настройки вида документа, определённые в *Справочнике видов карточек*.
`DocumentSignatureSetting`	Содержит параметры подписания основной части карточки документа.
`ExtensionSettingAttribute`	Данным атрибутом отмечаются свойства принадлежащие к списку параметров вида карточки.
`PropertyObject`	Предоставляет инфраструктуру для расширенных свойств объекта.
`PropertyObjectCollection<T>`	Предоставляет методы для работы с коллекцией `PropertyObject`.
`SignatureLabelSetting`	Настройки метки подписи вида карточки документ.
`TaskAutoCompletionSetting`	Параметры автоматического завершения задания при завершении подчиненных заданий.
`TaskChildCopySetting`	Список полей, значение которых будет скопировано из родительского задания в подчиненное при создании.
`TaskChildKindSetting`	Предоставляет дерево видов подчиненных заданий.

Класс	Описание
TaskCompletionMainSetting	Определяет необходимость отправки отчёта, и обязательность добавления файла отчёта.
TaskCompletionParameterItemSetting	Предоставляет поля, обязательные для заполнения при завершении задания.
TaskCompletionParametersSetting	Предоставляет настройки вариантов завершения задания, отображаемых на ленте.
TaskCompletionSetting	Содержит ссылки на типы, определяющие все настройки завершение задания.
TaskDelegationSetting	Предоставляет настройки делегирования задания.
TaskDependingOnRelatedTasksCompletionSetting	Содержит настройки завершения задания при невозможности завершения связанных заданий.
TaskEmailNotificationSetting	Определяет настройки почтовых уведомлений, отправляемых при выборе типов маршрутизации "Ссылка на задание" и "Письмо с описанием задания".
TaskGroupChildKindSetting	Предоставляет дерево видов задания, которые могут быть использованы в качестве подчиненного задания.
TaskGroupChildKindSettings	Содержит настройки создания подчиненного задания, и коллекцию видов подчиненного задания.
TaskGroupReferenceAllowedCardKindSetting	Предоставляет пользовательский вид карточки задания, которое будет получено из группы заданий.
TaskGroupReferenceAllowedCardTypeSetting	Определяет разрешённый тип карточки для документов, которые можно добавлять в группу заданий.

Класс	Описание
TaskGroupReferenceLinkTypeSetting	Содержит тип ссылки, которую можно добавлять в группу заданий.
TaskGroupReferenceSetting	Содержит коллекцию разрешённых, для группы заданий, типов ссылок и карточек.
TaskGroupSetting	Содержит настройки вида группы заданий.
TaskOperationSignatureSetting	Предоставляет параметры подписания операция карточки Задание, заданные в Справочнике видов карточек.
TaskPerformingEmployeeSetting	Предоставляет сотрудника для списка исполнителей задания, при варианте выбора исполнителя задания из списка.
TaskPerformingGroupSetting	Предоставляет группу для списка исполнителей задания, при варианте выбора исполнителя задания из списка.
TaskPerformingRoleSetting	Предоставляет роль для списка исполнителей задания, при варианте выбора исполнителя задания из списка.
TaskPerformingSearchWordSetting	Предоставляет поисковое слово для списка исполнителей задания, при варианте выбора исполнителя задания из списка.
TaskPerformingSetting	Содержит настройки выбора исполнителя задания.
TaskPerformingUnitSetting	Предоставляет подразделение для списка исполнителей задания, при варианте выбора исполнителя задания из списка.

Класс	Описание
TaskReferenceLinkTypeSetting	Содержит тип ссылки для основного документа задания.
TaskReferenceSetting	Содержит настройки типов ссылок для вида карточки задания.
TaskRejectionSetting	Содержит настройки отклонения задания.
TaskRoutingSetting	Содержит настройки маршрутизации задания.
TaskSetting	Предоставляет настройки определённого вида задания.

Перечисления

Перечисление	Описание
ChildTaskKindType	Определяет вид подчиненного задания у карточки задания или группы заданий.
CompletionParameterDemand	Определяет признаки обязательности заполнения поля до завершения задания.
CompletionParameterDisplayType	Определяет форматы отображения вариантов завершения задания.
DocumentLoadSource	Определяет источники загрузки основных файлов карточки Документ.
ExtensionSettingType	Определяет типы атрибута ExtensionSettingAttribute.
TaskAutoCompletionType	Определяет варианты автоматического завершения задания при завершении подчиненных заданий.
TaskDependingCompletionType	Определяет варианты завершения задания при невозможности завершения связанных заданий.

Перечисление	Описание
TaskRoutingType	Определяет допустимые типы маршрутизации задания.

CardFieldSetting — класс

Предоставляет параметры атрибута, включаемого в подпись и определённого в Справочнике видов карточек.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class CardFieldSetting : PropertyObject
```

Свойства

Имя	Описание
FieldAlias	Задаёт или возвращает псевдоним поля.
FieldId	Задаёт или возвращает уникальный идентификатор поля.
FieldName	Задаёт или возвращает локализованное название поля.
Order	Задаёт или возвращает порядковый номер.
SectionName	Задаёт или возвращает локализованное название секции.
SectionPath	Задаёт или возвращает последовательность строку, содержащую последовательность секций, полученную из идентификаторов Sections.

Имя	Описание
`Sections`	Задаёт или возвращает список идентификаторов секций, представляющую дерево от родительской секции до секции, содержащей поле.

Методы

Имя	Описание
`BuildSectionPath`	Создаёт из `Sections` строку для `SectionPath`.
`BuildSectionPath(Stack<Guid>)`	Создаёт из заданной последовательности строку для `SectionPath`.
`GetSectionField(IMetadataProvider, Guid)`	Получает из `CardFieldSetting` поле типа `SectionField`.
`RebuildSectionPath(String)`	Создаёт из строки последовательность для `Sections`.

Заметки

Параметры `SectionPath` и `Sections`, как правило, содержат единственный идентификатор секции, содержащей поле.

DocumentSetting — класс

Предоставляет настройки вида документа, определённые в *Справочнике видов карточек*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class DocumentSetting : ContextualPropertyObject
```

Свойства

Имя	Описание
DefaultAttachedFiles	Возвращает настройки для файла, загружаемого в документ, по умолчанию
DefaultMainFile	Задаёт или возвращает идентификатор основного файла, загружаемого в документ, по умолчанию.
DefaultMainFileIsTemplate	Задаёт или возвращает признак того, что основной файл по умолчанию является шаблоном.
DefaultMainFileName	Задаёт или возвращает название основного файла, загружаемого в документ, по умолчанию.
DefaultMainFiles	Возвращает настройки основных файлов, загружаемых в документ, по умолчанию.
DocumentLoadSource	Задаёт или возвращает источник загрузки основных файлов.
DocumentSignature	Возвращает параметры подписания документа.
Export	Возвращает настройки экспорта документа.
OperationSignatures	Возвращает параметры подписания операций документа.
PreviewDisabled	Задаёт или возвращает признак запрета на предварительный просмотр файлов документа.
Properties	Возвращает свойства документа.
RootCategory	Задаёт или возвращает идентификатор корневой директории.
UniqueAttributes	Возвращает настройки уникальных атрибутов карточки.

Имя	Описание
VersioningType	Задаёт или возвращает тип версионирования для файлов документа.

TaskGroupSetting — класс

Содержит настройки вида группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class TaskGroupSetting : ContextualPropertyObject
```

Свойства

Имя	Описание
DefaultTaskCardKind	Задаёт или возвращает идентификатор вида задания по умолчанию.
Reference	Задаёт или возвращает идентификатор типа ссылок и карточек для документов.
TaskCardKindResponsiblePerformer	Задаёт или возвращает идентификатор вида задания, отправляемого ответственному исполнителю, по умолчанию.
TasksLinkType	Задаёт или возвращает идентификатор типа ссылок для заданий.
UrlLinkType	Задаёт или возвращает идентификатор типа ссылок для Url.

ChildTaskKindType — перечисление

Определяет вид подчиненного задания у карточки задания или группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum ChildTaskKindType
```

Члены

Имя члена	Описание
ParentTaskKind	Карточка дочернего задания должна иметь вид родительского задания.
AnyKind	Карточка дочернего задания может быть любого вида.
KindFromList	Вид карточки задание будет указан в списке.

CompletionParameterDemand — перечисление

Определяет признаки обязательности заполнения поля до завершения задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum CompletionParameterDemand
```

Члены

Имя члена	Описание
Required	Значение заполняется обязательно.
Prompt	Заполнение значения поля — желательно.

Заметки

Для пользовательских видов карточек Задание в разметку карточки могут быть добавлены поля, которые должны быть заполнены до завершения задания.

CompletionParameterDisplayType — перечисление

Определяет форматы отображения вариантов завершения задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum CompletionParameterDisplayType
```

Члены

Имя члена	Описание
Buttons	Варианты завершения будут представлены в виде набора кнопок.
DropDown	Варианты завершения будут отображаться списком.

Заметки

Список вариантов завершения задания будет отображаться на ленте инструментов.

DocumentLoadSource — перечисление

Определяет источники загрузки основных файлов карточки Документ.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum DocumentLoadSource
```

Члены

Имя члена	Описание
`Manual`	Предусмотрена ручная загрузка файлов в документ.
`FileSystem`	Файлы будут загружаться с файловой системы.
`Scanner`	Источником файлов для документа будет сканер.

Заметки

Указанный тип источника будет вызван при открытии карточки Документа.

ExtensionSettingType — перечисление

Определяет типы атрибута `ExtensionSettingAttribute`.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum ExtensionSettingType
```

Члены

Имя члена	Описание
`Generic`	Универсальный.
`Card`	Карточка.
`File`	Файл.
`LargeText`	Текст.
`Image`	Изображение.

TaskAutoCompletionType — перечисление

Определяет варианты автоматического завершения задания при завершении подчиненных заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum TaskAutoCompletionType
```

Члены

Имя члена	Описание
NotComplete	Не завершать.
AfterCompletionAllChildren	Задание будет завершено при завершении всех дочерних заданий.
AfterCompletionAnyChildren	Задание будет завершено при завершении любого дочернего задания.

TaskDependingCompletionType — перечисление

Определяет варианты завершения задания при невозможности завершения связанных заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum TaskDependingCompletionType
```

Члены

Имя члена	Описание
Prevent	Не разрешать завершение основного задания.
Allow	Разрешать завершение задания.
UserDiscretion	На усмотрение пользователя.

TaskRoutingType — перечисление

Определяет допустимые типы маршрутизации задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum TaskRoutingType
```

Члены

Имя члена	Описание
ByDefault	По умолчанию.
OnlineTask	Онлайн задание.
OutlookTask	Задача Microsoft Outlook.
TaskLink	Ссылка на задание.
DescriptionLetter	Письмо с описанием задания.

TaskSetting — класс

Предоставляет настройки определённого вида задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities.KindSetting
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class TaskSetting : ContextualPropertyObject
```

Свойства

Имя	Описание
AuthorAdditionalSearchMode	Задаёт или возвращает признак разрешения выбора автора из справочника сотрудников.

Имя	Описание
ChildCopy	Возвращает настройки копирования подчиненного задания.
ChildKind	Возвращает настройки видов подчиненного задания.
Completion	Возвращает настройки завершения задания.
Delegation	Возвращает настройки делегирования задания.
EmailNotification	Возвращает настройки почтовых уведомлений, отправляемых при типах маршрутизации "Ссылка на задание" и "Письмо с описанием задания".
GroupChildKind	Возвращает настройки создания подчиненного задания.
OperationSignatures	Возвращает настройки подписания операций.
Performing	Возвращает настройки выбора исполнителей.
Reference	Возвращает настройки типов ссылок для основного документа, дополнений и отчётов.
Rejection	Возвращает настройки отправки комментария при отклонении задания.
Routing	Возвращает настройки маршрутизации.
UseAsContainer	Задаёт или возвращает признак использования карточки задания в виде контейнера для отображения вложенных карточек. При открытии задания, вместо него открывается вложенная карточка.

Имя	Описание
`UseBusinessCalendar`	Задаёт или возвращает признак разрешения использования настроек рабочего времени, заданных в календаре, при расчете дат начала, завершения и длительности задания.

BaseCardSignatureVerification — класс

Возвращает информацию по проверке подписи на карточке.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class BaseCardSignatureVerification
```

Свойства

Имя	Описание
`IsExpiredCertificate`	Срок действия сертификата истек.
`IsNotAllMainFilesSigned`	Не все основные файлы подписаны.
`IsValid`	Проверка пройдена.
`IsValidCertificate`	Сертификат действителен для подписи.
`Results`	Возвращает коллекцию результатов проверки каждой части подписи.

Заметки

Результат проверки может содержать не все значения, результат заполняется на усмотрение метода, т.е. может быть не задано значение `IsValid` или `IsNotAllMainFilesSigned`, или всех.

MailNotificationInfo — класс

Предоставляет данные электронного сообщения для сервиса отправки электронной почты.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class MailNotificationInfo
```

Конструкторы

Имя	Описание
MailNotificationInfo(String, String, String)	Инициализирует новый экземпляр класса MailNotificationInfo с помощью указанного значения.

Свойства

Имя	Описание
Body	Возвращает содержимое письма.
Email	Возвращает адрес электронной почты получателя.
Theme	Возвращает тему письма.

Заметки

Объект типа MailNotificationInfo используется для формирования данных перед отправкой электронного сообщения с использованием сервиса для работы с базовыми объектами IBaseCardService.SendMailNotification(IEnumerable<MailNotificationInfo>).

MailNotificationInfo — конструктор (String, String, String)

Инициализирует новый экземпляр класса MailNotificationInfo с помощью указанного значения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public MailNotificationInfo(string email, string theme, string body)
```

Параметры

email

Тип: System.String

Адрес электронной почты получателя сообщения

theme

Тип: System.String

Тема письма

body

Тип: System.String

Содержимое письма

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр email, theme или body.

PerformerSearchWordsManager — класс

Представляет статические методы работы с поисковыми словами.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public static class PerformerSearchWordsManager
```

Методы

Имя	Описание
GetEmployeesSearchWord(ObjectContext, StaffEmployee, Guid)	Возвращает список сотрудников, выбранных поисковым словом.

Имя	Описание
`GetPerformerSearchWordDescription(Guid)`	Возвращает описание поискового слова.
`GetPerformerSearchWords`	Возвращает коллекцию идентификаторов и описаний поисковых слов, зарегистрированных в классе `PerformerSearchWordsManager`.

Поля

Имя	Описание
`SearchWordDepartmentEmployees`	Сотрудники департамента автора.
`SearchWordDeputyFor`	Замещаемые.
`SearchWordManagedUnits`	Подразделения, в которых автор — руководитель.
`SearchWordManager`	Руководитель.
`SearchWordMyDeputies`	Заместители.
`SearchWordSubordinates`	Подчинённые.
`SearchWordSubordinatesAll`	Все подчинённые.

PowerOfAttorneyData — класс

Базовый класс сведений о доверенности.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public abstract class PowerOfAttorneyData
```

Методы

Имя	Описание
`GetPowerOfAttorneyNumber()`	Возвращает номер доверенности

Имя	Описание
GetPowerOfAttorneyStartDate()	Возвращает дату совершения (выдачи) доверенности
GetPowerOfAttorneyEndDate()	Возвращает дату окончания действия доверенности
GetRetrustType()	Возвращает признак возможности оформления передоверия
GetPrincipal()	Возвращает строковое представление поручителя
GetPowerOfAttorneyMachineReadableFileID()	Возвращает сформированное имя файла доверенности
GetRepresentative()	Возвращает строковое представление представителя в формате ФИО (или ФИ) — для физического лица, или названия организации — для юридического лица
GetRepresentativesPowers()	Возвращает полномочия представителя
GetSigner()	Возвращает строковое представление подписанта в формате ФИО (или ФИ)

PowerOfAttorneyDigest — класс

Представляет собой сведения о доверенности

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyDigest
```

Свойства

Имя	Описание
Id	Идентификатор доверенности.
DisplayString	Отображаемое имя доверенности.

Имя	Описание
RegTransferStatus	Статус регистрации доверенности в ФНС.

PowerOfAttorneyEMCHDData — класс

Сведения о доверенности в формате единой формы доверенности

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PowerOfAttorneyEMCHDData : PowerOfAttorneyData
```

Свойства

Имя	Описание
PowerOfAttorneyDocument	Состав и структура документа.
EsiaOID	Внешний идентификатор объекта ЕСИА.
OtherInformation	Иные сведения.
RecipientID	Идентификатор получателя файла доверенности.
FinalRecipientID	Идентификатор конечного получателя файла доверенности.
SenderID	Идентификатор отправителя информации.

Конструкторы

Имя	Описание
PowerOfAttorneyEMCHDData()	Инициализирует Сведения о доверенности в формате единой формы доверенности (без установки обязательных разделов).

Имя	Описание
`PowerOfAttorneyEMCHDData(bool, bool, bool)`	Инициализирует Сведения о доверенности в формате единой формы доверенности с включением проверки дополнительных разделов. *Параметры:* • `withEsia` — доверенность для получения государственных и муниципальных услуг с использованием платформы полномочий Госуслуг (ЕСИА). • `withNotary` — доверенность удостоверена нотариусом. • `withTax` — доверенность для взаимодействия с налоговыми органами.

Классы

Имя	Описание
`AddressInfo`	Сведения об адресе (АдрТип)
`BranchOfRussianEntityInfo`	Сведения о филиале (обособленном подразделении) юридического лица (ФилПерПолн)
`ConfirmationOfAuthorityDocument`	Документ, подтверждающий полномочия лица, действующего без доверенности (ДокПдтвТип)
`DelegatedPowerOfAttorneyPrincipal`	Сведения о лице, передавшем полномочия (ПередПолн)
`FIO`	Фамилия, имя, отчество (при наличии) (ФИОТип)
`ForeignEntityInfo`	Сведения об иностранном юридическом лице (СвИнОргТип)
`HandwrittenSignature`	Рукописная подпись (ПодпРукопис)

Имя	Описание
IndividualInfo	Сведения о физическом лице (СведФЛТип)
IndividualPrincipalInfo1	Сведения о физическом лице (ФЛПерПолн)
IndividualPrincipalInfo	Сведения о физическом лице (доверителе) (ФЛДоверТип)
IrrevocablePowerOfAttorneyInfo	Сведения о безотзывной доверенности (Безотзыв)
LegalEntityInfo	Сведения о юридическом лице (или филиале (обособленном подразделении) юридического лица) (СвОргТип)
MachineReadablePowerRestrictions	Сведения об ограничении (ограничениях) к полномочию (ОгрСвПолн)
MachineReadablePowersInfo	Сведения о машиночитаемом полномочии (машиночитаемых полномочиях) (МашПолн)
NotaryCertificateInfo	Сведения о нотариальном удостоверении (СвНотУдТип)
NotaryDeputyInfo	ВРИО нотариуса (ВрионотТип)
NotaryInfo	Сведения о нотариусе, совершившем нотариальное действие (СвНотДействТип)
PowerOfAttorneyDocument	Состав и структура документа (Документ)
PowerOfAttorneyDocumentData	Доверенность (Довер)
PowerOfAttorneyInfo	Сведения о доверенности (СвДовТип)
PrincipalInfo	Сведения о доверителе (Доверит)
PrincipalsInfo	Сведения о доверителе (доверителях) (СвДоверит)

Имя	Описание
RepresentativesInfo	Сведения о представителе (представителях) (СвУпПредТип)
RepresentativePowersInfo	Сведения о полномочиях представителя (представителей) (СвПолнТип)
RepresentativesInfo	Сведения о представителе (представителях) (СвУпПредТип)

Перечисления

Имя	Описание
AuthorityType	Тип полномочия
CitizenshipType	Признак наличия гражданства
EntityType	Тип лица, передавшего полномочия / тип представителя
Gender	Пол
JointRepresentationType	Признак совместных полномочий
NotarialActionParticipantStatus	Статус участника нотариального действия
PowerOfAttorneyForm	Форма доверенности
PowerOfAttorneyKind	Вид доверенности
PowerOfAttorneyLossOfAuthorityType	Признак утраты полномочий при передоверии
PowerOfAttorneyOption	Признак доверенности
PrincipalType	Тип доверителя
RevocationCondition	Условие отзыва доверенности
RevocationPossibleType	Признак безотзывной доверенности
SoleExecutiveAuthorityType	Вид полномочий единоличного исполнительного органа

PowerOfAttorneyEMCHDData.AddressInfo — класс

Сведения об адресе (АдрТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class AddressInfo
```

Свойства

Имя	Описание
SubjectOfRussia	Субъект Российской Федерации
FiasCode	Идентификатор адреса по ФИАС
Address	Адрес в Российской Федерации
FiasAddress	ФИАС адрес в Российской Федерации

PowerOfAttorneyEMCHDData.BranchOfRussianEntityInfo — класс

Сведения о филиале (обособленном подразделении) юридического лица (ФилПерПолн)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class BranchOfRussianEntityInfo
```

Свойства

Имя	Описание
FilialInfo	Сведения о филиале (обособленном подразделении) юридического лица
Manager	Сведения о руководителе филиала (обособленном подразделении) юридического лица

PowerOfAttorneyEMCHDData.ConfirmationOfAuthorityDocument — класс

Документ, подтверждающий полномочия лица, действующего без доверенности (ДокПдтвТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class ConfirmationOfAuthorityDocument
```

Свойства

Имя	Описание
DocumentName	Наименование документа
IssueDate	Дата выдачи документа
Issuer	Наименование органа, выдавшего документ
IdentityOfDocument	Сведения об удостоверении документа

PowerOfAttorneyEMCHDData.DelegatedAuthorityPrincipalsInfo — класс

Сведения о лице (лицах), передавшем (передавших) полномочия (СвПередПолн)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class DelegatedAuthorityPrincipalsInfo
```

Свойства

Имя	Описание
PrincipalType	Тип лица, передавшего полномочия
PrincipalInfo	Сведения о лице, передавшем полномочия

PowerOfAttorneyEMCHDData.DelegatedPowerOfAttorneyPrincipal — класс

Сведения о лице, передавшем полномочия (ПередПолн)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class DelegatedPowerOfAttorneyPrincipal
```

Свойства

Имя	Описание
RussianEntity	Сведения о юридическом лице
SoleProprietor	Сведения об индивидуальном предпринимателе
Individual	Сведения о физическом лице
BranchOfRussianEntity	Сведения о филиале (обособленном подразделении) юридического лица
BranchOfForeignEntity	Сведения о филиале (аккредитованном представительстве) иностранного юридического лица

PowerOfAttorneyEMCHDData.FIO — класс

Фамилия, имя, отчество (при наличии) (ФИОТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class FIO
```

Свойства

Имя	Описание
LastName	Фамилия

Имя	Описание
FirstName	Имя
MiddleName	Отчество (при наличии)

PowerOfAttorneyEMCHDData.ForeignEntityInfo — класс

Сведения об иностранном юридическом лице (СвИнОргТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class ForeignEntityInfo
```

Свойства

Имя	Описание
ParticipantStatus	Статус участника нотариального действия
Name	Наименование иностранного юридического лица / филиала (аккредитованного представительства) иностранного юридического лица
Inn	ИНН иностранного юридического лица / филиала (аккредитованного представительства) иностранного юридического лица
Kpp	КПП филиала (аккредитованного представительства) иностранного юридического лица
AccreditationRecordNumber	Номер записи об аккредитации
RegistrationCountry	Страна регистрации (инкорпорации)
RegisteringAuthorityName	Наименование регистрирующего органа
RegNumber	Регистрационный номер в стране регистрации (инкорпорации)

Имя	Описание
TaxpayerCode	Код налогоплательщика в стране регистрации (инкорпорации) или аналог
Phone	Контактный телефон
EMail	Адрес электронной почты
LegalAddress	Адрес юридического лица на территории государства, в котором оно зарегистрировано
ActualAddress	Адрес (место нахождения) на территории Российской Федерации

PowerOfAttorneyEMCHDData.ForeignLegalEntityPrincipalInfo — класс

Сведения об иностранном юридическом лице (или филиале (аккредитованном представительстве) иностранного юридического лица) (ИнОргДоверТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class SoleProprietorInfo0
```

Свойства

Имя	Описание
ForeignEntityInfo	Сведения об иностранном юридическом лице / Сведения о филиале (аккредитованном представительстве) иностранного юридического лица
BranchManagers	Сведения о руководителе иностранного юридического лица / Сведения о руководителе филиала (аккредитованного представительства) иностранного юридического лица

PowerOfAttorneyEMCHDData.HandwrittenSignature — класс

Рукописная подпись (ПодпРукопис)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class HandwrittenSignature
```

Свойства

Имя	Описание
PdfDocumentHash	Хеш PDF документа
SignatureImage	Изображение подписи
SignatureHash	Хеш подписи
SignatureDate	Дата и время подписи
Fio	ФИО лица, подписавшего доверенность

PowerOfAttorneyEMCHDData.IndividualInfo — класс

Сведения о физическом лице (СведФЛТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class IndividualInfo
```

Свойства

Имя	Описание
Gender	Пол физического лица
CitizenshipType	Признак наличия гражданства
UnifiedPopulationRegisterNumber	Номер записи единого регистра населения

Имя	Описание
BirthDate	Дата рождения
BirthPlace	Место рождения
Citizenship	Гражданство
ContactPhone	Контактный телефон
EMail	Адрес электронной почты
Fio	ФИО доверителя (или представителя)
ResidenceAddress	Адрес места жительства
IdentityCard	Сведения о документе, удостоверяющем личность физического лица

PowerOfAttorneyEMCHDData.IndividualPrincipalInfo1 — класс

Сведения о физическом лице (ФЛПерПолн)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class IndividualPrincipalInfo1
```

Свойства

Имя	Описание
ParticipantStatus	Статус участника нотариального действия
ApplicantExist	Признак наличия рукоприкладчика
Inn	ИНН физического лица
Snils	СНИЛС
ApplicantInfo	Сведения о рукоприкладчике
IndividualInfo	Сведения о физическом лице

PowerOfAttorneyEMCHDData.IndividualPrincipalInfo — класс

Сведения о физическом лице (доверителе) (ФЛДоверТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class IndividualPrincipalInfo
```

Свойства

Имя	Описание
ParticipantStatus	Статус участника нотариального действия
HasFullCapacity	Признак наличия полной гражданской дееспособности доверителя
ApplicantExist	Признак наличия рукоприкладчика
DocumentOfIncapacity	Документ, подтверждающий недееспособность доверителя
Inn	ИНН физического лица
Snils	СНИЛС
IndividualInfo	Сведения о физическом лице
RepresentativeInfo	Сведения о законном представителе физического лица / Сведения о рукоприкладчике

PowerOfAttorneyEMCHDData.IdentityCardOfIndividual — класс

Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛ)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class IdentityCardOfIndividual
```

Свойства

Имя	Описание
DocumentKindCode	Код вида документа
DocumentSerialNumber	Серия и номер документа
IssueDate	Дата выдачи документа
Issuer	Наименование органа, выдавшего документ
IssuerCode	Код подразделения органа, выдавшего документ
ExpDate	Дата истечения срока действия документа, удостоверяющего личность

PowerOfAttorneyEMCHDData.IrrevocablePowerOfAttorneyInfo — класс

Сведения о безотзывной доверенности (Безотзыв)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class IrrevocablePowerOfAttorneyInfo
```

Свойства

Имя	Описание
RevocationPossibleType	Признак передоверия безотзывной доверенности
RevocationCondition	Условие отзыва безотзывной доверенности
RevocationConditionDescription	Описание условия отзыва безотзывной доверенности

PowerOfAttorneyEMCHDData.LegalEntityInfo — класс

Сведения о юридическом лице (или филиале (обособленном подразделении) юридического лица) (СвОргТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class LegalEntityInfo
```

Свойства

Имя	Описание
ParticipantStatus	Статус участника нотариального действия
Name	Наименование юридического лица / Наименование филиала (обособленного подразделения) юридического лица
Inn	ИНН юридического лица / филиала (обособленного подразделения) юридического лица
Kpp	КПП юридического лица / филиала (обособленного подразделения) юридического лица
Ogrn	ОГРН
RegistrationNumber	Регистрационный номер филиала (обособленного подразделения)
ConstituentDocument	Контактный телефон
Phone	Адрес электронной почты
EMail	Документ, подтверждающий полномочия лица, действующего без доверенности
ConfirmationOfAuthorityDocument	Адрес юридического лица / филиала (обособленного подразделения) юридического лица в Российской Федерации
LegalAddress	Юридический адрес

PowerOfAttorneyEMCHDData.MachineReadablePowerRestrictions — класс

Сведения об ограничении (ограничениях) к полномочию (ОгрСвПолн)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class MachineReadablePowerRestrictions
```

Свойства

Имя	Описание
RestrictionsIndex	Порядковый номер ограничения
RestrictionsCode	Код ограничения
RestrictionsName	Наименование ограничения
RestrictionsText	Текстовое значение для ограничения
RestrictionsCodeValue	Код значения для ограничения
RestrictionsCodeValueName	Наименование значения для ограничения

PowerOfAttorneyEMCHDData.MachineReadablePowersInfo — класс

Сведения о машиночитаемом полномочии (машиночитаемых полномочиях) (МашПолн)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class MachineReadablePowersInfo
```

Свойства

Имя	Описание
PowersCode	Код полномочия

Имя	Описание
Restrictions	Сведения об ограничении (ограничениях) к полномочию

PowerOfAttorneyEMCHDData.NotaryCertificateInfo — класс

Сведения о нотариальном удостоверении (СвНотУдТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class NotaryCertificateInfo
```

Свойства

Имя	Описание
Place	Место совершения доверенности
FromPrincipalToEPGU	Направление сформированного электронного документа в ЛК доверителя на сайте ЕПГУ
FromAttorneyToEPGU	Направление сформированного электронного документа в ЛК поверенного на сайте ЕПГУ
FromDeclarantToFNP	Направление сформированного электронного документа в ЛК заявителя на сайте ФНП
FromAttorneyToFNP	Направление сформированного электронного документа в ЛК поверенного на сайте ФНП
NotaryPaymentPaid	Сведения об уплате за совершение нотариального действия
NotaryPaymentDiscount	Предоставленная льгота на сумму
ElectronicDocumentTransferMethod	Способ выдачи электронного нотариального документа

Имя	Описание
OtherDocumentTransferMethod	Иной способ выдачи
AdditionalInformation	Дополнительные сведения
OtherInformationOfAuthenticationInscription	Иные сведения удостоверительной надписи
NotaryInfo	Сведения о нотариусе, совершившем нотариальное действие
HandwrittenSignature	Рукописная подпись
NotaryDeputyInfo	ВРИО нотариуса

PowerOfAttorneyEMCHDData.NotaryDeputyInfo — класс

ВРИО нотариуса (ВрионотТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class NotaryDeputyInfo
```

Свойства

Имя	Описание
Position	Должность
RegistrationNumber	Регистрационный номер лица, сдавшего квалификационный экзамен в Минюсте
Fio	ФИО ВРИО нотариуса

PowerOfAttorneyEMCHDData.NotaryInfo — класс

Сведения о нотариусе, совершившем нотариальное действие (СвНотДействТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class NotaryInfo
```

Свойства

Имя	Описание
Position	Должность
RegistrationNumber	Регистрационный номер нотариуса в Минюсте
Fio	ФИО нотариуса

PowerOfAttorneyEMCHDData.PowerOfAttorneyDocument — класс

Состав и структура документа (Документ)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PowerOfAttorneyDocument
```

Свойства

Имя	Описание
KND	Код формы по КНД
PowerOfAttorneyData	Доверенность
RetrustPowerOfAttorneyData	Передоверие

PowerOfAttorneyEMCHDData.PowerOfAttorneyDocumentData — класс

Доверенность (Довер)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PowerOfAttorneyDocumentData
```

Свойства

Имя	Описание
PowerOfAttorney	Сведения о доверенности
Principal	Сведения о доверителе
Representative	Сведения о представителе(представителях)
RepresentativePowers	Сведения о полномочиях представителя (представителей)
NotaryCertificateInfo	Сведения о нотариальном удостоверении

PowerOfAttorneyEMCHDData.PrimaryPowerOfAttorneyInfo — класс

Сведения о первоначальной доверенности (или доверенности, на основании которой осуществляется передоверие) (СвПервДоверТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PrimaryPowerOfAttorneyInfo
```

Методы

Имя	Описание
PrimaryPowerOfAttorneyInfo()	Инициализирует экземпляр PrimaryPowerOfAttorneyInfo. Обязательные данные должны быть заполнены.

Имя	Описание
`PrimaryPowerOfAttorneyInfo(PowerOfAttorney)`	Инициализирует экземпляр `PrimaryPowerOfAttorneyInfo` данными переданной СКД. *Параметры:* • `primaryPowerOfAttorney` — СКД первоначальной доверенности или доверенности, на основании которой осуществляется передоверие. Инициализация является отложенной и будет выполнена на этапе формирования МЧД. Заполнять самостоятельно информацию в данной секции доверенности при этом не требуется.

Свойства

Имя	Описание
`PowerOfAttorneyOption`	Признак доверенности
`PowerOfAttorneyForm`	Форма доверенности
`PowerOfAttorneyStartDate`	Дата совершения (выдачи) доверенности
`PowerOfAttorneyEndDate`	Срок действия доверенности
`PrimaryPowerOfAttorneyInternalNumber`	Внутренний номер первоначальной доверенности
`ParentPowerOfAttorneyInternalNumber`	Внутренний номер доверенности, на основании которой осуществляется передоверие
`PrimaryPowerOfAttorneyNumber`	Единый регистрационный номер первоначальной доверенности
`ParentPowerOfAttorneyNumber`	Единый регистрационный номер доверенности, на основании которой осуществляется передоверие

Имя	Описание
PrimaryPowerOfAttorneyNotaryNumber	Регистрационный номер первоначальной доверенности в реестре нотариальных действий
ParentPowerOfAttorneyNotaryNumber	Регистрационный номер доверенности в реестре нотариальных действий, на основании которой осуществляется передоверие
PrimaryPowerOfAttorneyPrincipals	Сведения о доверителе (доверителях) первоначальной доверенности
NotaryInfo	Сведения о нотариусе, совершившем нотариальное действие
NotaryDeputyInfo	ВРИО нотариуса

PowerOfAttorneyEMCHDData.PrimaryPowerOfAttorneyPrincipal — класс

Сведения о доверителе (доверителях) первоначальной доверенности (СвДоверПерв)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PrimaryPowerOfAttorneyPrincipal
```

Свойства

Имя	Описание
RussianEntity	Сведения о юридическом лице
ForeignEntity	Сведения об иностранном юридическом лице
SoleProprietor	Сведения об индивидуальном предпринимателе
Individual	Сведения о физическом лице

PowerOfAttorneyEMCHDData.PrimaryPowerOfAttorneyPrincipalInfo — класс

Сведения о доверителе (доверителях) первоначальной доверенности (СвДоверПерв)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PrimaryPowerOfAttorneyPrincipalInfo
```

Свойства

Имя	Описание
PrincipalType	Тип доверителя
PrimaryPrincipalInfo	Сведения о доверителе первоначальной доверенности

PowerOfAttorneyEMCHDData.PowerOfAttorneyInfo — класс

Сведения о доверенности (СвДовТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PowerOfAttorneyInfo
```

Свойства

Имя	Описание
PowerOfAttorneyKind	Вид доверенности
RetrustType	Признак возможности оформления передоверия
PowerOfAttorneyInternalNumber	Внутренний номер доверенности
PowerOfAttorneyNumber	Единый регистрационный номер доверенности

Имя	Описание
PowerOfAttorneyNotaryNumber	Регистрационный номер доверенности в реестре нотариальных действий
PowerOfAttorneyAdditionalNumber	Дополнительный идентификатор доверенности
PowerOfAttorneyInternalRegistrationDate	Дата внутренней регистрации доверенности
PowerOfAttorneyStartDate	Дата совершения (выдачи) доверенности
PowerOfAttorneyEndDate	Срок действия доверенности
SubmittedPowerOfAttorneyTaxCode	Код налогового органа, в который представляется доверенность
TaxCode	Код налогового органа, в отношении которого действует доверенность
InformationSystemName	Сведения об информационной системе, в которой формируется доверенность
IrrevocablePowerOfAttorneyInfo	Сведения о безотзывной доверенности

PowerOfAttorneyEMCHDData.PrincipalInfo — класс

Сведения о доверителе (Доверит)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PrincipalInfo
```

Свойства

Имя	Описание
RussianEntity	Сведения о юридическом лице
ForeignLegalEntityPrincipal	Сведения об иностранном юридическом лице

Имя	Описание
SoleProprietorPrincipal	Сведения об индивидуальном предпринимателе
Individual	Сведения о физическом лице

PowerOfAttorneyEMCHDData.PrincipalsInfo — класс

Сведения о доверителе (доверителях) (СвДоверит)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PrincipalsInfo
```

Свойства

Имя	Описание
PrincipalType	Тип доверителя
PrincipalInfo	Сведения о доверителе

PowerOfAttorneyEMCHDData.PrincipalWithoutPowerOfAttorneyInfo — класс

Сведения о лице (лицах), в том числе законном представителе, действующем (действующих) без доверенности (ЛицоБезДовТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PrincipalWithoutPowerOfAttorneyInfo
```

Свойства

Имя	Описание
AuthorityType	Вид полномочий единоличного исполнительного органа

Имя	Описание
ManagementCompanyInfo	Сведения о единоличном исполнительном органе – управляющей компании
IndividualInfo	Сведения о единоличном исполнительном органе - физическом лице
SoleProprietorInfo	Сведения о единоличном исполнительном органе — индивидуальном предпринимателе

PowerOfAttorneyEMCHDData.RepresentativeInfo — класс

Сведения о доверителе (Доверит)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class RepresentativeInfo
```

Свойства

Имя	Описание
RussianEntity	Сведения о юридическом лице
SoleProprietor	Сведения об индивидуальном предпринимателе
Individual	Сведения о физическом лице
BranchOfRussianEntity	Сведения о филиале (обособленном подразделении) юридического лица
BranchOfForeignEntity	Сведения о филиале (аккредитованном представительстве) иностранного юридического лица

PowerOfAttorneyEMCHDData.RepresentativesInfo — класс

Сведения о представителе (представителях) (СвУпПредТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class RepresentativesInfo
```

Свойства

Имя	Описание
RepresentativeType	Тип представителя
Representative	Сведения о представителе

PowerOfAttorneyEMCHDData.RetrustPowerOfAttorneyDocumentData — класс

Передоверие (Передов)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class RetrustPowerOfAttorneyDocumentData
```

Свойства

Имя	Описание
PowerOfAttorney	Сведения о доверенности
Representative	Сведения о лице (лицах), получившем (получивших) полномочия
RepresentativePowers	Сведения о передаваемом (передаваемых) полномочии (полномочиях)
NotaryCertificateInfo	Сведения о нотариальном удостоверении
PrimaryPowerOfAttorney	Сведения о первоначальной доверенности

Имя	Описание
ParentPowerOfAttorney	Сведения о доверенности, на основании которой осуществляется передоверие
DelegatedAuthority	Сведения о лице (лицах), передавшем (передавших) полномочия

PowerOfAttorneyEMCHDData.RepresentativePowersInfo — класс

Сведения о полномочиях представителя (представителей) (СвПолнТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class RepresentativePowersInfo
```

Свойства

Имя	Описание
AuthorityType	Тип полномочия
JointRepresentationType	Признак совместного осуществления полномочий
LossOfAuthorityType	Признак утраты полномочий при передоверии
PowersTextContent	Текстовое содержание полномочия
MachineReadablePowersInfo	Сведения о машиночитаемом полномочии (машиночитаемых полномочиях)

PowerOfAttorneyEMCHDData.RussianLegalEntityPrincipalInfo — класс

Сведения о юридическом лице (РосОргДоверТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class RussianLegalEntityPrincipalInfo
```

Свойства

Имя	Описание
SoleExecutiveIsManagementCompany	Единоличным исполнительным органом выступает управляющая компания
SoleExecutiveIsIndividual	Единоличным исполнительным органом выступает физическое лицо
SoleExecutiveIsSoleProprietor	Единоличным исполнительным органом выступает индивидуальный предприниматель
EntityInfo	Сведения о юридическом лице
PrincipalsWithoutPowerOfAttorneyInfo	Сведения о лице (лицах), действующем (действующих) от имени юридического лица без доверенности

PowerOfAttorneyEMCHDData.SoleExecutiveIndividualInfo — класс

Сведения о физическом лице / руководителе юридического лица (иностранного юридического лица (или филиала (аккредитованного представительства) иностранного юридического лица)) (СвФЛТип).

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class SoleExecutiveIndividualInfo
```

Свойства

Имя	Описание
ParticipantStatus	Статус участника нотариального действия
Inn	ИНН физического лица

Имя	Описание
Snils	СНИЛС
Position	Должность
ConfirmationDocument	Документ, подтверждающий полномочия лица, действующего без доверенности / руководителя иностранного юридического лица / руководителя филиала (аккредитованного представительства) иностранного юридического лица / руководителя филиала (обособленного подразделения) юридического лица
IndividualInfo	Сведения о физическом лице

PowerOfAttorneyEMCHDData.SoleExecutiveManagementCompanyInfo — класс

Сведения о единоличном исполнительном органе – управляющей компании (СВЮЛ)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class SoleExecutiveManagementCompanyInfo
```

Свойства

Имя	Описание
LegalEntityInfo	Вид полномочий единоличного исполнительного органа
IndividualsInfo	Сведения о единоличном исполнительном органе – управляющей компании

PowerOfAttorneyEMCHDData.SoleProprietorInfo0 — класс

Сведения об индивидуальном предпринимателе (СведИПТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class SoleProprietorInfo0
```

Свойства

Имя	Описание
ParticipantStatus	Статус участника нотариального действия
Name	Наименование индивидуального предпринимателя
Ogrnip	ОГРНИП
Inn	ИНН индивидуального предпринимателя
Snils	СНИЛС
AuthorityDocument	Документ, подтверждающий полномочия единоличного исполнительного органа
IndividualInfo	Сведения о физическом лице

AuthorityType — перечисление

Тип полномочия

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum AuthorityType
```

Члены

Имя члена	Описание
`Text`	Текстовое человекочитаемое полномочие
`Code`	Машиночитаемое полномочие

PowerOfAttorneyEMCHDData.CitizenshipType — перечисление

Признак наличия гражданства

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum CitizenshipType
```

Члены

Имя члена	Описание
`Russian = 1`	Гражданин Российской Федерации
`Foreign = 2`	Иностранный гражданин
`Stateless = 3`	Лицо без гражданства

PowerOfAttorneyEMCHDData.EntityType — перечисление

Тип лица, передавшего полномочия / тип представителя

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum EntityType
```

Члены

Имя члена	Описание
`RussianEntity`	Юридическое лицо

Имя члена	Описание
SoleProprietor	Индивидуальный предприниматель
Individual	Физическое лицо
BranchOfEntity	Филиал (обособленное подразделение) юридического лица
BranchOfForeignEntity	Филиал (аккредитованное представительство) иностранного юридического лица

PowerOfAttorneyEMCHDData.Gender — перечисление

Пол

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum Gender
```

Члены

Имя члена	Описание
Male = 1	Мужской
Female = 2	Женский

PowerOfAttorneyEMCHDData.JointRepresentationType — перечисление

Признак совместных полномочий

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum JointRepresentationType
```

Члены

Имя члена	Описание
`Individual = 1`	Полномочия индивидуальны
`Joint = 2`	Полномочия совместные

PowerOfAttorneyEMCHDData.NotarialActionParticipantStatus — перечисление

Статус участника нотариального действия

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum NotarialActionParticipantStatus
```

Члены

Имя члена	Описание
`Declarant = 101`	Заявитель
`RepresentativeDeclarant = 102`	Заявитель, являющийся одновременно представителем
`Representative = 299`	Представитель
`Applicant = 301`	Рукоприкладчик
`Translator = 303`	Переводчик
`OtherPerson = 399`	Иное лицо, принимавшее участие в совершении нотариального действия

PowerOfAttorneyVerificationResult — перечисление

Результат проверки действительности доверенности.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum PowerOfAttorneyVerificationResult
```

Члены

Имя члена	Описание
Valid	Доверенность действительна.
Invalid	Доверенность недействительна.
NotUsed	Доверенность не использовалась.

PowerOfAttorneyEMCHDData.PowerOfAttorneyForm — перечисление

Форма доверенности

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum PowerOfAttorneyForm
```

Члены

Имя члена	Описание
Electronic	В электронной форме в машиночитаемом виде
Paper	На бумажном носителе

PowerOfAttorneyEMCHDData.PowerOfAttorneyKind — перечисление

Вид доверенности

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum PowerOfAttorneyKind
```

Члены

Имя члена	Описание
`Regular`	Доверенность с возможностью отзыва
`Irrevocable`	Безотзывная доверенность (без возможности отзыва)

PowerOfAttorneyEMCHDData.PowerOfAttorneyLossOfAuthorityType — перечисление

Признак утраты полномочий при передоверии

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum PowerOfAttorneyLossOfAuthorityType
```

Члены

Имя члена	Описание
`Lost = 1`	Полномочия утрачиваются в процессе передоверия
`NotLost = 2`	Полномочия не утрачиваются в процессе передоверия

PowerOfAttorneyEMCHDData.PowerOfAttorneyOption — перечисление

Признак доверенности

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum PowerOfAttorneyOption
```

Члены

Имя члена	Описание
`Primary`	Первоначальная доверенность
`Parent`	Доверенность, выданная в порядке передоверия ("предыдущая" доверенность)

PowerOfAttorneyEMCHDData.PrincipalType — перечисление

Тип доверителя

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum PrincipalType
```

Члены

Имя члена	Описание
`RussianEntity`	Юридическое лицо
`ForeignEntity`	Иностранное юридическое лицо
`SoleProprietor`	Индивидуальный предприниматель
`Individual`	Физическое лицо

PowerOfAttorneyEMCHDData.RevocationCondition — перечисление

Условие отзыва доверенности

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum RevocationCondition
```

Члены

Имя члена	Описание
`Expiration = 1`	По истечению срока действия
`Other = 2`	Прочие условия безотзывной доверенности

PowerOfAttorneyEMCHDData.RevocationPossibleType — перечисление

Признак безотзывной доверенности

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum RevocationPossibleType
```

Члены

Имя члена	Описание
`Possible = 1`	Отзыв возможен
`Impossible = 2`	Без возможности отзыв

PowerOfAttorneyEMCHDData.SoleExecutiveAuthorityType — перечисление

Вид полномочий единоличного исполнительного органа

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum SoleExecutiveAuthorityType
```

Члены

Имя члена	Описание
`Individual`	Индивидуальные
`Joint`	Совместные

PowerOfAttorneyFNSData — класс

Базовый класс сведений о доверенности в формате ФНС.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public abstract class PowerOfAttorneyFNSData : PowerOfAttorneyData
```

Перечисления

Имя	Описание
`RevocationCondition`	*Условие отзыва доверенности:* • По истечению срока действия — `Expiration = 1` • Прочие условия безотзывной доверенности — `Other = 2`
`RevocationPossibleType`	*Признак безотзывной доверенности:* • Отзыв возможен — `Possible = 1` • Без возможности отзыва `Impossible = 2`
`RetrustRevocationPossibleType`	*Признак передоверия безотзывной доверенности:* • Возможно — `Possible = 1` • Невозможно `Impossible = 2`
`CitizenshipType`	*Признак наличия гражданства:* • Гражданин Российской Федерации — `Russian = 1` • Иностранный гражданин `Foreign = 2` • Лицо без гражданства `Stateless = 3`

Имя	Описание
`PrincipalType`	*Полномочное лицо:* • Физическое лицо — `Individual` • Индивидуальный предприниматель — `SoleProprietor` • Российское юридическое лицо — `RussianEntity`
`PrincipalType0`	*Тип доверителя* • Физическое лицо — `Individual` • Индивидуальный предприниматель — `SoleProprietor` • Российское юридическое лицо — `RussianEntity` • Иностранное юридическое лицо — `ForeignEntity`
`PowerOfAttorneyLossOfAuthorityType`	*Признак утраты полномочий при передоверии:* • Полномочия утрачиваются в процессе передоверия — `Lost = 1` • Полномочия не утрачиваются в процессе передоверия — `NotLost = 2`
`JointRepresentationType`	*Признак совместных полномочий:* • Полномочия индивидуальны — `Individual = 1` • Полномочия совместные `Joint = 2`
`Gender`	*Пол* • Мужской — `Male = 1` • `Женский` — `Female = 2`

Имя	Описание
`DocumentKindCode`	*Код вида документа:* • Свидетельство о рождении — (Код 03) — `BirthCertificate` • Военный билет (Код 07) — `MilitaryTicket` • Паспорт иностранного гражданина (Код 10) — `ForeignPassport` • Свидетельство о рассмотрении ходатайства о признании лица беженцем на территории Российской Федерации по существу (Код 11) — `Petitions` • Вид на жительство в Российской Федерации (Код 12) — `Residence` • Удостоверение беженца (Код 13) — `RefugeeCertificate` • Разрешение на временное проживание в Российской Федерации (Код 15) — `TemporaryResidence` • Свидетельство о предоставлении временного убежища на территории Российской Федерации (Код 19) — `TemporaryShelter` • Паспорт гражданина Российской Федерации (Код 21) — `Passport` • Удостоверение личности военнослужащего Российской Федерации (Код 24) — `MilitaryIdentityCard`

PowerOfAttorneyFNSDOVBBData — класс

Сведения о доверенности ФНС в формате DOVBB.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PowerOfAttorneyFNSDOVBBData : PowerOfAttorneyFNSData
```

Методы

Имя	Описание
PowerOfAttorneyFNSDOVBBData(Guid, PowerOfAttorneyDocument, string)	Заполняет Сведения о доверенности *Параметры:* • powerOfAttorneyNumber — Единый регистрационный номер доверенности • document — Состав и структура документа • otherInformation — Иные сведения
override Guid GetPowerOfAttorneyNumber()	Заполняет номер доверенности
override DateTime GetPowerOfAttorneyStartDate()	Заполняет дату выдачи доверенности
override DateTime? GetPowerOfAttorneyEndDate()	Заполняет дату окончания срок действия доверенности
override PowerOfAttorneyRetrustType GetRetrustType()	Заполняет тип передоверия
override string GetPrincipal()	Заполняет сведения о доверителе
override string GetRepresentative()	Заполняет сведения о лице, получившем полномочия
override string GetSigner()	Заполняет сведения о полписанте
override string GetPowerOfAttorneyMachineReadableFileID()	Заполняет идентификатор файла МЧД
override List<Powers> GetRepresentativesPowers()	Заполняет сведения о полномочиях представителя

Свойства

Имя	Описание
AddressInfo	Сведения об адресе (АдрТип)
BasicPowerOfAttorneyInfo	Сведения об Основной доверенности (СвОснДовер)
BasicPowerOfAttorneyPrincipalInfo	Сведения о доверителе Основной доверенности (СвДовер0)
BranchManagerInfo	Сведения о руководителе обособленного подразделения (СвРукОП)
ConfirmationOfAuthorityDocument	Реквизиты документа, подтверждающего полномочия (РеквДокПдтвТип)
DelegatedAuthorityPrincipalInfo	Сведения о лице, передавшем полномочия (СвЛицПередПолн)
ElectronicDocumentTransferMethod	Способ передачи электронного нотариального документа (СпПрдЭНотДок)
FIO	Фамилия, имя, отчество (при наличии) (ФИОТип)
ForeignEntityInfo	Сведения об иностранном юридическом лице (СвИнОргТип)
ForeignLegalEntityPrincipalInfo	Сведения о доверителе – иностранном юридическом лице (ИнОргДовер)
HandwrittenSignature	Рукописная подпись (ПодпРукопис)
IdentityCardOfIndividual	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип)
IndividualDelegatedAuthorityInfo	Сведения о лице, передавшем полномочия – физическом лице (ФЛПрдПолн)
IndividualInfo0	Сведения по физическому лицу (СвФЛ)

Имя	Описание
IndividualInfo1	Сведения по физическому лицу (СвПоФЛ)
IndividualInfo2	Сведения о физическом лице (СведФизЛТип)
IndividualInfo	Сведения о физическом лице (СведФЛТип)
IndividualInfoBase	Управляет получением сведений о физическом лице.
IndividualPrincipalInfo	Сведения о доверителе – физическом лице (ФЛДоверТип)
IrrevocablePowerOfAttorneyInfo	Сведения о безотзывной доверенности (БезотзывТип)
LegalEntityInfo	Сведения об организации (СвОргТип)
LegalRepresentativeInfo	Сведения о законном представителе физического лица (СвЗакПредТип)
NotaryCertificateInfo	Сведения о нотариальном удостоверении (СвНотУдТип)
NotaryDeputyInfo	ВРИО нотариуса (ВрионНот)
NotaryInfo	Сведения о нотариусе, совершившем нотариальное действие (СвНотДейств)
NotaryPaymentInfo	Сведения об оплате за совершение нотариального действия (ОплатНотДейст)
OrganizationInfo	Сведения об организации (СвОрг)
PowerOfAttorneyDocument	Состав и структура документа (Документ)
PowerOfAttorneyDocumentData	Доверенность (Довер)
PowerOfAttorneyInfo	Сведения доверенности (СвДовТип)
PrincipalInfo	Сведения о доверителе (СвДоверит)

Имя	Описание
PrincipalWithoutPowerOfAttorneyInfo	Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов)
RepresentativeInfo	Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип)
RepresentativePowerInfo	Сведения о полномочиях представителя (представителей) (СвПолнТип)
RetrustPowerOfAttorneyInfo	Сведения доверенности, выданной в порядке передоверия (СвДовПер)
RetrustPowerOfAttorneyInfoData	Передоверие (Передов)
RussianEntityInfo	Сведения о российском юридическом лице (СвРосОргТип)
RussianLegalEntityPrincipalInfo	Сведения о доверителе – российском юридическом лице (РосОргДовер)
SoleProprietorInfo0	Сведения об индивидуальном предпринимателе (СведИПТип)
SoleProprietorInfo1	Сведения об индивидуальном предпринимателе (СвИПТип)

PowerOfAttorneyFNSDOVBBData.AddressInfo — класс

Сведения об адресе (АдрТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class AddressInfo
```

Методы

Имя	Описание
`AddressInfo(string, string)`	Заполняет Сведения об адресе (АдрТип) *Параметры* • `regionCode` — Субъект Российской Федерации (код) • `address` — Адрес в Российской Федерации

PowerOfAttorneyFNSDOVBBData.BasicPowerOfAttorneyInfo — класс

Сведения об Основной доверенности (СвОснДовер)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class BasicPowerOfAttorneyInfo
```

Методы

Имя	Описание
`BasicPowerOfAttorneyInfo(Guid, List<BasicPowerOfAttorneyPrincipalInfo>)`	Заполняет Сведения об Основной доверенности (СвОснДовер) *Параметры* • `basicPowerOfAttorneyID` — Единый регистрационный номер Основной доверенности • `principalInfo` — Сведения о доверителе Основной доверенности

PowerOfAttorneyFNSDOVBBData.BasicPowerOfAttorneyPrincipalInfo — класс

Сведения о доверителе Основной доверенности (СвДовер0)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`

- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class BasicPowerOfAttorneyPrincipalInfo
```

Методы

Имя	Описание
`BasicPowerOfAttorneyPrincipalInfo(RussianEntityInfo)`	Заполняет Сведения о доверителе Основной доверенности (СвДовер0) *Параметры:* • `russianEntity` — Сведения о доверителе – российском юридическом лице
`BasicPowerOfAttorneyPrincipalInfo(ForeignEntityInfo)`	Заполняет Сведения о доверителе Основной доверенности (СвДовер0) *Параметры:* • `foreignEntity` — Сведения о доверителе – иностранном юридическом лице
`BasicPowerOfAttorneyPrincipalInfo(SoleProprietorInfo1)`	Заполняет Сведения о доверителе Основной доверенности (СвДовер0) *Параметры:* • `soleProprietor` — Сведения о доверителе – индивидуальном предпринимателе
`BasicPowerOfAttorneyPrincipalInfo(IndividualPrincipalInfo)`	Заполняет Сведения о доверителе Основной доверенности (СвДовер0) *Параметры:* • `individualPrincipal` — Сведения о доверителе – физическом лице

PowerOfAttorneyFNSDOVBBData.BranchManagerInfo — класс

Сведения о руководителе обособленного подразделения (СвРукОП)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class BranchManagerInfo
```

Методы

Имя	Описание
BranchManagerInfo(string, IndividualInfo, string, ConfirmationOfAuthorityDocument)	Заполняет Сведения о руководителе обособленного подразделения *Параметры:* • authorityDocument — Наименование документа, подтверждающего полномочия • individualInfo — Сведения о физическом лице • inn — ИНН физического лица — руководителя обособленного подразделения • authorityDocumentRequisites — Реквизиты документа, подтверждающего полномочия

PowerOfAttorneyFNSDOVBBData.ConfirmationOfAuthorityDocument — класс

Реквизиты документа, подтверждающего полномочия (РеквДокПдтвТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ConfirmationOfAuthorityDocument
```

Методы

Имя	Описание
`ConfirmationOfAuthorityDocument(DateTime, string, string)`	Заполняет Реквизиты документа, подтверждающего полномочия (РеквДокПдтвТип) *Параметры:* • `issueDate` — Дата выдачи • `issuer` — Кем выдан • `identityOfDocument` — Сведения об удостоверении документа, если он удостоверен

PowerOfAttorneyFNSDOVBBData.DelegatedAuthorityPrincipalInfo — класс

Сведения о лице, передавшем полномочия (СвЛицПередПолн)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class DelegatedAuthorityPrincipalInfo
```

Методы

Имя	Описание
`DelegatedAuthorityPrincipalInfo(LegalEntityInfo, FIO)`	Заполняет Сведения о лице, передавшем полномочия (СвЛицПередПолн) *Параметры:* • `legalEntity` — Сведения о лице, передавшем полномочия – российском юридическом лице • `signer` — Сведения о физическом лице, подписывающем доверенность от имени лица, передавшего полномочия(российской организации) или сведения о физическом лице, подписывающем доверенность от своего имени
`DelegatedAuthorityPrincipalInfo(SoleProprietorInfo0, FIO)`	Заполняет Сведения о лице, передавшем полномочия (СвЛицПередПолн) *Параметры:* • `soleProprietor` — Сведения о лице, передавшем полномочия – индивидуальном предпринимателе • `signer` — Сведения о физическом лице, подписывающем доверенность от имени лица, передавшего полномочия(российской организации) или сведения о физическом лице, подписывающем доверенность от своего имени

Имя	Описание
`DelegatedAuthorityPrincipalInfo(IndividualDelegatedAuthorityInfo, FIO)`	Заполняет Сведения о лице, передавшем полномочия (СвЛицПередПолн) *Параметры:* • `individual` — Сведения о лице, передавшем полномочия – физическом лице • `signer` — Сведения о физическом лице, подписывающем доверенность от имени лица, передавшего полномочия(российской организации) или сведения о физическом лице, подписывающем доверенность от своего имени

PowerOfAttorneyFNSDOVBBData.ElectronicDocumentTransferMethod — класс

Способ передачи электронного нотариального документа (СпПрдЭНотДок)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class ElectronicDocumentTransferMethod
```

Методы

Имя	Описание
`ElectronicDocumentTransferMethod(bool?, bool?, bool?, bool?, string, string)`	Заполняет Способ передачи электронного нотариального документа (СпПрдЭНотДок) *Параметры:* • `fromPrincipalToEPGU` — Направление сформированного электронного документа в ЛК доверителя на сайте ЕПГУ • `fromAttorneyToEPGU` — Направление сформированного электронного документа в ЛК поверенного на сайте ЕПГУ • `fromDeclarantToFNP` — Направление сформированного электронного документа в ЛК заявителя на сайте ФНП • `fromAttorneyToFNP` — Направление сформированного электронного документа в ЛК поверенного на сайте ФНП • `otherSystem` — Другая информационная система • `otherIssueWay` — Другой способ выдачи

PowerOfAttorneyFNSDOVBBData.FIO — класс

Фамилия, имя, отчество (при наличии) (ФИОТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class FIO
```

Методы

Имя	Описание
`FIO(string, string, string)`	Заполняет Фамилия, имя, отчество (при наличии) (ФИОТип) *Параметры:* • `lastName` — Фамилия • `firstName` — Имя • `middleName` — Отчество

PowerOfAttorneyFNSDOVBBData.ForeignEntityInfo — класс

Сведения об иностранном юридическом лице (СвИнОргТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class ForeignEntityInfo
```

Методы

Имя	Описание
`ForeignEntityInfo(string, string registrationCountry, string, string, string, string, string, string, string)`	Заполняет Сведения об иностранном юридическом лице (СвИнОргТип) *Параметры:* • `name` — Наименование иностранной организации • `registrationCountry` — Страна регистрации (инкорпорации) • `regNumber` — Регистрационный номер • `legalAddress` — Адрес юридического лица на территории государства, в котором оно зарегистрировано • `inn` — ИНН организации • `kpp` — КПП обособленного подразделения • `accreditationRecordNumber` — Номер записи об аккредитации • `registeringAuthorityName` — Наименование регистрирующего орган • `actualAddress` — Адрес фактического места нахождения • `phone` — Контактный телефон

PowerOfAttorneyFNSDOVBBData.ForeignLegalEntityPrincipalInfo — класс

Сведения о доверителе – иностранном юридическом лице (ИнОргДовер)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class ForeignLegalEntityPrincipalInfo
```

Методы

Имя	Описание
`ForeignLegalEntityPrincipalInfo(ForeignEntityInfo, BranchManagerInfo)`	Заполняет Сведения о доверителе – иностранном юридическом лице *Параметры:* • `foreignEntityInfo` — Сведения об иностранном юридическом лице • `branchManagerInfo` — Сведения о руководителе обособленного подразделения

PowerOfAttorneyFNSDOVBBData.HandwrittenSignature — класс

Рукописная подпись (ПодпРукопис)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class HandwrittenSignature
```

Методы

Имя	Описание
`HandwrittenSignature(byte[], byte[], byte[], DateTime, FIO)`	Заполняет Рукописная подпись (ПодпРукопис) *Параметры:* • `pdfDocumentHash` — Хеш PDF документа • `signatureImage` — Изображение подписи • `signatureHash` — Хеш подписи • `signatureDate` — Дата и время подписи • `fio` — Фамилия, имя, отчество(при наличии) лица, подписавшего доверенность

PowerOfAttorneyFNSDOVBBData.IdentityCardOfIndividual — класс

Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IdentityCardOfIndividual
```

Методы

Имя	Описание
`IdentityCardOfIndividual(DocumentKindCode, string, DateTime, ё string, string)`	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип) .Параметры: * `documentKindCode` — Код вида документ * `documentSerialNumber` — Серия и номер документа * `issueDate` — Дата выдачи документа * `issuer` — Наименование органа, выдавшего документ * `issuerCode` — Код подразделения органа, выдавшего документ

PowerOfAttorneyFNSDOVBBData.IndividualDelegatedAuthorityInfo — класс

Сведения о лице, передавшем полномочия – физическом лице (ФЛПрдПолн)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualDelegatedAuthorityInfo : IndividualInfoBase
```

Методы

Имя	Описание
`IndividualDelegatedAuthorityInfo(string, string, IndividualInfo, FIO, LegalRepresentativeInfo) : base()`	Заполняет Сведения о лице, передавшем полномочия – физическом лице (ФЛПрдПолн) *Параметры:* • `inn` — ИНН физического лица • `snils` — СНИЛС • `individualInfo` — Сведения о физическом лице • `fio` — ФИО физического лица • `representativeInfo` — Сведения о законном представителе физического лица

PowerOfAttorneyFNSDOVBBData.IndividualInfo0 — класс

Сведения по физическому лицу (СвФЛ)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualInfo0 : IndividualInfoBase
```

Методы

Имя	Описание
`IndividualInfo0(string, IndividualInfo, string, string, string, ConfirmationOfAuthorityDocument) : base()`	Заполняет Сведения по физическому лицу (СвФЛ) *Параметры:* • `inn` — ИНН физического лица • `snils` — СНИЛС • `individualInfo` — Сведения о физическом лице • `position` — Должность • `authorityConfirmingDocumentName` — Наименование документа, подтверждающего полномочия • `authorityConfirmingDocument` — Реквизиты документа, подтверждающего полномочия

PowerOfAttorneyFNSDOVBBData.IndividualInfo1 — класс

Сведения по физическому лицу (СвПоФЛ)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualInfo1 : IndividualInfoBase
```

Методы

Имя	Описание
`IndividualInfo1(string, string, IndividualInfo, string, ConfirmationOfAuthorityDocument) : base()`	Заполняет Сведения по физическому лицу (СвПоФЛ) *Параметры:* • `inn` — ИНН физического лица • `snils` — СНИЛС • `individualInfo` — Сведения о физическом лице • `incapacityDocumentName` — Наименование документа, подтверждающего недееспособность доверителя физического лица • `authorityConfirmingDocument` — Реквизиты документа, подтверждающего полномочия законного представителя физического лица

PowerOfAttorneyFNSDOVBBData.IndividualInfo2 — класс

Сведения о физическом лице (СведФизЛТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualInfo2 : IndividualInfoBase
```

Методы

Имя	Описание
`IndividualInfo2(string, string, IndividualInfo, FIO) : base()`	Заполняет Сведения о физическом лице (СведФизЛТип) *Параметры:* • `inn` — ИНН физического лица • `snils` — СНИЛС • `individualInfo` — Сведения о физическом лице • `fio` — Фамилия, имя, отчество (при наличии) физического лица

PowerOfAttorneyFNSDOVBBData.IndividualInfo — класс

Сведения о физическом лице (СведФЛТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualInfo
```

Методы

Имя	Описание
`IndividualInfo(DateTime, CitizenshipType, string , string, Gender?, string, AddressInfo, IdentityCardOfIndividual)`	Заполняет Сведения о физическом лице (СведФЛТип) *Параметры:* • `birthDate` — Дата рождения • `citizenshipType` — Признак наличия гражданства • `birthPlace` — Место рождения • `contactPhone` — Контактный телефон • `gender` — Пол физического лица • `citizenship` — Гражданство (для иностранного гражданина) • `residenceAddress` — Адрес места жительства • `identityCard` — Сведения о документе, удостоверяющем личность физического лица

PowerOfAttorneyFNSDOVBBData.IndividualInfoBase — класс

Управляет получением сведений о физическом лице.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public abstract class IndividualInfoBase
```

Методы

Имя	Описание
`IndividualInfoBase(string, string, IndividualInfo)`	Заполняет сведения о физическом лице *Параметры:* • `inn` — ИНН физического лица • `snils` — СНИЛС физического лица • `individualInfo` — Сведения о физическом лице

PowerOfAttorneyFNSDOVBBData.IndividualPrincipalInfo — класс

Сведения о доверителе – физическом лице (ФЛДоверТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualPrincipalInfo : IndividualDelegatedAuthorityInfo
```

Методы

Имя	Описание
`IndividualPrincipalInfo(bool, string, IndividualInfo, string, string, FIO, LegalRepresentativeInfo) : base()`	Заполняет Сведения о лице, передавшем полномочия – физическом лице (ФЛПрдПолн) *Параметры:* • `hasFullCapacity` — Признак наличия полной гражданской дееспособности доверителя – физического лица • `inn` — ИНН физического лица • `snils` — СНИЛС • `documentOfIncapacity` — Наименование и реквизиты документа, подтверждающего недееспособность доверителя физического лица • `fio` — ФИО физического лица • `individualInfo` — Сведения о физическом лице • `representativeInfo` — Сведения о законном представителе физического лица

PowerOfAttorneyFNSDOVBBData.IrrevocablePowerOfAttorneyInfo — класс

Сведения о безотзывной доверенности (БезотзывТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IrrevocablePowerOfAttorneyInfo
```

Методы

Имя	Описание
`IrrevocablePowerOfAttorneyInfo(RevocationPossibleType, RetrustRevocationPossibleType?, RevocationCondition?, string)`	Заполняет Сведения о безотзывной доверенности (БезотзывТип) *Параметры:* • `revocationPossibleType` — Признак безотзывной доверенности • `retrustRevocationPossibleType` — Признак передоверия безотзывной доверенности • `revocationCondition` — Условие отзыва • `revocationConditionDescription` — Описание условия отзыва

PowerOfAttorneyFNSDOVBBData.LegalEntityInfo — класс

Сведения об организации (СвОргТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class LegalEntityInfo
```

Методы

Имя	Описание
`LegalEntityInfo(string, string, string, string, string, string, string, AddressInfo, AddressInfo)`	Заполняет Сведения об организации (СвОргТип) *Параметры:* • `name` — Наименование организации / Наименование обособленного подразделения организации • `inn` — ИНН организации • `kpp` — КПП организации (обособленного подразделения) • `ogrn` — ОГРН • `constituentDocument` — Наименование учредительного документа • `phone` — Контактный телефон • `registrationNumber` — Регистрационный номер обособленного подразделения (для нерезидентов) • `registrationAddress` — Адрес регистрации • `actualAddress` — Адрес фактического места нахождения

PowerOfAttorneyFNSDOVBBData.LegalRepresentativeInfo — класс

Сведения о законном представителе физического лица (СвЗакПредТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class LegalRepresentativeInfo
```

Методы

Имя	Описание
`LegalRepresentativeInfo(IndividualInfo1, LegalEntityInfo)`	Заполняет Сведения о законном представителе физического лица (СвЗакПредТип) *Параметры:* • `legalEntityInfo` — Сведения о юридическом лице • `individualInfo` — ведения о физическом лице

PowerOfAttorneyFNSDOVBBData.NotaryCertificateInfo — класс

Сведения о нотариальном удостоверении (СвНотУдТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class NotaryCertificateInfo
```

Методы

Имя	Описание
`NotaryCertificateInfo(string, string, NotaryInfo, NotaryPaymentInfo, List<ElectronicDocumentTransferMethod>, NotaryDeputyInfo, HandwrittenSignature, string)`	Заполняет Сведения о нотариальном удостоверении (СвНотУдТип) *Параметры:* • `registrationNumber` — Реестровый номер доверенности • `place` — Место совершения доверенности • `notaryInfo` — Сведения о нотариусе, совершившем нотариальное действие • `notaryPaymentInfo` — Сведения об оплате за совершение нотариального действия • `electronicDocumentTransferMethods` — Сп особ передачи электронного нотариального документа • `notaryDeputyInfo` — ВРИО нотариуса • `handwrittenSignature` — Рукописная подпись • `otherInfo` — Иные сведения удостоверительной надписи

PowerOfAttorneyFNSDOVBBData.NotaryDeputyInfo — класс

ВРИО нотариуса (ВрионНот)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class NotaryDeputyInfo
```

Методы

Имя	Описание
`NotaryDeputyInfo(string, string, FIO)`	Заполняет ВРИО нотариуса (ВрионНот) *Параметры* • `position` — Должность • `registrationNumber` — Регистрационный номер лица, сдавшего квалификационный экзамен в Минюсте • `fio` — Фамилия, имя, отчество (при наличии)

PowerOfAttorneyFNSDOVBBData.NotaryInfo — класс

Сведения о нотариусе, совершившем нотариальное действие (СвНотДейств)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class NotaryInfo
```

Методы

Имя	Описание
`NotaryInfo(string, string, FIO)`	Заполняет Сведения о нотариусе, совершившем нотариальное действие (СвНотДейств) *Параметры:* • `position` — Должность • `registrationNumber` — Регистрационный номер нотариуса в Минюсте • `fio` — Фамилия, имя, отчество (при наличии)

PowerOfAttorneyFNSDOVBBData.NotaryPaymentInfo — класс

Сведения об оплате за совершение нотариального действия (ОплатНотДейст)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class NotaryPaymentInfo
```

Методы

Имя	Описание
`NotaryPaymentInfo(decimal, decimal?)`	Заполняет Сведения об оплате за совершение нотариального действия (ОплатНотДейст) *Параметры:* • `paid` — Уплачено за совершение нотариального действия • `discount` — Предоставлена льгота на сумму

PowerOfAttorneyFNSDOVBBData.OrganizationInfo — класс

Сведения об организации (СвОрг)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class OrganizationInfo
```

Методы

Имя	Описание
`OrganizationInfo(LegalEntityInfo, List<IndividualInfo2>)`	Заполняет Сведения об организации (СвОрг) *Параметры:* • `legalEntityInfo` — Сведения о юридическом лице • `individualInfo` — Сведения по физическому лицу

PowerOfAttorneyFNSDOVBBData.PowerOfAttorneyDocument — класс

Состав и структура документа (Документ)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PowerOfAttorneyDocument
```

Методы

Имя	Описание
`PowerOfAttorneyDocument(PowerOfAttorneyDocumentData)`	Создание первичной доверенности *Параметры:* • `documentData` — Данные доверенности
`PowerOfAttorneyDocument(RetrustPowerOfAttorneyInfoData)`	Создание доверенности в рамках передоверия *Параметры:* • `retrustPowerInfo` — Данные доверенности в рамках передоверия

PowerOfAttorneyFNSDOVBBData.PowerOfAttorneyDocumentData — класс

Доверенность (Довер)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyDocumentData
```

Методы

Имя	Описание
PowerOfAttorneyDocumentData(PowerOfAttorneyInfo, PrincipalInfo, RepresentativeInfo, List<RepresentativePowerInfo>, NotaryCertificateInfo)	Заполняет Доверенность (Довер) *Параметры:* • powerOfAttorneyInfo — Сведения доверенности • principalInfo — Сведения о доверителе • representativeInfo — Сведения об уполномоченном представителе • representativesPowers — Сведения о полномочиях представителя • notaryCertificateInfo — Сведения о нотариальном удостоверении

PowerOfAttorneyFNSDOVBBData.PowerOfAttorneyInfo — класс

Сведения доверенности (СвДовТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyInfo
```

Методы

Имя	Описание
`PowerOfAttorneyInfo(Guid, DateTime, DateTime, PowerOfAttorneyRetrustType, JointRepresentationType, IrrevocablePowerOfAttorneyInfo, string, string, DateTime?, PowerOfAttorneyLossOfAuthorityType?)`	Заполняет Сведения доверенности (СвДовТип) *Параметры:* • `powerOfAttorneyNumber` — Единый регистрационный номер доверенности • `powerOfAttorneyStartDate` — Дата совершения (выдачи) доверенности • `powerOfAttorneyEndDate` — Дата окончания действия доверенности • `retrustType` — Признак возможности оформления передоверия • `jointRepresentation` — Признак совместных полномочий • `irrevocablePowerOfAttorneyInfo` — Сведения о безотзывной доверенности • `organizationName` — Название организации-владельца информационной системы • `powerOfAttorneyInternalNumber` — Внутренний регистрационный номер доверенности • `powerOfAttorneyInternalRegistrationDate` — Дата внутренней регистрации доверенности • `lossOfAuthorityType` — Признак утраты полномочий при передоверии

PowerOfAttorneyFNSDOVBBData.PrincipalInfo — класс

Сведения о доверителе (СвДоверит)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`

- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PrincipalInfo
```

Методы

Имя	Описание
`PrincipalInfo(RussianLegalEntityPrincipalInfo, FIO)`	Заполняет Сведения о доверителе (СвДоверит) *Параметры:* • `russianLegalEntityPrincipal` — Сведения о доверителе – российском юридическом лице • `signer` — Сведения о физическом лице, подписывающем доверенность от имени доверителя (российской организации / иностранной организации) без доверенности или сведения о физическом лице, подписывающем доверенность от своего имени/законном представителе физического лица

Имя	Описание
`PrincipalInfo(ForeignLegalEntityPrincipalInfo, FIO)`	Заполняет Сведения о доверителе (СвДоверит) *Параметры:* • `foreignLegalEntityPrincipal` — Сведения о доверителе – иностранном юридическом лице • `signer` — Сведения о физическом лице, подписывающем доверенность от имени доверителя (российской организации / иностранной организации) без доверенности или сведения о физическом лице, подписывающем доверенность от своего имени/законном представителе физического лица
`PrincipalInfo(SoleProprietorInfo0, FIO)`	Заполняет Сведения о доверителе (СвДоверит) *Параметры:* • `soleProprietor` — Сведения о доверителе – индивидуальном предпринимателе • `signer` — Сведения о физическом лице, подписывающем доверенность от имени доверителя (российской организации / иностранной организации) без доверенности или сведения о физическом лице, подписывающем доверенность от своего имени/законном представителе физического лица

Имя	Описание
`PrincipalInfo(IndividualPrincipalInfo, FIO)`	Заполняет Сведения о доверителе (СвДоверит) *Параметры:* • `individualPrincipal` — Сведения о доверителе – физическом лице • `signer` — Сведения о физическом лице, подписывающем доверенность от имени доверителя (российской организации / иностранной организации) без доверенности или сведения о физическом лице, подписывающем доверенность от своего имени/законном представителе физического лица

PowerOfAttorneyFNSDOVBBData.PrincipalWithoutPowerOfAttorneyInfo — класс

Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PrincipalWithoutPowerOfAttorneyInfo
```

Методы

Имя	Описание
`PrincipalWithoutPowerOfAttorneyInfo(IndividualInfo0, LegalEntityInfo)`	Заполняет Сведения о лице, действующем от имени юридического лица без доверенности *Параметры:* • `individualInfo` — Сведения по физическому лицу • `legalEntityInfo` — Сведения о юридическом лице

PowerOfAttorneyFNSDOVBBData.RepresentativeInfo — класс

Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RepresentativeInfo
```

Методы

Имя	Описание
`RepresentativeInfo(OrganizationInfo)`	Заполняет Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип) *Параметры:* • `organizationInfo` — Сведения об организации

Имя	Описание
`RepresentativeInfo(SoleProprietorInfo1)`	Заполняет Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип) *Параметры:* • `soleProprietorInfo` — Сведения об индивидуальном предпринимателе
`RepresentativeInfo(IndividualInfo2)`	Заполняет Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип) *Параметры:* • `individualInfo` — Сведения о физическом лице

PowerOfAttorneyFNSDOVBBData.RepresentativePowerInfo — класс

Сведения о полномочиях представителя (представителей) (СвПолнТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RepresentativePowerInfo
```

Методы

Имя	Описание
`RepresentativePowerInfo(PowersCode, string, string)`	Заполняет Сведения о полномочиях представителя (представителей) (СвПолнТип) *Параметры:* • `powersCode` — Код полномочий уполномоченного представителя • `content` — Содержание полномочий уполномоченного представителя • `additionalInfo` — Дополнительное описание полномочий уполномоченного представителя
`RepresentativePowerInfo(string, string, string)`	Заполняет Сведения о полномочиях представителя (представителей) (СвПолнТип) *Параметры:* • `powersDescription` — Текстовое описание полномочий уполномоченного представителя • `content` — Содержание полномочий уполномоченного представителя • `additionalInfo` — Дополнительное описание полномочий уполномоченного представителя

PowerOfAttorneyFNSDOVBBData.RetrustPowerOfAttorneyInfo — класс

Сведения доверенности, выданной в порядке передоверия (СвДовПер)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RetrustPowerOfAttorneyInfo
```

Методы

Имя	Описание
`RetrustPowerOfAttorneyInfo(BasicPowerOfAttorneyInfo, Guid, PowerOfAttorneyInfo)`	Заполняет Сведения доверенности, выданной в порядке передоверия (СвДовПер) *Параметры:* • `basicPowerOfAttorneyInfo` — Сведения об Основной доверенности (СвОснДовер) • `parentPowerOfAttorneyID` — Единый регистрационный номер доверенности, на основании которой осуществляется передоверие • `powerOfAttorneyInfo` — Сведения о доверенности, выданной в порядке передоверия

PowerOfAttorneyFNSDOVBBData.RetrustPowerOfAttorneyInfoData — класс

Передоверие (Передов)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RetrustPowerOfAttorneyInfoData
```

Методы

Имя	Описание
`RetrustPowerOfAttorneyInfoData(RetrustPowerOfAttorneyInfo, DelegatedAuthorityPrincipalInfo, RepresentativeInfo, List<RepresentativePowerInfo>, NotaryCertificateInfo)`	Заполняет Передоверие (Передов) *Параметры:* • `retrustPowerOfAttorneyInfo` — Сведения доверенности, выданной в порядке передоверия • `delegatedAuthorityPrincipalInfo` — Сведения о лице, передавшем полномочия • `representativePrincipalInfo` — Сведения о лице, получившем полномочия • `representativePowersInfo` — Сведения о передаваемых полномочиях • `notaryCertificateInfo` — Сведения о нотариальном удостоверении

PowerOfAttorneyFNSDOVBBData.RussianEntityInfo — класс

Сведения о российском юридическом лице (СвРосОргТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RussianEntityInfo
```

Методы

Имя	Описание
`RussianEntityInfo(string, string, string, string, string, string, string, string)`	Заполняет Сведения о российском юридическом лице (СвРосОргТип) *Параметры:* • `name` — Наименование организации • `ogrn` — ОГРН • `inn` — ИНН организации • `kpp` — КПП организации • `legalAddress` — Адрес юридического лица в Российской Федерации • `actualAddress` — Адрес фактического места нахождения • `constituentDocument` — Наименование учредительного документа • `phone` — Контактный телефон

PowerOfAttorneyFNSDOVBBData.RussianLegalEntityPrincipalInfo — класс

Сведения о доверителе – российском юридическом лице (РосОргДовер)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RussianLegalEntityPrincipalInfo
```

Методы

Имя	Описание
`RussianLegalEntityPrincipalInfo(RussianEntityInfo, PrincipalWithoutPowerOfAttorneyInfo)`	Заполняет Сведения о доверителе – российском юридическом лице (РосОргДовер) *Параметры:* • `russianEntityInfo` — Сведения о российском юридическом лице • `principalWithoutPowerOfAttorneyInfo` — Сведения о лице, действующем от имени юридического лица без доверенности

PowerOfAttorneyFNSDOVBBData.SoleProprietorInfo0 — класс

Сведения об индивидуальном предпринимателе (СведИПТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class SoleProprietorInfo0
```

Методы

Имя	Описание
`SoleProprietorInfo0(string, string, string, string, IndividualInfo)`	Заполняет Сведения об индивидуальном предпринимателе (СвИПТип) *Параметры:* • `name` — Наименование индивидуального предпринимателя • `inn` — ИНН индивидуального предпринимателя • `ogrnip` — ОГРНИП • `snils` — СНИЛС • `individualInfo` — Сведения о физическом лице

PowerOfAttorneyFNSDOVBBData.SoleProprietorInfo1 — класс

Сведения об индивидуальном предпринимателе (СвИПТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class SoleProprietorInfo1 : SoleProprietorInfo0
```

Методы

Имя	Описание
`SoleProprietorInfo1(string, string, string, string, IndividualInfo, FIO) : base()`	Заполняет Сведения об индивидуальном предпринимателе (СвИПТип) • `name` — Наименование индивидуального предпринимателя • `inn` — ИНН индивидуального предпринимателя • `ogrnip` — ОГРНИП индивидуального предпринимателя • `snils` — СНИЛС индивидуального предпринимателя • `individualInfo` — Сведения о физическом лице • `fio` — Фамилия, имя, отчество (при наличии)

PowerOfAttorneyFNSDOVEL502Data.PowerOfAttorneyFNSDOVEL502Data — класс

Данные доверенности в формате, подтверждающем полномочия представителя налогоплательщика, версия 5.02.

Формат 5.02 устарел. Формируйте доверенности в формате 5.03, PowerOfAttorneyFNSDOVEL503Data])

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PowerOfAttorneyFNSDOVEL502Data : PowerOfAttorneyFNSData
```

Свойства

Имя	Описание
`FormatId`	Идентификатор формата доверенности

Имя	Описание
`Document`	Состав и структура документа (Обязательный)
`RecipientID`	Идентификатор получателя файла доверенности (Обязательный)
`FinalRecipientID`	Идентификатор конечного получателя файла доверенности (Обязательный) В случае передачи файла от отправителя к конечному получателю при отсутствии налоговых органов, осуществляющих передачу на промежуточных этапах, значения `RecipientID` и `FinalRecipientID` совпадают.
`SenderID`	Идентификатор отправителя информации (Обязательный) *Значение:* • Для организаций — 19-разрядный код (ИНН и КПП организации). • Для физических лиц — 12-разрядный код (ИНН физического лица, при наличии. При отсутствии ИНН – последовательность из двенадцати нулей).

Методы

Имя	Описание
`override DateTime GetPowerOfAttorneyStartDate()`	Заполняет дату выдачи доверенности
`override DateTime? GetPowerOfAttorneyEndDate()`	Заполняет дату окончания срок действия доверенности
`override string GetPowerOfAttorneyMachineReadableFileID()`	Заполняет идентификатор файла МЧД
`override Guid GetPowerOfAttorneyNumber()`	Возвращает номер доверенности

Имя	Описание
`override string GetPrincipal()`	Заполняет сведения о доверителе
`override string GetRepresentative()`	Заполняет сведения о лице, получившем полномочия
`override List<Powers> GetRepresentativesPowers()`	Заполняет сведения о полномочиях представителя
`override PowerOfAttorneyRetrustType GetRetrustType()`	Заполняет тип передоверия
`override string GetSigner()`	Заполняет сведения о подписанте

Классы

Имя	Описание
`PowerOfAttorneyDocument`	Состав и структура документа (Документ). Код формы по КНД принимает значение 1110310 в соответствии с форматом и не может быть изменён.
`PowerOfAttorneyDocumentData`	Доверенность (Довер) — Таблица 4.3
`PowerOfAttorneyInfo`	Сведения доверенности (СвДов) — Таблица 4.4
`PrincipalInfo`	Сведения о доверителе (СвДоверит) — Таблица 4.5
`RussianCompanyInfo`	Сведения о российской организации (НПЮЛ) — Таблица 4.6
`PrincipalWithoutPowerOfAttorneyInfo`	Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов) — Таблица 4.7
`IndividualInfo`	Сведения по физическому лицу (СвФЛ) — Таблица 4.8
`ForeignCompanyInfo`	Сведения об иностранной организации (ИО) — Таблица 4.9

Имя	Описание
SeparateSubdivisionManagerInfo	Сведения о руководителе обособленного подразделения (СвРукОП) — Таблица 4.10
RetrustPowerOfAttorneyDocumentData	Передоверие (Передов) — Таблица 4.11
RetrustPowerOfAttorneyInfo	Сведения доверенности, совершённой (выданной) в рамках передоверия (СвДовП) — Таблица 4.12
PrimaryPowerOfAttorneyInfo	Сведения об основной доверенности (СвОснДов) — Таблица 4.13
PrimaryPrincipalInfo	Сведения о доверителе основной доверенности (СвДоверитОсн) — Таблица 4.14
ForeignCompanyInfo1	Сведения о доверителе – иностранном юридическом лице (ДоверитИО) — Таблица 4.15
ParentPrincipalInfo	Сведения о доверителе из доверенности, на основании которой осуществляется передоверие (СвДоверщП) — Таблица 4.16
RetrustPrincipalInfo	Сведения о доверителе в порядке передоверия (СвДоверщ) — Таблица 4.17
OrganizationInfo	Сведения об организации (СвОргТип) — Таблица 4.18
RussianCompanyInfo1	Сведения о российском юридическом лице (СведЮЛТип) — Таблица 4.19
RepresentativeInfo	Сведения об уполномоченном представителе (уполномоченных представителях) (СвПредТип) — Таблица 4.20

Имя	Описание
IndividualInfo2	Сведения о физическом лице, в том числе индивидуальном предпринимателе (СвФизЛицТип) — Таблица 4.21
IndividualInfo3	Сведения по физическому лицу (СведФЛТип) — Таблица 4.22
AddressInfo	Адрес доверителя в доверенности (АдрДовТип) — Таблица 4.23
IdentityCardInfo	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип) — Таблица 4.24
FIO	Фамилия, имя, отчество (ФИОТип) — Таблица 4.25

Перечисления

Имя	Описание
RetrustType	Признак возможности оформления передоверия: • Followed = 1 — передоверие возможно. • None = 2 — без права передоверия.

PowerOfAttorneyFNSDOVEL502Data.PowerOfAttorneyDocument — класс

Состав и структура документа (Документ). Код формы по КНД принимает значение 1110310 в соответствии с форматом и не может быть изменён.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyDocument
```

Свойства

Имя	Описание
PowerOfAttorneyNumber	Единый регистрационный номер доверенности (Обязательный).
TaxAuthorityCode	Код налогового органа, в который представляется доверенность, созданная в электронной форме (Обязательный). При оформлении доверенности в отношении одного налогового органа принимает значение кода этого налогового органа. При оформлении доверенности в отношении нескольких налоговых органов принимает значение кода налогового органа постановки налогоплательщика на учет по месту нахождения (месту жительства).
PowerOfAttorneyData	Доверенность (Обязательный с условием).
RetrustPowerOfAttorneyData	Передоверие (Обязательный с условием). Обязательный при создании доверенности в рамках передоверия.
Signer	Сведения о физическом лице, подписывающем доверенность от имени доверителя (российской организации / иностранной организации) без доверенности или подписывающем доверенность от своего имени, в том числе при передаче полномочий другому лицу в порядке передоверия (Обязательный).

PowerOfAttorneyFNSDOVEL502Data.PowerOfAttorneyDocumentData — класс

Доверенность (Довер) — Таблица 4.3.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyDocumentData
```

Свойства

Имя	Описание
PowerOfAttorney	Сведения доверенности (Обязательный).
Principal	Сведения о доверителе (Обязательный). Сведения о лице, доверяющем представителю выступать от своего имени в отношениях, регулируемых законодательством Российской Федерации о налогах и сборах.
Representative	Сведения об уполномоченном представителе (Обязательный). Расхождение с форматом: поддерживается только один уполномоченный представитель.
RepresentativePowers	Признак (код) области полномочий уполномоченного представителя (Обязательный).

PowerOfAttorneyFNSDOVEL502Data.PowerOfAttorneyInfo — класс

Сведения доверенности (СвДов) — Таблица 4.4.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyInfo
```

Свойства

Имя	Описание
InternalPowerOfAttorneyNumber	Номер доверенности. Внутренний номер доверенности.
PowerOfAttorneyStartDate	Дата совершения (выдачи) доверенности (Обязательный).
PowerOfAttorneyEndDate	Дата окончания срока действия доверенности (Обязательный).
RetrustType	Признак возможности оформления передоверия (Обязательный).
TaxCodes	Код налогового органа, в отношении которого действует доверенность. При оформлении доверенности в отношении всех налоговых органов Российской Федерации не заполняется.

PowerOfAttorneyFNSDOVEL502Data.PrincipalInfo — класс

Сведения о доверителе (СвДоверит) — Таблица 4.5.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrincipalInfo
```

Свойства

Имя	Описание
RussianCompany	Сведения о российской организации (Обязательный с условием). Обязательный, если доверителем является российская организация.

Имя	Описание
ForeignCompany	Сведения об иностранной организации (Обязательный с условием). Обязательный, если доверителем является иностранная организация.
Individual	Сведения о физическом лице (Обязательный с условием). Обязательный, если доверителем является физическое лицо.

PowerOfAttorneyFNSDOVEL502Data.RetrustPrincipalInfo — класс

Сведения о доверителе в порядке передоверия (СвДоверщ) - 4.17.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RetrustPrincipalInfo
```

Свойства

Имя	Описание
Organization	Сведения об организации (Обязательный с условием). Обязательный, если доверяющий действует от имени юридического лица.
Individual	Сведения по физическому лицу (Обязательный). Указываются сведения об индивидуальном предпринимателе или физическом лице, в том числе, когда в доверенности в качестве доверяющего указывается юридическое лицо.

PowerOfAttorneyFNSDOVEL502Data.PrincipalWithoutPowerOfAttorneyInfo — класс

Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов) — Таблица 4.7.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrincipalWithoutPowerOfAttorneyInfo
```

Свойства

Имя	Описание
Organization	Сведения об организации (Обязательный с условием). Обязательный, если полномочия единоличного исполнительного органа хозяйственного товарищества или общества, или в соответствии с законодательством о банкротстве переданы управляющей организации.
Individual	Указываются сведения о физическом лице, действующем без доверенности от имени юридического лица или без доверенности от имени управляющей организации или в соответствии с законодательством о банкротстве.

PowerOfAttorneyFNSDOVEL502Data.IndividualInfo — класс

Сведения по физическому лицу (СвФЛ) — Таблица 4.8.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class IndividualInfo
```

Свойства

Имя	Описание
Inn	ИНН физического лица.
Snils	СНИЛС (Обязательный).
Citizenship	Гражданство. Принимает значение в соответствии с Общероссийским классификатором стран мира.
BirthDate	Дата рождения.
Position	Должность.

PowerOfAttorneyFNSDOVEL502Data.ForeignCompanyInfo — класс

Сведения об иностранной организации (ИО) — Таблица 4.9.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ForeignCompanyInfo
```

Свойства

Имя	Описание
Name	Наименование иностранной организации (Обязательный).
Inn	ИНН организации (Обязательный).
Kpp	КПП обособленного подразделения (Обязательный).
RegistrationCountry	Страна регистрации (инкорпорации) (Обязательный). Принимает значение в соответствии с Общероссийским классификатором стран мира.
RegisteringAuthorityName	Наименование регистрирующего органа.

Имя	Описание
RegNumber	Регистрационный номер в стране регистрации (инкорпорации) (Обязательный).
TaxpayerCode	Код налогоплательщика в стране регистрации (инкорпорации) или аналог.
LegalAddress	Полный адрес иностранной организации в стране регистрации (инкорпорации) (Обязательный).
Manager	Сведения о руководителе обособленного подразделения (Обязательный).

PowerOfAttorneyFNSDOVEL502Data.SeparateSubdivisionManagerInfo — класс

Сведения о руководителе обособленного подразделения (СвРукОП) — Таблица 4.10.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class SeparateSubdivisionManagerInfo
```

Свойства

Имя	Описание
Inn	ИНН физического лица - руководителя обособленного подразделения.
Gender	Пол физического лица (Обязательный).
BirthDate	Дата рождения (Обязательный).
BirthPlace	Место рождения.
CitizenshipType	Признак наличия гражданства (Обязательный).

Имя	Описание
Citizenship	Гражданство (Обязательный с условием). Обязательный для иностранного гражданина. Принимает значение в соответствии с Общероссийским классификатором стран мира.

PowerOfAttorneyFNSDOVEL502Data.RetrustPowerOfAttorneyDocumentData — класс

Передоверие (Передов) — Таблица 4.11.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RetrustPowerOfAttorneyDocumentData
```

Свойства

Имя	Описание
PowerOfAttorney	Сведения доверенности, совершённой (выданной) в рамках передоверия (Обязательный).
Principal	Сведения о доверителе в порядке передоверия (Обязательный). Сведения о лице, передающем полномочия другому лицу в порядке передоверия.
Representative	Сведения об уполномоченном представителе (Обязательный). Расхождение с форматом: поддерживается только один уполномоченный представитель.
RepresentativePowers	Признак (код) области полномочий уполномоченного представителя (Обязательный).

PowerOfAttorneyFNSDOVEL502Data.RetrustPowerOfAttorneyInfo — класс

Сведения доверенности, совершённой (выданной) в рамках передоверия (СвДовП) — Таблица 4.12.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RetrustPowerOfAttorneyInfo
```

Свойства

Имя	Описание
`ParentPowerOfAttorneyNumber`	Регистрационный номер доверенности, в отношении которой производится передоверие (Обязательный). Если передоверие осуществляется на основании основной доверенности, значение совпадает с регистрационным номером основной доверенности.
`InternalPowerOfAttorneyNumber`	Номер доверенности. Внутренний номер доверенности.
`PowerOfAttorneyStartDate`	Дата совершения (выдачи) доверенности (Обязательный).
`PowerOfAttorneyEndDate`	Дата окончания срока действия доверенности (Обязательный).
`RetrustType`	Признак возможности оформления передоверия (Обязательный).
`PrimaryPowerOfAttorney`	Сведения об основной доверенности (Обязательный).

Имя	Описание
ParentPrincipal	Сведения о доверителе из доверенности, на основании которой осуществляется передоверие (Обязательный с условием). Элемент должен отсутствовать, если передоверие осуществляется на основании основной доверенности.
TaxCodes	Код налогового органа, в отношении которого действует доверенность. При оформлении доверенности в отношении всех налоговых органов Российской Федерации не заполняется.

PowerOfAttorneyFNSDOVEL502Data.PrimaryPowerOfAttorneyInfo — класс

Сведения об основной доверенности (СвОснДов) — Таблица 4.13.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrimaryPowerOfAttorneyInfo
```

Свойства

Имя	Описание
PowerOfAttorneyNumber	Единый регистрационный номер основной доверенности (Обязательный)
Principal	Сведения о доверителе основной доверенности (Обязательный)

PowerOfAttorneyFNSDOVEL502Data.PrimaryPrincipalInfo — класс

Сведения о доверителе основной доверенности (СвДоверитОсн) — Таблица 4.14.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrimaryPrincipalInfo
```

Свойства

Имя	Описание
RussianCompany	Сведения о доверителе — российском юридическом лице (Обязательный с условием). Обязательный, если доверителем является российская организация.
ForeignCompany	Сведения о доверителе — иностранном юридическом лице (Обязательный с условием). Обязательный, если доверителем является иностранная организация.
Individual	Сведения о доверителе — физическом лице (Обязательный с условием). Обязательный, если доверителем является физическое лицо.

PowerOfAttorneyFNSDOVEL502Data.ForeignCompanyInfo1 — класс

Сведения о доверителе – иностранном юридическом лице (ДоверитИО) — Таблица 4.15.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ForeignCompanyInfo1
```

Свойства

Имя	Описание
Name	Наименование иностранной организации (Обязательный)
Inn	ИНН организации
Kpp	КПП обособленного подразделения

PowerOfAttorneyFNSDOVEL502Data.ParentPrincipalInfo — класс

Сведения о доверителе из доверенности, на основании которой осуществляется передоверие (СвДоверщП) — Таблица 4.16.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ParentPrincipalInfo
```

Свойства

Имя	Описание
RussianCompany	Сведения о доверителе – российском юридическом лице (Обязательный с условием). Обязательный, если доверителем является российская организация.
Individual	Сведения о доверителе – физическом лице (Обязательный с условием). Обязательный, если доверителем является физическое лицо.

PowerOfAttorneyFNSDOVEL502Data.RetrustPrincipalInfo — класс

Сведения о доверителе в порядке передоверия (СвДоверщ) - 4.17

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RetrustPrincipalInfo
```

Свойства

Имя	Описание
`Organization`	Сведения об организации (Обязательный с условием). Обязательный, если доверяющий действует от имени юридического лица.
`Individual`	Сведения по физическому лицу (Обязательный). Указываются сведения о физическом лице, в том числе, когда в доверенности в качестве доверяющего указывается юридическое лицо.

PowerOfAttorneyFNSDOVEL502Data.OrganizationInfo — класс

Сведения об организации (СвОргТип) — Таблица 4.18.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class OrganizationInfo
```

Свойства

Имя	Описание
`Name`	Наименование организации / Наименование обособленного подразделения организации (Обязательный)
`Inn`	ИНН организации (Обязательный)

Имя	Описание
Kpp	КПП организации (Обязательный)
Ogrn	ОГРН

PowerOfAttorneyFNSDOVEL502Data.RussianCompanyInfo1 — класс

Сведения о российском юридическом лице (СведЮЛТип) — Таблица 4.19.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RussianCompanyInfo1
```

Свойства

Имя	Описание
Name	Наименование организации (Обязательный).
Inn	ИНН организации (Обязательный).
Kpp	КПП организации (Обязательный). Указывается КПП, присвоенный организации по месту ее нахождения.
Ogrn	ОГРН (Обязательный).

PowerOfAttorneyFNSDOVEL502Data.RepresentativeInfo — класс

Сведения об уполномоченном представителе (уполномоченных представителях) (СвПредТип) — Таблица 4.20.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RepresentativeInfo
```

Свойства

Имя	Описание
`Organization`	Сведения об организации (Обязательный с условием). Обязательный, если в доверенности в качестве уполномоченного представителя требуется указать юридическое лицо.
`Individual`	Сведения о физическом лице, в том числе индивидуальном предпринимателе (Обязательный). Указываются сведения об уполномоченном физическом лице (в том числе, когда в доверенности в качестве уполномоченного представителя указывается юридическое лицо) или индивидуальном предпринимателе. Расхождение с форматом: поддерживается только один уполномоченный представитель.

PowerOfAttorneyFNSDOVEL502Data.IndividualInfo2 — класс

Сведения о физическом лице, в том числе индивидуальном предпринимателе (СвФизЛицТип) — Таблица 4.21.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualInfo2
```

Свойства

Имя	Описание
`Inn`	ИНН физического лица.

Имя	Описание
`Ogrnip`	ОГРНИП (Обязательный с условием). Обязателен, если лицо является индивидуальным предпринимателем.
`Snils`	СНИЛС (Обязательный).
`Citizenship`	Гражданство. Принимает значение в соответствии с Общероссийским классификатором стран мира.
`BirthDate`	Дата рождения.
`Fio`	Фамилия, имя, отчество физического лица (Обязательный).
`IdentityCard`	Сведения о документе, удостоверяющем личность физического лица (Обязательный).

PowerOfAttorneyFNSDOVEL502Data.IndividualInfo3 — класс

Сведения по физическому лицу (СведФЛТип) — Таблица 4.22.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualInfo3
```

Свойства

Имя	Описание
`Inn`	ИНН физического лица.
`Ogrnip`	ОГРНИП (Обязательный с условием). Обязателен, если лицо является индивидуальным предпринимателем.
`Snils`	СНИЛС (Обязательный).

Имя	Описание
`Citizenship`	Гражданство. Принимает значение в соответствии с Общероссийским классификатором стран мира.
`BirthDate`	Дата рождения.
`IdentityCard`	Сведения о документе, удостоверяющем личность физического лица (Обязательный).

PowerOfAttorneyFNSDOVEL502Data.AddressInfo — класс

Адрес доверителя в доверенности (АдрДовТип) — Таблица 4.23

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class AddressInfo
```

Свойства

Имя	Описание
`AddressKyr`	Адрес в русской транскрипции (Обязательный с условием). Обязательный, если указывается адрес российской организации.
`AddressLat`	Адрес в латинской транскрипции (Обязательный с условием). Обязательный, если не указан `Address`.

PowerOfAttorneyFNSDOVEL502Data.IdentityCardInfo — класс

Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип) — Таблица 4.24.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IdentityCardInfo
```

Свойства

Имя	Описание
DocumentKindCode	Код вида документа (Обязательный).
DocumentSerialNumber	Серия и номер документа (Обязательный).
IssueDate	Дата выдачи документа (Обязательный).
Issuer	Наименование органа, выдавшего документ (Обязательный с условием). Обязательный, если вид документа - Паспорт гражданина Российской Федерации.
IssuerCode	Код подразделения органа, выдавшего документ.

PowerOfAttorneyFNSDOVEL502Data.FIO — класс

Фамилия, имя, отчество (ФИОТип) - 4.25

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class FIO
```

Свойства

Имя	Описание
LastName	Фамилия (Обязательный).
FirstName	Имя (Обязательный).
MiddleName	Отчество.

PowerOfAttorneyFNSDOVEL502Data.RetrustType — перечисление

Признак возможности оформления передоверия.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum RetrustType
```

Свойства

Имя	Описание
Followed = 1	Передоверие возможно
None = 2	Без права передоверия

PowerOfAttorneyFNSDOVEL503Data.PowerOfAttorneyFNSDOVEL503Data — класс

Данные доверенности в формате, подтверждающем полномочия представителя налогоплательщика, версия 5.03.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyFNSDOVEL503Data : PowerOfAttorneyFNSData
```

Свойства

Имя	Описание
FormatId	Идентификатор формата доверенности
Document	Состав и структура документа (Обязательный)
RecipientID	Идентификатор получателя файла доверенности (Обязательный)

Имя	Описание
`FinalRecipientID`	Идентификатор конечного получателя файла доверенности (Обязательный) В случае передачи файла от отправителя к конечному получателю при отсутствии налоговых органов, осуществляющих передачу на промежуточных этапах, значения `RecipientID` и `FinalRecipientID` совпадают
`SenderID`	Идентификатор отправителя информации (Обязательный) *Значение:* • Для организаций — 19-разрядный код (ИНН и КПП организации). • Для физических лиц — 12-разрядный код (ИНН физического лица, при наличии. При отсутствии ИНН – последовательность из двенадцати нулей).

Методы

Имя	Описание
`override DateTime GetPowerOfAttorneyStartDate()`	Заполняет дату выдачи доверенности
`override DateTime? GetPowerOfAttorneyEndDate()`	Заполняет дату окончания срок действия доверенности
`override string GetPowerOfAttorneyMachineReadableFileID()`	Заполняет идентификатор файла МЧД
`override Guid GetPowerOfAttorneyNumber()`	Возвращает номер доверенности
`override string GetPrincipal()`	Заполняет сведения о доверителе
`override string GetRepresentative()`	Заполняет сведения о лице, получившем полномочия

Имя	Описание
`override List<Powers> GetRepresentativesPowers()`	Заполняет сведения о полномочиях представителя
`override PowerOfAttorneyRetrustType GetRetrustType()`	Заполняет тип передоверия
`override string GetSigner()`	Заполняет сведения о подписанте

Классы

Имя	Описание
`PowerOfAttorneyDocument`	Состав и структура документа (Документ). Код формы по КНД принимает значение 1110310 в соответствии с форматом и не может быть изменён.
`PowerOfAttorneyDocumentData`	Доверенность (Довер) — Таблица 4.3
`PowerOfAttorneyInfo`	Сведения доверенности (СвДов) — Таблица 4.4
`PrincipalInfo`	Сведения о доверителе (СвДоверит) — Таблица 4.5
`RussianCompanyInfo`	Сведения о российской организации (НПЮЛ) — Таблица 4.6
`PrincipalWithoutPowerOfAttorneyInfo`	Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов) — Таблица 4.7
`IndividualInfo`	Сведения по физическому лицу (СвФЛ) — Таблица 4.8
`ForeignCompanyInfo`	Сведения об иностранной организации (ИО) — Таблица 4.9
`SeparateSubdivisionManagerInfo`	Сведения о руководителе обособленного подразделения (СвРукОП) — Таблица 4.10

Имя	Описание
RetrustPowerOfAttorneyDocumentData	Передоверие (Передов) — Таблица 4.11
RetrustPowerOfAttorneyInfo	Сведения доверенности, совершённой (выданной) в рамках передоверия (СвДовП) — Таблица 4.12
PrimaryPowerOfAttorneyInfo	Сведения об основной доверенности (СвОснДов) — Таблица 4.13
PrimaryPrincipalInfo	Сведения о доверителе основной доверенности (СвДоверитОсн) — Таблица 4.14
ForeignCompanyInfo1	Сведения о доверителе – иностранном юридическом лице (ДоверитИО) — Таблица 4.15
ParentPrincipalInfo	Сведения о доверителе из доверенности, на основании которой осуществляется передоверие (СвДоверщП) — Таблица 4.16
RetrustPrincipalInfo	Сведения о доверителе в порядке передоверия (СвДоверщ) — Таблица 4.17
OrganizationInfo	Сведения об организации (СвОргТип) — Таблица 4.18
RussianCompanyInfo1	Сведения о российском юридическом лице (СведЮЛТип) — Таблица 4.19
RepresentativeInfo	Сведения об уполномоченном представителе (уполномоченных представителях) (СвПредТип) — Таблица 4.20
IndividualInfo2	Сведения о физическом лице, в том числе индивидуальном предпринимателе (СвФизЛицТип) — Таблица 4.21
IndividualInfo3	Сведения по физическому лицу (СведФЛТип) — Таблица 4.22

Имя	Описание
`AddressInfo`	Адрес доверителя в доверенности (АдрДовТип) — Таблица 4.23
`IdentityCardInfo`	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип) — Таблица 4.24
`FIO`	Фамилия, имя, отчество (ФИОТип) — Таблица 4.25

Перечисления

Имя	Описание
`RetrustType`	Признак возможности оформления передоверия: • `Followed = 1` — передоверие возможно. • `None = 2` — без права передоверия.

PowerOfAttorneyFNSDOVEL503Data.PowerOfAttorneyDocument — класс

Состав и структура документа (Документ). Код формы по КНД принимает значение 1110310 в соответствии с форматом и не может быть изменён.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PowerOfAttorneyDocument
```

Свойства

Имя	Описание
`PowerOfAttorneyNumber`	Единый регистрационный номер доверенности (Обязательный).

Имя	Описание
TaxAuthorityCode	Код налогового органа, в который представляется доверенность, созданная в электронной форме (Обязательный). При оформлении доверенности в отношении одного налогового органа принимает значение кода этого налогового органа. При оформлении доверенности в отношении нескольких налоговых органов принимает значение кода налогового органа постановки налогоплательщика на учет по месту нахождения (месту жительства).
PowerOfAttorneyData	Доверенность (Обязательный с условием).
RetrustPowerOfAttorneyData	Передоверие (Обязательный с условием). Обязательный при создании доверенности в рамках передоверия.
Signer	Сведения о физическом лице, подписывающем доверенность от имени доверителя (российской организации / иностранной организации) без доверенности или подписывающем доверенность от своего имени, в том числе при передаче полномочий другому лицу в порядке передоверия (Обязательный).

PowerOfAttorneyFNSDOVEL503Data.PowerOfAttorneyDocumentData — класс

Доверенность (Довер) — Таблица 4.3.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyDocumentData
```

Свойства

Имя	Описание
PowerOfAttorney	Сведения доверенности (Обязательный).
Principal	Сведения о доверителе (Обязательный). Сведения о лице, доверяющем представителю выступать от своего имени в отношениях, регулируемых законодательством Российской Федерации о налогах и сборах.
Representative	Сведения об уполномоченном представителе (Обязательный). Расхождение с форматом: поддерживается только один уполномоченный представитель.
RepresentativePowers	Признак (код) области полномочий уполномоченного представителя (Обязательный).

PowerOfAttorneyFNSDOVEL503Data.PowerOfAttorneyInfo — класс

Сведения доверенности (СвДов) — Таблица 4.4.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyInfo
```

Свойства

Имя	Описание
InternalPowerOfAttorneyNumber	Номер доверенности. Внутренний номер доверенности.
PowerOfAttorneyStartDate	Дата совершения (выдачи) доверенности (Обязательный).
PowerOfAttorneyEndDate	Дата окончания срока действия доверенности (Обязательный).
RetrustType	Признак возможности оформления передоверия (Обязательный).
TaxCodes	Код налогового органа, в отношении которого действует доверенность. При оформлении доверенности в отношении всех налоговых органов Российской Федерации не заполняется.

PowerOfAttorneyFNSDOVEL503Data.PrincipalInfo — класс

Сведения о доверителе (СвДоверит) — Таблица 4.5.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrincipalInfo
```

Свойства

Имя	Описание
RussianCompany	Сведения о российской организации (Обязательный с условием). Обязательный, если доверителем является российская организация.
ForeignCompany	Сведения об иностранной организации (Обязательный с условием). Обязательный, если доверителем является иностранная организация.

Имя	Описание
Individual	Сведения о физическом лице (Обязательный с условием). Обязательный, если доверителем является физическое лицо.

PowerOfAttorneyFNSDOVEL503Data.RetrustPrincipalInfo — класс

Сведения о доверителе в порядке передоверия (СвДоверщ) - 4.17.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RetrustPrincipalInfo
```

Свойства

Имя	Описание
Organization	Сведения об организации (Обязательный с условием). Обязательный, если доверяющий действует от имени юридического лица.
Individual	Сведения по физическому лицу (Обязательный). Указываются сведения об индивидуальном предпринимателе или физическом лице, в том числе, когда в доверенности в качестве доверяющего указывается юридическое лицо.

PowerOfAttorneyFNSDOVEL503Data.PrincipalWithoutPowerOfAttorneyInfo — класс

Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов) — Таблица 4.7.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrincipalWithoutPowerOfAttorneyInfo
```

Свойства

Имя	Описание
Organization	Сведения об организации (Обязательный с условием). Обязательный, если полномочия единоличного исполнительного органа хозяйственного товарищества или общества, или в соответствии с законодательством о банкротстве переданы управляющей организации.
Individual	Указываются сведения о физическом лице, действующем без доверенности от имени юридического лица или без доверенности от имени управляющей организации или в соответствии с законодательством о банкротстве.

PowerOfAttorneyFNSDOVEL503Data.IndividualInfo — класс

Сведения по физическому лицу (СвФЛ) — Таблица 4.8.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class IndividualInfo
```

Свойства

Имя	Описание
Inn	ИНН физического лица.

Имя	Описание
RecordNumberOfFederalPopulationRegister	Номер записи федерального регистра сведений о населении
Snils	СНИЛС (Обязательный).
Citizenship	Гражданство. Принимает значение в соответствии с Общероссийским классификатором стран мира.
BirthDate	Дата рождения.
Position	Должность.

PowerOfAttorneyFNSDOVEL503Data.ForeignCompanyInfo — класс

Сведения об иностранной организации (ИО) — Таблица 4.9.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ForeignCompanyInfo
```

Свойства

Имя	Описание
Name	Наименование иностранной организации (Обязательный).
Inn	ИНН организации (Обязательный).
Kpp	КПП обособленного подразделения (Обязательный).
RegistrationCountry	Страна регистрации (инкорпорации) (Обязательный). Принимает значение в соответствии с Общероссийским классификатором стран мира.
RegisteringAuthorityName	Наименование регистрирующего органа.

Имя	Описание
RegNumber	Регистрационный номер в стране регистрации (инкорпорации) (Обязательный).
TaxpayerCode	Код налогоплательщика в стране регистрации (инкорпорации) или аналог.
LegalAddress	Полный адрес иностранной организации в стране регистрации (инкорпорации) (Обязательный).
Manager	Сведения о руководителе обособленного подразделения (Обязательный).

PowerOfAttorneyFNSDOVEL503Data.SeparateSubdivisionManagerInfo — класс

Сведения о руководителе обособленного подразделения (СвРукОП) — Таблица 4.10.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class SeparateSubdivisionManagerInfo
```

Свойства

Имя	Описание
Inn	ИНН физического лица - руководителя обособленного подразделения.
Gender	Пол физического лица (Обязательный).
BirthDate	Дата рождения (Обязательный).
BirthPlace	Место рождения.
CitizenshipType	Признак наличия гражданства (Обязательный).

Имя	Описание
Citizenship	Гражданство (Обязательный с условием). Обязательный для иностранного гражданина. Принимает значение в соответствии с Общероссийским классификатором стран мира.

PowerOfAttorneyFNSDOVEL503Data.RetrustPowerOfAttorneyDocumentData — класс

Передоверие (Передов) — Таблица 4.11.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RetrustPowerOfAttorneyDocumentData
```

Свойства

Имя	Описание
PowerOfAttorney	Сведения доверенности, совершённой (выданной) в рамках передоверия (Обязательный).
Principal	Сведения о доверителе в порядке передоверия (Обязательный). Сведения о лице, передающем полномочия другому лицу в порядке передоверия.
Representative	Сведения об уполномоченном представителе (Обязательный). Расхождение с форматом: поддерживается только один уполномоченный представитель.
RepresentativePowers	Признак (код) области полномочий уполномоченного представителя (Обязательный).

PowerOfAttorneyFNSDOVEL503Data.RetrustPowerOfAttorneyInfo — класс

Сведения доверенности, совершённой (выданной) в рамках передоверия (СвДовП) — Таблица 4.12.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RetrustPowerOfAttorneyInfo
```

Свойства

Имя	Описание
`ParentPowerOfAttorneyNumber`	Регистрационный номер доверенности, в отношении которой производится передоверие (Обязательный). Если передоверие осуществляется на основании основной доверенности, значение совпадает с регистрационным номером основной доверенности.
`InternalPowerOfAttorneyNumber`	Номер доверенности. Внутренний номер доверенности.
`PowerOfAttorneyStartDate`	Дата совершения (выдачи) доверенности (Обязательный).
`PowerOfAttorneyEndDate`	Дата окончания срока действия доверенности (Обязательный).
`RetrustType`	Признак возможности оформления передоверия (Обязательный).
`PrimaryPowerOfAttorney`	Сведения об основной доверенности (Обязательный).

Имя	Описание
ParentPrincipal	Сведения о доверителе из доверенности, на основании которой осуществляется передоверие (Обязательный с условием). Элемент должен отсутствовать, если передоверие осуществляется на основании основной доверенности.
TaxCodes	Код налогового органа, в отношении которого действует доверенность. При оформлении доверенности в отношении всех налоговых органов Российской Федерации не заполняется.

PowerOfAttorneyFNSDOVEL503Data.PrimaryPowerOfAttorneyInfo — класс

Сведения об основной доверенности (СвОснДов) — Таблица 4.13.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrimaryPowerOfAttorneyInfo
```

Свойства

Имя	Описание
PowerOfAttorneyNumber	Единый регистрационный номер основной доверенности (Обязательный)
Principal	Сведения о доверителе основной доверенности (Обязательный)

PowerOfAttorneyFNSDOVEL503Data.PrimaryPrincipalInfo — класс

Сведения о доверителе основной доверенности (СвДоверитОсн) — Таблица 4.14.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrimaryPrincipalInfo
```

Свойства

Имя	Описание
RussianCompany	Сведения о доверителе — российском юридическом лице (Обязательный с условием). Обязательный, если доверителем является российская организация.
ForeignCompany	Сведения о доверителе — иностранном юридическом лице (Обязательный с условием). Обязательный, если доверителем является иностранная организация.
Individual	Сведения о доверителе — физическом лице (Обязательный с условием). Обязательный, если доверителем является физическое лицо.

PowerOfAttorneyFNSDOVEL503Data.ForeignCompanyInfo1 — класс

Сведения о доверителе – иностранном юридическом лице (ДоверитИО) — Таблица 4.15.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ForeignCompanyInfo1
```

Свойства

Имя	Описание
Name	Наименование иностранной организации (Обязательный)
Inn	ИНН организации
Kpp	КПП обособленного подразделения

PowerOfAttorneyFNSDOVEL503Data.ParentPrincipalInfo — класс

Сведения о доверителе из доверенности, на основании которой осуществляется передоверие (СвДоверщП) — Таблица 4.16.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ParentPrincipalInfo
```

Свойства

Имя	Описание
RussianCompany	Сведения о доверителе – российском юридическом лице (Обязательный с условием). Обязательный, если доверителем является российская организация.
Individual	Сведения о доверителе – физическом лице (Обязательный с условием). Обязательный, если доверителем является физическое лицо.

PowerOfAttorneyFNSDOVEL503Data.RetrustPrincipalInfo — класс

Сведения о доверителе в порядке передоверия (СвДоверщ) - 4.17

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RetrustPrincipalInfo
```

Свойства

Имя	Описание
Organization	Сведения об организации (Обязательный с условием). Обязательный, если доверяющий действует от имени юридического лица.
Individual	Сведения по физическому лицу (Обязательный). Указываются сведения о физическом лице, в том числе, когда в доверенности в качестве доверяющего указывается юридическое лицо.

PowerOfAttorneyFNSDOVEL503Data.OrganizationInfo — класс

Сведения об организации (СвОргТип) — Таблица 4.18.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class OrganizationInfo
```

Свойства

Имя	Описание
Name	Наименование организации / Наименование обособленного подразделения организации (Обязательный)
Inn	ИНН организации (Обязательный)

Имя	Описание
Kpp	КПП организации (Обязательный)
Ogrn	ОГРН

PowerOfAttorneyFNSDOVEL503Data.RussianCompanyInfo1 — класс

Сведения о российском юридическом лице (СведЮЛТип) — Таблица 4.19.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RussianCompanyInfo1
```

Свойства

Имя	Описание
Name	Наименование организации (Обязательный).
Inn	ИНН организации (Обязательный).
Kpp	КПП организации (Обязательный). Указывается КПП, присвоенный организации по месту ее нахождения.
Ogrn	ОГРН (Обязательный).

PowerOfAttorneyFNSDOVEL503Data.RepresentativeInfo — класс

Сведения об уполномоченном представителе (уполномоченных представителях) (СвПредТип) — Таблица 4.20.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RepresentativeInfo
```

Свойства

Имя	Описание
`Organization`	Сведения об организации (Обязательный с условием). Обязательный, если в доверенности в качестве уполномоченного представителя требуется указать юридическое лицо.
`Individual`	Сведения о физическом лице, в том числе индивидуальном предпринимателе (Обязательный). Указываются сведения об уполномоченном физическом лице (в том числе, когда в доверенности в качестве уполномоченного представителя указывается юридическое лицо) или индивидуальном предпринимателе. Расхождение с форматом: поддерживается только один уполномоченный представитель.

PowerOfAttorneyFNSDOVEL503Data.IndividualInfo2 — класс

Сведения о физическом лице, в том числе индивидуальном предпринимателе (СвФизЛицТип) — Таблица 4.21.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualInfo2
```

Свойства

Имя	Описание
`Inn`	ИНН физического лица.

Имя	Описание
RecordNumberOfFederalPopulationRegister	Номер записи федерального регистра сведений о населении
Ogrnip	ОГРНИП (Обязательный с условием). Обязателен, если лицо является индивидуальным предпринимателем.
Snils	СНИЛС (Обязательный).
Citizenship	Гражданство. Принимает значение в соответствии с Общероссийским классификатором стран мира.
BirthDate	Дата рождения.
Fio	Фамилия, имя, отчество физического лица (Обязательный).
IdentityCard	Сведения о документе, удостоверяющем личность физического лица (Обязательный).

PowerOfAttorneyFNSDOVEL503Data.IndividualInfo3 — класс

Сведения по физическому лицу (СведФЛТип) — Таблица 4.22.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class IndividualInfo3
```

Свойства

Имя	Описание
Inn	ИНН физического лица.
RecordNumberOfFederalPopulationRegister	Номер записи федерального регистра сведений о населении

Имя	Описание
`Ogrnip`	ОГРНИП (Обязательный с условием). Обязателен, если лицо является индивидуальным предпринимателем.
`Snils`	СНИЛС (Обязательный).
`Citizenship`	Гражданство. Принимает значение в соответствии с Общероссийским классификатором стран мира.
`BirthDate`	Дата рождения.
`IdentityCard`	Сведения о документе, удостоверяющем личность физического лица (Обязательный).

PowerOfAttorneyFNSDOVEL503Data.AddressInfo — класс

Адрес доверителя в доверенности (АдрДовТип) — Таблица 4.23

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class AddressInfo
```

Свойства

Имя	Описание
`AddressKyr`	Адрес в русской транскрипции (Обязательный с условием). Обязательный, если указывается адрес российской организации.
`AddressLat`	Адрес в латинской транскрипции (Обязательный с условием). Обязательный, если не указан `Address`.

PowerOfAttorneyFNSDOVEL503Data.IdentityCardInfo — класс

Сведения о документе, удостоверяющем личность физического лица

(УдЛичнФЛТип) — Таблица 4.24.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class IdentityCardInfo
```

Свойства

Имя	Описание
DocumentKindCode	Код вида документа (Обязательный).
DocumentSerialNumber	Серия и номер документа (Обязательный).
IssueDate	Дата выдачи документа (Обязательный).
Issuer	Наименование органа, выдавшего документ (Обязательный с условием). Обязательный, если вид документа - Паспорт гражданина Российской Федерации.
IssuerCode	Код подразделения органа, выдавшего документ.

PowerOfAttorneyFNSDOVEL503Data.FIO — класс

Фамилия, имя, отчество (ФИОТип) - 4.25

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class FIO
```

Свойства

Имя	Описание
LastName	Фамилия (Обязательный).
FirstName	Имя (Обязательный).
MiddleName	Отчество.

PowerOfAttorneyFNSDOVEL503Data.RetrustType — перечисление

Признак возможности оформления передоверия.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum RetrustType
```

Свойства

Имя	Описание
Followed = 1	Передоверие возможно
None = 2	Без права передоверия

PowerOfAttorneyFNSDOVELData — класс

Сведения о доверенности ФНС в формате DOVEL.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyFNSDOVELData : PowerOfAttorneyFNSData
```

Методы

Имя	Описание
`PowerOfAttorneyFNSDOVELData(Guid, string, string, string, PowerOfAttorneyDocumentData)`	Инициализация данных доверенности *Параметры:* • `powerOfAttorneyId` — Идентификатор доверенности • `recipientID` — Идентификатор получателя, которому направляется файл обмена • `finalRecipientID` — Идентификатор конечного получателя, для которого предназначена информация из данного файла обмена • `senderID` — Идентификатор отправителя информации • `document` — Состав и структура документа

Классы

Имя	Описание
`PowerOfAttorneyDocumentData`	Доверенность (Довер)
`PowerOfAttorneyInfo`	Сведения доверенности (СвДовТип)
`PrincipalInfo`	Сведения о доверителе (СвДоверит)
`RussianEntityPrincipalInfo`	Сведения о российской организации (НПЮЛ)
`PrincipalWithoutPowerOfAttorneyInfo`	Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов)
`IndividualPrincipalInfo`	Сведения о доверителе — физическом лице (ФЛДоверТип)
`ForeignLegalEntityPrincipalInfo`	Сведения о доверителе — иностранном юридическом лице (ИнОргДовер)

Имя	Описание
PowerOfAttorneyFNSDOVELData.HeadOfSeparateDivisionInfo_CL.adoc	Сведения о руководителе обособленного подразделения (СвРукОП)
IndividualPrincipalInfo	Сведения о доверителе — физическом лице (ФЛДоверТип)
RepresentativeAndAuthorityInfo	Сведения о доверителе — физическом лице (ФЛДоверТип)
RepresentativeInfo	Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип)
IndividualInfo	Сведения о физическом лице (СведФЛТип)
OrganizationInfo	Сведения об организации (СвОрг)
AddressInfo	Сведения об адресе (АдрТип)
IdentityDocumentInfo	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип)
FIO	Фамилия, имя, отчество (при наличии) (ФИОТип)

PowerOfAttorneyFNSDOVELData.PowerOfAttorneyDocument — класс

Доверенность (Довер)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyDocumentData
```

Методы

Имя	Описание
`PowerOfAttorneyDocumentData(PowerOfAttorneyInfo, PrincipalInfo, RepresentativeInfo, List<RepresentativePowerInfo>, NotaryCertificateInfo)`	Заполняет Доверенность (Довер) .Параметры: * `powerOfAttorneyInfo` — Сведения доверенности * `principalInfo` — Сведения о доверителе * `representativeInfo` — Сведения об уполномоченном представителе * `representativesPowers` — Сведения о полномочиях представителя * `notaryCertificateInfo` — Сведения о нотариальном удостоверении

PowerOfAttorneyFNSDOVELData.PowerOfAttorneyInfo — класс

Сведения доверенности (СвДовТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PowerOfAttorneyInfo
```

Методы

Имя	Описание
`PowerOfAttorneyInfo(Guid, DateTime, DateTime, PowerOfAttorneyRetrustType, JointRepresentationType, IrrevocablePowerOfAttorneyInfo, string, string, DateTime?, PowerOfAttorneyLossOfAuthorityType?)`	Заполняет Сведения доверенности (СвДовТип) .Параметры: * `powerOfAttorneyNumber` — Единый регистрационный номер доверенности * `powerOfAttorneyStartDate` — Дата совершения (выдачи) доверенности * `powerOfAttorneyEndDate` — Дата окончания действия доверенности * `retrustType` — Признак возможности оформления передоверия * `jointRepresentation` — Признак совместных полномочий * `irrevocablePowerOfAttorneyInfo` — Сведения о безотзывной доверенности * `organizationName` — Название организации-владельца информационной системы * `powerOfAttorneyInternalNumber` — Внутренний регистрационный номер доверенности * `powerOfAttorneyInternalRegistrationDate` — Дата внутренней регистрации доверенности * `lossOfAuthorityType` — Признак утраты полномочий при передоверии

PowerOfAttorneyFNSDOVELData.PrincipalInfo — класс

Сведения о доверителе (СвДоверит)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PrincipalInfo
```

Методы

Имя	Описание
`PrincipalInfo(RussianLegalEntityPrincipalInfo, FIO)`	Заполняет Сведения о доверителе (СвДоверит) .Параметры: * `russianLegalEntityPrincipal` — Сведения о доверителе — российском юридическом лице * `signer` — Сведения о физическом лице, подписывающем доверенность от имени доверителя (российской организации / иностранной организации) без доверенности или сведения о физическом лице, подписывающем доверенность от своего имени/законном представителе физического лица
`PrincipalInfo(ForeignLegalEntityPrincipalInfo, FIO)`	Заполняет Сведения о доверителе (СвДоверит) .Параметры: * `foreignLegalEntityPrincipal` — Сведения о доверителе — иностранном юридическом лице * `signer` — Сведения о физическом лице, подписывающем доверенность от имени доверителя (российской организации / иностранной организации) без доверенности или сведения о физическом лице, подписывающем доверенность от своего имени/законном представителе физического лица

Имя	Описание
`PrincipalInfo(SoleProprietorInfo0, FIO)`	Заполняет Сведения о доверителе (СвДоверит) .Параметры: * `soleProprietor` — Сведения о доверителе — индивидуальном предпринимателе * `signer` — Сведения о физическом лице, подписывающем доверенность от имени доверителя (российской организации / иностранной организации) без доверенности или сведения о физическом лице, подписывающем доверенность от своего имени/законном представителе физического лица</param>
`PrincipalInfo(IndividualPrincipalInfo, FIO)`	Заполняет Сведения о доверителе (СвДоверит) .Параметры: * `individualPrincipal` — Сведения о доверителе — физическом лице * `signer` — Сведения о физическом лице, подписывающем доверенность от имени доверителя (российской организации / иностранной организации) без доверенности или сведения о физическом лице, подписывающем доверенность от своего имени/законном представителе физического лица

PowerOfAttorneyFNSDOVELData.RussianEntityPrincipalInfo — класс

Сведения о российской организации (НПЮЛ)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RussianEntityPrincipalInfo
```

Свойства

Имя	Описание
name	Название
inn	ИНН
kpp	КПП
ogrn	ОГРН
address	Сведения об адресе (АдрТип)
principalWithoutPowerOfAttorney	Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов)

Методы

Имя	Описание
RussianEntityPrincipalInfo(string, string, string, string, AddressInfo, PrincipalWithoutPowerOfAttorneyInfo)	Заполняет Сведения о российской организации (НПЮЛ) *Параметры:* • Наименование организации • ИНН организации • КПП организации • ОГРН • Сведения о лице, действующем от имени юридического лица без доверенности

PowerOfAttorneyFNSDOVELData.PrincipalWithoutPowerOfAttorneyInfo — класс

Сведения о лице, действующем от имени юридического лица без доверенности (ЛицоБезДов)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrincipalWithoutPowerOfAttorneyInfo
```

Методы

Имя	Описание
PrincipalWithoutPowerOfAttorneyInfo(IndividualInfo0, LegalEntityInfo)	Заполняет Сведения о лице, действующем от имени юридического лица без доверенности .Параметры: * individualInfo — Сведения по физическому лицу * legalEntityInfo — Сведения о юридическом лице

PowerOfAttorneyFNSDOVELData.IndividualPrincipalInfo — класс

Сведения о доверителе — физическом лице (ФЛДоверТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class IndividualPrincipalInfo : IndividualDelegatedAuthorityInfo
```

Поля

Имя	Описание
inn	ИНН
Ogrnip	ОГРНИП.
Snils	СНИЛС
Citizenship	Гражданство
birthDate	Дата рождения
identityDocument	Документ, удостоверяющий личность

Методы

Имя	Описание
`IndividualPrincipalInfo(bool, string, IndividualInfo, string, string, FIO, LegalRepresentativeInfo) : base()`	Заполняет Сведения о лице, передавшем полномочия — физическом лице (ФЛПрдПолн) .Параметры: * `hasFullCapacity` — Признак наличия полной гражданской дееспособности доверителя — физического лица * `inn` — ИНН физического лица * `snils` — СНИЛС * `documentOfIncapacity` — Наименование и реквизиты документа, подтверждающего недееспособность доверителя физического лица * `fio` — ФИО физического лица * `individualInfo` — Сведения о физическом лице * `representativeInfo` — Сведения о законном представителе физического лица

PowerOfAttorneyFNSDOVELData.ForeignLegalEntityPrincipalInfo — класс

Сведения о доверителе — иностранном юридическом лице (ИнОргДовер)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class ForeignLegalEntityPrincipalInfo
```

Поля

Имя	Описание
`name`	Название
`inn`	ИНН
`kpp`	КПП

Имя	Описание
`country`	Страна регистрации
`registrationAuthority`	Регистрирующий орган
`registrationNumber`	Регистрационный номер
`taxCode`	Код налогоплательщика в стране регистрации
`address`	Полный адрес иностранной организации в стране регистрации
`headOfSeparateDivision`	Сведения о руководителе обособленного подразделения (СвРукОП)

Методы

Имя	Описание
`ForeignLegalEntityPrincipalInfo(string, string, string, string, string, AddressInfo, HeadOfSeparateDivisionInfo, string, string)`	Заполняет Сведения об иностранной организации (ИО) *Параметры:* • `name` — Наименование иностранной организации • `inn` — ИНН организации • `kpp` — КПП обособленного подразделения • `country` — Страна регистрации(инкорпорации) • `registrationNumber` — Регистрационный номер • `address` — Полный адрес иностранной организации в стране регистрации(инкорпорации) • `headOfSeparateDivision` — Сведения о руководителе обособленного подразделения • `registrationAuthority` — Наименование регистрирующего органа • `taxCode` — Код налогоплательщика в стране регистрации(инкорпорации) или аналог

PowerOfAttorneyFNSDOVELData.HeadOfSeparateDivisionInfo — класс

Сведения о руководителе обособленного подразделения (СвРукОП)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class HeadOfSeparateDivisionInfo
```

Поля

Имя	Описание
`inn`	ИНН
`kpp`	КПП
`gender`	Пол
`birthDate`	Дата рождения
`birthPlace`	Место рождения
`citizenshipType`	Признак наличия гражданства
`citizenship`	Гражданство (для иностранного гражданина)

Методы

Имя	Описание
`HeadOfSeparateDivisionInfo(Gender, DateTime, CitizenshipType, string, string, string)`	Заполняет Сведения о руководителе обособленного подразделения (СвРукОП) *Параметры:* • `gender` — Пол • `birthDate` — Дата рождения • `citizenshipType` — Признак наличия гражданства • `inn` — ИНН физического лица - руководителя обособленного подразделения • `birthPlace` — Место рождения • `citizenship` — Гражданство (для иностранного гражданина)

/dv6/programmer/dv6/BackOffice-ObjectModel-Services-Entities/Entities/PowerOfAttorneyFNSDOVELData.IndividualPrincipalInfo_CL/[PowerOfAttorneyFNSDOVELData.IndividualPrincipalInfo — класс]

PowerOfAttorneyFNSDOVELData.RepresentativeAndAuthorityInfo — класс

Сведения об уполномоченном представителе и его полномочиях (СвУпПред)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RepresentativeAndAuthorityInfo
```

Поля

Имя	Описание
`authorityCode`	Название
`representative`	ИНН

Методы

Имя	Описание
`RepresentativeAndAuthorityInfo(RepresentativeInfo, List<PowersCode>)`	Заполняет Сведения о российской организации (НПЮЛ) *Параметры:* • `representative` — Сведения об уполномоченном представителе • `authorityCode` — Признак (код) области полномочий уполномоченного представителя

PowerOfAttorneyFNSDOVELData.RepresentativeInfo — класс

Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RepresentativeInfo
```

Методы

Имя	Описание
`RepresentativeInfo(OrganizationInfo)`	Заполняет Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип) .Параметры: * `organizationInfo` — Сведения об организации
`RepresentativeInfo(SoleProprietorInfo1)`	Заполняет Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип) .Параметры: * `soleProprietorInfo` — Сведения об индивидуальном предпринимателе
`RepresentativeInfo(IndividualInfo2)`	Заполняет Сведения об уполномоченном представителе (уполномоченных представителях) (СвУпПредТип) .Параметры: * `individualInfo` — Сведения о физическом лице

PowerOfAttorneyFNSDOVELData.IndividualInfo — класс

Сведения о физическом лице (СведФЛТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualInfo
```

Методы

Имя	Описание
`IndividualInfo(DateTime, CitizenshipType, string, string, Gender?, string, AddressInfo, IdentityCardOfIndividual)`	Заполняет Сведения о физическом лице (СведФЛТип) .Параметры: * `birthDate` — Дата рождения * `citizenshipType` — Признак наличия гражданства * `birthPlace` — Место рождения * `contactPhone` — Контактный телефон * `gender` — Пол физического лица * `citizenship` — Гражданство (для иностранного гражданина) * `residenceAddress` — Адрес места жительства * `identityCard` — Сведения о документе, удостоверяющем личность физического лиц

PowerOfAttorneyFNSDOVELData.IndividualPrincipalInfo1 — класс

Сведения по физическому лицу (СвФЛ)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualPrincipalInfo1
```

Свойства

Имя	Описание
`inn`	ИНН
`kpp`	КПП
`snils`	СНИЛС
`citizenship`	Гражданство
`birthDate`	Дата рождения

Имя	Описание
position	Должность

Методы

Имя	Описание
IndividualPrincipalInfo1(string, string, string, DateTime?, string)	Заполняет Сведения по физическому лицу (СвФЛ) .Параметры: * inn — ИНН физического лица * snils — СНИЛС * citizenship — Гражданство физического лица * birthDate — Дата рождения физического лица * position — Должность

PowerOfAttorneyFNSDOVELData.OrganizationInfo — класс

Сведения об организации (СвОрг)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class OrganizationInfo
```

Методы

Имя	Описание
OrganizationInfo(LegalEntityInfo, List<IndividualInfo2>)	Заполняет Сведения об организации (СвОрг) .Параметры: * legalEntityInfo — Сведения о юридическом лице * individualInfo — Сведения по физическому лицу

PowerOfAttorneyFNSDOVELData.AddressInfo — класс

Сведения об адресе (АдрТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class AddressInfo
```

Методы

Имя	Описание
AddressInfo(string) : this()	Заполняет Сведения об адресе (АдрТип) *Параметры:* • address — Адрес в Российской Федерации.
AddressInfo(string, addressType)	Заполняет Адрес доверителя в доверенности (АдрДовТип) *Параметры:* • address — Адрес в Российской Федерации.

PowerOfAttorneyFNSDOVELData.IdentityDocumentInfo — класс

Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип)

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class IdentityDocumentInfo
```

Поля

Имя	Описание
documentKindCode	Код вида документа

Имя	Описание
`documentSeriesAndNumber`	Серия и номер документа
`issueDate`	Дата выдачи документа
`issuer`	Наименование органа, выдавшего документ
`issuerCode`	Код подразделения органа, выдавшего документ

Методы

Имя	Описание
`IdentityDocumentInfo(DocumentKindCode, string, DateTime, string, string)`	Заполняет Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип) .Параметры: * `documentKindCode` — Код вида документа * `documentSeriesAndNumber` — Серия и номер документа * `issueDate` — Дата выдачи документа * `issuer` — Наименование органа, выдавшего документ * `issuerCode` — Код подразделения органа, выдавшего документ

PowerOfAttorneyFNSDOVELData.FIO — класс

Фамилия, имя, отчество (при наличии) (ФИОТип)

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class FIO
```

Методы

Имя	Описание
`FIO(string, string, string)`	Заполняет Фамилия, имя, отчество (при наличии) (ФИОТип) .Параметры: * `lastName` — Фамилия * `firstName` — Имя * `middleName` — Отчество

PowerOfAttorneyMachineReadableInfo — класс

Содержит информацию о МЧД.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PowerOfAttorneyMachineReadableInfo
```

Конструкторы

Имя	Описание
`PowerOfAttorneyMachineReadableInfo(Guid, DateTime, DateTime?, PowerOfAttorneyRetrustType, string, string, string, string, Encoding, List<string>)`	Создаёт новый экземпляр класса

Свойства

Имя	Описание
`PowerOfAttorneyNumber`	Номер доверенности
`PowerOfAttorneyStartDate`	Дата совершения доверенности
`PowerOfAttorneyEndDate`	Срок действия доверенности
`RetrustType`	Признак возможности оформления передоверия
`Principal`	Доверитель
`PowerOfAttorneyMachineReadableFileID`	Идентификатор файла МЧД

Имя	Описание
`Encoding PowerOfAttorneyMachineReadableFileEncoding`	Кодировка файла МЧД
`RepresentativesSnils`	СНИЛСы представителей
`PrincipalsSnils`	СНИЛСы доверителей
`Representative`	Строковое представление представителя в формате ФИО (или ФИ) — для физлица, или названия организации — для юридического лица
`Signer`	Строковое представление подписанта в формате ФИО (или ФИ)
`RepresentativePowers`	Полномочия представителя

Методы

Имя	Описание
`PowerOfAttorneyMachineReadableInfo(Guid, DateTime, DateTime?, PowerOfAttorneyRetrustType, string, string, string, string, Encoding, List<string>, List<string>, List<string>)`	Создаёт новый экземпляр класса `PowerOfAttorneyMachineReadableInfo`. *Параметры:* • `powerOfAttorneyNumber` — Номер доверенности • `powerOfAttorneyStartDate` — Дата совершения доверенности • `powerOfAttorneyEndDate` — Срок действия доверенности • `retrustType` — Признак возможности передоверия • `representative` — Представитель • `signer` — Подписант • `principal` — Доверитель • `powerOfAttorneyMachineReadableFileID` — Идентификатор файла доверенности • `powerOfAttorneyMachineReadableFileEncoding` — Кодировка файла МЧД • `representativesSnils` — СНИЛСы представителей • `principalsSnils` — СНИЛСы поручителей • `principalsInn` — ИННы доверителей
`AddRepresentativePowers(string, bool)`	Добавить полномочие. *Параметры:* • `power` — код или текстовое описание полномочия • `isTextPower` — содержит текстовое описание полномочия

PowerOfAttorneyHandlingFlags — перечисление

Флаги обработки доверенностей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum PowerOfAttorneyHandlingFlags
```

Члены

Имя члена	Описание
None = 0	Нет.
SkipEntityMatchCheck = 1	Не проверять соответствие сущностей доверенности со справочниками.
SupportDistributedRegistryFederalTaxService = 2	Включить формирование доверенности для ЦПРР.
DisableAutoAssignmentReadRightsToRepresentative = 4	Отключить автоматическую выдачу права на СКД представителю-физлицу.
DisableSignerSnilsValidation = 8	Отключить проверку соответствия СНИЛСа подписанта данным из МЧД, заявления на отзыв МЧД.
EnableLegacyPoaFormat = 16	Отключить генерацию ошибки про формировании устаревшего формата доверенности

PowerOfAttorneyVerification — класс

Содержит результат проверки действительности доверенности.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PowerOfAttorneyVerification
```

Свойства

Имя	Описание
PowerOfAttorneyVerificationResult	Результат проверки .Доверенность действительна при выполнении условий: - СНИЛС в МЧД соответствует СНИЛС в подписи МЧД - Доверенность подписана действующим сертификатом (по дате) - Дата подписания соответствует дате формирования подписи - Дата выдачи доверенности попадает в срок действия сертификата подписи - Если доверенность является дочерней — все полномочия (ссылающиеся на Справочник полномочий) есть в родительской
Messages	Сообщения об ошибках проверки действительности доверенности

ImportESNSIResults — класс

Результат импорта полномочий из ЕСНСИ

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ImportESNSIResults
```

Свойства

Имя	Описание
FileIsEmpty	Пустой или некорректный файл
NothingLoaded	Ничего не загружено

Имя	Описание
NotFoundGroups	Список идентификаторов ЕСНСИ групп, которые не были найдены в справочнике
UpdatedGroups	Группы с обновлёнными кодами полномочий
AlreadyExistsGroups	Список групп с не изменившимися кодами полномочий
DeactivatedCodes	Деактивированные коды полномочий
GroupsWithDeactivatedCodes	Группы с деактивированными кодами полномочий

IPowersService — интерфейс

Сервис справочника полномочий

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IPowersService
```

Методы

Имя	Описание
FindPowerOfAttorneyCardsWithPowerCodes(IEnumerable<PowersCode>)	Выполняет поиск системных карточек доверенностей, использующих указанные коды полномочий. *Параметры:* • codes — коды полномочий Возвращаемое значение: FindPowerOfAttorneyCardsResults — системные карточки доверенностей, использующие указанные коды полномочий.

Имя	Описание
`ImportESNSIPowers(Stream)`	Импортирует полномочия из ЕСНСИ *Параметры:* • `stream` — поток данных. Возвращаемое значение: `ImportESNSIResults`
`ImportESNSIPowers(string)`	Импортирует полномочия из ЕСНСИ *Параметры:* • `filePath` — путь к файлу. Возвращаемое значение: `ImportESNSIResults`

FindPowerOfAttorneyCardsResults — класс

Содержит результаты поиска использования кодов полномочий в СКД

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class FindPowerOfAttorneyCardsResults
```

Методы

Имя	Описание
`IsPowerCodeUsed(Guid)`	`Возвращает признак использования кода полномочия` *Параметры:* • `powerCodeId` — идентификатор кода полномочий. Возвращаемое значение: `bool`

Имя	Описание
AnyPowerCodeUsed()	Возвращает признак использования любого кода полномочия из списка проверенных Возвращаемое значение: bool
GetPowerCodeUsings(Guid)	Возвращает информацию по СКД, в которых используется код полномочий *Параметры:* • powerCodeId — идентификатор кода полномочий. Возвращаемое значение: IReadOnlyList

FindPowerOfAttorneyCardsResultsItem — класс

Результат поиска использования кода полномочий в СКД.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class FindPowerOfAttorneyCardsResultsItem
```

Свойства

Имя	Описание
InstanceID	Идентификатор системной карточки доверенности
PowerOfAttorneyNumber	Номер доверенности
PowerOfAttorneyStartDate	Дата совершения доверенности

StartBusinessProcessErrorInfo — класс

Представляет содержимое ошибки запуска бизнес-процесса.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class StartBusinessProcessErrorInfo
```

Конструкторы

Имя	Описание
`StartBusinessProcessErrorInfo(Card, KindsCardProcess, BusinessProcessErrorType)`	Инициализирует экземпляр класса `StartBusinessProcessErrorInfo`.

Свойства

Имя	Описание
`ErrorMessage`	Возвращает текст ошибки.
`ErrorType`	Возвращает тип ошибки.
`InitialCard`	Возвращает тип карточки зарегистрировавшей ошибку.
`IsCriticalError`	Возвращает true, если ошибка является критической.
`ProcessSettings`	Возвращает параметры бизнес-процесса вызвавшего ошибку.

TaskCopyResultsOptions — класс

Предоставляет параметры переноса отчёта об исполнении подчиненного задания в родительское.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class TaskCopyResultsOptions
```

Конструкторы

Имя	Описание
`TaskCopyResultsOptions()`	Инициализирует новый экземпляр класса `TaskCopyResultsOptions`.

Свойства

Имя	Описание
`CopyReferences`	Задаёт или возвращает признак необходимости копировать ссылки.
`CopyReport`	Задаёт или возвращает признак необходимости копировать отчёт.
`References`	Возвращает список ссылок для переноса.

TaskStopExecutionInfo — класс

Представляет содержимое ошибки остановки исполнения задания.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
[Serializable]
public class TaskStopExecutionInfo
```

Конструкторы

Имя	Описание
`TaskStopExecutionInfo()`	Инициализирует новый экземпляр класса `TaskStopExecutionInfo`.
`TaskStopExecutionInfo(ErrorInfo, IEnumerable<LockedTaskInfo>)`	Инициализирует новый экземпляр класса `TaskStopExecutionInfo` с указанными параметрами.

Свойства

Имя	Описание
`CompletedTasks`	Задаёт или возвращает информацию по завершённым заданиям.
`ErrorInfo`	Задаёт или возвращает содержимое ошибки остановки завершения задания.
`LockedTasks`	Задаёт или возвращает информацию по заблокированным заданиям.

ILongProcessManager — интерфейс

Добавляется в производный класс возможности менеджера длительных процессов.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services.Entities`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface ILongProcessManager : IMessageLogger
```

Свойства

Имя	Описание
`CancellationPending`	Возвращает значение, показывающее, запросило ли приложение отмену фоновой операции.
`PausePending`	Возвращает значение, показывающее, запросило ли приложение приостановку фоновой операции.

Методы

Имя	Описание
`Cancel`	Завершает фоновую операцию.

Заметки

Менеджер длительных процессов позволяет выполнить принудительное

завершение процесса, поддерживающего работу с менеджером.

BusinessProcessErrorType — перечисление

Определяет типы ошибок формируемых при проверке бизнес-процесса.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum BusinessProcessErrorType
```

Члены

Имя члена	Описание
None	Нет ошибки.
OperationNotSet	В настройках бизнес-процесса отсутствует операция.
OperationNotAllowed	Операция не разрешена к выполнению.
TooMuchInstances	Превышено разрешённое число экземпляров бизнес-процесса.
TemplateNotExists	Отсутствует карточка бизнес-процесса, указанная в качестве шаблона.

Заметки

Ошибка может быть возвращена в результате проверки возможности запуска бизнес-процесса, выполняемая методом IBaseCardService.CanStartBusinessProcess

DocumentPropertyDirection — перечисление

Определяет направление синхронизации свойств документа и полей карточки.

Это перечисление имеет атрибут FlagsAttribute, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
[Flags]
public enum DocumentPropertyDirection
```

Члены

Имя члена	Описание
None	Не задано. Значение 0x00.
Forward	Из карточки. Значение 0x01.
Backward	В карточку. Значение 0x02.
Both	В обе стороны. Значение 0x03.

GridViewFieldCollectionType — перечисление

Определяет тип отображаемого поля справочника сотрудников или контрагентов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum GridViewFieldCollectionType
```

Члены

Имя члена	Описание
UnitDepartment	Отображаемое поле подразделения.
UnitEmployee	Отображаемое поле сотрудника.
GroupElement	Отображаемое поле группы.

PerformerType — перечисление

Определяет тип исполнителя задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum PerformerType
```

Члены

Имя члена	Описание
None	Исполнитель не задан.
Employee	Сотрудник.
Unit	Подразделение.
Group	Группа сотрудников.
Role	Роль.
SearchWord	Поисковое слово.

TaskTreeNodeType — перечисление

Определяет тип узла в дереве заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum TaskTreeNodeType
```

Члены

Имя члена	Описание
Task	Задание.
Delegate	Было проведено делегирование.
TaskGroup	Группа заданий.

CheckSignatureResult — класс

Представляет результат проверки подписи.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class CheckSignatureResult
```

Конструкторы

Имя	Описание `
CheckSignatureResult(string, bool, bool, DateTime?, DateTime?, DateTime?)	Инициализирует новый экземпляр класса CheckSignatureResult.

CheckSignatureResult — конструктор (string, bool, bool, DateTime?, DateTime?, DateTime?)

Инициализирует новый экземпляр класса CheckSignatureResult.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services.Entities
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
internal CheckSignatureResult(string result, bool isExpiredCertificate, bool
isRevokedCertificate, DateTime? signatureDate, DateTime? clrEntryDate, DateTime?
expireDate)
```

Параметры

result

Тип: System.String

Адрес электронной почты получателя сообщения

isExpiredCertificate

Тип: System.Boolean

Проверка срока действия сертификата подписи: true -— срок действия истёк, false — сертификат действителен.

isRevokedCertificate

Тип: System.Boolean

Проверка подлинности сертификата: true -— сертификат отозван и больше

не действителен, `false` — сертификат действителен.

signatureDate

Тип: `System.DateTime`

Дата создания подписи.

clrEntryDate

Тип: `System.DateTime`

Дата и время записи об отзыве сертификата в CRL. Если подпись была сделана до этой даты, она действительна, если после — то нет.

expireDate

Тип: `System.DateTime`

Срок действия подписи.

BackOfficeServiceFactory — класс

Фабрика сервисов для библиотеки BackOffice.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class BackOfficeServiceFactory : ServiceFactory
```

Конструкторы

Имя	Описание
`BackOfficeServiceFactory()`	Инициализирует новый экземпляр класса BackOfficeServiceFactory.

Методы

Имя	Описание
`GetService(Type)`	Возвращает сервис указанного типа.

Интерфейсы

Имя	Описание
SignatureService	Реализует ISignatureService.

BackOfficeServiceFactory — конструктор

Инициализирует новый экземпляр класса BackOfficeServiceFactory.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public BackOfficeServiceFactory()
```

Заметки

BackOfficeServiceFactory предоставляет сервисы библиотеки *Базовые объекты*: документы, задания, справочник категорий и т.д

ServiceError — класс

Содержит статические методы формирования исключений, возникающих при работе сервисов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public static class ServiceError
```

Методы

Имя	Описание
ArgumentNull(String)	Возвращает исключение типа System.ArgumentNullException#.
ArgumentOutOfRange(String)	Возвращает исключение типа System.ArgumentOutOfRangeException.

Имя	Описание
`FileNotFound(String)`	Возвращает исключение типа `System.IO.FileNotFoundException`.
`InvalidOperation(String)`	Возвращает исключение типа `System.InvalidOperationException`.
`InvalidOperation(String, Object[])`	Возвращает исключение типа `System.InvalidOperationException`. Текст ошибки формируется согласно переданному шаблону.
`ObjectLockedException(String)`	Возвращает исключение типа `System.Exception`.
`OperationAccessDenied(String)`	Возвращает исключение типа `System.MethodAccessException`.
`PathTooLong(String)`	Возвращает исключение типа `System.IO.PathTooLongException`.
`ServiceNotAvailable(String)`	Возвращает исключение типа `DocsVision.Platform.ObjectModel.ServiceNotAvailableException`.
`SettingsForKindAreNull`	Возвращает исключение типа `System.Exception`.
`TaskHierarchySearchException(String)`	Возвращает исключение типа `DocsVision.BackOffice.ObjectModel.Services.Entities.TaskHierarchySearchException`.
`TaskInactivePerformersException(String)`	Возвращает исключение типа `DocsVision.BackOffice.ObjectModel.Services.Entities.TaskInactivePerformersException`.
`WrongStatePassed`	Возвращает исключение типа `System.Exception`.

IAccessCheckingService — интерфейс

Предоставляет методы получения списка ролей и доступных операций для сотрудника, а также методы сброса кэша ролевой модели.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
[ResDescription("IAccessCheckingService"), ResName("IAccessCheckingService")]
public interface IAccessCheckingService
```

Свойства

Имя	Описание
EditMode	Задаёт или возвращает признак того, что сервис работает в режиме записи/чтения.

Методы

Имя	Описание
GetUserOperations(BaseCard, StaffEmployee)	Возвращает список всех операций, доступных указанному сотруднику для карточки.
GetUserRoles(BaseCard)	Получает список ролей карточки для текущего сотрудника.
GetUserRoles(BaseCard, Guid)	Получает список ролей карточки для указанного сотрудника.
GetUserRoles(BaseCard, StaffEmployee)	Получает список ролей карточки для указанного сотрудника.
IsOperationAllowed(BaseCard, BuiltInOperation)	Возвращает признак разрешения выполнения встроенной операции для заданной карточки.
IsOperationAllowed(BaseCard, StatesOperation)	Возвращает признак разрешения выполнения операции для заданной карточки.
ResetRolesCache	Сбрасывает локальный кэш сервиса.
ResetRolesCache(BaseCard)	Удаляет из кэша информацию полученную для указанной карточки.

Имя	Описание
ResetRolesCache(Guid)	Удаляет из кэша информацию полученную для карточки с заданным идентификатором.

Заметки

Метод ResetRolesCache сбрасывает только локальный кэш ролевой модели. Поэтому его желательно вызывать вместе с IServerExtensionProxyService.ResetRolesCache, который сбрасывает кэш ролевой модели на сервере.

Примеры

В следующем примере запрашивается список ролей для документа с идентификатором 00000000-0000-0000-0000-000000000000

```
①

IAccessCheckingService accessCheckService = objectContext.GetService
<IAccessCheckingService>(); ②

IStaffService staffService = objectContext.GetService<IStaffService>(); ③

BaseCard card = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ④

IEnumerable<RoleModelRole> roles = accessCheckService.GetUserRoles(card); ⑤

foreach (var item in roles) ⑥
{
  Console.WriteLine(item.Name);
}
```

① Инициализация контекста объектов.

② Получение сервиса проверки прав доступа.

③ Получение сервиса для работы со справочником сотрудников.

④ Получение документа с идентификатором 00000000-0000-0000-0000-000000000000.

⑤ Получение списка ролей для карточки.

⑥ Выведение в консоль названий ролей.

IAccessCheckingService.EditMode — свойство

Задаёт или возвращает признак того, что сервис работает в режиме записи/чтения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool EditMode { get; set; }
```

Значение свойства

Тип: System.Boolean

true — разрешено изменение данных

Заметки

Свойство зарезервировано и в текущей версии платформы не используется.

IAccessCheckingService.GetUserOperations — метод (BaseCard, StaffEmployee)

Возвращает список всех операций, доступных указанному сотруднику для карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<StatesOperation> GetUserOperations(BaseCard card, StaffEmployee employee)
```

Параметры

card

Тип: BaseCard

Карточка, для которой запрашиваются доступные операции

employee

Тип: StaffEmployee

Сотрудник, проверяемый на права доступа к операциям карточки

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<StatesOperation>

Список доступных операций типа StatesOperation

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр card или employee.

IAccessCheckingService.GetUserRoles — метод (BaseCard)

Получает список ролей карточки для текущего сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<RoleModelRole> GetUserRoles(BaseCard card)
```

Параметры

card

Тип: BaseCard

Карточка библиотеки *Базовые объекты*

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<RoleModelRole>

Коллекция объектов типа RoleModelRole

Исключения

Исключение	Условие
System.ArgumentNullException	Возникает при передаче в метод NULL.

Заметки

В первую очередь производиться поиск в локальном кэше. В случае возникновения ошибки, метод вернет пустую коллекцию.

IAccessCheckingService.GetUserRoles — метод (BaseCard, Guid)

Получает список ролей карточки для указанного сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<RoleModelRole> GetUserRoles(BaseCard card, Guid employeeId)
```

Параметры

card

Тип: BaseCard

Карточка библиотеки *Базовые объекты*

employeeId

Тип: System.Guid

Идентификатор сотрудника

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<RoleModelRole>

Коллекция объектов типа RoleModelRole

Заметки

В первую очередь производиться поиск в локальном кэше. В случае возникновения ошибки, метод вернет пустую коллекцию.

IAccessCheckingService.GetUserRoles — метод (BaseCard, StaffEmployee)

Получает список ролей карточки для указанного сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<RoleModelRole> GetUserRoles(BaseCard card, StaffEmployee employee)
```

Параметры

card

Тип: BaseCard

Карточка библиотеки *Базовые объекты*

employee

Тип: StaffEmployee

Сотрудник подразделения

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<RoleModelRole>

Коллекция объектов типа RoleModelRole

Заметки

В первую очередь производиться поиск в локальном кэше. В случае возникновения ошибки, метод вернет пустую коллекцию.

IAccessCheckingService.IsOperationAllowed — метод (BaseCard, BuiltInOperation)

Возвращает признак разрешения выполнения встроенной операции для заданной карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool IsOperationAllowed(BaseCard card, BuiltInOperation operation)
```

Параметры

card

Тип: BaseCard

Карточка из библиотеки *Базовые объекты*

operation

Тип: BackOffice-ObjectModel-BuiltIn:BuiltInOperation_CL.adoc[BuiltInOperation]

Встроенная операция

Возвращаемое значение

Тип: System.Boolean

`true` — операция разрешена, иначе — `false`

Заметки

Если операция не запрещена, то она отмечается разрешённой.

Примеры

Ниже приведён пример определения наличия прав у роли текущего сотрудника, достаточных для редактирования основного файла некого документа

```
①

IAccessCheckingService accessCheckingService = objectContext.GetService
<IAccessCheckingService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

bool allowed = accessCheckingService.IsOperationAllowed(document, Document
.MainFileEditOperation); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ `true` — операция редактирования разрешена.

IAccessCheckingService.IsOperationAllowed — метод (BaseCard, StatesOperation)

Возвращает признак разрешения выполнения операции для заданной карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool IsOperationAllowed(BaseCard card, StatesOperation operation)
```

Параметры

card

Тип: `BaseCard`

Карточка из библиотеки *Базовые объекты*

operation

Тип: `StatesOperation`

Встроенная операция

Возвращаемое значение

Тип: `System.Boolean`

`true` — операция разрешена, иначе — `false`

Заметки

Если операция не запрещена, то она отмечается разрешённой.

IAccessCheckingService.ResetRolesCache — метод

Сбрасывает локальный кэш сервиса.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void ResetRolesCache()
```

Заметки

Реализованный сервис имеет два кэша, которые пополняются и используются по мере вызова методов GetUserRoles и IsOperationAllowed. В ряде случаев (например, изменение ролевой модели) кэш может быть заполнен неверными данными. Сброс кэша принуждает методы обращаться за данными к серверу.

IAccessCheckingService.ResetRolesCache — метод (BaseCard)

Удаляет из кэша информацию полученную для указанной карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ResetRolesCache(BaseCard card)
```

Параметры

card

Тип: BaseCard

Карточка

IAccessCheckingService.ResetRolesCache — метод (Guid)

Удаляет из кэша информацию полученную для карточки с заданным идентификатором.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ResetRolesCache(Guid cardId)
```

Параметры

cardId

Тип: System.Guid

Идентификатор карточки

ITaskGroupMessagesEventHandlerService — класс

Предоставляет методы для работы с почтовыми уведомлениями.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskGroupMessagesEventHandlerService : MessagesEventHandlerService,
ITaskGroupMessagesEventHandlerService
```

Методы

Имя	Описание
GetNotificationCc(Task, StaffEmployee)	Позволяет вычислить адресатов в CC.
GetNotificationBcc(Task, StaffEmployee)	Позволяет вычислить адресатов в BCC.

GetNotificationBcc(TaskGroup, StaffEmployee)

Позволяет вычислить адресатов в BCC.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
protected virtual string GetNotificationBcc(TaskGroup taskGroup, StaffEmployee employee
```

Параметры

taskGroup

Тип: TaskGroup

Группа заданий

employee

Тип: StaffEmployee

Сотрудник организации

Возвращаемое значение

Тип: null

Пустое значение

GetNotificationCc(TaskGroup, StaffEmployee)

Позволяет вычислить адресатов в CC.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
protected virtual string GetNotificationCc(TaskGroup taskGroup, StaffEmployee employee)
```

Параметры

taskGroup

Тип: TaskGroup

Группа заданий

employee

Тип: StaffEmployee

Сотрудник организации

Возвращаемое значение

Тип: null

Пустое значение

ITaskMessagesEventHandlerService — класс

Содержит методы для работы с почтовыми уведомлениями.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskMessagesEventHandlerService : MessagesEventHandlerService,
ITaskMessagesEventHandlerService
```

Методы

Имя	Описание
GetNotificationCc(Task, StaffEmployee)	Позволяет вычислить адресатов в CC.
GetNotificationBcc(Task, StaffEmployee)	Позволяет вычислить адресатов в BCC.

GetNotificationBcc — метод (Task, StaffEmployee)

Позволяет вычислить адресатов в BCC.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
protected virtual string GetNotificationBcc(Task task, StaffEmployee employee)
```

Параметры

task

Тип: Task

Задание

employee

Тип: StaffEmployee

Сотрудник организации

Возвращаемое значение

Тип: null

Пустое значение

GetNotificationCc — метод (Task, StaffEmployee)

Позволяет вычислить адресатов в CC.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
protected virtual string GetNotificationCc(Task task, StaffEmployee employee)
```

Параметры

task

Тип: Task

Задание

employee

Тип: `StaffEmployee`

Сотрудник организации

Возвращаемое значение

Тип: null

Пустое значение

IBarcodeService — интерфейс

Определяет методы генерации и печати штрих-кодов карточки

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface IBarcodeService
```

Методы

Имя	Описание
`ChangeBarcode(Document, String)`	Задаёт значение штрих-кода карточки документа.
`GenerateBarcode(Document)`	Формирует штрих-код для заданной карточки документа.
`GenerateBarcode(Document, DateTime)`	Формирует штрих-код для заданной карточки документа.

Заметки

Настройки формата осуществляются в справочнике системных настроек, раздел "Штрих-коды".

Примеры

Пример генерации штрих-кода для карточки "Документ" (вид карточки должен поддерживать использование штрих-кода, иначе при назначении вернется

ошибка).

```
IBarcodeService barcodeService = objectContext.GetService<IBarcodeService>(); ①

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-000000000000")); ②

string barcode = barcodeService.GenerateBarcode(document); ③

barcodeService.ChangeBarcode(document, barcode); ④
```

① Инициализация контекста объектов и получение сервиса штрих-кодов.

② Получение некого существующего Документа с идентификатором `00000000-0000-0000-0000-000000000000`.

③ Получаем штрих-код на основе настроек из справочника системных настроек.

④ Присваиваем полученный штрих-код документу.

Заметки

Методы `GetBarcodePrintSettings` и `PrintBarcode` windows-специфичные и недоступны в Linux.

IBarcodeService.ChangeBarcode — метод (Document, String)

Задаёт значение штрих-кода карточки документа.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void ChangeBarcode(Document document, string newBarcode)
```

Параметры

document

Тип: `Document`

Документ

newBarcode

Тип: `System.String`

Штрих-код

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document.
System.MethodAccessException	Ошибка возвращается в случае, если у сотрудника недостаточно прав для изменения штрих-кода.

IBarcodeService.GenerateBarcode — метод (Document)

Формирует штрих-код для заданной карточки документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string GenerateBarcode(Document document)
```

Параметры

document

Тип: Document

Документ

Возвращаемое значение

Тип: System.String

Штрих-код

Заметки

Штрих-код формируется в соответствии с системными настройками. За основу берется текущая дата и время.

IBarcodeService.GenerateBarcode — метод (Document, DateTime)

Формирует штрих-код для заданной карточки документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string GenerateBarcode(Document document, DateTime documentDate)
```

Параметры

document

Тип: Document

Документ

documentDate

Тип: System.DateTime

Дата и время, используемые при формировании штрих-кода

Возвращаемое значение

Тип: System.String

Штрих-код

Заметки

Штрих-код формируется в соответствии с системными настройками. За основу берется дата и время DateTime.

IBaseCardService — интерфейс

Данный сервис определяет общие для всех базовых объектов методы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IBaseCardService
```

Методы

Имя	Описание
`AddBusinessProcess(BaseCard, Guid, KindsCardProcess)`	Добавляет к карточке бизнес-процесс с переданным идентификатором.
`AddSignature(SignatureList, X509Certificate2, String, StatesState)`	Добавляет в переданный список подписей ЭП.
`AddSignaturePart(BaseCardSignature, Guid, Byte[], String, Guid, Guid, BaseCard)`	Инициализирует и добавляет в указанную подпись часть подписи.
`AddSignaturePart(BaseCardSignature, Guid, Byte[], String, Guid, Guid, BaseCard, DateTime?, BaseCardSignaturePartStatus)`	Инициализирует и добавляет в указанную подпись часть подписи, прошедшую проверку валидности.
`CanStartBusinessProcess(BaseCard, KindsCardProcess)`	Проверяет допустимость запуска бизнес-процесса и возвращает объект, содержащий ошибку.
`CanStartBusinessProcesses(BaseCard, IEnumerable<KindsCardProcess>)`	Проверяет допустимость запуска коллекции бизнес-процессов и возвращает список ошибок.
`ComputeSignature(Stream, X509Certificate2)`	Подписывает поток данных.
`CopySignatures(SignatureList, SignatureList)`	Создаёт список подписей на основе имеющегося.
`ExtractCertificate(BaseCardSignaturePart)`	Извлекает объект типа `X509Certificate2` из части подписи.
`FindCardsByUniqueAttributes(Guid, SearchParametersInfo)`	Возвращает список идентификаторов карточек, имеющих указанные уникальные атрибуты.
`GenerateDigest(BaseCard, IXmlExportable, String)`	Генерирует дайджест на основе данных карточки.
`GetActiveBusinessProcessInstances(BaseCard, KindsCardProcess)`	Получает список активных бизнес-процессов Workflow для заданной карточки.
`GetAllBusinessProcessInstances(BaseCard, KindsCardProcess)`	Получает список всех бизнес-процессов Workflow указанного вида для заданной карточки.

Имя	Описание
GetBaseCardSectionRow(BaseCard, CardSection)	Получает строку секции карточки.
GetBaseCardSectionRows(BaseCard, CardSection)	Возвращает коллекцию всех строк секции карточки.
GetBusinessProcessesToStart(BaseCard)	Получает список бизнес-процессов, которые могут быть запущены из карточки.
GetBusinessProcessesToStop(BaseCard)	Получает список активных бизнес-процессов для карточки.
GetBusinessProcessesToView(BaseCard)	Получает список бизнес-процессов у которых установлено разрешение на просмотр из карточки.
GetCardTypeIcon(KindsCardKind)	Возвращает иконку для карточки указанного вида карточки.
GetSignaturePartData(BaseCardSignaturePart)	Извлекает данные из части подписи.
InitializeBusinessProcessVariables(Process, Dictionary<String, Object>)	Выполняет инициализацию переменных бизнес-процесса.
InitializeSystemInfo(BaseCard, KindsCardKind)	Инициализирует системную информацию карточки.
IsSignaturePartOperation(BaseCardSignaturePart)	Возвращает true, если указанной частью подписи подписана операция.
RemoveSignature(SignatureList, BaseCardSignature)	Удалят указанную подпись из списка подписей.
RemoveSignaturePart(BaseCardSignaturePart)	Удаляет указанную часть из подписи.
StartBusinessProcess(BaseCard, KindsCardProcess)	Запускает бизнес-процесс указанного вида
SendMailNotification(IEnumerable<MailNotificationInfo>)	Отправляет оповещение по электронной почте.

Имя	Описание
SignatureInfo GetPartSignatureInfo(BaseCardSignaturePart signaturePart, Stream contentStream, X509Certificate2 certificate)	Представляет информацию о части подписи.
AddExternalPowerOfAttorney(SignatureList, Guid, BaseCardExternalPowerOfAttorneyStatus);	Добавляет стороннюю МЧД
AddExternalPowerOfAttorney(SignatureList, Guid, BaseCardExternalPowerOfAttorneyStatus, string);	Добавляет стороннюю МЧД
StopBusinessProcess(BaseCard, Process)	Останавливает выполнение бизнес-процесса.
UpdateBusinessProcessActiveInstancesProperties(BaseCard)	Сохраняет в карточку значение переменных активных бизнес-процессов карточки.
UpdateBusinessProcessActiveInstancesVariables(BaseCard)	Загружает из карточки значения переменных активных бизнес-процессов карточки.
VerifySignature(BaseCardSignature, IDictionary<BaseCardSignaturePart, Stream>, X509Certificate2)	Проверяет корректность подписи.

События

Имя	Описание
BeforeBusinessProcessStart	Происходит перед непосредственным запуском бизнес-процесса.
BusinessProcessVariableInitialized	Предоставляет метод для обработки события, возникающего после инициализации переменных бизнес-процесса.

Имя	Описание
ResolveProcessVariableBindedElement	Предоставляет метод обработки события, возникающего перед получением названия переменной бизнес-процесса для разметки.

IBaseCardService.AddBusinessProcess — метод (BaseCard, Guid, KindsCardProcess)

Добавляет к карточке бизнес-процесс с переданным идентификатором.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardProcess AddBusinessProcess(BaseCard baseCard, Guid processId, KindsCardProcess
kindProcessSettings)
```

Параметры

baseCard

Тип: BaseCard

Карточка к которой будет добавлен бизнес-процесс

processId

Тип: System.Guid

Идентификатор экземпляра бизнес-процесса

kindProcessSettings

Тип: KindsCardProcess

Параметры экземпляра бизнес-процесса

Возвращаемое значение

Тип: BaseCardProcess

Предоставляет ссылку на экземпляр и настройки бизнес-процесса

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard или processId.

IBaseCardService.AddSignature — метод (SignatureList, X509Certificate2, String, StatesState)

Инициализирует электронную подпись и добавляет её в список подписей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardSignature AddSignature(SignatureList signatureList, X509Certificate2 certificate,
string cardDescription, StatesState cardState)
```

Параметры

signatureList

Тип: SignatureList

Список подписей

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат

cardDescription

Тип: System.String

Примечание к подписи

cardState

Тип: StatesState

Состояние карточки

Возвращаемое значение

Тип: BaseCardSignature

Объект содержащий полную информацию по установленной ЭП

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр signatureList.

Заметки

Если certificate не задан, то будет проставлена простая подпись.

IBaseCardService.AddSignaturePart — метод (BaseCardSignature, Guid, Byte[], String, Guid, Guid, BaseCard)

Инициализирует и добавляет в указанную подпись часть подписи.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardSignature AddSignaturePart(BaseCardSignature signature, Guid typeId, byte[]
signatureData, string description, Guid fileId, Guid fileVersionId, BaseCard partObject)
```

Параметры

signature

Тип: BaseCardSignature

Электронная подпись

typeId

Тип: System.Guid

Идентификатор типа части подписи

signatureData

Тип: System.Byte[]

Содержимое добавляемой части подписи

description

Тип: System.String

Примечание к части подписи

fileId

Тип: `System.Guid`

Идентификатор подписываемого файла

fileVersionId

Тип: `System.Guid`

Идентификатор версии подписываемого файла

partObject

Тип: `BaseCard`

Базовый объект

Возвращаемое значение

Тип: `BaseCardSignature`

Итоговая подпись, включающая добавляемую часть подписи

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `signature` или `signatureData`.

Заметки

Как правило, вызывать данный метод нет необходимости — соответствующий вызов делает сервис. Например, метод `IDocumentService.ComputeAttachmentSignature` вызовет `AddSignaturePart` с необходимыми параметрами.

IBaseCardService.AddSignaturePart — метод (BaseCardSignature, Guid, Byte[], String, Guid, Guid, BaseCard, DateTime?, BaseCardSignaturePartStatus)

Инициализирует и добавляет в указанную подпись часть подписи, прошедшую проверку срока действия.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
BaseCardSignature AddSignaturePart(BaseCardSignature signature, Guid typeId, byte[]
signatureData, string description, Guid fileId, Guid fileVersionId, BaseCard partObject,
DateTime? expireDate, BaseCardSignaturePartStatus status)
```

Параметры

signature

Тип: BaseCardSignature

Электронная подпись

typeId

Тип: System.Guid

Идентификатор типа части подписи

signatureData

Тип: System.Byte[]

Содержимое добавляемой части подписи

description

Тип: System.String

Примечание к части подписи

fileId

Тип: System.Guid

Идентификатор подписываемого файла

fileVersionId

Тип: System.Guid

Идентификатор версии подписываемого файла

partObject

Тип: BaseCard

Базовый объект

expireDate

Тип: `System.DateTime`

Дата и время окончания срока действия сертификата части подписи

status

Тип: `BaseCardSignaturePartStatus`

Результат проверки подписи. Может принимать значения из: `Unknown = 0`, `Valid = 1`, `Invalid = 2`, `Expired = 3`, `Error = 4`, `Expiring = 5`

Возвращаемое значение

Тип: `BaseCardSignature`

Итоговая подпись, включающая добавляемую часть подписи после прохождения проверки срока действия.

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `signature` или `signatureData`.

Заметки

Как правило, вызывать данный метод нет необходимости — соответствующий вызов делает сервис. Например, метод `IDocumentService.ComputeAttachmentSignature` вызовет `AddSignaturePart` с необходимыми параметрами.

IBaseCardService.CanStartBusinessProcess — метод (BaseCard, KindsCardProcess)

Проверяет допустимость запуска бизнес-процесса и возвращает объект содержащий ошибку.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
StartBusinessProcessErrorInfo CanStartBusinessProcess(BaseCard baseCard, KindsCardProcess processSettings)
```

Параметры

baseCard

Тип: `BaseCard`

Проверяемая карточка

processSettings

Тип: `KindsCardProcess`

Настройки бизнес-процесса

Возвращаемое значение

Тип: `StartBusinessProcessErrorInfo`

Содержимое ошибки

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `baseCard` или `processSettings`.

Заметки

Метод в любом случае вернет объект StartBusinessProcessErrorInfo, но в случае отсутствия ошибки — `StartBusinessProcessErrorInfo.ErrorType` будет равен `BusinessProcessErrorType.None`. Фактически возвращается первая встреченная ошибка, таким образом необходимо произвести исправление и повторить проверку.

IBaseCardService.CanStartBusinessProcesses — метод (BaseCard, IEnumerable<KindsCardProcess>)

Проверяет допустимость запуска коллекции бизнес-процессов и возвращает список ошибок.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
IEnumerable<StartBusinessProcessErrorInfo> CanStartBusinessProcesses(BaseCard baseCard,
```

```
IEnumerable<KindsCardProcess> kindsCardProcesses)
```

Параметры

baseCard

Тип: BaseCard

Проверяемая карточка

kindsCardProcesses

Тип: System.Collections.Generic.IEnumerable<KindsCardProcess>

Коллекция видов бизнес-процессов типа KindsCardProcess

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<StartBusinessProcessErrorInfo>

Коллекция ошибок типа StartBusinessProcessErrorInfo

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard или processSettings.

Заметки

Метод выполняет последовательный вызов метода IBaseCardService.CanStartBusinessProcess для каждого элемента коллекции kindsCardProcesses.

IBaseCardService.ComputeSignature — метод (Stream, X509Certificate2)

Подписывает поток данных.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
byte[] ComputeSignature(Stream contentStream, X509Certificate2 certificate)
```

Параметры

contentStream

Тип: `System.IO.Stream`

Подписываемый объект

certificate

Тип: `System.Security.Cryptography.X509Certificates.X509Certificate2`

Сертификат подписи

Возвращаемое значение

Тип: `System.Byte[]`

Закодированный в сообщение CMS/PKCS 7 поток. Если параметр `certificate` не задан, метод вернет хэш потока.

IBaseCardService.CopySignatures — метод (SignatureList, SignatureList)

Создаёт список подписей на основе имеющегося.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void CopySignatures(SignatureList sourceSignatureList, SignatureList
destinationSignatureList)
```

Параметры

sourceSignatureList

Тип: `SignatureList`

Исходный список подписей

destinationSignatureList

Тип: `SignatureList`

Конечный список подписей

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр sourceSignatureList или destinationSignatureList.

Заметки

При копировании создаются объекты типа BaseCardSignature и BaseCardSignaturePart, которые заполняются данными существующих подписей и частей подписи.

IBaseCardService.ExtractCertificate — метод (BaseCardSignaturePart)

Извлекает объект типа X509Certificate2 из части подписи.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
X509Certificate2 ExtractCertificate(BaseCardSignaturePart signaturePart)
```

Параметры

signaturePart

Тип: BaseCardSignaturePart

Часть подписи

Возвращаемое значение

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат

Заметки

Метод извлекает из объекта информацию по последней подписи и возвращает её вызвавшей стороне.

IBaseCardService.FindCardsByUniqueAttributes — метод (Guid, SearchParametersInfo)

Возвращает список идентификаторов карточек, имеющих указанные уникальные

атрибуты.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<Guid> FindCardsByUniqueAttributes(Guid cardKindId, SearchParametersInfo
searchParametersInfo)
```

Параметры

cardKindId

Тип: System.Guid

Идентификатор вида карточки

searchParametersInfo

Тип: SearchParametersInfo

Параметры поиска, представленные в виде массива полей карточки

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<Guid>

Коллекция идентификаторов карточек, удовлетворяющих условиям поиска

Заметки

Для выполнения данного запроса у карточки заданного вида должны быть настроены уникальные атрибуты в *Справочнике видов карточек*. Для поиска необходимо передать значение всех полей, обозначенных как уникальные атрибуты.

Примеры

Для примера можно рассмотреть поиск по уникальным атрибутам карточек вида "Заявка" (идентификатор 7DB9044E-91B7-447D-8CAA-5E5A4967B8D4). Для данного вида карточки в Справочнике видов карточек добавлены уникальные атрибуты: "Дата регистрации" и "Регистратор".

```
①
```

```
SearchParametersInfo searchParametersInfo = new SearchParametersInfo(); ②

SearchParameterInfo registrator = new SearchParameterInfo() { ③
 Name = "MainInfoRegistrar", ④
 Type = (int)DocsVision.Platform.Data.Metadata.CardModel.FieldType.RefId, ⑤
 Value = new Guid("00000000-0000-0000-0000-000000000000").ToString("B") }; ⑥

SearchParameterInfo regDate = new DocsVision.BackOffice.Xml.Schemas.SearchParameterInfo()
{  ⑦
 Name = "MainInfoRegDate",
 Type = (int)DocsVision.Platform.Data.Metadata.CardModel.FieldType.Date,
 Value = "2015.01.01" };

searchParametersInfo.SearchParameterInfo = new SearchParameterInfo[] { registrator,
regDate };

IBaseCardService baseCardService = objectContext.GetService<IBaseCardService>(); ⑧

IEnumerable<Guid> cards = baseCardService.FindCardsByUniqueAttributes(new Guid("7DB9044E-
91B7-447D-8CAA-5E5A4967B8D4"), searchParametersInfo); ⑨
```

① Инициализация контекста объектов.

② Инициализация массива параметров поиска.

③ Добавление значения параметра поиска по регистратору документа.

④ Название поля состоит из названия секции (`MainInfo`) и названия поля (`Registrar`).

⑤ Тип параметра указывается в соответствии с типом поля.

⑥ Регистратор — сотрудник с идентификатор, к примеру, `00000000-0000-0000-0000-000000000000.`

⑦ Добавление значения параметра поиска по дате регистрации документа.

⑧ Получение сервиса для работы с базовыми объектами.

⑨ Осуществления поиска документов с указанными уникальными атрибутами.

С помощью аналогичного запроса можно выявить сотрудников нарушивших уникальность документов, для которых предусмотрено ограничение на число экземпляров.

IBaseCardService.GenerateDigest — метод (BaseCard, IXmlExportable, String)

Генерирует дайджест на основе данных карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string GenerateDigest(BaseCard baseCard, IXmlExportable cardData, string defaultDigest)
```

Параметры

baseCard

Тип: BaseCard

Карточка

cardData

Тип: IXmlExportable

Данные карточки

defaultDigest

Тип: System.String

Базовое значение, которое будет возвращено в случае невозможности создания дайджеста из данных карточки

Возвращаемое значение

Тип: System.String

Дайджест

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard или cardData.

Примеры

В качестве примера использования данного сервиса, можно привести код, который позволяет сгенерировать описание (дайджест) карточки, на основе её данных. Это может понадобиться, например, при программном изменении имени

документа (свойство `MainInfo.Name`).

```
①

IBaseCardService baseCardService = objectContext.GetService<IBaseCardService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

CardData cardData = userSession.CardManager.GetCardData(new Guid("00000000-0000-0000-
0000-000000000000")); ④

string digest = baseCardService.GenerateDigest(document, cardData, "Дайджест по
умолчанию"); ⑤

document.Description = digest; ⑥

objectContext.AcceptChanges(); ⑦
```

① Инициализация контекста объектов.

② Получение базового сервиса для работы с карточками.

③ Получение документа для которого мы генерируем дайджест..

④ Получение данных документа.

⑤ При генерации метод формирует из данных карточки дайджест, согласно шаблону хранимого в базовом объекте. Если данных для генерации недостаточно, будет использован последний аргумент метода.

⑥ Дайджест Документа сохраняется в свойство `Description`.

⑦ Сохранение всего контекста.

IBaseCardService.GetActiveBusinessProcessInstances — метод (BaseCard, KindsCardProcess)

Получает список активных бизнес-процессов Workflow для заданной карточки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
IEnumerable<Process> GetActiveBusinessProcessInstances(BaseCard baseCard,
KindsCardProcess kindsCardProcess)
```

Параметры

baseCard

Тип: BaseCard

Карточка

kindsCardProcess

Тип: KindsCardProcess

Вид бизнес-процесса

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<Process>

Коллекция бизнес-процессов Workflow типа DocsVision.Workflow.Objects.Process.

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard или kindsCardProcess.

IBaseCardService.GetAllBusinessProcessInstances — метод (BaseCard, KindsCardProcess)

Получает список всех бизнес-процессов Workflow указанного вида для заданной карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<DocsVision.Workflow.Objects.Process> GetAllBusinessProcessInstances(BaseCard baseCard, KindsCardProcess kindsCardProcess)
```

Параметры

baseCard

Тип: BaseCard

Карточка

kindsCardProcess

Тип: KindsCardProcess

Вид бизнес-процесса

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<Process>

Коллекция бизнес-процессов Workflow типа DocsVision.Workflow.Objects.Process.

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard или kindsCardProcess.

IBaseCardService.GetBaseCardSectionRow — метод (BaseCard, CardSection)

Получает строку секции карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardSectionRow GetBaseCardSectionRow(BaseCard card, CardSection section)
```

Параметры

card

Тип: BaseCard

Карточка

section

Тип: CardSection

Секция карточки

Возвращаемое значение

Тип: BaseCardSectionRow

Строка секции

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр card или section.

IBaseCardService.GetBaseCardSectionRows — метод (BaseCard, CardSection)

Возвращает коллекцию всех строк секции карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<BaseCardSectionRow> GetBaseCardSectionRows(BaseCard card, CardSection
section)
```

Параметры

card

Тип: BaseCard

Карточка

section

Тип: CardSection

Секция карточки

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<BaseCardSectionRow>

Коллекция строк секции типа BaseCardSectionRow

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр card или section.

IBaseCardService.GetBusinessProcessesToStart — метод (BaseCard)

Получает список видов бизнес-процесса, которые могут быть запущены из карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<KindsCardProcess> GetBusinessProcessesToStart(BaseCard baseCard)
```

Параметры

baseCard

Тип: BaseCard

Карточка

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<KindsCardProcess>

Коллекция видов типа KindsCardProcess

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard.

IBaseCardService.GetBusinessProcessesToStop — метод (BaseCard)

Получает список видов бизнес-процесса карточки, который может быть остановлен.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<KindsCardProcess> GetBusinessProcessesToStop(BaseCard baseCard)
```

Параметры

baseCard

Тип: BaseCard

Карточка

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<KindsCardProcess>

Коллекция видов типа KindsCardProcess

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard.

Заметки

Возможность остановки БП определяется исходя из наличия у карточки активных экземпляров БП.

IBaseCardService.GetBusinessProcessesToView — метод (BaseCard)

Получает список бизнес-процессов у которых установлено разрешение на просмотр из карточки

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<KindsCardProcess> GetBusinessProcessesToView(BaseCard baseCard)
```

Параметры

baseCard

Тип: BaseCard

Карточка

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<KindsCardProcess>

Коллекция видов типа `KindsCardProcess`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `baseCard`.

IBaseCardService.GetCardTypeIcon — метод (KindsCardKind)

Возвращает иконку для карточки указанного вида карточки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
Image GetCardTypeIcon(KindsCardKind cardKind)
```

Параметры

cardKind

Тип: `KindsCardKind`

Вид карточки

Возвращаемое значение

Тип: `System.Drawing.Image`

Изображение карточки

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `cardKind`.

IBaseCardService.GetSignaturePartData — метод (BaseCardSignaturePart)

Извлекает данные из части подписи.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
byte[] GetSignaturePartData(BaseCardSignaturePart singaturePart)
```

Параметры

singaturePart

Тип: BaseCardSignaturePart

Часть подписи

Возвращаемое значение

Тип: System.Byte[]

Содержимое подписи

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр singaturePart.

Заметки

Полученные данные могут быть использованы, например, в методе IBaseCardService.AddSignaturePart.

IBaseCardService.InitializeBusinessProcessVariables — метод (Process, Dictionary<String, Object>)

Выполняет инициализацию переменных бизнес-процесса.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void InitializeBusinessProcessVariables(Process process, Dictionary<string, object>
variablesKeyValuePairs)
```

Параметры

process

Тип: Process

Бизнес-процессы

variablesKeyValuePairs

Тип: System.Collections.Generic.Dictionary<String, Object>

Коллекция содержащая название переменной и её значение

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр process или variablesKeyValuePairs.

Заметки

Метод выполняет поиск в бизнес-процессе переменной с заданным именем, определяет наличие у неё признака переменной с набором значений и записывает переданное значение в переменную.

Примеры

В приведенном примере создаётся экземпляр бизнес-процесса, на основе существующего шаблона, после чего выполняется инициализация двух его переменных (var1 и var2)

```
①

Library WorkflowLibrary = new Library(userSession, 1, 1, false); ②

DocsVision.Workflow.Objects.Process processTemplate = WorkflowLibrary.GetProcess(new
Guid("00000000-0000-0000-0000-000000000000")); ③

DocsVision.Workflow.Objects.Process processInstance = WorkflowLibrary.CreateProcess
(processTemplate); ④

IBaseCardService baseCardService = objectContext.GetService<IBaseCardService>();
IStaffService staffService = objectContext.GetService<IStaffService>(); ⑤

baseCardService.InitializeBusinessProcessVariables(processInstance, new System
.Collections.Generic.Dictionary<string, object>{ ⑥
```

```
 {"var1", "Значение строковой переменной"},
 {"var2", new object[]{1,2,4}} ⑦
});

processInstance.Start(staffService.GetCurrentEmployee().AccountName, WorkflowLibrary
.Dictionary, ExecutionModeEnum.Automatic, true); ⑧
```

① Инициализация контекста объектов.

② Инициализация объекта для работы с библиотекой карточкой СУБП.

③ Получение шаблона бизнес-процесса.

④ Создание экземпляра бизнес-процесса.

⑤ Получение необходимых сервисов.

⑥ Инициализация двух переменных.

⑦ `var2` является переменной с набором значений.

⑧ Запуск экземпляра бизнес-процесса.

IBaseCardService.InitializeSystemInfo — метод (BaseCard, KindsCardKind)

Инициализирует системную информацию карточки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void InitializeSystemInfo(BaseCard baseCard, KindsCardKind cardKind)
```

Параметры

baseCard

Тип: `BaseCard`

Карточка

cardKind

Тип: `KindsCardKind`

Вид карточки. Используется в случае невозможности получения данных вида карточки из самой карточки

Исключения

Исключение	Условие
`System.InvalidOperationException`	Ошибка возникает в случае невозможности получения вида карточки — не определён `baseCard.SystemInfo.CardKind` и не задан `cardKind`. Может также возникнуть, если в Конструкторе состояний не указано начальное состояние вида карточки.

Заметки

Метод выполняется в том случае, если `baseCard.SystemInfo` не является `null`. Фактически метод задаёт значение вида карточки, если не задано, и устанавливает текущее состояние карточки в начальное значение.

IBaseCardService.RemoveSignature — метод (SignatureList, BaseCardSignature)

Удалят указанную подпись из списка подписей.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void RemoveSignature(SignatureList signatureList, BaseCardSignature signature)
```

Параметры

signatureList

Тип: `SignatureList`

Список, содержащий удаляемую подпись

signature

Тип: `BaseCardSignature`

Подпись

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр signatureList или signature.

IBaseCardService.RemoveSignaturePart — метод (BaseCardSignaturePart)

Удаляет указанную часть из подписи.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void RemoveSignaturePart(BaseCardSignaturePart signaturePart)
```

Параметры

signaturePart

Тип: BaseCardSignaturePart

Часть подписи

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр signaturePart.

IBaseCardService.SendMailNotification — метод (IEnumerable<MailNotificationInfo>)

Отправляет оповещение по электронной почте.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void SendMailNotification(IEnumerable<MailNotificationInfo> mailNotificationInfos)
```

Параметры

mailNotificationInfos

Тип: System.Collections.Generic.IEnumerable<MailNotificationInfo>

Коллекция сообщений, подготовленных к отправке

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр mailNotificationInfos.

Заметки

Данный метод формирует экземпляр бизнес-процесса "УД Отправка почтовых уведомлений", который выполняет отправку электронного сообщения. Для возможности отправки сообщений должны быть произведены соответствующие настройки шлюза Workflow (см. документ "Руководство по настройке").

Примеры

Ниже приведён пример отправки почтового уведомления с использованием сервиса базовых объектов

```
①

IBaseCardService baseCardService = objectContext.GetService<IBaseCardService>(); ②

string eMailFormat = "<root ID=\"{0}\" Email=\"{1}\" Name=\"{1}\"></root>"; ③

string eMail = string.Format(eMailFormat, Guid.NewGuid(), "sample@domain.com"); ④

MailNotificationInfo mailNotificationInfo = new MailNotificationInfo(eMail,
"Уведомление", "Данное письмо является информационным"); ⑤

baseCardService.SendMailNotification(new MailNotificationInfo[] { mailNotificationInfo
}); ⑥
```

① Инициализация контекста объектов.

② Получение сервиса для работы с базовыми объектами.

③ Специальный формат для почтового адреса.

④ Получение допустимого почтового адреса из "sample@domain.com".

⑤ Инициализация нового информационного письма с тестовым адресом.

⑥ Отправка сообщения.

IBaseCardService.GetPartSignatureInfo — метод (BaseCardSignaturePart signaturePart, Stream contentStream, X509Certificate2 certificate)

Возвращает информацию о части подписи документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
SignatureInfo GetPartSignatureInfo(BaseCardSignaturePart signaturePart, Stream
contentStream, X509Certificate2 certificate);
```

Параметры

signaturePart

Тип: BaseCardSignaturePart

Часть подписи, для которой запрашивается информация

contentStream

Тип: Stream

Поток подписываемых данных

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат подписи

Возвращаемое значение

Тип: SignatureInfo

Информация о подписи.

IBaseCardService.StartBusinessProcess — метод (BaseCard, KindsCardProcess)

Запускает бизнес-процесс указанного вида.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
Process StartBusinessProcess(BaseCard baseCard, KindsCardProcess kindsCardProcess)
```

Параметры

baseCard

Тип: BaseCard

Карточка

kindsCardProcess

Тип: KindsCardProcess

Вид бизнес-процесса

Возвращаемое значение

Тип: Process

Бизнес-процесс

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard или kindsCardProcess.
DocsVision.Platform.StorageServer.StorageServerException	Возвращается в случае появления ошибки на стороне сервера.

Заметки

Если текущий сотрудник не определён, то бизнес-процесс будет создан, но не будет запущен. Запуск БП может быть отменен по результатам обработки события IBaseCardService.BeforeBusinessProcessStart, но только в случае запуска процесса из сервиса IBaseCardService.

Примеры

Ниже приведён пример запуска бизнес-процесса, определённого в виде карточки

```
①

IBaseCardService baseCardService = objectContext.GetService<IBaseCardService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

KindsCardProcess kindsCardProcess = objectContext.GetObject<KindsCardProcess>(new Guid
("00000000-0000-0000-0000-000000000001")); ④

baseCardService.StartBusinessProcess(document, kindsCardProcess); ⑤
```

① Инициализация контекста объектов.

② Получаем необходимый сервис.

③ Получаем документ, вид которого позволяет запускать бизнес-процесс вида `00000000-0000-0000-0000-000000000001`.

④ Получение вида бизнес-процесса.

⑤ Запускаем БП.

IBaseCardService.StopBusinessProcess — метод (BaseCard, Process)

Останавливает выполнение бизнес-процесса.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void StopBusinessProcess(BaseCard baseCard, Process processInstance)
```

Параметры

baseCard

Тип: `BaseCard`

Карточка

processInstance

Тип: `Process`

Экземпляр бизнес-процесса

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard или processInstance.

Заметки

Бизнес-процесс переводится в статус "Остановлен". При невозможности определения текущего сотрудника (IStaffService.GetCurrentEmployee()), что возможно при некорректном открытии сессии, — метод не отработает.

IBaseCardService.UpdateBusinessProcessActiveInstancesProperties — метод (BaseCard)

Сохраняет значения переменных запущенных бизнес-процессов указанной карточки в самой карточке.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void UpdateBusinessProcessActiveInstancesProperties(BaseCard baseCard)
```

Параметры

baseCard

Тип: BaseCard

Карточка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard.

Заметки

Системная информация карточки (baseCard.SystemInfo) и вид карточки не должны быть пустыми, иначе метод не отработает.

IBaseCardService.UpdateBusinessProcessActiveInstancesVariables — метод (BaseCard)

Загружает из карточки значения переменных активных бизнес-процессов карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void UpdateBusinessProcessActiveInstancesVariables(BaseCard baseCard)
```

Параметры

baseCard

Тип: BaseCard

Карточка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard.

Заметки

Системная информация карточки (baseCard.SystemInfo) и вид карточки не должны быть пустыми, иначе метод не отработает.

IBaseCardService.VerifySignature — метод (BaseCardSignature, IDictionary<BaseCardSignaturePart, Stream>, X509Certificate2)

Проверяет корректность подписи.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardSignatureVerification VerifySignature(BaseCardSignature signature, IDictionary
<BaseCardSignaturePart, Stream> contentStreams, X509Certificate2 certificate)
```

Параметры

signature

Тип: `BaseCardSignature`

Подпись

contentStreams

Тип: `System.Collections.Generic.IDictionary<BaseCardSignaturePart, Stream>`

Пара типа `BaseCardSignaturePart` и `System.IO.Stream`, представляющая подписанный объект и соответствующую ему часть подписи

certificate

Тип: `System.Security.Cryptography.X509Certificates.X509Certificate2`

Сертификат, на основе которого проверяется корректность подписей

Возвращаемое значение

Тип: `BaseCardSignatureVerification`

Результат проверки

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `signature`.

Заметки

Метод последовательно проверяет соответствие сертификата (`certificate`) и подписи (`signature`), подписанной операции и всех элементов переданного словаря (`contentStreams`).

IBaseCardService.BeforeBusinessProcessStart — событие

Происходит перед непосредственным запуском бизнес-процесса.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
event EventHandler<BusinessProcessCancelEventArgs> BeforeBusinessProcessStart
```

Заметки

Аргумент события содержит свойство Cancel, установка которого в значение `true` позволяет предотвратить запуск БП

Примеры

Ниже приведён пример перехвата запуска БП у которого приоритет выше 2.

```
①

IBaseCardService baseCardService = objectContext.GetService<IBaseCardService>(); ②

baseCardService.BeforeBusinessProcessStart += (s, e) => ③
{
 if (e.Process.Priority > 2) ④
 {
  e.Cancel = false;
 }
};

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ⑤

KindsCardProcess kindsCardProcess = objectContext.GetObject<KindsCardProcess>(new Guid
("00000000-0000-0000-0000-000000000001")); ⑥

baseCardService.StartBusinessProcess(document, kindsCardProcess); ⑦
```

① Инициализация контекста объектов.

② Получаем необходимые сервисы.

③ Подключаемся к событию.

④ Определяем приоритет и если выше `2`, блокируем запуск.

⑤ Получаем документ, вид которого позволяет запускать бизнес-процесс вида `00000000-0000-0000-0000-000000000001`.

⑥ Получаем вид бизнес-процесса.

⑦ запуск БП.

IBaseUniversalService — интерфейс

Описывает сервис для работы с *Конструктором справочников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IBaseUniversalService
```

Методы

Имя	Описание
AddItem(BaseUniversalItemType, BaseUniversalItem)	Добавляет строку в указанный узел справочника.
AddNewItem(BaseUniversalItemType)	Создаёт новую строку в указанном узле справочника.
AddNewItemType(BaseUniversalItemType)	Создаёт дочерний узел в указанном узле справочника.
AddNewViewField(String, SectionField, BaseUniversalItemType)	Создаёт отображаемое поле в коллекции полей, и возвращает на него ссылку.
CanCopyItemType(BaseUniversalItemType, BaseUniversalItemType)	Определяет возможность создания копии узла в заданном узле конструктора справочников.
CanMoveItemType(BaseUniversalItemType, BaseUniversalItemType)	Определяет возможность перемещения узла в заданный узел конструктора справочников.
CopyItem(BaseUniversalItem, BaseUniversalItemType)	Создаёт копию строки в заданным узле конструктора справочников.
CopyItemType(BaseUniversalItemType, BaseUniversalItemType)	Создаёт копию узла в заданным узле конструктора справочников.
DeleteItem(BaseUniversalItem)	Удаляет заданную строку справочника.
DeleteItemType(BaseUniversalItemType)	Удаляет заданный узел справочника.

Имя	Описание
`DeleteViewField(BaseUniversalItemTypeViewField, BaseUniversalItemType)`	Удаляет переданное отображаемое поле из узла справочника.
`FindItemTypeWithSameName(String, BaseUniversalItemType)`	Осуществляет поиск узла с заданным именем в переданном родительском узле.
`FindItemWithSameName(String, BaseUniversalItemType)`	Осуществляет поиск строки с заданным именем в переданном родительском узле.
`GetFieldDisplayName(SectionField)`	Возвращает, по возможности локализованное, название отображаемого поля.
`GetItemSectionField(BaseUniversalItemTypeViewField)`	Возвращает поле строки секции справочника, соответствующее заданному отображаемому полю.
`GetItemTypeCardKind(BaseUniversalItemType)`	Определяет вид карточки строки справочника.
`MoveItem(BaseUniversalItem, BaseUniversalItemType)`	Перемещает строку в заданный узел справочника.
`MoveItems(IEnumerable<BaseUniversalItem>, BaseUniversalItemType)`	Перемещает строки из коллекции в заданный узел справочника.
`MoveItemType(BaseUniversalItemType, BaseUniversalItemType)`	Перемещает узел справочника в переданный узел.
`OpenOrCreateItemCard(BaseUniversalItem)`	Возвращает карточку строки справочника, если не существует, то создаст пустую.

IBaseUniversalService.AddItem — метод (BaseUniversalItemType, BaseUniversalItem)

Добавляет строку в указанный узел справочника.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
BaseUniversalItem AddItem(BaseUniversalItemType itemType, BaseUniversalItem item)
```

Параметры

itemType

Тип: `BaseUniversalItemType`

Узел справочника

item

Тип: `BaseUniversalItem`

Добавляемая строка

Возвращаемое значение

Тип: `BaseUniversalItem`

Ссылка на строку

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `itemType` или `item`.

Заметки

В случае, если узел itemType уже содержит значение item, будет возвращен null.

IBaseUniversalService.AddNewItem — метод (BaseUniversalItemType)

Создаёт новую строку в указанном узле справочника.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
BaseUniversalItem AddNewItem(BaseUniversalItemType itemType)
```

Параметры

itemType

Тип: BaseUniversalItemType

Узел справочника

Возвращаемое значение

Тип: BaseUniversalItem

Добавленная строка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр itemType.

Примеры

В примере, сервис используется для добавления строки в существующий узел

```
①

IBaseUniversalService baseUniversalService = objectContext.GetService
<IBaseUniversalService>(); ②

BaseUniversalItemType baseUniversalItemType = objectContext.GetObject
<BaseUniversalItemType>(new Guid("00000000-0000-0000-0000-000000000000")); ③

BaseUniversalItem baseUniversalItem = baseUniversalService.AddNewItem
(baseUniversalItemType);
baseUniversalItem.Name = "Строка справочника";
baseUniversalItem.Description = "Строка справочника созданная из кода"; ④
objectContext.SaveObject<BaseUniversalItem>(baseUniversalItem);
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение узла, в который будет произведено добавление строки.

④ Добавление строки.

IBaseUniversalService.AddNewItemType — метод (BaseUniversalItemType)

Создаёт дочерний узел в указанном узле справочника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseUniversalItemType AddNewItemType(BaseUniversalItemType parentItemType)
```

Параметры

parentItemType

Тип: BaseUniversalItemType

Родительский узел

Возвращаемое значение

Тип: BaseUniversalItemType

Добавленный узел

Заметки

Если parentItemType не задан, то узел будет создан в корневом узле дерева справочника.

Примеры

```
①

IBaseUniversalService baseUniversalService = objectContext.GetService
<IBaseUniversalService>(); ②

BaseUniversalItemType baseUniversalItemType = baseUniversalService.AddNewItemType(null);
baseUniversalItemType.Name = "Узел справочника";
baseUniversalItemType.Description = "Узел справочника созданный из кода"; ③

baseUniversalItemType.UseOrder = true; ④

objectContext.SaveObject<BaseUniversalItemType>(baseUniversalItemType); ⑤
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Создаем новый узел в корне справочника.

④ Элементы узла будут использовать порядковые номера.

⑤ Сохранение.

IBaseUniversalService.AddNewViewField — метод (String, SectionField, BaseUniversalItemType)

Создаёт отображаемое поле в коллекции полей, и возвращает на него ссылку.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseUniversalItemTypeViewField AddNewViewField(string fieldName, SectionField
sectionField, BaseUniversalItemType itemType)
```

Параметры

fieldName

Тип: System.String

Название поля для которого создаётся отображение

sectionField

Тип: SectionField

Поле секции для которого создаётся отображение

itemType

Тип: BaseUniversalItemType

Узел справочника к которому добавляется отображаемое поле

Возвращаемое значение

Тип: BaseUniversalItemTypeViewField

Отображаемое поле

Примеры

В приведенном ниже примере создаётся отображаемое поле в узле конструктора справочников

```
①

IBaseUniversalService baseUniversalService = objectContext.GetService
<IBaseUniversalService>(); ②

IMetadataProvider iMetadataProvider = objectContext.GetService<IMetadataProvider>(); ③

CardSection cardSectionMetadata = iMetadataProvider.GetSection(new Guid("1B1A44FB-1FB1-
4876-83AA-95AD38907E24")); ④

BaseUniversalItemType baseUniversalItemType = objectContext.GetObject
<BaseUniversalItemType>(new Guid("00000000-0000-0000-0000-000000000000")); ⑤

SectionField sectionField = cardSectionMetadata.Fields[new Guid("8B45BC58-6831-40C1-B654-
E0C0C6E36115")]; ⑥

BaseUniversalItemTypeViewField baseUniversalItemTypeViewField = baseUniversalService
.AddNewViewField(sectionField.Name, sectionField, baseUniversalItemType);
baseUniversalItemTypeViewField.Order = baseUniversalItemType.ViewFields.Count; ⑦

objectContext.SaveObject(baseUniversalItemTypeViewField); ⑧
```

① Инициализация контекста объектов.

② Получение необходимых сервисов.

③ Получение поставщика метаданных.

④ Получение секции, которая содержит отображаемое поле (в нашем случае секция "Строка справочника" карточки "Конструктор справочников").

⑤ Получение узла, для которого добавляется отображаемое поле.

⑥ Получение поля "Описание".

⑦ Создание нового отображаемого поля и установка порядкового номера.

⑧ Сохранение.

IBaseUniversalService.CanCopyItemType — метод (BaseUniversalItemType, BaseUniversalItemType)

Определяет возможность создания копии узла в заданном узле конструктора справочников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanCopyItemType(BaseUniversalItemType copiedItemType, BaseUniversalItemType
targetItemType)
```

Параметры

copiedItemType

Тип: BaseUniversalItemType

Узел, который проверяется на возможность копирования

targetItemType

Тип: BaseUniversalItemType

Узел, который является получателем копии

Возвращаемое значение

Тип: System.Boolean

true — копирование допустимо, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр copiedItemType.

Заметки

Метод проходит дерево от targetItemType до его родительского узла, проверяя наличии copiedItemType среди родительских узлов.

IBaseUniversalService.CanMoveItemType — метод (BaseUniversalItemType, BaseUniversalItemType)

Определяет возможность перемещения узла в заданный узел конструктора справочников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanMoveItemType(BaseUniversalItemType movedItemType, BaseUniversalItemType
```

```
targetItemType)
```

Параметры

movedItemType

Тип: `BaseUniversalItemType`

Узел, который проверяется на возможность перемещения

targetItemType

Тип: `BaseUniversalItemType`

Узел, в который планируется перемещение

Возвращаемое значение

Тип: `System.Boolean`

`true` — перемещение допустимо, иначе — `false`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `movedItemType`.

Заметки

Метод проходит дерево от targetItemType до его родительского узла, проверяя наличии movedItemType среди родительских узлов.

IBaseUniversalService.CopyItem — метод (BaseUniversalItem, BaseUniversalItemType)

Создаёт копию строки в заданном узле конструктора справочников.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
BaseUniversalItem CopyItem(BaseUniversalItem copiedItem, BaseUniversalItemType
targetItemType)
```

Параметры

copiedItem

Тип: BaseUniversalItem

Копируемая строка

targetItemType

Тип: BaseUniversalItemType

Узел в который производится копирование

Возвращаемое значение

Тип: BaseUniversalItem

Созданная копия строки

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр copiedItem или targetItemType.

IBaseUniversalService.CopyItemType — метод (BaseUniversalItemType, BaseUniversalItemType)

Создаёт копию узла в заданном узле конструктора справочников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseUniversalItemType CopyItemType(BaseUniversalItemType copiedItemType,
BaseUniversalItemType targetItemType)
```

Параметры

copiedItemType

Тип: BaseUniversalItemType

Копируемый узел

targetItemType

Тип: BaseUniversalItemType

Узел в который производится копирование

Возвращаемое значение

Тип: BaseUniversalItemType

Созданная копия узла

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр copiedItemType.

Примеры

```
①

IBaseUniversalService baseUniversalService = objectContext.GetService
<IBaseUniversalService>(); ②

BaseUniversalItemType baseUniversalItemTypeToCopy = objectContext.GetObject
<BaseUniversalItemType>(new Guid("00000000-0000-0000-0000-000000000000")); ③

BaseUniversalItemType baseUniversalItemTypeParent = objectContext.GetObject
<BaseUniversalItemType>(new Guid("00000000-0000-0000-0000-000000000001")); ④

BaseUniversalItemType copiedItem = baseUniversalService.CopyItemType
(baseUniversalItemTypeToCopy, baseUniversalItemTypeParent);
copiedItem.Name = string.Format("Копия {0}", baseUniversalItemTypeToCopy.Name); ⑤

objectContext.SaveObject<BaseUniversalItemType>(copiedItem); ⑥
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Копируемый узел справочника.

④ Узел в который будет произведено копирование.

⑤ Создание копии.

⑥ Сохранение.

IBaseUniversalService.DeleteItem — метод (BaseUniversalItem)

Удаляет заданную строку справочника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteItem(BaseUniversalItem item)
```

Параметры

item

Тип: BaseUniversalItem

Строка справочника

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр item.

ICalendarService — интерфейс

Сервис для работы с бизнес-календарем.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface ICalendarService
```

Методы

Имя	Описание
AddCalendarDay(Int32)	Создаёт объект типа CalendarDay с заданным значением календарного дня.

Имя	Описание
`AddCalendarDefaultWorkTime(DateTime, DateTime)`	Создаёт объект типа `CalendarDefaultWorkTime` с заданными значениями начального и конечного времени.
`AddCalendarWorkTime(DateTime, DateTime)`	Создаёт объект типа `CalendarWorkTime` с заданными значениями начального и конечного времени.
`AddCalendarYear(Int32)`	Создаёт объект типа `CalendarYear` с заданным значением года.
`GetCalendarID(IEnumerable<ObjectBase>)`	Возвращает идентификатор бизнес-календаря.
`GetCalendarID(IEnumerable<Guid>)`	Возвращает идентификатор бизнес-календаря.
`GetDuration(Guid, DateTime, DateTime)`	Рассчитывает длительность (в часах) заданного периода времени с учетом плана в бизнес-календаре.
`GetEndDate(Guid, DateTime, Double)`	Рассчитывает конечное время с учетом плана в бизнес-календаре и указанной длительности.
`GetStartDate(Guid, DateTime, Double)`	Рассчитывает начальное время с учетом плана в бизнес-календаре и указанной длительности.

ICategoriesService — интерфейс

Сервис для работы со *Справочником категорий*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface ICategoriesService
```

Методы

Имя	Описание
AddCategory(CategoriesCategory)	Добавляет новую категорию в *Справочник категорий*, в указанную родительскую категорию.
CanMoveCategory(CategoriesCategory, CategoriesCategory)	Определяет возможность перемещения категории в новую родительскую категорию.
ChangeLinkedCategories(CategoriesCategory, IEnumerable<CategoriesCategory>, IEnumerable<CategoriesCategory>)	Добавляет или удаляет связанные категории.
DeleteAllFolders	Удаляет все папки категорий.
DeleteCategory(CategoriesCategory)	Удаляет указанную категорию.
FindCategoryByName(CategoriesCategory, String)	Выполняет поиск категории с заданным именем.
IsCategoryChildLockedByAnotherUser(CategoriesCategory)	Определяет наличие заблокированных другими пользователями дочерних категорий в заданной родительской.
IsDependentCategoryLockedbyAnotherUser(CategoriesCategory)	Определяет наличие блокировки у категории, с которой связана переданная категория.
MoveCategory(CategoriesCategory, CategoriesCategory)	Перемещает подкатегорию в новую категорию.
OtherCategoryWithSameNameExists(CategoriesCategory, String)	Определяет наличие в указанной категории подкатегории с заданным именем.
PasteCategory(CategoriesCategory, CategoriesCategory)	Создаёт копию указанной категории с сохранением прав доступа и существующих подкатегорий.
PropagateNotAvailable(CategoriesCategory)	Распространяет признак "Не показывать при выборе категории" на подкатегории заданной категории.
RebuildFolders(ILongProcessManager)	Перестраивает дерева папок.

Имя	Описание
`RenameCategory(CategoriesCategory, String)`	Переименовывает категорию.

ICategoriesService.AddCategory — метод (CategoriesCategory)

Добавляет новую категорию в *Справочник категорий*, в указанную родительскую категорию.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
CategoriesCategory AddCategory(CategoriesCategory parentCategory)
```

Параметры

parentCategory

Тип: `CategoriesCategory`

Родительская категория

Возвращаемое значение

Тип: `CategoriesCategory`

Созданная категория

Заметки

Если параметр `parentCategory` не задан (имеет значение `null`), то в качестве родительской категории будет использована корневая категория *Справочника категорий*.

Примеры

```
①

ICategoriesService categoriesService = objectContext.GetService<ICategoriesService>(); ②

CategoriesCategory category = categoriesService.AddCategory(null); ③
category.Name = "Новая категория";
category.Comment = "Пример создания категории";
```

```
category.View = objectContext.GetObject<SavedViewsView>(new Guid("00000000-0000-0000-
0000-000000000000")); ④

objectContext.SaveObject(category); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса для работы с сервисами.

③ Добавление категории в корневую категорию.

④ Получение и присвоение представления. Если представление не задано, то будет использоваться Дайджест.

⑤ Сохранение.

ICategoriesService.FindCategoryByName — метод (CategoriesCategory, String)

Выполняет поиск категории с заданным именем.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
CategoriesCategory FindCategoryByName(CategoriesCategory parentCategory, string name)
```

Параметры

parentCategory

Тип: CategoriesCategory

Категория, в которой осуществляется поиск

name

Тип: System.String

Название категории

Возвращаемое значение

Тип: CategoriesCategory

Искомая категория

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `name`.

Заметки

Поиск игнорирует регистр значения параметра name. Если параметр `parentCategory` не задан (значение `null`), то будет выполнен поиск в корневой категории. Поиск выполняется только среди непосредственных потомков `parentCategory` — поиск в подкатегориях не выполняется.

Примеры

```
①

ICategoriesService categoriesService = objectContext.GetService<ICategoriesService>(); ②

CategoriesCategory category = categoriesService.FindCategoryByName(null, "Новая
категория"); ③
```

① Инициализация контекста объектов.

② Получение сервиса для работы со Справочником категорий.

③ Поиск категории с названием "Новая категория" (регистр не имеет значения) в корневой категории.

ICategoriesService.PasteCategory — метод (CategoriesCategory, CategoriesCategory)

Создаёт копию указанной категории с сохранением прав доступа и существующих подкатегорий.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
CategoriesCategory PasteCategory(CategoriesCategory parentCategory, CategoriesCategory
categoryToCopy)
```

Параметры

parentCategory

Тип: `CategoriesCategory`

Категория, в которую будет помещена копия категории

categoryToCopy

Тип: `CategoriesCategory`

Копируемая категория

Возвращаемое значение

Тип: `CategoriesCategory`

Готовая копия категории

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `categoryToCopy`.

Заметки

Примеры

```
①

ICategoriesService categoriesService = objectContext.GetService<ICategoriesService>(); ②

CategoriesCategory copiedCategory = categoriesService.FindCategoryByName(null,
"Копируемая категория"); ③
CategoriesCategory parentCategory = categoriesService.FindCategoryByName(null,
"Родительская категория");

CategoriesCategory copy = categoriesService.PasteCategory(parentCategory,
copiedCategory); ④
objectContext.SaveObject(copiedCategory);
```

① Инициализация контекста объектов.

② Получение сервиса для работы со Справочником категорий.

③ Получение копируемой категории и категории, в которую будет помещена копия.

④ Создание копии.

ICategoryListService — интерфейс

Сервис для работы со списком категорий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface ICategoryListService
```

Методы

Имя	Описание
AddCategory(CategoryList, CategoriesCategory)	Добавляет категорию в заданный список категорий.
CreateCategoryList	Создаёт новый список категорий.

ICategoryListService.AddCategory — метод (CategoryList, CategoriesCategory)

Добавляет категорию в заданный список категорий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
CategoryListCategory AddCategory(CategoryList categoryList, CategoriesCategory category)
```

Параметры

categoryList

Тип: CategoryList

Список категорий, в который осуществляется добавление

category

Тип: CategoriesCategory

Категория, в которую осуществляется добавление

Возвращаемое значение

Тип: CategoryListCategory

Добавленная категория, представленная в списке категорий

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр categoryList или category.

ICategoryListService.CreateCategoryList — метод

Создаёт новый список категорий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
CategoryList CreateCategoryList()
```

Возвращаемое значение

Тип: CategoryList

Список категорий

ICryptService — интерфейс

Сервис для работы с шифрованием базовых объектов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface ICryptService
```

Методы

Имя	Описание
AddCryptAccount(X509Certificate2, Guid, String)	Предоставляет доступ указанному сотруднику к зашифрованному файлу.
DecryptFile(X509Certificate2, Guid, Stream)	Расшифровывает файл в переданный поток.
EncryptFile(X509Certificate2, Guid)	Выполняет шифрование файла с помощью переданного сертификата.
RemoveCryptAccount(X509Certificate2, Guid, String)	Отменяет доступ указанного сотрудника к зашифрованному файлу.
RemoveEncryption(X509Certificate2, Guid)	Расшифровывает файл и удаляет связанные с ним крипто-объекты.

Заметки

Сервис шифрования базовых объектов доступен для вызова исключительно из скрипта карточки, унаследованной от класса BaseCardControl.

Шифрование файлов и сервис шифрования доступны начиная с версии 5.4 платформы.

ICryptService.AddCryptAccount — метод (X509Certificate2, Guid, String)

Предоставляет доступ указанному сотруднику к зашифрованному файлу.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void AddCryptAccount(X509Certificate2 certificate, Guid fileId, string accountName)
```

Параметры

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат шифрования, которым зашифрован файл

fileId

Тип: System.Guid

Идентификатор зашифрованного файла

accountName

Тип: `System.String`

Учетная запись сотрудника, которому предоставляется доступ к зашифрованному файлу

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `certificate` или `accountName`.
`System.InvalidOperationException`	Ошибка возвращается в случае, если файл с указанным идентификатором не существует, либо у указанного сотрудника уже имеется доступ к файлу, либо у указанного сотрудника отсутствует открытый ключ шифрования.

Заметки

В том случае, если файл не является зашифрованным, будет произведено шифрование сертификатом `certificate`, поэтому вызывать метод `ICryptService.EncryptFile` не требуется.

ICryptService.DecryptFile — метод (X509Certificate2, Guid, Stream)

Расшифровывает файл в переданный поток.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void DecryptFile(X509Certificate2 certificate, Guid fileId, Stream destStream)
```

Параметры

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат шифрования, которым зашифрован файл

fileId

Тип: System.Guid

Идентификатор зашифрованного файла

destStream

Тип: System.IO.Stream

Поток для записи расшифрованных данных

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр certificate или destStream.
System.InvalidOperationException	Ошибка возвращается в случае, если файл с указанным идентификатором не существует.

ICryptService.EncryptFile — метод (X509Certificate2, Guid)

Выполняет шифрование файла с помощью переданного сертификата.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void EncryptFile(X509Certificate2 certificate, Guid fileId)
```

Параметры

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат сотрудника, выполняющего шифрование

fileId

Тип: System.Guid

Идентификатор шифруемого файла

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр certificate.
System.InvalidOperationException	Ошибка возвращается в случае, если файл с указанным идентификатором не существует.

Заметки

В том случае, если указанный файл уже зашифрован, шифрование будет отменено.

Параметры шифрования должны быть определены в системных настройках Docsvision.

ICryptService.RemoveCryptAccount — метод (X509Certificate2, Guid, String)

Отменяет доступ указанного сотрудника к зашифрованному файлу.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void RemoveCryptAccount(X509Certificate2 certificate, Guid fileId, string accountName)
```

Параметры

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат шифрования, которым зашифрован файл

fileId

Тип: System.Guid

Идентификатор зашифрованного файла

accountName

Тип: `System.String`

Учетная запись сотрудника, у которого отбирается доступ к зашифрованному файлу

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `certificate` или `accountName`.
`System.InvalidOperationException`	Ошибка возвращается в случае, если файл с указанным идентификатором не существует.

Заметки

В том случае, если доступ к файлу имеет только владелец, то файл будет расшифрован.

ICryptService.RemoveEncryption — метод (X509Certificate2, Guid)

Расшифровывает файл и удаляет связанные с ним крипто-объекты.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void RemoveEncryption(X509Certificate2 certificate, Guid fileId)
```

Параметры

certificate

Тип: `System.Security.Cryptography.X509Certificates.X509Certificate2`

Сертификат шифрования, которым зашифрован файл

fileId

Тип: System.Guid

Идентификатор зашифрованного файла

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр certificate.
System.InvalidOperationException	Ошибка возвращается в случае, если файл с указанным идентификатором не существует.

Заметки

Данный метод, в отличие от метода ICryptService.DecryptFile, удаляет крипто-объекты, используемые для расшифровки файла сотрудниками, которым предоставлен соответствующий доступ.

IDocumentService — интерфейс

Представляет сервис для работы с документами.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IDocumentService
```

Методы

Имя	Описание
AddAdditionalFile(Document, String)	Добавляет в документ дополнительный файл.
AddAdditionalFile(Document, VersionedFileCard)	Добавляет в документ дополнительный файл.
AddAdditionalFiles(Document, IEnumerable<string>)	Добавляет дополнительные файлы в документ.

Имя	Описание
`AddMainFile(Document, String)`	Добавляет в документ основной файл.
`AddMainFileComment(Document, DocumentFile, String)`	Добавляет комментарий к основному файлу документа.
`AddMainFiles(Document document, IEnumerable<String>)`	Добавляет несколько основных файлов в документ.
`AddMainFileVersionComment(Document, DocumentFile, Guid, String)`	Добавляет комментарий к указанной версии основного файла документа.
`AddSignature(Document, X509Certificate2, Boolean, ICollection<CardFieldSetting>)`	Добавляет новую подпись в список подписей карточки и устанавливает ЭП на поля, основные и дополнительные файлы документа.
`AddSignature(Document, X509Certificate2, SignatureLabel, StaffEmployee, String, Boolean, ICollection<CardFieldSetting>)`	Данный метод предназначен для выполнения подписания операции в документе.
`AddSignatures(Document, `SignedCms)`	Данный метод добавляет коллекцию новых подписей в список подписей карточки, и устанавливает ЭП на поля, основные и дополнительные файлы документа. Позволяет использовать для подписания криптографическое сообщение типа `SignedCms`.
`AddSignatures – метод (Document, DocumentFile, `SignedCms)`	Добавляет коллекцию новых подписей в список подписей карточки. Определяет, к какому файлу будет добавлена подпись. Позволяет использовать для подписания криптографическое сообщение типа `SignedCms`.

Имя	Описание
AddSignatures – метод (Document, DocumentFile, `SignedCms, bool)`	Добавляет коллекцию новых подписей в список подписей карточки. Определяет, к какому файлу будет добавлена подпись. Позволяет использовать для подписания криптографическое сообщение типа SignedCms. Позволяет проверять срок действия добавляемых подписей.
AddSignatures – метод (Document, `SignedCms, bool)`	Добавляет коллекцию новых подписей в список подписей карточки. Устанавливает ЭП на поля, основные и дополнительные файлы документа. Позволяет использовать для подписания криптографическое сообщение типа SignedCms. Позволяет проверять срок действия добавляемых подписей.
ApplyKindSettings(Document, DocumentSetting)	Устанавливает у документа определённые настройки вида карточки.
ApplyKindSettings(Document, KindsCardKind)	Устанавливает у документа настройки указанного вида карточки.
AttachMainFile(Document, String)	Загружает в документ основной файл.
AttachMainFile(Document, VersionedFileCard)	Добавляет в документ карточку версионного файла в качестве основного файла документа.
AttachMainFile(Document, string, `SignedCms, bool)`	Добавляет в документ подпись в качестве основного файла документа, проверяет срок действия подписи.
CanAttachMainFile(Document, VersionedFileCard)	Определяет возможность добавления основного файла в документ.
CanAttachMainFile(Document, VersionedFileCard, `SignedCms)`	Определяет возможность добавления основного файла в документ.

Имя	Описание
CanCancelLockAnyMainFile(Document)	Определяет возможность снятия блокировки с основных файлов.
CanCancelLockMainFile(DocumentFile)	Определяет возможность снятия блокировки с основного файла.
CancelLockMainFile(Document, DocumentFile)	Снимает блокировку с основного файла документа.
CancelLockMainFileIfNeeded(Document, DocumentFile)	Снимает блокировку с основного файла документа, если это необходимо.
CanEditAnyMainFile(Document)	Определяет наличие в документе основного файла, который может быть открыт для редактирования.
CanEditMainFile(Document, DocumentFile, Boolean)	Определяет возможность выгрузки указанного основного файла для редактирования.
CanLockAnyMainFile(Document)	Определяет наличие по меньшей мере одного основного файла документа, который может быть заблокирован.
CanLockMainFile(Document, DocumentFile, Boolean)	Определяет возможность блокировки указанного основного файла документа.
CanRemoveMainFile(Document, DocumentFile)	Определяет возможность удаления указанного основного файла.
CanRemoveMainFiles(Document document, IEnumerable<DocumentFile>)	Определяет возможность удаления указанных основных файлов.
CanUnlockAnyMainFile(Document)	Определяет наличие в документе основных файлов, которые могут быть разблокированы.
CanUnlockMainFile(Document, DocumentFile)	Определяет возможность разблокировки определённого основного файла документа.
CanUnlockMainFile(Document, DocumentFile, LockStatus)	Определяет возможность разблокировки определённого основного файла документа.

Имя	Описание
CanUnlockMainFile(Document, DocumentFile, String)	Определяет возможность разблокировки определённого основного файла документа.
CheckActualSignature(Document, ICollection<CardFieldSetting>, BaseCardSignature)	Определяет актуальность ЭП по отношению к данным документа.
CheckAllMainFilesSigned(Document, BaseCardSignature)	Определяет корректность ЭП, установленной на основных файлах документа.
CheckMainFileSigned(Document, DocumentFile)	Определяет наличие подписи у указанного основного файла документ.
ComputeAttachmentSignature(Document, BaseCardSignature, X509Certificate2)	Подписывает дополнительные файлы документа.
ComputeFieldsSignature(Document, BaseCardSignature, X509Certificate2, ICollection<CardFieldSetting>)	Подписывает атрибуты документа.
ComputeMainFileSignature(Document, BaseCardSignature, X509Certificate2)	Подписывает основные файлы документа.
CreateDocument	Создаёт экземпляр нового документа.
CreateDocument(String)	Создаёт экземпляр нового документа с основным файлом.
CreateDocument(String, KindsCardKind)	Создаёт экземпляр нового документа с указанным видом, с основным файлом.
CreateDocument(String, KindsCardKind, DocumentVersioningType)	Создаёт документ с заданным видом карточки и добавляет основной файл с указанным типов версионирования.
CreateListCards(Document)	Инициализирует списки, используемые документом.
CreateSignatureList	Инициализирует новый экземпляр списка подписей документа.

Имя	Описание
DecodeSignedDocument(String, Document, `SignedCms)`	Расшифровывает сообщение CMS/PKCS 7, которое содержится в файле, расположенном по указанному пути.
DownloadAdditionalFile(Document, DocumentFile)	Выгружает в файловую систему указанный дополнительный файл документа.
DownloadAdditionalFile(Document, DocumentFile, String)	Выгружает дополнительный файл документа в указанный каталог.
DownloadFile(Guid)	Сохраняет файл с указанным идентификатором на файловую систему.
DownloadMainFile(Document, DocumentFile)	Выгружает основной файл документа на диск.
DownloadMainFile(Document, DocumentFile, String)	Выгружает основной файл документа в указанный каталог.
DownloadMainFileVersion(Document, DocumentFile, Guid)	Выгружает определённую версию основного файла документа.
DownloadMainFileVersion(Document, DocumentFile, Guid, String)	Выгружает указанную версию основного файла документа в заданный каталог.
EditMainFile(Document, DocumentFile)	Устанавливает временную блокировку и выгружает файл в файловую систему.
EditMainFile(Document, DocumentFile, String)	Устанавливает временную блокировку и выгружает файл в заданный каталог.
EditMainFileComment(Document, DocumentFile, FileComment, String)	Редактирует комментарий к основному файлу.
Export(Document, DocumentExportTransformSetting)	Экспортирует документ в формате XML.
ExportFromTransform(Document, Stream, DocumentExportTransformSetting)	Экспортирует документ в указанный поток.
ExportToHtml(Document)	Экспортирует документ и возвращает сформированный HTML.

Имя	Описание
ExportToHtmlFile(Document)	Экспортирует документ в формате HTML и возвращает путь к сформированному файлу.
ExportToPDF(String, Document)	Отправляет на печать основные файлы.
ExportToPDF(String, Document, DocumentFile)	Отправляет на печать указанный файл документа.
ExportToPDF(String, Document, IEnumerable<DocumentFile>)	Отправляет на печать коллекцию файлов документа.
ExportToZip(String, Dictionary<String, Stream>)	Архивирует файлы, представленные в виде потока, в формате zip.
GetAdditionalFile(DocumentFile)	Возвращает карточку файл с версиями из дополнительного файла документа.
GetAdditionalFiles(Document)	Получает все дополнительные файлы заданного документа.
GetFiles(Document)	Получает все файлы документа.
GetFileTypeLocalizedName(DocumentFile)	Возвращает локализованное название типа файла документа (основной, дополнительный).
GetKindSettings(KindsCardKind)	Возвращает настройки заданного вида карточки Документ.
GetMainFile(DocumentFile)	Возвращает карточку файл с версиями из основного файла документа.
GetMainFileChanged(DocumentFile)	Возвращает признак того, что в указанном файле есть несохранённые изменения.
GetMainFileLockStatus(DocumentFile, LockStatus)	Возвращает статус блокировки основного файла.
GetMainFileLockStatus(DocumentFile, LockStatus, String)	Возвращает статус блокировки основного файла и учетную запись заблокировавшего сотрудника.
GetMainFiles(Document)	Возвращает все основные файлы документа.

Имя	Описание
`GetMainFileVersionNumber(DocumentFile, Guid)`	Возвращает строковое представление указанной версии основного файла.
`HasAnyMainFileChanged(Document)`	Определяет наличие основного файла карточки с несохранёнными изменениями.
`HasEditedMainFile(DocumentFile)`	Определяет наличие редактируемого основного файла карточки.
`HasEditedMainFile(DocumentFile, LockStatus)`	Определяет наличие редактируемого основного файла карточки. Текущий статус блокировки файла указывается принудительно.
`HasFreeMainFile(DocumentFile)`	Возвращает `true`, если указанный основной файл свободен.
`HasFreeMainFiles(IEnumerable<DocumentFile>)`	Возвращает true, если все основное файлы не заблокированы.
`HasLockedMainFile(DocumentFile)`	Возвращает `true`, если основной файл выгружен для редактирования, либо заблокирован владельцем сессии.
`HasLockedMainFile(DocumentFile, LockStatus)`	Возвращает true, если основной файл выгружен для редактирования, либо заблокирован владельцем сессии. Текущий статус блокировки файла указывается принудительно.
`HasMainFile(DocumentFile)`	Определяет, что указанный файл документа является основным файлом.
`IsDocumentSignaturePartAttachments(BaseCardSignaturePart)`	Возвращает `true`, если переданной частью ЭП подписан дополнительный файл.
`IsDocumentSignaturePartFields(BaseCardSignaturePart)`	Возвращает `true`, если переданной частью ЭП подписаны поля карточки.
`IsDocumentSignaturePartMainFile(BaseCardSignaturePart)`	Возвращает `true`, если переданной частью ЭП подписан основной файл.

Имя	Описание
IsEncodedSignedDocument(String)	Определяет соответствие указанного файла криптографическому сообщению CMS/PKCS 7.
IsFileNotEmpty(Document, DocumentFile)	Возвращает true, если файл документа не нулевой.
LockMainFile(Document, DocumentFile)	Выгружает файл карточки и ставит постоянную блокировку.
LockMainFile(Document, DocumentFile, String)	Выгружает файл из документа на диск с установкой постоянной блокировки.
MakeCurrentMainFileVersion(Document, DocumentFile, Guid)	Задаёт последнюю версию файла.
MakeLockedMainFileForceChanged(Document, DocumentFile)	Устанавливает признака наличия изменений для выгруженного основного файла документа.
PrepareToVerifySignature(Document, IDictionary<BaseCardSignaturePart, Stream>, IEnumerable<BaseCardSignaturePart>, Boolean, Boolean, ICollection<CardFieldSetting>)	Заполняет коллекцию IDictionary<BaseCardSignaturePart, Stream>, где ключ соответствует части ЭП, а значение коллекции — сопоставленные частям ЭП значения полей, основных или дополнительных файлов.
PurgeFileCache(Document)	Очищает кэш файлов документ.
RemoveAdditionalFile(Document, DocumentFile)	Удаляет заданный дополнительный файл из документа.
RemoveMainFile(Document, DocumentFile)	Удаляет заданный основной файл из документа.
RemoveMainFile(Document, DocumentFile)	Удаляет заданный основной файл из документа. Если файл заблокирован, то блокировка отменяется.
RemoveMainFileComments(Document, DocumentFile, FileCommentCollection, IEnumerable<FileComment>)	Удаляет комментарии к файлу документа.

Имя	Описание
`RemoveMainFileVersion(Document, DocumentFile, Guid)`	Удаляет указанную версию файла документа.
`RemoveMainFileVersionComments(Document, DocumentFile, Guid, IEnumerable<FileComment>)`	Удаляет комментарии к заданной версии файла.
`RenameAdditionalFile(Document, DocumentFile, String)`	Переименовывает дополнительный файл документа.
`RenameMainFile(Document, DocumentFile, String)`	Переименовывает основной файл документа.
`ReplaceMainFile(Document, DocumentFile, String, `SignedCms)`	Заменяет основной файл документа. Новый файл должен иметь название аналогичное старому.
`SignatureInfo GetPartSignatureInfo(BaseCardSignaturePart signaturePart)`	Возвращает информацию о части подписи документа.
`SyncPropertiesFromFile(Document, IList<IDocumentPropertySetting>, String)`	Загружает свойства документа из файла.
`SyncPropertiesFromFiles(Document, IList<IDocumentPropertySetting>, IEnumerable<string>)`	Загружает свойства документа из нескольких файлов.
`SyncPropertiesToFile(Document, DocumentFile, IList<IDocumentPropertySetting>)`	Выполняет синхронизацию значений полей карточки *Документ* со свойствами основного файла данного документа. Передача осуществляется в сторону вложенного файла.
`UnlockMainFile(Document, DocumentFile)`	Разблокирует указанный файл. При наличии изменений в файле документа, номер версии будет увеличен.
`UnlockMainFile(Document, DocumentFile, String)`	Разблокирует файл, расположенный по заданному пути. В случае наличия изменений в файле документа, увеличивает номер версии.

Имя	Описание
UnlockMainFile(Document, DocumentFile, Int32, String, Boolean)	Разблокирует файл. В случае наличия изменений в файле документа, увеличивает номер версии. К файлу или версии файла добавляется комментарий.
UnlockMainFile(Document, DocumentFile, Int32, String, Boolean, String)	Разблокирует файл, расположенный по заданному пути. В случае наличия изменений в файле документа, увеличивает номер версии. К файлу или версии файла добавляется комментарий.
VerifySignature(Document, BaseCardSignature, X509Certificate2, Boolean, Boolean, ICollection<CardFieldSetting>)	Выполняет проверку ЭП, установленной на документе, файлах документа или полях, и возвращает результат проверки.

События

Имя	Описание
AdditionalFileAdded	Событие срабатывает после добавлении дополнительного файла.
AdditionalFileRemoved	Событие срабатывает после удаления дополнительного файла.
MainFileAdded	Событие срабатывает после добавлении основного файла.
MainFileCancelLocked	Событие срабатывает после отмены блокировки основного файла.
MainFileCurrentVersionChanged	Событие срабатывает после изменения текущей версии основного файла.
MainFileLocked	Событие срабатывает после блокировки основного файла.
MainFileRemoved	Событие срабатывает после удаления основного файла.

Имя	Описание
MainFileUnlocked	Событие срабатывает после разблокировки основного файла.
MainFileUnlocking	Вызывается до разблокировки основного файла.
SignatureAdded	Событие срабатывает после подписания документ ЭП.
SynchronizationFromFileCompleted	Событие срабатывает после синхронизации данных карточки со свойствами основного файла.

IDocumentService.AddAdditionalFile — метод (Document, String)

Добавляет в документ дополнительный файл.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
DocumentFile AddAdditionalFile(Document document, string filePath)
```

Параметры

document

Тип: Document

Документ, в который добавляется файл

filePath

Тип: System.String

Путь к добавляемому файлу

Возвращаемое значение

Тип: DocumentFile

Файл документа

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или filePath.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для добавления дополнительного файла.
System.InvalidOperationException	Ошибка возвращается в случае, если невозможно загрузить дополнительный файл, в.т.ч. по причине наличия в документе дополнительного файла с подобным именем.

Примеры

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

documentService.AddAdditionalFile(document, @"Z:\Служебная записка на ремонт
компьютера.docx"); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ Добавление дополнительного файла.

IDocumentService.AddAdditionalFile — метод (Document, VersionedFileCard)

Добавляет в документ дополнительный файл.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
DocumentFile AddAdditionalFile(Document document, VersionedFileCard versionedFileCard)
```

Параметры

document

Тип: Document

Документ, в который добавляется файл

versionedFileCard

Тип: VersionedFileCard

Карточка версионного файла

Возвращаемое значение

Тип: DocumentFile

Файл документа

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или versionedFileCard.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для добавления дополнительного файла.
System.InvalidOperationException	Ошибка возвращается в случае, если невозможно загрузить дополнительный файл, в.т.ч. по причине наличия в документе дополнительного файла с подобным именем.

IDocumentService.AddAdditionalFiles — метод (Document, IEnumerable<string>)

Добавляет дополнительные файлы в документ.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<DocumentFile> AddAdditionalFiles(Document document, IEnumerable<string>
filePaths)
```

Параметры

document

Тип: Document

Документ, в который будут добавлены файлы

filePaths

Тип: System.Collections.Generic.IEnumerable<System.String>

Расположение файлов

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<DocumentFile>

Список файлов документа

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или filePaths.
System.MethodAccessException	Ошибка возвращается в случае, если текущее состояние документа не позволяет выполнить операцию добавления.
System.InvalidOperationException	Ошибка возникает при наличии в документе файла с аналогичным добавляемому названием, либо в случае превышения размера вложений.

Заметки

Перед непосредственным добавлением файлов будет выполнена проверка на возможность добавления, в результате чего может быть возвращена ошибка.

Примеры

В приведенном примере к документу добавляются два дополнительных файла с файловой системы

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

string[] filePaths = new string[] { @"X:\Document.docx", @"X:\Document2.docx" };
documentService.AddAdditionalFiles(document, filePaths); ④

objectContext.SaveObject(document); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ Добавление дополнительных файлов.

⑤ Сохранение.

IDocumentService.AddMainFile — метод (Document, String)

Добавляет в документ основной файл.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
DocumentFile AddMainFile(Document document, string filePath)
```

Параметры

document

Тип: Document

Документ, в который добавляется файл

filePath

Тип: System.String

Путь к добавляемому файлу

Возвращаемое значение

Тип: DocumentFile

Файл документа

Примеры

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

documentService.AddMainFile(document, @"Z:\Инструкция пожарной безопасности.docx"); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ Добавление основного файла.

IDocumentService.AddMainFileComment — метод (Document, DocumentFile, String)

Добавляет комментарий к основному файлу документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool AddMainFileComment(Document document, DocumentFile file, string comment)
```

Параметры

document

Тип: `Document`

Документ, содержащий файл `file`

file

Тип: `DocumentFile`

Объектная модель файла документа

comment

Тип: `System.String`

Комментарий к файлу

Возвращаемое значение

Тип: `System.Boolean`

`true` — комментарий был успешно добавлен, иначе — `false`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `document`, `file` или `comment`.

Примеры

В приведенном примере комментарий добавляется к первому основному файлу

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

DocumentFile file = document.Files.First(t => t.FileType = DocumentFileType.Main); ④

documentService.AddMainFileComment(document, file, "Данный файл был проверен"); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ Получение первого основного файла.

⑤ Добавление комментария.

IDocumentService.AddMainFiles — метод (Document document, IEnumerable<String>)

Добавляет несколько основных файлов в документ.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<DocumentFile> AddMainFiles(Document document, IEnumerable<string> filePaths)
```

Параметры

document

Тип: Document

Документ, в который будут добавлены файлы

filePaths

Тип: System.Collections.Generic.IEnumerable<System.String>

Расположение основных файлов

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<DocumentFile>

Список файлов документа

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или filePaths.

Исключение	Условие
System.MethodAccessException	Ошибка возвращается в случае, если текущее состояние документа не позволяет выполнить операцию добавления, либо у сотрудника недостаточно прав для выполнения операции.
System.InvalidOperationException	Ошибка возникает при наличии в документе файла с аналогичным добавляемому названием, либо в случае, если добавление файла недопустимо.

Заметки

Перед непосредственным добавлением файлов будет выполнена проверка на возможность добавления, в результате чего может быть возвращена ошибка.

Примеры

В приведенном примере к документу добавляются два основных файла с файловой системы

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

string[] filePaths = new string[] { @"X:\Document.docx", @"X:\Document2.docx" };
documentService.AddMainFiles(document, filePaths); ④

objectContext.SaveObject(document); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ Добавление основных файлов.

⑤ Сохранение.

IDocumentService.AddMainFilesAndSignatures — метод (Document, IEnumerable<string>, bool, out IEnumerable<DocumentFile>, out IEnumerable<BaseCardSignature>)

Импортирует файлы с подписью в карточку документа

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void AddMainFilesAndSignatures(Document document, IEnumerable<string> filePaths, bool verifySignatures, out IEnumerable<DocumentFile> files, out IEnumerable<BaseCardSignature> signatures)
```

Параметры

document

Тип: Document

Подписываемый документ

filePaths

Тип: System.String

Находит импортируемые файлы по расширению

verifySignatures

Тип: System.Boolean

При true

files

Добавляет найденные файлы как основные

signatures

Импортирует файлы подписи

IDocumentService.AddMainFileVersionComment — метод ([Тип параметра перегрузки])

Добавляет комментарий к указанной версии основного файла документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool AddMainFileVersionComment(Document document, DocumentFile file, Guid versionId,
string comment)
```

Параметры

document

Тип: Document

Документ, содержащий основной файл file

file

Тип: DocumentFile

Объектная модель файла документа

versionId

Тип: System.Guid

Идентификатор версии, к которой добавляется комментарий

comment

Тип: System.String

Комментарий к версии основного файла

Возвращаемое значение

Тип: System.Boolean

true — комментарий добавлен успешно, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document, file или comment.

Примеры

В приведенном примере добавляется комментарий к первой версии основного файла документа

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②
IVersionedFileCardService versionedFileCardService = objectContext.GetService
<IVersionedFileCardService>();

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

DocumentFile file = document.Files.First(t => t.FileType = DocumentFileType.Main); ④

VersionedFileCard versionedFileCard = versionedFileCardService.OpenCard(file.FileId);

FileVersion version = versionedFileCard.Versions.First(); ⑤

documentService.AddMainFileVersionComment(document, file, version.VersionId, "Первая
версия основного файла"); ⑥
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ Получение первого основного файла.

⑤ Получение первой версии файла.

⑥ Добавление комментария.

IDocumentService.AddSignature — метод (Document, X509Certificate2, Boolean, ICollection<CardFieldSetting>)

Добавляет новую подпись в список подписей карточки и устанавливает ЭП на поля, основные и дополнительные файлы документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardSignature AddSignature(Document document, X509Certificate2 cert, bool
signAttachments, ICollection<CardFieldSetting> fields)
```

Параметры

document

Тип: `Document`

Документ

cert

Тип: `System.Security.Cryptography.X509Certificates.X509Certificate2`

Сертификат, которым будет выполнено подписание

signAttachments

Тип: `System.Boolean`

`true` — подписывать дополнительные файлы, иначе — `false`

fields

Тип: `System.Collections.Generic.ICollection<CardFieldSetting>`

Коллекция подписываемых полей типа `CardFieldSetting`, определённых для карточки в Справочнике видов карточек

Возвращаемое значение

Тип: `BaseCardSignature`

Подпись

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `document`.

Заметки

Сертификат сотрудника может быть получен из карточки Настроек пользователя. Данный метод инициализирует новую подпись в виде объекта типа `BaseCardSignature`, которую использует при непосредственном подписании данных карточки *Документ*.

IDocumentService.AddSignature — метод (Document, X509Certificate2, SignatureLabel, StaffEmployee, String, Boolean, ICollection<CardFieldSetting>)

Данный метод предназначен для выполнения операции подписания в документе.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardSignature AddSignature(Document document, X509Certificate2 cert, bool
signAttachments, ICollection<CardFieldSetting> fields, bool signDocument, StatesOperation
operation, Guid signatureType)
```

Параметры

document

Тип: Document

Документ

cert

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат сотрудника

signAttachments

Тип: System.Boolean

Признак подписания дополнительных файлов документа: true — подписывать, иначе — false

fields

Тип: System.Collections.Generic.ICollection<CardFieldSetting>

Поля документа, требующие подписания

signDocument

Тип: System.Boolean

Признак подписания основных файлов документа: true — подписывать, иначе — false

operation

Тип: StatesOperation

Подписываемая операция

signatureType

Тип: `System.Guid`

Идентификатор вида подписи из *Справочника видов карточек*

Возвращаемое значение

Тип: `BaseCardSignature`

Созданная подпись

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `document` или `operation`.

Заметки

Чтобы использовать возможность выполнять подписание операций, необходимо добавить соответствующую настройку в *Справочнике видов карточек*. Она регистрируется во вкладке **Подпись** в разделе **Подписание операций**.

Примеры

Ниже приведён пример использования метода `AddSignature` при подписании операции перехода "Согласован" для карточки типа *Документ*

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>();
IStateService stateService = objectContext.GetService<IStateService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

KindsCardKind kind = document.SystemInfo.CardKind;
DocumentSetting documentSetting = documentService.GetKindSettings(kind); ④

StatesOperation statesOperation = stateService.GetOperations(kind).First(t => t
.DefaultName.Equals("Is approved")); ⑤

DocumentOperationSignatureSetting operationSignatureSetting = documentSetting
.OperationSignatures.First(t => t.SignedOperations.Contains(statesOperation)); ⑥
```

```
X509Certificate2 certificate = GetCertificate(); ⑦

BaseCardSignature signature = documentService.AddSignature(document, certificate,
operationSignatureSetting.SignAttachments,
operationSignatureSetting.Fields, operationSignatureSetting.SignDocument,
statesOperation, new Guid(operationSignatureSetting.Name)); ⑧
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение документа.

④ Получение вида и настроек данного вида.

⑤ Получение операции перехода "Согласован" из Конструктора состояний.

⑥ Получение первой настройки подписания для операции перехода "Согласован".

⑦ Получение сертификата сотрудника, выполняющего подписание.

⑧ Подписание операции. Используются параметры подписания, определённые в Справочнике видов карточек.

Для получения сертификата сотрудника используется следующий метод:

```
static X509Certificate2 GetCertificate()
{
 X509Certificate2 certificate = new X509Certificate2();
 X509Store store = new X509Store(StoreName.My, StoreLocation.CurrentUser);
 store.Open(OpenFlags.OpenExistingOnly);
 X509Certificate2Enumerator enumerator = store.Certificates.GetEnumerator();
 while (enumerator.MoveNext())
 {
  if (enumerator.Current.HasPrivateKey)
  {
   return enumerator.Current;
  }
 }
 return null;
}
```

IDocumentService.AddSignatures — метод (Document, SignedCms)

Данный метод добавляет коллекцию новых подписей в список подписей

карточки, и устанавливает ЭП на поля, основные и дополнительные файлы документа. Позволяет использовать для подписания криптографическое сообщение типа SignedCms.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<BaseCardSignature> AddSignatures(Document document, System.Security
.Cryptography.Pkcs.SignedCms signedCms)
```

Параметры

document

Тип: Document

Подписываемый документ

signedCms

Тип: SignedCms

Сообщение CMS/PKCS 7

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<BaseCardSignature>

Список подписей

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или signedCms.

Заметки

Данный метод может быть использован для импорта подписанного файла, сохранённого в виде сообщения CMS/PKCS 7, в карточку. Docsvision экспортирует подписанные файлы в один файл с расширением .p7s, либо в два файла: экспортируемый файл и отдельно сообщение CMS/PKCS 7 в виде файла с

расширением .p7s. Для извлечения исходного файла из сообщения предназначен метод IDocumentService.DecodeSignedDocument.

IDocumentService.AddSignatures — метод (Document, DocumentFile, SignedCms)

Добавляет коллекцию новых подписей в список подписей карточки. Определяет, к какому файлу будет добавлена подпись. Позволяет использовать для подписания криптографическое сообщение типа SignedCms.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<BaseCardSignature> AddSignatures(Document document, DocumentFile mainFile,
SignedCms signedCms);
```

Параметры

document

Тип: Document

Подписываемый документ

mainFile

DocumentFile`

Определяет, к какому файлу будет добавлена подпись

signedCms

Тип: SignedCms

Сообщение CMS/PKCS 7

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<BaseCardSignature>

Список подписей

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или signedCms.

Заметки

Данный метод может быть использован для импорта подписанного файла, сохранённого в виде сообщения CMS/PKCS 7, в карточку. Docsvision экспортирует подписанные файлы в один файл с расширением .p7s, либо в два файла: экспортируемый файл и отдельно сообщение CMS/PKCS 7 в виде файла с расширением .p7s. Для извлечения исходного файла из сообщения предназначен метод IDocumentService.DecodeSignedDocument.

IDocumentService.AddSignatures — метод (Document, DocumentFile, SignedCms, bool)

Добавляет коллекцию новых подписей в список подписей карточки. Определяет, к какому файлу будет добавлена подпись. Позволяет использовать для подписания криптографическое сообщение типа SignedCms. Позволяет проверять срок действия добавляемых подписей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<BaseCardSignature> AddSignatures(Document document, DocumentFile mainFile,
SignedCms signedCms, bool verifySignatures);
```

Параметры

document

Тип: Document

Подписываемый документ

mainFile

DocumentFile`

Определяет, к какому файлу будет добавлена подпись

signedCms

Тип: `SignedCms`

Сообщение CMS/PKCS 7

verifySignatures

Тип: `System.Boolean`

При `true`, проверяется срок действия подписей

Возвращаемое значение

Тип: `System.Collections.Generic.IEnumerable<BaseCardSignature>`

Список подписей

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `document` или `signedCms`.

Заметки

Данный метод может быть использован для импорта подписанного файла, сохранённого в виде сообщения CMS/PKCS 7, в карточку. Docsvision экспортирует подписанные файлы в один файл с расширением `.p7s`, либо в два файла: экспортируемый файл и отдельно сообщение CMS/PKCS 7 в виде файла с расширением `.p7s`. Для извлечения исходного файла из сообщения предназначен метод `IDocumentService.DecodeSignedDocument`.

IDocumentService.AddSignatures — метод (Document, SignedCms, bool)

Добавляет коллекцию новых подписей в список подписей карточки. Устанавливает ЭП на поля, основные и дополнительные файлы документа. Позволяет использовать для подписания криптографическое сообщение типа `SignedCms`. Позволяет проверять срок действия добавляемых подписей.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
IEnumerable<BaseCardSignature> AddSignatures(Document document, SignedCms signedCms, bool verifySignatures)
```

Параметры

document

Тип: Document

Подписываемый документ

signedCms

Тип: SignedCms

Сообщение CMS/PKCS 7

verifySignatures

Тип: System.Boolean

При true, проверяется срок действия подписей

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<BaseCardSignature>

Список подписей

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или signedCms.

Заметки

Данный метод может быть использован для импорта подписанного файла, сохранённого в виде сообщения CMS/PKCS 7, в карточку. Docsvision экспортирует подписанные файлы в один файл с расширением .p7s, либо в два файла: экспортируемый файл и отдельно сообщение CMS/PKCS 7 в виде файла с расширением .p7s. Для извлечения исходного файла из сообщения предназначен метод IDocumentService.DecodeSignedDocument.

IDocumentService.ApplyKindSettings — метод (Document, DocumentSetting)

Устанавливает у документа определённые настройки вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ApplyKindSettings(Document document, DocumentSetting documentSetting)
```

Параметры

document

Тип: Document

Документ, к которому применяются настройки вида карточки

documentSetting

Тип: DocumentSetting

Настройки документа

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или documentSetting.
System.Exception	Ошибка возвращается в случае, если невозможно прикрепить основной файл, если его добавление определено настройками documentSetting.

IDocumentService.ApplyKindSettings — метод (Document, KindsCardKind)

Устанавливает у документа настройки указанного вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ApplyKindSettings(Document document, KindsCardKind kind)
```

Параметры

document

Тип: Document

Документ, к которому применяются настройки вида карточки

kind

Тип: KindsCardKind

Вид карточки, из которого будут получены настройки

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или kind.

Заметки

Применение настроек вида карточки включает прикрепление к документу основных и дополнительных файлов, заданных в настройках указанного вида kind.

IDocumentService.AttachMainFile — метод (Document, String)

Загружает в документ основной файл.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
DocumentFile AttachMainFile(Document document, string fileName)
```

Параметры

document

Тип: Document

Документ, к которому добавляется файл

fileName

Тип: System.String

Путь к загружаемому файлу

Возвращаемое значение

Тип: DocumentFile

Объектная модель файла документа

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document.
System.IO.FileNotFoundException	Ошибка возвращается при отсутствии файла, указанного в fileName.
System.InvalidOperationException	Ошибка возвращается, если сообщение CMS/PKCS 7, полученное из загружаемого файла, не содержит подписанного файла. Также данная ошибка возникает при попытке добавления файла, существующего в документе, либо при прочих ошибках, связанных с добавлением файла.
System.MethodAccessException	Ошибка возвращается в случае, если у сотрудника недостаточно прав для добавления основного файла.

Заметки

Данный метод, в отличие от метода AddMainFile, позволяет корректно загружать подписанные файлы.

IDocumentService.AttachMainFile — метод (Document, VersionedFileCard)

Добавляет в документ карточку версионного файла в качестве основного файла документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
DocumentFile AttachMainFile(Document document, VersionedFileCard versionedFileCard)
```

Параметры

document

Тип: Document

Документ

versionedFileCard

Тип: VersionedFileCard

Карточка файла с версиями

Возвращаемое значение

Тип: DocumentFile

Файл документа

Исключения

Исключение	Условие
System.Exception	Ошибка будет возвращена, если по каким либо причинам добавление основного файла невозможно.

Заметки

Данный метод, в отличие от метода AddMainFile, позволяет корректно загружать подписанные файлы.

Примеры

Ниже приведён пример использования метода AttachMainFile при добавлении существующей карточки файла с версиями в документ

```
①
```

```
IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

VersionedFileCard versionedFileCard = (VersionedFileCard)userSession.CardManager.GetCard
(new Guid("00000000-0000-0000-0000-000000000001")); ④

documentService.AttachMainFile(document, versionedFileCard);
objectContext.SaveObject(document); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ Получение карточки файла с версиями.

⑤ Добавление файла.

IDocumentService.AttachMainFile — метод (Document, string, SignedCms, bool)

Добавляет в документ подпись в качестве основного файла документа, проверяет срок действия подписи.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
DocumentFile AttachMainFile(Document document, string fileName, SignedCms signedCms, bool
verifySignatures)
```

Параметры

document

Тип: Document

Документ

fileName

Тип: VersionedFileCard

Карточка файла с подписями

signedCms

Тип: SignedCms

Сообщение CMS/PKCS 7

verifySignatures

Тип: System.Boolean

При true, проверяется срок действия подписей

Возвращаемое значение

Тип: DocumentFile

Файл документа

Исключения

Исключение	Условие
System.Exception	Ошибка будет возвращена, если по каким либо причинам добавление основного файла невозможно.

Заметки

Данный метод, в отличие от метода AddMainFile, позволяет корректно загружать подписанные файлы.

IDocumentService.CanAttachMainFile — метод (Document, VersionedFileCard)

Определяет возможность добавления основного файла в документ.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanAttachMainFile(Document document, VersionedFileCard versionedFileCard)
```

Параметры

document

Тип: Document

Документ, для которого определяется возможность добавления основного файла

versionedFileCard

Тип: `VersionedFileCard`

Карточка основного файла

Возвращаемое значение

Тип: `System.Boolean`

`true` — указанный файл может быть добавлен как основной, иначе — `false`

IDocumentService.CanAttachMainFile — метод (Document, VersionedFileCard, SignedCms)

Определяет возможность добавления основного файла в документ.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
bool CanAttachMainFile(Document document, string fileName, SignedCms signedCms)
```

Параметры

document

Тип: `Document`

Документ, для которого определяется возможность добавления основного файла

versionedFileCard

Тип: `VersionedFileCard`

Карточка основного файла

signedCms

Тип: `SignedCms`

Сообщение CMS/PKCS 7

Возвращаемое значение

Тип: System.Boolean

true — указанный файл может быть добавлен как основной, иначе — false

IDocumentService.CanCancelLockAnyMainFile — метод (Document)

Определяет возможность снятия блокировки с основных файлов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanCancelLockAnyMainFile(Document document)
```

Параметры

document

Тип: Document

Документ

Возвращаемое значение

Тип: System.Boolean

true — если есть основные файлы, с которых можно снять блокировку, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document.

IDocumentService.CanCancelLockMainFile — метод (DocumentFile)

Определяет возможность снятия блокировки с основного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanCancelLockMainFile(DocumentFile file)
```

Параметры

file

Тип: DocumentFile

Файл документа

Возвращаемое значение

Тип: System.Boolean

true — блокировка может быть снята, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр file.

IDocumentService.CancelLockMainFile — метод (Document, DocumentFile)

Снимает блокировку с основного файла документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void CancelLockMainFile(Document document, DocumentFile file)
```

Параметры

document

Тип: Document

Документ

file

Тип: DocumentFile

Файл документа

IDocumentService.CancelLockMainFileIfNeeded — метод (Document, DocumentFile)

Снимает блокировку с основного файла документа, если это необходимо.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void CancelLockMainFile(Document document, DocumentFile file)
```

Параметры

document

Тип: Document

Документ

file

Тип: DocumentFile

Файл документа

IDocumentService.CanEditAnyMainFile — метод (Document)

Определяет наличие в документе основного файла, который может быть открыт для редактирования.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanEditAnyMainFile(Document document)
```

Параметры

document

Тип: Document

Документ

Возвращаемое значение

Тип: System.Boolean

true — если в документе есть основные файлы, которые могут быть открыты для изменения, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document.

IDocumentService.CanEditMainFile — метод (Document, DocumentFile, Boolean)

Определяет возможность выгрузки указанного основного файла для редактирования.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanEditMainFile(Document document, DocumentFile file, bool forceUnlock)
```

Параметры

document

Тип: Document

Документ

file

Тип: DocumentFile

Файл документа

forceUnlock

Тип: System.Boolean

true — принудительная разблокировка

Возвращаемое значение

Тип: System.Boolean

true — блокировка допустима, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или file.

IDocumentService.CanLockAnyMainFile — метод (Document)

Определяет наличие по меньшей мере одного основного файла документа, который может быть заблокирован.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanLockAnyMainFile(Document document)
```

Параметры

document

Тип: Document

Документ

Возвращаемое значение

Тип: System.Boolean

true — есть основные файлы с возможностью установки блокировки, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document.

IDocumentService.CanLockMainFile — метод (Document, DocumentFile, Boolean)

Определяет возможность блокировки указанного основного файла документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanLockMainFile(Document document, DocumentFile file, bool forceUnlock)
```

Параметры

document

Тип: Document

Документ

file

Тип: DocumentFile

Файл документа

forceUnlock

Тип: System.Boolean

true — принудительная разблокировка

Возвращаемое значение

Тип: System.Boolean

true — блокировка файла возможна, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или file.
System.InvalidOperationException	Ошибка возвращается, если карточка версионного файла была удалена или не существует.

IDocumentService.CanRemoveMainFile — метод (Document, DocumentFile)

Определяет возможность удаления указанного основного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanRemoveMainFile(Document document, DocumentFile file)
```

Параметры

document

Тип: Document

Документ

file

Тип: DocumentFile

Файл документа

Возвращаемое значение

Тип: System.Boolean

true — файл может быть удалён, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или file.

IDocumentService.CanRemoveMainFiles — метод (Document document, IEnumerable<DocumentFile>)

Определяет возможность удаления указанных основных файлов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanRemoveMainFiles(Document document, IEnumerable<DocumentFile> files)
```

Параметры

document

Тип: Document

Документ

files

Тип: System.Collections.Generic.IEnumerable<DocumentFile>

Список удаляемых файлов документа типа DocumentFile

Возвращаемое значение

Тип: System.Boolean

true — все файлы могут быть удалены, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр files.

IDocumentService.CanUnlockAnyMainFile — метод (Document)

Определяет наличие в документе основных файлов, которые могут быть разблокированы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanUnlockAnyMainFile(Document document)
```

Параметры

document

Тип: Document

Документ

Возвращаемое значение

Тип: System.Boolean

true — если есть файлы, которые могут быть разблокированы, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document.

IDocumentService.CanUnlockMainFile — метод (Document, DocumentFile)

Определяет возможность разблокировки определённого основного файла документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanUnlockMainFile(Document document, DocumentFile file)
```

Параметры

document

Тип: Document

Документ

file

Тип: DocumentFile

Файл документа

Возвращаемое значение

Тип: System.Boolean

true — файл может быть разблокирован, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или file.
System.InvalidOperationException	Ошибка возвращается в случае, если невозможно сопоставить файл версионной карточки, указанному файлу документа — file.

IDocumentService.CanUnlockMainFile — метод (Document, DocumentFile, LockStatus)

Определяет возможность разблокировки определённого основного файла документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanUnlockMainFile(Document document, DocumentFile file, LockStatus lockStatus)
```

Параметры

document

Тип: Document

Документ

file

Тип: DocumentFile

Файл документа

lockStatus

Тип: LockStatus

Собственное значение состояние блокировки файла документа, если настоящее состояние блокировки не должно учитываться

Возвращаемое значение

Тип: System.Boolean

`true` — файл может быть разблокирован, иначе — `false`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `document` или `file`.
`System.InvalidOperationException`	Ошибка возвращается в случае, если невозможно сопоставить файл версионной карточки, указанному файлу документа — `file`.

IDocumentService.CanUnlockMainFile — метод (Document, DocumentFile, String)

Определяет возможность разблокировки определённого основного файла документа.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
bool CanUnlockMainFile(Document document, DocumentFile file, out string failReason)
```

Параметры

document

Тип: `Document`

Документ

file

Тип: `DocumentFile`

Файл документа

failReason

Тип: `System.String`

Причина, по которой невозможно разблокировать основной файла

Возвращаемое значение

Тип: System.Boolean

true — файл может быть разблокирован, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или file.
System.InvalidOperationException	Ошибка возвращается в случае, если невозможно сопоставить файл версионной карточки, указанному файлу документа — file.

IDocumentService.CheckActualSignature — метод (Document, ICollection<CardFieldSetting>, BaseCardSignature)

Определяет актуальность ЭП по отношению к данным документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CheckActualSignature(Document document, ICollection<CardFieldSetting> fields,
BaseCardSignature signature)
```

Параметры

document

Тип: Document

Документ, проверяемые тестируемые данные

fields

Тип: System.Collections.Generic.ICollection<CardFieldSetting>

Коллекция полей типа CardFieldSetting, у которых должна быть проверена подпись. Список полей определён в Справочнике видов карточек

signature

Тип: BaseCardSignature

ЭП, для которой производится проверка

Возвращаемое значение

Тип: System.Boolean

true — подпись актуальна, иначе — false

Заметки

Данный метод проверяет части указанной подписи на соответствие текущим данным (версия основного файла) документа, без проверки подлинности самой ЭП.

IDocumentService.CheckAllMainFilesSigned — метод (Document, BaseCardSignature)

Определяет корректность ЭП, установленной на основных файлах документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CheckAllMainFilesSigned(Document document, BaseCardSignature signature)
```

Параметры

document

Тип: Document

Документ, для которого проводится проверка

signature

Тип: BaseCardSignature

Подпись карточки, содержащая актуальные ЭП

Возвращаемое значение

Тип: System.Boolean

true — все подписи корректны, либо документ не содержит основных файлов,

иначе — `false`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `document` или `signature`.

Примеры

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

BaseCardSignature lastSignature = document.MainInfo.SignatureList.Signatures.Last(); ④

bool result = documentService.CheckAllMainFilesSigned(document, lastSignature); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса для работы с документами.

③ Получение документа.

④ Получение последней подписи.

⑤ Выполнение проверки.

IDocumentService.CheckMainFileSigned — метод (Document, DocumentFile)

Определяет наличие подписи у указанного основного файла документ.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
bool CheckMainFileSigned(Document document, DocumentFile mainFile)
```

Параметры

document

Тип: `Document`

Документ

mainFile

Тип: `DocumentFile`

Файл документа

Возвращаемое значение

Тип: `System.Boolean`

`true` — файл подписан, иначе — `false`

Примеры

Далее приведён пример использования метода `CheckMainFileSigned` при определении наличия подписи у первого основного файла документа.

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

DocumentFile documentFile = document.Files.First(t => t.FileType = DocumentFileType.
Main); ④

bool result = documentService.CheckMainFileSigned(document, documentFile); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ Получение первого основного файла.

⑤ Проверка подписи.

IDocumentService.ComputeAttachmentSignature — метод (Document, BaseCardSignature, X509Certificate2)

Подписывает дополнительные файлы документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ComputeAttachmentSignature(Document document, BaseCardSignature signature,
X509Certificate2 cert)
```

Параметры

document

Тип: Document

Документ

signature

Тип: BaseCardSignature

ЭП

cert

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат сотрудника

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document.

IDocumentService.ComputeFieldsSignature — метод (Document, BaseCardSignature, X509Certificate2, ICollection<CardFieldSetting>)

Подписывает атрибуты документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ComputeFieldsSignature(Document document, BaseCardSignature signature,
```

```
X509Certificate2 cert, ICollection<CardFieldSetting> properties)
```

Параметры

document

Тип: Document

Документ

signature

Тип: BaseCardSignature

ЭП

cert

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат сотрудника

properties

Тип: System.Collections.ICollection<CardFieldSetting>

Список подписываемых атрибутов (полей карточки) типа CardFieldSetting

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document.

IDocumentService.ComputeMainFileSignature — метод (Document, BaseCardSignature, X509Certificate2)

Подписывает основные файлы документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ComputeMainFileSignature(Document document, BaseCardSignature signature,
X509Certificate2 cert)
```

Параметры

document

Тип: Document

Документ

signature

Тип: BaseCardSignature

ЭП

cert

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат сотрудника

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document.

IDocumentService.CreateDocument — метод

Создаёт экземпляр нового документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
Document CreateDocument()
```

Возвращаемое значение

Тип: Document

Карточка типа "Документ"

Заметки

Для созданного экземпляра карточки будут использованы настройки корневого вида карточки.

Примеры

Ниже приведён пример создания карточки типа Документ и изменение её описания.

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = documentService.CreateDocument(); ③

document.Description = "Документ созданный из кода"; ④

objectContext.SaveObject<Document>(document); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса для работы с документами.

③ Создание документа.

④ Изменение описания.

⑤ Сохранение изменений в контексте.

IDocumentService.CreateDocument — метод (String)

Создаёт экземпляр нового документа с основным файлом.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
Document CreateDocument(string filePath)
```

Параметры

filePath

Тип: System.String

Путь к загружаемому файлу документа

Возвращаемое значение

Тип: Document

Документ

Заметки

В качестве вида документа будет установлен корневой вид (обычно Документ) документа в *Справочнике видов карточек*.

Примеры

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = documentService.CreateDocument(@"x:\Sample.docx"); ③

objectContext.SaveObject(document);
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Создание документа с указанием местоположения основного файла.

IDocumentService.CreateDocument — метод (String, KindsCardKind, DocumentVersioningType)

Создаёт документ с заданным видом карточки и добавляет основной файл с указанным типов версионирования.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
Document CreateDocument(string filePath, KindsCardKind kind, DocumentVersioningType
versioningType)
```

Параметры

filePath

Тип: System.String

Путь к основному файлу

kind

Тип: KindsCardKind

Вид документа

versioningType

Тип: DocumentVersioningType

Вид документа

Возвращаемое значение

Тип: Document

Созданный документ

Исключения

Исключение	Условие
System.InvalidOperationException	Ошибка возникает в том случае, если добавляемый файл является подписанным, но в соответствующем сообщении отсутствуют файлы.

Примеры

Далее приведён пример создания карточки документ с основным файлом, у которого автоматический тип версионирования указан

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("7DB9044E-91B7-447D-8CAA-5E5A4967B8D4")); ③

Document document = documentService.CreateDocument(@"x:\Sample.docx", kind,
DocumentVersioningType.Auto);
objectContext.SaveObject(document); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение вида карточки (в примере, Заявка).

④ Создание документа.

IDocumentService.CreateDocument — метод (String, KindsCardKind)

Создаёт экземпляр нового документа с указанным видом, с основным файлом.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
[WorkflowVisible(WorkflowVisibility.Visible)]
Document CreateDocument(string filePath, KindsCardKind kind)
```

Параметры

filePath

Тип: System.String

Путь к основному файлу

kind

Тип: KindsCardKind

Вид карточки

Возвращаемое значение

Тип: Document

Экземпляр карточки Документ

Примеры

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("0538E317-FAB6-4BB7-
85C5-2303375A4EE1")); ③

Document document = documentService.CreateDocument(@"..\Sample.docx", kind); ④

document.Description = "Пример документа"; ⑤

objectContext.SaveObject(document); ⑥
```

① Инициализация контекста объектов.

② Получение сервиса для работы с документами.

③ Получение вида карточки.

④ Создание документа с основным файлом.

⑤ Изменение описания.

⑥ Сохранение изменений в контексте.

IDocumentService.CreateListCards — метод (Document)

Инициализирует списки, используемые документом.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void CreateListCards(Document document)
```

Параметры

document

Тип: `Document`

Документ

Заметки

Метод `CreateListCards` создаёт в документе экземпляры списков: Список подписей, Задания, Ссылки, Категории, Опрос; если они не были проинициализированы ранее. Может быть использован при создании нового экземпляра документа методом `CreateDocument`.

IDocumentService.CreateSignatureList — метод

Инициализирует новый экземпляр списка подписей документа.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
SignatureList CreateSignatureList()
```

Возвращаемое значение

Тип: SignatureList

Список подписей

Заметки

Метод создаёт и *сохраняет* в контексте объектов экземпляр сущности типа SignatureList, который может быть присвоен соответствующему свойству документа.

IDocumentService.DecodeSignedDocument — метод (String, Document, SignedCms)

Расшифровывает сообщение CMS/PKCS 7, которое содержится в файле, расположенном по указанному пути.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string DecodeSignedDocument(string filePath, Document document, out System.Security
.Cryptography.Pkcs.SignedCms signedCms)
```

Параметры

filePath

Тип: System.String

Путь к файлу с криптографическим сообщением

document

Тип: Document

Документ, в который производится добавление содержимого сообщения. Если документ содержит основной файл с именем идентичным имени файла (без учета расширения .p7s) загружаемого сообщения, то содержимое основного файла будет загружено в сообщение

signedCms

Тип: SignedCms

Сообщение CMS/PKCS 7. Если не было проинициализировано, то будет выполнена инициализация

Возвращаемое значение

Тип: System.String

Расположение временного файла, содержащего расшифрованный объект

IDocumentService.DownloadAdditionalFile — метод (Document, DocumentFile)

Выгружает в файловую систему указанный дополнительный файл документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string DownloadAdditionalFile(Document document, DocumentFile additionalFile)
```

Параметры

document

Тип: Document

Документ

additionalFile

Тип: DocumentFile

Дополнительный файл документа

Возвращаемое значение

Тип: System.String

Путь, по которому был выгружен файл

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр additionalFile.
System.MethodAccessException	Ошибка возвращается в случае, если у сотрудника недостаточно прав на чтение дополнительного файла.

Примеры

Ниже приведён пример использования метода DownloadAdditionalFile при выгрузке дополнительного файла документа

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-000000000000")); ③

string path = documentService.DownloadAdditionalFile(document, document.Files.First(t => t.FileType = DocumentFileType.Additional)); ④
```

① Инициализация контекста объектов.

② Получение сервиса для работы с документами.

③ Получение документа.

④ Выгрузка первого дополнительного файла на диск, по пути path.

IDocumentService.DownloadAdditionalFile — метод (Document, DocumentFile, String)

Выгружает дополнительный файл документа в указанный каталог.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DownloadAdditionalFile(Document document, DocumentFile additionalFile, string path)
```

Параметры

document

Тип: Document

Документ, содержащий выгружаемый файл

additionalFile

Тип: DocumentFile

Возвращаемый файл документа

path

Тип: System.String

Путь для выгрузки файла

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр additionalFile.
System.ArgumentOutOfRangeException	Ошибка возвращается в случае, если не задан параметр path.
System.MethodAccessException	Ошибка возвращается в случае, если у сотрудника недостаточно прав для чтения основного файла.

Примеры

В примере демонстрируется использование метода DownloadAdditionalFile при выгрузке первого дополнительного файла документа

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000"));
DocumentFile firstAdditionalFile = document.Files.First(t=>t.FileType = DocumentFileType
.Additional); ③

documentService.DownloadAdditionalFile(document, firstAdditionalFile, @"..\" +
```

```
firstAdditionalFile.FileName); ④
```

① Инициализация контекста объектов.

② Получение сервиса для работы с документами.

③ Получение первого дополнительного файла документа.

④ Выгрузка файла на файловую систему.

IDocumentService.DownloadFile — метод (Guid)

Сохраняет файл с указанным идентификатором на файловую систему.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string DownloadFile(Guid versionedFileCardId)
```

Параметры

versionedFileCardId

Тип: System.Guid

Идентификатор карточки версионного файла

Возвращаемое значение

Тип: System.String

Путь, по которому был выгружен файл

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр versionedFileCardId.
System.InvalidOperationException	Ошибка возвращается, если карточка версионного файла была удалена или не существует.

Примеры

Ниже приведён пример использования метода DownloadFile при выгрузке файла документа

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

string pathFile = documentService.DownloadFile(new Guid("00000000-0000-0000-0000-
000000000000")); ③
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Выгрузка файла по пути `pathFile`.

IDocumentService.DownloadMainFile — метод (Document, DocumentFile)

Выгружает основной файл документа на диск.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
string DownloadMainFile(Document document, DocumentFile file)
```

Параметры

document

Тип: `Document`

Документ

file

Тип: `DocumentFile`

Файл документа

Возвращаемое значение

Тип: `System.String`

Путь, по которому был выгружен файл

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или file.
System.InvalidOperationException	Ошибка возвращается в случае, если файл документа не указывает на карточку версионного файла.
System.MethodAccessException	Ошибка возвращается в случае, если у сотрудника недостаточно прав для чтения основного файла.

IDocumentService.DownloadMainFile — метод (Document, DocumentFile, String)

Выгружает основной файл документа в указанный каталог.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
[WorkflowVisible(WorkflowVisibility.Visible)]
void DownloadMainFile(Document document, DocumentFile file, string path)
```

Параметры

document

Тип: Document

Документ

file

Тип: DocumentFile

Файл документа

path

Тип: System.String

Путь для сохранения файла

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или file.
System.ArgumentOutOfRangeException	Ошибка возвращается в случае, если не задан параметр path.
System.MethodAccessException	Ошибка возвращается в случае, если у сотрудника недостаточно прав для чтения основного файла.

Примеры

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

documentService.DownloadMainFile(document, document.Files.FirstOrDefault(t=>t.FileType =
DocumentFileType.Main), @"..\SampleDownload.docx"); ④
```

① Инициализация контекста объектов.

② Получение сервиса для работы с документами.

③ Получение документа с основным файлом.

④ Сохранение первого основного файла.

IDocumentService.DownloadMainFileVersion — метод (Document, DocumentFile, Guid)

Выгружает определённую версию основного файла документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string DownloadMainFileVersion(Document document, DocumentFile file, Guid versionId)
```

Параметры

document

Тип: `Document`

Документ

file

Тип: `DocumentFile`

Файл документа

versionId

Тип: `System.Guid`

Идентификатор версии файла

Возвращаемое значение

Тип: `System.String`

Путь, по которому был выгружен файл

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `document` или `file`.

IDocumentService.GetAdditionalFile — метод (DocumentFile)

Возвращает карточку файл с версиями из дополнительного файла документа.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
VersionedFileCard GetAdditionalFile(DocumentFile additionalFile)
```

Параметры

additionalFile

Тип: `DocumentFile`

Дополнительный файл документа

Возвращаемое значение

Тип: `VersionedFileCard`

Карточка файла с версиями

Примеры

Ниже приведён пример получения файла с версиями из документа

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

VersionedFileCard versionedFileCard = documentService.GetAdditionalFile(document.Files
.First(t => t.FileType = DocumentFileType.Additional)); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ Получение файла с версиями.

IDocumentService.GetKindSettings — метод (KindsCardKind)

Возвращает настройки заданного вида карточки Документ.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
DocumentSetting GetKindSettings(KindsCardKind kind)
```

Параметры

kind

Тип: `KindsCardKind`

Вид карточки Документ

Возвращаемое значение

Тип: `DocumentSetting`

Настройки вида документа

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `kind`.

Заметки

Настройки вида карточки задаются в *Справочнике видов карточек*.

Примеры

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("7DB9044E-91B7-447D-8CAA-5E5A4967B8D4")); ③

DocumentSetting documentSettings = documentService.GetKindSettings(kind); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение вида карточки (в примере "Заявка").

④ Получение настроек вида карточки.

IDocumentService.GetMainFile — метод (DocumentFile)

Возвращает карточку файл с версиями из основного файла документа.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
VersionedFileCard GetMainFile(DocumentFile file)
```

Параметры

file

Тип: `DocumentFile`

Основной файл документа

Возвращаемое значение

Тип: `VersionedFileCard`

Карточка файла с версиями

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `file`.

Примеры

Ниже приведён пример получения файла с версиями из первого основного файла документа

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

VersionedFileCard versionedFileCard = documentService.GetMainFile(document.Files.First(t
=> t.FileType = DocumentFileType.Main)); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ Получение файла с версиями.

IDocumentService.GetMainFiles — метод (Document)

Возвращает все основные файлы документа.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<DocumentFile> GetMainFiles(Document document)
```

Параметры

document

Тип: Document

Документ

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<DocumentFile>

Коллекция элементов типа DocumentFile

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document.

IDocumentService.LockMainFile — метод (Document, DocumentFile, String)

Выгружает файл из документа на диск с установкой постоянной блокировки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void LockMainFile(Document document, DocumentFile file, string path)
```

Параметры

document

Тип: Document

Документ

file

Тип: DocumentFile

Выгружаемый Файл документа

path

Тип: System.String

Каталог

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или file.
System.MethodAccessException	Ошибка возвращается в случае, если путь path не определён.

IDocumentService.MakeLockedMainFileForceChanged — метод (Document, DocumentFile)

Устанавливает признака наличия изменений для выгруженного основного файла документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void MakeLockedMainFileForceChanged(Document document, DocumentFile file)
```

Параметры

document

Тип: Document

Документ

file

Тип: DocumentFile

Файл документа

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или file.

Примеры

Далее приведён пример использования метода MakeLockedMainFileForceChanged при редактировании основного файла документа

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

DocumentFile documentFile = document.Files.First(t => t.FileType = DocumentFileType.
Main); ④

string path = documentService.EditMainFile(document, documentFile); ⑤

⑥

documentService.MakeLockedMainFileForceChanged(document, documentFile); ⑦

documentService.UnlockMainFile(document, documentFile); ⑧
```

① Инициализация контекста объектов.

② Получение сервиса для работы с документами.

③ Получение документа.

④ Получение основного файла документа.

⑤ Выгрузка файла для редактирования.

⑥ Редактирование файла.

⑦ Установка признака наличия изменений.

⑧ Разблокировка файла после редактирования с созданием новой версии файла.

IDocumentService.PurgeFileCache — метод (Document)

Очищает кэш файлов документ.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void PurgeFileCache(Document document)
```

Параметры

document

Тип: Document

Документ с файлами

IDocumentService.RemoveMainFileComments — метод (Document, DocumentFile, FileCommentCollection, IEnumerable<FileComment>)

Удаляет комментарии к файлу документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void RemoveMainFileComments(Document document, DocumentFile file, FileCommentCollection
collection, IEnumerable<FileComment> comments)
```

Параметры

document

Тип: Document

Документ

file

Тип: DocumentFile

Файл документа

collection

Тип: `FileCommentCollection`

Коллекция комментариев

comments

Тип: `System.Collections.Generic.IEnumerable<FileComment>`

Коллекция удаляемых комментариев

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `document`, `file`, `collection` или `comments`.

Примеры

В приведенном примере демонстрируется использование метода RemoveMainFileComments при удалении комментариев, содержащих слово "удалить", к первому основному файлу

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

DocumentFile file = document.Files.First(t => t.FileType = DocumentFileType.Main); ④

VersionedFileCard versionedFileCard = documentService.GetMainFile(file); ⑤

documentService.RemoveMainFileComments(document, file, versionedFileCard.Comments,
versionedFileCard.Comments.Where(t => t.Comment.Contains("удалить"))); ⑥
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа.

④ Получение первого основного файла.

⑤ Получение карточки файла с версией из основного файла.

⑥ Удаление комментария.

IDocumentService.RenameAdditionalFile — метод (Document, DocumentFile, String)

Переименовывает дополнительный файл документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void RenameAdditionalFile(Document document, DocumentFile file, string newFileName)
```

Параметры

document

Тип: Document

Документ, которому принадлежит файл

file

Тип: DocumentFile

Файл, название которого изменяется

newFileName

Тип: System.String

Новое название файла. Имя файла должно содержать расширение.

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document, file или newFileName.

Исключение	Условие
System.ArgumentException	Ошибка возвращается, если newFileName содержит только символы-разделители, либо содержит недопустимые (*, /, \, |, <, >, ", ?, :) символы.

Заметки

Чтобы сотрудник мог переименовывать файлы документа, ему должны быть предоставлены права в ролевой модели безопасности: Переименование файлов и Запись.

Примеры

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

documentService.RenameAdditionalFile(document, document.Files.First(t => t.FileType =
DocumentFileType.Additional), "Новое имя файла.docx"); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа с дополнительным файлом.

④ Изменение названия первого дополнительного файла.

Можно указывать новое название файла без расширения, а расширение получать из оригинального файла, например, так:

```
DocumentFile firstAdditionalFile = document.Files.First(t => t.FileType =
DocumentFileType.Additional);
string newFileName = string.Format("Новое название файла{0}", Path.GetExtension
(firstAdditionalFile.FileName));
```

IDocumentService.RenameMainFile — метод (Document, DocumentFile, String)

Переименовывает основной файл документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void RenameMainFile(Document document, DocumentFile file, string newFileName)
```

Параметры

document

Тип: Document

Документ, которому принадлежит файл

file

Тип: DocumentFile

Файл, название которого изменяется

newFileName

Тип: System.String

Новое название файла. Имя файла должно содержать расширение.

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document, file или newFileName.
System.ArgumentException	Ошибка возвращается, если newFileName содержит только символы-разделители, либо содержит недопустимые (*, /, \, |, <, >, ", ?, :) символы.

Заметки

Чтобы сотрудник мог переименовывать файлы документа, ему должны быть предоставлены права в ролевой модели безопасности: Переименование файлов и Запись.

Примеры

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

documentService.RenameMainFile(document, document.Files.First(t => t.FileType =
DocumentFileType.Main), "Новое имя файла.docx"); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа с основным файлом.

④ Изменение названия первого основного файла.

Можно указывать новое название файла без расширения, а расширение получать из оригинального файла, например, так:

```
DocumentFile firstMainFile = document.Files.First(t => t.FileType = DocumentFileType.
Main);
string newFileName = string.Format("Новое название файла{0}", Path.GetExtension
(firstMainFile.FileName));
```

IDocumentService.SyncPropertiesToFile — метод (Document, DocumentFile, IList<IDocumentPropertySetting>)

Выполняет синхронизацию значений полей карточки *Документ* со свойствами основного файла данного документа. Передача осуществляется в сторону вложенного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool SyncPropertiesToFile(Document document, DocumentFile file, IList
<IDocumentPropertySetting> propertySettings)
```

Параметры

document

Тип: `Document`

Документ

file

Тип: `DocumentFile`

Файл документа

propertySettings

Тип: `System.Collections.Generic.IList<IDocumentPropertySetting>`

Синхронизируемые свойства документа

Возвращаемое значение

Тип: `System.Boolean`

Значение всегда false

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `document` или `file`.

Заметки

Параметры синхронизации полей карточки и свойств основного документа задаются в *Справочнике видов карточек*.

Синхронизация производится только с файлами Microsoft Office.

Примеры

Ниже приведён пример использования метода `SyncPropertiesToFile` при синхронизации названия документа со свойствами основного файла, привязанного к документу

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②
```

```
Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

document.MainInfo.Name = "Изменение названия и синхронизация";
objectContext.SaveObject(document); ④

DocumentSetting documentSetting = documentService.GetKindSettings(document.SystemInfo
.CardKind);
IList<IDocumentPropertySetting> setting = documentSetting.Properties.Cast
<IDocumentPropertySetting>().ToList(); ⑤

documentService.SyncPropertiesToFile(document, document.Files.First(t => t.FileType =
DocumentFileType.Main), setting); ⑥
```

① Инициализация контекста объектов.

② Получение сервиса для работы с документами.

③ Получение документа.

④ Изменение данных.

⑤ Получение настроек для вида документа и приведение к требуемому типу.

⑥ Передача названия карточки в файл.

IDocumentService.GetPartSignatureInfo — метод (BaseCardSignaturePart signaturePart)

Возвращает информацию о части подписи документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
SignatureInfo GetPartSignatureInfo(Document document, BaseCardSignaturePart
signaturePart, X509Certificate2 certificate);
```

Параметры

document

Тип: Document

Документ, из которого будет получена информация о подписи

signaturePart

Тип: BaseCardSignaturePart

Часть подписи

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат подписи

Возвращаемое значение

Тип: SignatureInfo

Информация о подписи

IDocumentService.VerifySignature — метод (Document, BaseCardSignature, X509Certificate2, Boolean, Boolean, ICollection<CardFieldSetting>)

Выполняет проверку ЭП, установленной на документе, файлах документа или полях, и возвращает результат проверки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardSignatureVerification VerifySignature(Document document, BaseCardSignature
signature, X509Certificate2 certificate, bool signDocument, bool signAttachments,
ICollection<CardFieldSetting> fields);
```

Параметры

document

Тип: Document

Документ, содержащий подписанные данные

signature

Тип: BaseCardSignature

ЭП, установленная на данной карточке

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат ЭП

signDocument

Тип: System.Boolean

Выполнять проверку подписи, установленной на документе — проверяется подпись, установленная на основных файлах

signAttachments

Тип: System.Boolean

Выполнять проверку подписи, установленной на дополнительных файлах

fields

Тип: System.Collections.Generic.ICollection<CardFieldSetting>

Коллекция полей типа CardFieldSetting, у которых должна быть проверена подпись. Список подписываемых полей задаётся в Справочнике видов карточек

Возвращаемое значение

Тип: BaseCardSignatureVerification

Результат проверки ЭП

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр document или signature.

IDocumentService.AdditionalFileAdded — событие

Событие срабатывает после добавления дополнительного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event EventHandler<DocumentAdditionalFileEventArgs> AdditionalFileAdded
```

IDocumentService.AdditionalFileRemoved — событие

Событие срабатывает после удаления дополнительного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event EventHandler<DocumentAdditionalFileEventArgs> AdditionalFileRemoved
```

IDocumentService.MainFileAdded — событие

Событие срабатывает после добавления основного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event EventHandler<DocumentMainFileEventArgs> MainFileAdded
```

IDocumentService.MainFileCancelLocked — событие

Событие срабатывает после отмены блокировки основного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event EventHandler<DocumentCancelLockedEventArgs> MainFileCancelLocked
```

IDocumentService.MainFileCurrentVersionChanged — событие

Событие срабатывает после изменения текущей версии основного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event EventHandler<DocumentMainFileEventArgs> MainFileCurrentVersionChanged
```

IDocumentService.MainFileLocked — событие

Событие срабатывает после блокировки основного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event EventHandler<DocumentMainFileEventArgs> MainFileLocked
```

IDocumentService.MainFileRemoved — событие

Событие срабатывает после удаления основного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event EventHandler<DocumentMainFileEventArgs> MainFileRemoved
```

IDocumentService.MainFileUnlocked — событие

Событие срабатывает после разблокировки основного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event EventHandler<DocumentUnlockedEventArgs> MainFileUnlocked
```

IDocumentService.MainFileUnlocking — событие

Вызывается до разблокировки основного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event EventHandler<DocumentUnlockingEventArgs> MainFileUnlocking
```

IDocumentService.SignatureAdded — событие

Событие срабатывает после подписания документа с помощью ЭП.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event EventHandler<DocumentSignatureEventArgs> SignatureAdded
```

IDocumentService.SynchronizationFromFileCompleted — событие

Событие срабатывает после синхронизации данных карточки со свойствами основного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event EventHandler<EventArgs> SynchronizationFromFileCompleted
```

/dv6/programmer/dv6/BackOffice-ObjectModel-Services-IDocumentService/IDocumentService.GetMainFiles_MT/[IDocumentService.GetMainFiles — метод (Document)]

IKindService — интерфейс

Сервис для работы со справочником видов карточек.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IKindService
```

Свойства

Имя	Описание
KindsDictionary	Возвращает виды карточек, определённые в Справочнике видов карточек.

Методы

Имя	Описание
AddCardKind(KindsCardKind)	Добавляет к заданному виду карточки подвид и возвращает ссылку на созданный объект.
AddCardKind(KindsCardKind, String)	Добавляет к виду карточки подвид с указанным названием и возвращает на него ссылку.
AddCardKind(KindsCardType)	Добавляет дополнительный вид карточки, указанного типа, и возвращает на него ссылку.
AddCardKind(ObjectCollection<KindsCardKind>, String)	Добавляет к приведенной коллекции видов карточке подвид с указанным названием и возвращает на него ссылку.
AddCardType(Guid)	Добавляет новый тип карточки с предопределённым идентификатором.
AddCreationSettings(KindsCardKind, Guid, String, KindsCardKind, Boolean, StatesOperation)	Добавляет новый режим создания карточки к указанному виду карточки.
AddExtendedSettingGroup(KindsCardExtendedSettingGroupCollection)	Создаёт новую группу настроек в указанной коллекции групп настроек расширения карточки определённого вида.

Имя	Описание
AddExtension(String, String)	Добавляет расширение для типа карточки в *Справочник видов карточек*.
AddProcessOperation(KindsCardProcess, StatesOperation)	Добавляет бизнес-процесс к списку процессов, привязанных к операциям в карточке определённого вида.
AddProcessSettings(KindsCardKind, Guid, String)	Добавляет бизнес-процесс с настройками к указанному виду карточки.
AddProcessStateBranch(KindsCardProcess, StatesStateMachineBranch)	Добавляет переход состояний для процесса.
AddProcessVariable(KindsCardProcess, Guid, Guid, String, Boolean, KindsCardProcessVariableSync)	Добавляет переменную для бизнес-процесса.
GetCardDefaultLocationDescription(KindsCardCreationSettingLocation)	Возвращает название размещения по умолчанию для карточки определённого вида.
GetCardType(Guid)	Возвращает тип карточки с заданным идентификатором.
GetPreferredCardKind(KindsCardKind)	Возвращает вид карточки, имеющий все собственные настройки (наследование отключено).
GetPreferredCardKindByExtendedSettings(KindsCardKind)	Возвращает первого родителя указанного вида карточки, у которого установлен признак "Использовать собственные настройки расширений". Может быть возвращен вид карточки, переданный в метод в качестве параметра.

Имя	Описание
GetPreferredCardKindByLayout(KindsCardKind)	Возвращает первого родителя указанного вида карточки, который не использует наследование настроек (установлен признак "Использовать собственные разметки" или "Использовать собственные настройки вида, автомат состояний, ролевую модель и разметки"). Может быть возвращен вид карточки, переданный в метод в качестве параметра.
RemoveCardKind(KindsCardKind)	Удаляет указанный вид карточки.
RemoveCreationSettings(KindsCardKind, KindsCardCreationSetting)	Удаляет у вида карточки указанную настройку способа создания карточки вида.
RemoveExtension(KindsCardExtension)	Удаляет указанное расширение типа карточки.

Примеры

```
①

IKindService kindService = objectContext.GetService<IKindService>(); ②

KindsCardKind kindsCardKind = objectContext.GetObject<KindsCardKind>(new Guid("00000000-
0000-0000-0000-000000000000")); ③

KindsCardProcess kindsCardProcess = kindService.AddProcessSettings(kindsCardKind, new
Guid("00000000-0000-0000-0000-000000000001"), "Вызов бизнес-процесса"); ④
kindsCardProcess.Policy = KindsCardProcessPolicy.StateOperation;

IStateService stateService = objectContext.GetService<IStateService>(); ⑤

StatesOperation statesOperation = stateService.GetOperations(kindsCardKind).First(); ⑥

kindService.AddProcessOperation(kindsCardProcess, statesOperation); ⑦

objectContext.AcceptChanges(); ⑧
```

① Инициализация контекста объектов.

② Получение сервиса для работы со Справочником вида карточек.

③ Получение вида карточки.

④ Добавление настроек нового бизнес-процесса в процессы карточки указанного вида.

⑤ Получение сервиса для работы с Конструктором состояний.

⑥ Получение первого, доступного для карточки, состояния.

⑦ Добавление состояние к состояниям, вызывающим запуск бизнес-процесса.

IKindService.KindsDictionary — свойство

Возвращает виды карточек, определённые в Справочнике видов карточек.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
Kinds KindsDictionary { get; }
```

Значение свойства

Тип: Kinds

Виды карточек

IKindService.AddCardKind — метод (KindsCardKind)

Добавляет к заданному виду карточки подвид и возвращает ссылку на созданный объект.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
KindsCardKind AddCardKind(KindsCardKind parentKind)
```

Параметры

parentKind

Тип: KindsCardKind

Родительский вид карточки

Возвращаемое значение

Тип: `KindsCardKind`

Созданный вид карточки

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `parentKind`.

Заметки

Название создаваемого вида карточки будет сформировано на основе названия родительского вида карточки, либо на основе типа карточки.

Примеры

```
①

IKindService kindService = objectContext.GetService<IKindService>(); ②

KindsCardKind parentKind = objectContext.GetObject<KindsCardKind>(new Guid("7DB9044E-91B7-447D-8CAA-5E5A4967B8D4")); ③

KindsCardKind newKind = kindService.AddCardKind(parentKind); ④

objectContext.SaveObject(newKind); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса для работы со Справочником видов карточек.

③ Получение родительского вида "Заявка".

④ Добавление нового вида карточки.

⑤ Обязательное сохранение.

IKindService.AddCardKind — метод (KindsCardKind, String)

Добавляет к виду карточки подвид с указанным названием и возвращает на него ссылку.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
KindsCardKind AddCardKind(KindsCardKind parentKind, string name)
```

Параметры

parentKind

Тип: KindsCardKind

Родительский вид карточки

name

Тип: System.String

Название нового вида карточки

Возвращаемое значение

Тип: KindsCardKind

Добавленный вид карточки

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр parentKind.

IKindService.AddCardKind — метод (KindsCardType)

Добавляет дополнительный вид карточки, указанного типа, и возвращает на него ссылку.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
KindsCardKind AddCardKind(KindsCardType cardType)
```

Параметры

cardType

Тип: KindsCardType

Тип карточки, к которому осуществляется добавление вида карточки

Возвращаемое значение

Тип: KindsCardKind

Ссылка на новый вид карточки

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр cardType.

Заметки

Название нового вида карточки будет сформировано на основе названия типа карточки.

Примеры

```
①

IKindService kindService = objectContext.GetService<IKindService>(); ②

KindsCardType kindType = kindService.GetCardType(new Guid("B9F7BFD7-7429-455E-A3F1-
94FFB569C794")); ③

KindsCardKind newKind = kindService.AddCardKind(kindType); ④

objectContext.SaveObject(newKind); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса для работы со Справочником видов карточек.

③ Получение типа карточки "Документ".

④ Добавление нового вида карточки.

⑤ Обязательное сохранение.

IKindService.AddCardKind — метод (ObjectCollection<KindsCardKind>, String)

Добавляет к приведенной коллекции видов карточке подвид с указанным названием и возвращает на него ссылку.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
KindsCardKind AddCardKind(ObjectCollection<KindsCardKind> cardKinds, string name)
```

Параметры

cardKinds

Тип: ObjectCollection<KindsCardKind>

Коллекция видов карточек

name

Тип: System.String

Название нового вида карточек

Возвращаемое значение

Тип: KindsCardKind

Созданный вид карточки

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр cardKinds.

Заметки

Коллекция видов карточек может быть получена, например, из свойства типа карточки: KindsCardType.Kinds.

IKindService.AddCardType — метод (Guid)

Добавляет новый тип карточки с предопределённым идентификатором.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
KindsCardType AddCardType(Guid cardtypeId)
```

Параметры

cardtypeId

Тип: System.Guid

Идентификатор типа карточки

Возвращаемое значение

Тип: KindsCardType

Добавленный тип карточки

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр cardtypeId.

Заметки

Для типа карточки с указанным идентификатором должен быть реализован компонент, а также должна существовать соответствующая библиотека карточек.

Примеры

```
①

IKindService kindService = objectContext.GetService<IKindService>(); ②

KindsCardType newType = kindService.AddCardType(new Guid("00000000-0000-0000-0000-
000000000000")); ③

objectContext.SaveObject(newType); ④
```

① Инициализация контекста объектов.

② Получение сервиса для работы со Справочником видов карточек.

③ Добавление типа карточки с идентификатором `00000000-0000-0000-0000-000000000000` в библиотеке карточек.

④ Обязательное сохранение.

IKindService.AddCreationSettings — метод (KindsCardKind, Guid, String, KindsCardKind, Boolean, StatesOperation)

Добавляет новый режим создания карточки к указанному виду карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
KindsCardCreationSetting AddCreationSettings(KindsCardKind cardKind, Guid modeId, string displayName, KindsCardKind creatorKind, bool withAllChildren, StatesOperation operation)
```

Параметры

cardKind

Тип: KindsCardKind

Вид карточки, к которой будет добавлен способ создания

modeId

Тип: System.Guid

Идентификатор режима создания

displayName

Тип: System.String

Отображаемое название режима создания

creatorKind

Тип: KindsCardKind

Вид родительской карточки

withAllChildren

Тип: System.Boolean

Флаг, позволяющий использовать любой доступный вид карточки в качестве создающего вида карточки

operation

Тип: StatesOperation

Тип операция редактирования, которая будет использована при создании карточки. Данная информация используется, в том числе при добавлении события в историю карточки.

Возвращаемое значение

Тип: KindsCardCreationSetting

Параметры создания карточки данного вида

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр cardKind, modeId или displayName.

Заметки

Список доступных идентификаторов режима создания modeId может быть получен следующим образом

```
List<NavCardCreatorInfo> cardKindCreators = new List<NavCardCreatorInfo>();

foreach (INavCardCreatorExtension navCardCreatorExtension in this.CardHost
.ExtensionManager.CardCreatorExtensions)
{
 cardKindCreators.AddRange(navCardCreatorExtension.GetCreatorsForKind(new Guid("0538E317-
FAB6-4BB7-85C5-2303375A4EE1")));
}
```

Примеры

```
①

IKindService kindService = objectContext.GetService<IKindService>(); ②

KindsCardKind kindsCardKind = objectContext.GetObject<KindsCardKind>(new Guid("00000000-
```

```
0000-0000-0000-000000000000")); ③

KindsCardKind parentCardKind = objectContext.GetObject<KindsCardKind>(new Guid("00000000-
0000-0000-0000-000000000001")); ④

Guid modeID = new Guid("00000000-0000-0000-0000-000000000002"); ⑤

KindsCardCreationSetting setting = kindService.AddCreationSettings(kindsCardKind, modeID,
"Создание карточки", parentCardKind, true, null); ⑥

objectContext.SaveObject<KindsCardCreationSetting>(setting); ⑦
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение вида карточки, для которой производится добавление способа создания.

④ Получение вида карточки, из которой может быть вызвано создание данной карточки.

⑤ Идентификатор режима создания карточки.

⑥ Добавление способа создания.

⑦ Обязательное сохранение контекста.

IKindService.AddProcessOperation — метод (KindsCardProcess, StatesOperation)

Добавляет бизнес-процесс к списку процессов, привязанных к операциям в карточке определённого вида.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
KindsCardProcessOperation AddProcessOperation(KindsCardProcess process, StatesOperation
operation)
```

Параметры

process

Тип: KindsCardProcess

Параметры связываемого бизнес-процесса

operation

Тип: StatesOperation

Операция в карточке, приводящая к запуску бизнес-процесса

Возвращаемое значение

Тип: KindsCardProcessOperation

Операция процесса

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр process или operation.

IKindService.AddProcessSettings — метод (KindsCardKind, Guid, String)

Добавляет бизнес-процесс с настройками к указанному виду карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
KindsCardProcess AddProcessSettings(KindsCardKind cardKind, Guid templateId, string name)
```

Параметры

cardKind

Тип: KindsCardKind

Вид карточки, к которой осуществляется добавление

templateId

Тип: System.Guid

Идентификатор шаблона бизнес-процесса

Возвращаемое значение

Тип: KindsCardProcess

Бизнес-процесс, описанный в настройках карточки определённого вида

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр cardKind или templateId.

IKindService.GetCardDefaultLocationDescription — метод (KindsCardCreationSettingLocation)

Возвращает название размещения по умолчанию для карточки определённого вида.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string GetCardDefaultLocationDescription(KindsCardCreationSettingLocation location)
```

Параметры

location

Тип: KindsCardCreationSettingLocation

Размещение вида карточки

Возвращаемое значение

Тип: System.String

Название размещения

Заметки

Размещение по умолчанию для создаваемого экземпляра карточки определённого вида, регулируется в Справочнике видов карточек.

IKindService.GetPreferredCardKind — метод (KindsCardKind)

Возвращает вид карточки, имеющий все собственные настройки (наследование отключено).

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
KindsCardKind GetPreferredCardKind(KindsCardKind cardKind)
```

Параметры

cardKind

Тип: KindsCardKind

Вид карточки, начиная с которого начинается поиск (по дереву наследования) вида, имеющего собственные настройки

Возвращаемое значение

Тип: KindsCardKind

Вид карточки, не наследующий настроек

Заметки

Наследование настроек вида карточки устанавливается в *Справочнике видов карточек*, в секции Наследование.

ILayoutService — интерфейс

Описывает сервис для работы с *Конструктором разметок*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface ILayoutService
```

Свойства

Имя	Описание
Layouts	Возвращает коллекцию ссылок из справочника ссылок.

Методы

Имя	Описание
AddLayoutToDesignTree(LayoutsDesignTree, LayoutsLayout)	Добавляет узел с разметкой в дерево дизайнов.
AddRoleToDesignTree(LayoutsDesignTree, RoleModelRole)	Добавляет узел с ролью в дерево дизайнов.
AddStateToDesignTree(LayoutsDesignTree, StatesState)	Добавляет узел с состоянием в дерево дизайнов.
CanMoveDown(LayoutsDesignTree)	Определяет возможность перемещения поддерева вниз в дереве дизайнов
CanMoveDown(KindsCardKind, LayoutsDesignTree, LayoutsLayout)	Определяет возможность перемещения узла разметки вниз в дереве дизайнов.
CanMoveUp(LayoutsDesignTree)	Определяет возможность перемещения поддерева вверх в дереве дизайнов
CanMoveUp(KindsCardKind, LayoutsDesignTree, LayoutsLayout)	Определяет возможность перемещения узла разметки вверх в дереве дизайнов.
CreateCardKindSettings(KindsCardKind, Boolean)	Создаёт настройку для вида карточки с возможностью создания дерева дизайнов.
CreateDesignTree	Инициализирует объект типа LayoutsDesignTree.
CreateDesignTree(KindsCardKind)	Инициализирует объект типа LayoutsDesignTree для заданного вида карточки.
CreateLayout(LayoutsCardKindSettings, Boolean)	Создаёт разметку для вида карточки, с возможностью создания дерева дизайнов.
CreateLayout(LayoutsCardKindSettings, LayoutsLayout, Boolean)	Создаёт разметку для вида карточки, используя данные другой разметки.
CreateLayoutsProperty(LayoutsCardKindSettings)	Добавляет свойства к настройкам вида карточки.

Имя	Описание
`CreateNodeLayout`	Инициализирует объект типа LayoutsNodeLayout.
`CreateOperation(KindsCardKind, LayoutsProperty, String)`	Добавляет операцию к виду карточки, используя данные настроек указанной разметки.
`CreateTableColumn(LayoutsCardKindSettings, LayoutsProperty)`	Добавляет к настройкам разметки атрибут столбца.
`GetAnyDesignTree(KindsCardKind, LayoutsLayout)`	Возвращает первое дерево дизайнов для указанного вида карточки с наличием в нем заданной разметки.
`GetAvailableLayouts(KindsCardKind, StatesState, IEnumerable<RoleModelRole>)`	Возвращает список разметок для указанного вида карточки, удовлетворяющих заданным условиям (состояние и роли).
`GetCardKindSettings(KindsCardKind)`	Возвращает настройки разметки для заданного вида карточки.
`GetCardKindStateSettings(KindsCardKind)`	Возвращает настройки вида карточки из Конструктора состояний.
`GetCustomProperties(KindsCardKind, Boolean)`	Возвращает коллекцию дополнительных свойств из настроек вида карточки.
`GetCustomProperty(KindsCardKind, String, Boolean)`	Возвращает дополнительное свойство из настроек вида карточки.
`GetDesignTree(KindsCardKind)`	Возвращает дерево дизайнов для заданного вида карточки.
`GetFreePropertyName(KindsCardKind, LayoutsPropertyItem, String, List<String>)`	Формирует незанятое имя свойства для настроек вида карточки, с учетом заданного списка зарезервированных имён.
`GetKindsCardType(Guid)`	Получает тип для заданного вида карточки.
`GetKindsCardType(KindsCardKind)`	Получает тип для заданного вида карточки.

Имя	Описание
`GetLayoutAttribute(LayoutsProperty, LayoutsLayout)`	Получает атрибуты для заданной разметки.
`GetLayouts(KindsCardKind)`	Возвращает все разметки для заданного вида карточки.
`GetLinkedProperties(KindsCardKind, LayoutsProperty)`	Возвращает коллекцию ссылок на указанное свойство разметки.
`LowerPriority(LayoutsDesignTree)`	Понижает приоритет поддерева дизайнов.
`LowerPriority(LayoutsDesignTree, LayoutsLayout)`	Понижает приоритет разметки в заданном дереве дизайнов.
`NotifyBeforeSaveLayout`	Вызывает событие `LayoutSaving`.
`RaisePriority(LayoutsDesignTree)`	Повышает приоритет поддерева дизайнов.
`RaisePriority(LayoutsDesignTree, LayoutsLayout)`	Повышает приоритет разметки в заданном дереве дизайнов.
`RemoveCardKindSettings(KindsCardKind)`	Удаляет настройки разметки для указанного вида карточки.
`RemoveDesignTree(KindsCardKind, LayoutsDesignTree)`	Удаляет дерево дизайнов из настроек разметки указанного вида карточки.
`RemoveLayout(KindsCardKind, LayoutsLayout)`	Удаляет указанную разметку.
`RemoveLayoutFromDesignTree(LayoutsDesignTree, LayoutsLayout)`	Удаляет разметку из дерева дизайнов.

События

Имя	Описание
`LayoutSaving`	Событие может быть вызвано перед сохранением.

ILinkService — интерфейс

Представляет сервис для работы со *Справочником ссылок*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface ILinkService
```

Методы

Имя	Описание
CreateLink	Создаёт новый тип ссылки в *Справочнике ссылок*.
DeleteLink(LinksLinkType)	Удаляет тип ссылки из *Справочника ссылок*.
FindLink(String)	Возвращает тип ссылки с указанным названием.
FindLinkOnServer(String)	Выполняет поиск типа ссылки с указанным именем.
FindOppositeLink(LinksLinkType)	Возвращает тип обратной ссылки.

ILinkService.CreateLink — метод

Создаёт новый тип ссылки в *Справочнике ссылок*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
LinksLinkType CreateLink()
```

Возвращаемое значение

Тип: LinksLinkType

Тип ссылки

ILinkService.DeleteLink — метод (LinksLinkType)

Удаляет тип ссылки из *Справочника ссылок*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteLink(LinksLinkType link)
```

Параметры

link

Тип: LinksLinkType

Удаляемый тип ссылки

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр link.

ILinkService.FindLink — метод (String)

Возвращает тип ссылки с указанным названием.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
LinksLinkType FindLinkOnServer(string linkName)
```

Параметры

linkName

Тип: System.String

Название типа ссылки, как оно указано в *Справочнике ссылок*

Возвращаемое значение

Тип: LinksLinkType

Тип ссылки

ILinkService.FindLinkOnServer — метод (String)

Выполняет поиск типа ссылки с указанным именем.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
LinksLinkType FindLinkOnServer(string linkName)
```

Параметры

linkName

Тип: System.String

Название ссылки

Возвращаемое значение

Тип: LinksLinkType

Тип ссылки

ILinkService.FindOppositeLink — метод (LinksLinkType)

Возвращает тип обратной ссылки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
LinksLinkType FindOppositeLink(LinksLinkType link)
```

Параметры

link

Тип: LinksLinkType

Ссылка, для которой должна быть получена обратная ссылка

Возвращаемое значение

Тип: LinksLinkType

Тип ссылки

ILockService — интерфейс

Сервис управления блокировкой объектов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface ILockService
```

Методы

Имя	Описание
IsObjectLocked<T>(T, Boolean, String)	Определяет наличие и владельца блокировки, установленной на объект.
IsObjectLockedByAnotherUser<T>(T)	Определяет наличие у объекта блокировки, установленной другим сотрудником.
IsObjectLockedByAnotherUser<T>(T, String)	Определяет наличие у объекта блокировки, установленной другим сотрудником, а также учетную запись сотрудника, выполнившего блокировку.
IsObjectLockedByMe<T>(T)	Определяет наличие у объекта блокировки, установленной текущим пользователем.
LockObject<T>(T)	Устанавливает временную блокировку на объект.
LockObjectBase<T>(T)	Устанавливает временную блокировку на объект.
UnlockObject<T>(T)	Снимает временную блокировку с объекта.

ILockService.IsObjectLocked<T> — метод (T, Boolean, String)

Определяет наличие и владельца блокировки, установленной на объект.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool IsObjectLocked<T>(T obj, out bool byMe, out string ownerName) where T : ObjectBase;
```

Параметры типа

T

Тип объекта, состояние блокировки которого запрашивается

Параметры

obj

Тип: T

Объект, состояние блокировки которого запрашивается

byMe

Тип: System.Boolean

Если true, то блокировка установлена текущим пользователем

ownerName

Тип: System.String

Учетная запись сотрудника, заблокировавшего объект. Если заблокировано текущим пользователем, то возвращает пустую строку

Возвращаемое значение

Тип: System.Boolean

Если объект заблокирован, то возвращает true

Примеры

```
①
ILockService lockService = objectContext.GetService<ILockService>();
bool lockIsMy;
```

```
string lockOwner;

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ②

bool locked = lockService.IsObjectLocked<BaseCard>(document, out lockIsMy, out
lockOwner); ③
```

① Инициализация контекста объектов.

② Получение документа, для которого определяется блокировка.

③ Получение владельца блокировки.

ILockService.IsObjectLockedByAnotherUser<T> — метод (T)

Определяет наличие у объекта блокировки, установленной другим сотрудником.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool IsObjectLockedByAnotherUser<T>(T obj) where T : ObjectBase
```

Параметры типа

T

Тип заблокированного объекта

Параметры

obj

Тип: T

Объект, для которого проверяется блокировка

Возвращаемое значение

Тип: System.Boolean

Если объект заблокирован другим сотрудником, возвращает true. Если блокировка не установлена, либо установлена самим сотрудником, возвращает false

ILockService.IsObjectLockedByAnotherUser<T> — метод (T, String)

Определяет наличие у объекта блокировки, установленной другим сотрудником, а также учетную запись сотрудника, выполнившего блокировку.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool IsObjectLockedByAnotherUser<T>(T obj, out string ownerName) where T : ObjectBase
```

Параметры типа

T

Тип заблокированного объекта

Параметры

obj

Тип: T

Объект, для которого проверяется блокировка

ownerName

Тип: System.String

Учетная запись владельца блокировки

Возвращаемое значение

Тип: System.Boolean

Если объект заблокирован другим сотрудником, возвращает true. Если блокировка не установлена, либо установлена самим сотрудником, возвращает false

Примеры

```
①
ILockService lockService = objectContext.GetService<ILockService>();

Document document = objectContext.GetObject<Document>(); ②
string lockOwner;
```

```
bool locked = lockService.IsObjectLockedByAnotherUser<BaseCard>(document, out lockOwner);
③
```

① Инициализация контекста объектов.

② Получение документа.

③ Получение сведений о блокировке.

ILockService.IsObjectLockedByMe<T> — метод (T)

Определяет наличие у объекта блокировки, установленной текущим пользователем.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool IsObjectLockedByMe<T>(T obj) where T : ObjectBase
```

Параметры типа

T

Тип объекта, блокировка которого определяется

Параметры

obj

Тип: T

Объект, блокировка которого определяется

Возвращаемое значение

Тип: System.Boolean

Если блокировка установлена текущим сотрудником, возвращает true. Если объект не заблокирован или блокировка установлена другим сотрудником, возвращает false.

ILockService.LockObject<T> — метод (T)

Устанавливает временную блокировку на объект.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void LockObject<T>(T obj) where T : ObjectBase
```

Параметры типа

T

Тип блокируемого объекта

Параметры

obj

Тип: T

Блокируемый объект

Заметки

Для установки постоянной блокировки используйте метод LockObject.

Все блокировки, установленные при помощи ILockService.LockObject, автоматически снимаются при закрытии контекста объектов.

ILockService.LockObjectBase<T> — метод (T)

Устанавливает временную блокировку на объект.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool LockObjectBase<T>(T obj) where T : DocsVision.Platform.ObjectModel.ObjectBase
```

Параметры типа

T

Тип блокируемого объекта

Параметры

obj

Тип: `T`

Блокируемый объект

Возвращаемое значение

Тип: `System.Boolean`

Если блокировка установлена успешно, то возвращает — `true`, иначе — `false`

ILockService.UnlockObject<T> — метод (T)

Снимает временную блокировку с объекта.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void UnlockObject<T>(T obj) where T : ObjectBase
```

Параметры типа

T

Тип объекта, с которого снимается блокировка

Параметры

obj

Тип: `T`

Объект, с которого снимается блокировка

Заметки

Может быть снята только собственная блокировка.

ILogService — интерфейс

Сервис для работы с историей карточки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface ILogService
```

Методы

Имя	Описание
AddLogMessage(BaseCard, BuiltInOperation, LogItemDetailsCollection)	Добавляет детализированное сообщение об операции с карточкой в журнал. Используется для корректной локализации сообщений журнала.
AddLogMessage(BaseCard, BuiltInOperation, String)	Добавляет сообщение об операции с карточкой в журнал.
AddLogMessage(BaseCard, BuiltInOperation, String, Boolean)	Добавляет сообщение об операции с карточкой в журнал минуя кэш.
AddLogMessage(BaseCard, StatesOperation, LogItemDetailsCollection)	Добавляет детализированное сообщение об изменении статуса карточки в журнал. Сообщение должно быть определено в ресурсах. Используется для корректной локализации сообщений журнала.
AddLogMessage(BaseCard, StatesOperation, String)	Добавляет сообщение об изменении статуса карточки в журнал.
AddLogMessage(BaseCard, StatesOperation, String, Boolean)	Добавляет сообщение об изменении статуса карточки в журнал минуя кэш.
AddMessageAssembly(System.Reflection.Assembly)	Добавляет специальную сборку, содержащую менеджеров ресурсов, которые будут автоматически добавлены во временное хранилище ресурсов. Ресурсы будут использованы при получении готовых сообщений для журнала сообщений.
AddMessageResources(Guid, ResourceManager)	Добавляет менеджер ресурсов, предназначенный для получения деталей сообщения в текущем контексте.

Имя	Описание
`CommitLog`	Производит запись в журнал всех добавленных сообщений.
`GetLogMessages(DateTime?, DateTime?, BaseCard)`	Возвращает коллекцию отфильтрованных записей журнала.
`GetLogMessageDescription(LogMessage)`	Получает строку детализированного сообщения из объекта типа `LogMessage`.

События

Имя	Описание
`LogMessage`	Событие возникает после добавления сообщения в журнал методом `ILogService.AddLogMessage`.

Примеры

В примере, сообщение о редактировании добавляется в журнал документа с идентификатором `00000000-0000-0000-0000-000000000000`:

```
①

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-000000000000")); ②

ILogService logService = objectContext.GetService<ILogService>(); ③

logService.AddLogMessage(document, Document.EditOperation, "Документ был изменён из кода."); ④

logService.CommitLog(); ⑤
```

① Инициализация контекста объектов.

② Получение документа.

③ Получение сервиса журналирования.

④ Добавление в журнал сообщения.

⑤ Сохранение журнала.

INumerationRulesService — интерфейс

Сервис для работы с *конструктором правил нумерации*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface INumerationRulesService
```

Методы

Имя	Описание
`AddNewCondition(NumerationRulesRuleItem)`	Добавляет условие в строку правила нумерации в *конструкторе правил нумерации*.
`AddNewNumerator`	Создаёт новый нумератор.
`AddNewRule`	Создаёт новое правило нумерации в *конструкторе правил нумерации*.
`AddNewRuleItem(NumerationRulesRule)`	Добавляет новую строчку в правило нумерации.
`CreateNumber(CardData, BaseCard, NumerationRulesRule)`	Выделяет указанной карточке номер из нумератора следуя приложенному правилу нумерации.
`CreateTextNumber(BaseCard, String)`	Создаёт объект типа `BaseCardNumber`, содержащий текстовое представление номера.
`DeleteCondition(NumerationRulesRuleItemCondition, NumerationRulesRuleItem)`	Удаляет условие из строки правила нумерации.
`DeleteNumber(BaseCardNumber)`	Удаляет указанный номер из владеющей им карточки.
`DeleteNumerator(NumerationRulesNumerator)`	Удаляет нумератор из *конструктора правил нумерации*.
`DeleteRule(NumerationRulesRule)`	Удаляет правило нумерации из *конструктора правил нумерации*.

Имя	Описание
DeleteRuleItem(NumerationRulesRuleItem, NumerationRulesRule)	Удаляет строку правила нумерации из *конструктора правил нумерации*.
FindRuleBy(NumerationRulesRuleItem)	Возвращает правило нумерации, которому принадлежит указанная строка.
GetNumber(BaseCard, Guid)	Возвращает из карточки номер с указанным идентификатором.

INumerationRulesService.AddNewCondition — метод (NumerationRulesRuleItem)

Добавляет условие в строку правила нумерации в *конструкторе правил нумерации*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
NumerationRulesRuleItemCondition AddNewCondition(NumerationRulesRuleItem ruleItem)
```

Параметры

ruleItem

Тип: NumerationRulesRuleItem

Строка правила нумерации, в которую добавляется условие

Возвращаемое значение

Тип: NumerationRulesRuleItemCondition

Добавленное условие

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр ruleItem.

INumerationRulesService.AddNewNumerator — метод

Добавляет новый нумератор в *конструктор правил нумерации*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
NumerationRulesNumerator AddNewNumerator()
```

Возвращаемое значение

Тип: NumerationRulesNumerator

Нумератор в *конструкторе правил нумерации*

INumerationRulesService.AddNewRule — метод

Создаёт новое правило нумерации в *конструкторе правил нумерации*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
NumerationRulesRule AddNewRule()
```

Возвращаемое значение

Тип: NumerationRulesRule

Новое правило нумерации

INumerationRulesService.AddNewRuleItem — метод (NumerationRulesRule)

Добавляет новую строчку в правило нумерации.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
NumerationRulesRuleItem AddNewRuleItem(NumerationRulesRule rule)
```

Параметры

rule

Тип: NumerationRulesRule

Правило нумерации, в которое добавляется строка

Возвращаемое значение

Тип: NumerationRulesRuleItem

Новая строка в правиле нумерации

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр rule.

INumerationRulesService.CreateNumber — метод (CardData, BaseCard, NumerationRulesRule)

Выделяет указанной карточке номер из нумератора следуя приложенному правилу нумерации.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardNumber CreateNumber(CardData cardData, BaseCard baseCard, NumerationRulesRule
rule)
```

Параметры

cardData

Тип: CardData

Карточка, которой выделяется номер

baseCard

Тип: BaseCard

Объектная модель карточки, которой выделяется номер

rule

Тип: `NumerationRulesRule`

Правило нумерации, по которому будет выделен номер

Возвращаемое значение

Тип: `BaseCardNumber`

Выделенный номер

Примеры

Ниже приведён пример выделения номера из нумератора для карточки.

```
①

INumerationRulesService numerationRulesService = objectContext.GetService
<INumerationRulesService>(); ②

QueryObject query = new QueryObject(RefNumerationRules.Rules.RuleName, "УД Исходящий");
NumerationRulesRule numerationRule = objectContext.FindObject<NumerationRulesRule>(
query); ③

Guid cardId = new Guid(new Guid("00000000-0000-0000-0000-000000000000")); ④

numerationRulesService.CreateNumber(userSession.CardManager.GetCardData(cardId),
objectContext.GetObject<Document>(cardId), numerationRule); ⑤
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение правила нумерации, по которому выделяется номер.

④ Идентификатор карточки, которой выделяется номер.

⑤ Выделение номера.

Если необходимо использовать выделенный номер в качестве регистрационного номера документа, достаточно записать идентификатор выделенного номера в поле `RegNumber` карточки.

INumerationRulesService.CreateTextNumber — метод (BaseCard, String)

Создаёт объект типа `BaseCardNumber`, содержащий текстовое представление номера.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardNumber CreateTextNumber(BaseCard baseCard, string text)
```

Параметры

baseCard

Тип: BaseCard

Карточка, в которую добавляется номер

text

Тип: System.String

Текстовая метка добавляемого номера

Возвращаемое значение

Тип: BaseCardNumber

Номер, добавленный в номера карточки

INumerationRulesService.DeleteCondition — метод (NumerationRulesRuleItemCondition, NumerationRulesRuleItem)

Удаляет условие из строки правила нумерации.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteCondition(NumerationRulesRuleItemCondition condition, NumerationRulesRuleItem
ruleItem)
```

Параметры

condition

Тип: NumerationRulesRuleItemCondition

Удаляемое условие

ruleItem

Тип: NumerationRulesRuleItem

Строка, из которой удаляется условие

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр condition или ruleItem.

INumerationRulesService.DeleteNumber — метод (BaseCardNumber)

Удаляет указанный номер из владеющей им карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteNumber(BaseCardNumber domainObject)
```

Параметры

domainObject

Тип: BaseCardNumber

Удаляемый номер

Заметки

Метод самостоятельно определяет владельца резерва номера, удаляет из списка номеров владельца указанный номер, а также освобождает номер в нумераторе.

INumerationRulesService.DeleteNumerator — метод (NumerationRulesNumerator)

Удаляет нумератор из *конструктора правил нумерации*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool DeleteNumerator(NumerationRulesNumerator numerator)
```

Параметры

numerator

Тип: NumerationRulesNumerator

Удаляемый нумератор

Возвращаемое значение

Тип: System.Boolean

true — успешно удалено, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр numerator.

Заметки

Метод не удаляет соответствующую карточку нумератора.

INumerationRulesService.DeleteRule — метод (NumerationRulesRule)

Удаляет правило нумерации из *конструктора правил нумерации*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteRule(NumerationRulesRule rule)
```

Параметры

rule

Тип: NumerationRulesRule

Удаляемое правило нумерации

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр rule.

INumerationRulesService.DeleteRuleItem — метод (NumerationRulesRuleItem, NumerationRulesRule)

Удаляет строку правила нумерации из *конструктора правил нумерации*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteRuleItem(NumerationRulesRuleItem ruleItem, NumerationRulesRule rule)
```

Параметры

ruleItem

Тип: NumerationRulesRuleItem

Удаляемая строка правила нумерации

rule

Тип: NumerationRulesRule

Правило нумерации, из которого удаляется строка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр ruleItem или rule.

INumerationRulesService.FindRuleBy — метод (NumerationRulesRuleItem)

Возвращает правило нумерации, которому принадлежит указанная строка.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
NumerationRulesRule FindRuleBy(NumerationRulesRuleItem ruleItem)
```

Параметры

ruleItem

Тип: NumerationRulesRuleItem

Строка искомого правила нумерации

Возвращаемое значение

Тип: NumerationRulesRule

Правило нумерации, в котором присутствует строка ruleItem

INumerationRulesService.GetNumber — метод (BaseCard, Guid)

Возвращает из карточки номер с указанным идентификатором.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardNumber GetNumber(BaseCard baseCard, Guid numberId)
```

Параметры

baseCard

Тип: BaseCard

Карточка, из которой должен быть полу

numberId

Тип: System.Guid

Идентификатор номера в карточке

Возвращаемое значение

Тип: BaseCardNumber

Номер в карточке

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр baseCard или numberId.

Заметки

Идентификатор номера в карточке отличается от идентификатора номера в нумераторе.

INumeratorCardService — интерфейс

Сервис для работы с карточкой нумератора.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface INumeratorCardService
```

Методы

Имя	Описание
AddNumeratorCard	Создаёт экземпляр карточки нумератора.
AddNumeratorRange(NumeratorCard, Int32, Int32, Guid)	Добавляет в нумератор новый диапазон номеров.
AddNumeratorZone(NumeratorCard, String)	Создаёт дополнительную зону нумератора.
DeleteNumeratorCard(Guid)	Удаляет нумератор.
DeleteNumeratorCard(NumeratorCard)	Удаляет нумератор.
DeleteNumeratorRange(NumeratorRange, NumeratorCard)	Удаляет диапазон номеров из нумератора.
DeleteNumeratorZone(NumeratorZone, NumeratorCard)	Удаляет зону нумерации из нумератора.

Имя	Описание
GetNumeratorCard(Guid)	Возвращает карточку нумератора с указанным идентификатором.
NumeratorCardReadOnly(NumeratorCard)	Возвращает признак того, что нумератор доступен только на чтение.

INumeratorCardService.AddNumeratorCard — метод

Создаёт экземпляр карточки нумератора.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
NumeratorCard AddNumeratorCard()
```

Возвращаемое значение

Тип: NumeratorCard

Карточка нумератора

INumeratorCardService.AddNumeratorRange — метод (NumeratorCard, Int32, Int32, Guid)

Добавляет в нумератор новый диапазон номеров.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
NumeratorRange AddNumeratorRange(NumeratorCard card, int startNumber, int endNumber, Guid
ownerId)
```

Параметры

card

Тип: NumeratorCard

Нумератор, в который добавляется новый диапазон

startNumber

Тип: System.Int32

Начальный номер диапазона

endNumber

Тип: System.Int32

Конечный номер диапазона

ownerId

Тип: System.Guid

Начальный номер диапазона

Возвращаемое значение

Тип: NumeratorRange

Диапазон нумератора

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр card.

INumeratorCardService.AddNumeratorZone — метод (NumeratorCard, String)

Добавляет в нумератор новую зону нумерации.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
NumeratorZone AddNumeratorZone(NumeratorCard card, string name)
```

Параметры

card

Тип: NumeratorCard

Нумератор, в который добавляется новая зона

name

Тип: System.String

Название добавляемой зоны

Возвращаемое значение

Тип: NumeratorZone

Зона нумерации

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр card.

INumeratorCardService.DeleteNumeratorCard — метод (Guid)

Удаляет нумератор.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteNumeratorCard(NumeratorCard card)
```

Параметры

card

Тип: NumeratorCard

Удаляемый нумератор

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр card.

INumeratorCardService.DeleteNumeratorCard — метод (NumeratorCard)

Удаляет нумератор.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteNumeratorCard(Guid cardId)
```

Параметры

cardId

Тип: System.Guid

Идентификатор удаляемого нумератора

INumeratorCardService.DeleteNumeratorRange — метод (NumeratorRange, NumeratorCard)

Удаляет диапазон номеров из нумератора.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteNumeratorRange(NumeratorRange range, NumeratorCard card)
```

Параметры

range

Тип: NumeratorRange

Удаляемый диапазон

card

Тип: NumeratorCard

Нумератор, из которого удаляется диапазон номеров

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр range или card.

INumeratorCardService.DeleteNumeratorZone — метод (NumeratorZone, NumeratorCard)

Удаляет зону нумерации из нумератора.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteNumeratorZone(NumeratorZone zone, NumeratorCard card)
```

Параметры

zone

Тип: NumeratorZone

Удаляемая зона нумерации

card

Тип: NumeratorCard

Нумератор, из которого удаляется зона нумерации

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр zone или card.

INumeratorCardService.GetNumeratorCard — метод (Guid)

Возвращает карточку нумератора с указанным идентификатором.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
NumeratorCard GetNumeratorCard(Guid cardId)
```

Параметры

cardId

Тип: `System.Guid`

Идентификатор карточки нумератора

Возвращаемое значение

Тип: `NumeratorCard`

Нумератор

INumeratorCardService.NumeratorCardReadOnly — метод (NumeratorCard)

Возвращает признак того, что нумератор доступен только на чтение.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
bool NumeratorCardReadOnly(NumeratorCard card)
```

Параметры

card

Тип: `NumeratorCard`

Нумератор, для которого выполняется проверка

Возвращаемое значение

Тип: `System.Boolean`

`true` — нумератор доступен только на чтение, иначе — `false`

IOcspService — интерфейс

Описывает функциональность сервиса для работы с центром установления статуса сертификатов по протоколу OCSP.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface IOcspService
```

Методы

Имя	Описание
`Verify(X509Certificate2)`	Проверяет статус сертификата. Если сертификат отозван, либо недействителен, должно быть создано исключение.

Заметки

Для работы с сервисом (возможно только из скрипта карточки), должны быть выполнены настройки, приведенные в /dv6/backoffice/6.1/admin/system-settings/#signature-cypher[документации по администрированию] модуля *Базовые объекты*.

IPartnersService — интерфейс

Описывает сервис для работы со справочником контрагентов.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface IPartnersService
```

Методы

Имя	Описание
`AddNewBankAccount(PartnersCompany)`	Добавляет контрагенту банковские реквизиты.
`AddNewCode(PartnersCompany)`	Добавляет новый элемент в коды контрагента.
`AddNewContact(PartnersCompany)`	Добавляет контактное лицо.
`AddNewEmployee(PartnersCompany)`	Добавляет нового сотрудника в указанную организацию/подразделение.

Имя	Описание
AddNewEmployeeFormatItem(PartnersCompany)	Добавляет дополнительное условие в формат отображения данных о сотрудниках.
AddNewGroup(PartnersGroup)	Создаёт новую партнерскую группу.
AddNewNameCase(PartnersEmployee, PartnersNameCaseNameCase)	Создаёт у сотрудника заданный падеж имени.
AddNewOrgType	Добавляет новый тип юридического лица.
AddNewPosition	Добавляет новую должность.
AddNewTitle	Добавляет форму обращения к сотрудникам контрагентов.
AddNewUnit(PartnersCompany)	Добавляет подразделение в заданную организацию контрагента.
AddNewViewField(ObjectBase, GridViewFieldCollectionType)	Добавляет отображаемое поле для сотрудников, групп или подразделений.
AddUnitToGroup(PartnersGroup, PartnersCompany)	Добавляет компанию в указанную группу компаний.
ApplyDisplayFormatToEmployees(PartnersCompany, Boolean)	Применяет установленный формат отображения данных о сотрудниках к сотрудникам организации. Возможно применение ко всем подчиненным подразделениям.
CanMoveGroup(PartnersGroup, PartnersGroup)	Определяет возможность перемещения группы.
CanMoveUnit(PartnersCompany, PartnersCompany)	Определяет возможность перемещения подразделения.
CanMoveViewField(IGridViewField, GridViewFieldCollectionType, Boolean)	Определяет возможность перемещения отображаемого поля.
CopyGroup(PartnersGroup, PartnersGroup)	Копирует группу в указанную группу.
DeleteBankAccount(PartnersBankAccount)	Удаляет заданные банковские реквизиты.

Имя	Описание
DeleteCode(PartnersCode)	Удаляет указанные элемент из кодов контрагента.
DeleteContact(PartnersContact)	Удаляет контакт.
DeleteEmployee(PartnersEmployee)	Удаляет сотрудника контрагента.
DeleteEmployeeFormatItem(PartnersCompany, PartnersEmployeesFormat)	Удаляет условие из формата отображения данных о сотрудниках.
DeleteGroup(PartnersGroup)	Удаляет группу.
DeleteOrgType(PartnersOrgType)	Удаляет тип юридического лица.
DeletePosition(PartnersPosition)	Удаляет указанную должность.
DeleteTitle(PartnersTitle)	Удаляет форму обращения к сотрудникам контрагента.
DeleteUnit(PartnersCompany)	Удаляет заданное подразделение.
DeleteViewField(IGridViewField, GridViewFieldCollectionType)	Удаляет указанное отображаемое поле.
FindCompanyByNameOnServer(PartnersCompany, String)	Осуществляет поиск подразделения с заданным названием.
FindCompanyWithSameNameOnServer(PartnersCompany, String)	Выполняет поиск организации с именем, аналогичным приведенному.
FindDepartmentWithSameNameOnServer(PartnersCompany, String)	Выполняет поиск подразделения с именем, аналогичным приведенному.
FindDublicateCompaniesOnServer(PartnersCompany)	Осуществляет поиск дубликатов контрагента. В возвращаемом значении указаны название атрибутов по которым обнаружено дублирование. Уникальные атрибуты должны быть заданы в *Справочнике контрагентов*.
FindSameCompanyOnServer(PartnersCompany, String, String)	Выполняет поиск организации с указанным именем, либо заданным ИНН.

Имя	Описание
FindUnitGroups(PartnersCompany)	Возвращает список всех групп, в которые входит заданное подразделение контрагента.
FindViewField(ViewCardFieldsGroup, IGridViewField)	Осуществляет поиск отображаемого поля.
Get(Guid)	Возвращает сотрудника контрагента с заданным идентификатором.
GetAllEmployeeViewFields	Возвращает весь список отображаемых полей сотрудников.
GetAllUnitViewFields	Возвращает весь список отображаемых полей групп.
GetCompany(Guid)	Возвращает подразделение с заданным идентификатором.
GetEmployeeAllFormatFields	Возвращает весь список, составленный на основе схемы метаданных карточки, отображаемых полей сотрудников.
GetEmployeeCardFieldValue(PartnersEmployee, Guid, String)	Возвращает значение указанного поля карточки сотрудника контрагента.
GetEmployeeCardKind(PartnersEmployee)	Возвращает вид карточки сотрудника контрагента.
GetEmployeeDefaultViewFields	Возвращает список базовых отображаемых полей сотрудников.
GetEmployeeDisplayString(PartnersCompany, PartnersEmployee)	Возвращает отображаемое имя сотрудника, полученное на основе параметров указанной организации.
GetEmployeeFieldDisplayName(Guid, String)	Возвращает отображаемое название указанного поля карточки сотрудников.
GetEmployeesCount(PartnersCompany, Boolean, Boolean)	Получает количество сотрудников контрагента с учетом иерархии.
GetEmployeesDisplayFormat(PartnersCompany)	Возвращает формат отображения данных о сотрудниках для указанной компании.

Имя	Описание
GetGroupInheritedViewFields(PartnersGroup)	Возвращает унаследованные от родительской группы отображаемые поля.
GetGroupItems(PartnersGroup, Boolean)	Возвращает список подгрупп заданной группы.
GetUnitCardFieldValue(PartnersCompany, Guid, String)	Возвращает значение отображаемого поля подразделения.
GetUnitCardKind(PartnersCompany)	Возвращает вид карточки подразделения.
GetUnitDefaultViewFields	Возвращает базовый список отображаемых полей подразделения.
GetUnitEmployeeCardKind(PartnersCompany)	Возвращает вид карточки сотрудника установленный для заданного подразделения.
GetUnitEmployees(PartnersCompany, Boolean)	Возвращает коллекцию сотрудников из указанного подразделения.
GetUnitEmployees(PartnersCompany, Boolean, Boolean)	Возвращает коллекцию сотрудников из указанного подразделения, а также из всех подчиненных подразделений.
GetUnitFieldDisplayName(Guid, String)	Возвращает отображаемое имя поля карточки подразделения контрагента.
GetUnitInheritedViewFields(PartnersCompany, Boolean)	Возвращает унаследованные отображаемые поля подразделения.
GetUnits(PartnersCompany, Boolean)	Предоставляет коллекцию подразделений, входящих в указанное подразделение.
GetUnits(PartnersCompany, Boolean, Boolean)	Предоставляет коллекцию подразделений, входящих в указанное подразделение, с учетом иерархии.

Имя	Описание
`GetViewFields(ObjectBase, GridViewFieldCollectionType)`	Возвращает отображаемые поля для указанного объекта (группа, компания или сотрудник). Все возвращаемые записи будут приведены к типу IGridViewField.
`GetViewFieldsAsObject(ObjectBase, GridViewFieldCollectionType)`	Возвращает отображаемые поля для указанного объекта (группа, компания или сотрудник) без приведения типов.
`MoveEmployees(IEnumerable<PartnersEmployee>, PartnersCompany)`	Перемещает выбранных сотрудников в выбранное подразделение.
`MoveGroup(PartnersGroup, PartnersGroup)`	Перемещает выбранную подгруппу в заданную группу.
`MoveUnit(PartnersCompany, PartnersCompany)`	Перемещает выбранное подразделение.
`MoveViewField(IGridViewField, GridViewFieldCollectionType, Boolean)`	Перемещает выбранное отображаемое поле.
`OpenOrCreateDepartmentCard(PartnersCompany, Boolean)`	Возвращает карточку подразделения контрагента; при отсутствии карточки, она будет создана.
`OpenOrCreateEmployeeCard(PartnersEmployee, Boolean)`	Возвращает карточку сотрудника контрагента. В случае отсутствии карточки, будет создана новая.
`OrgTypeWithSameNameExists(PartnersOrgType, String)`	Возвращает признак наличия в *справочнике контрагентов* типа юридического лица с указанным названием, но не эквивалентным переданному объекту.
`OtherPositionWithSameNameExists(PartnersPosition, String)`	Возвращает признак наличия в *справочнике контрагентов* должности с указанным названием, но не эквивалентным переданному объекту.

Имя	Описание
`OtherTitleWithSameNameExists(PartnersTitle, String)`	Возвращает признак наличия в *справочнике контрагентов* обращения (к сотруднику) с указанным названием, но не эквивалентным переданному объекту.
`PropagateNotAvailable(PartnersCompany)`	Распространяет значение флага "Не показывать при выборе", установленное у заданного подразделения, на все дочерние элементы.
`RemoveGroupItems(PartnersGroup, IEnumerable<PartnersGroupGroup>)`	Удаляет заданные элементы из группы контрагентов.
`RemoveGroupUnit(PartnersGroup, PartnersCompany)`	Удаляет из группы указанного контрагента.
`RemoveGroupUnits(PartnersGroup, IEnumerable<PartnersCompany>)`	Удаляет список контрагентов из группы.
`SetCompanyUniqueAttributes(IEnumerable<CardFieldInfo>)`	Задаёт атрибуты организации, проверяемые на уникальность.
`SetCompanyUniqueAttributesUnion(PartnersCompanyUniqueAttributesOperation)`	Задаёт тип объединения для атрибутов организации, проверяемых на уникальность.

Заметки

Тип контейнера (организация, либо подразделение) определяется значением свойства `StaffUnit.Type`.

IPartnersService.AddNewEmployee — метод (PartnersCompany)

Добавляет нового сотрудника в указанную организацию/подразделение.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
PartnersEmployee AddNewEmployee(PartnersCompany unit)
```

Параметры

unit

Тип: PartnersCompany

Подразделение, в которое добавляется сотрудник

Возвращаемое значение

Тип: PartnersEmployee

Сотрудник, добавленный в организацию

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр unit.

Примеры

```
①

IPartnersService partnersService = objectContext.GetService<IPartnersService>(); ②

PartnersCompany partnersCompany = objectContext.GetObject<PartnersCompany>(new Guid
("00000000-0000-0000-0000-000000000000")); ③

PartnersEmployee partnersEmployee = partnersService.AddNewEmployee(partnersCompany);
partnersEmployee.LastName = "Иванов";
objectContext.SaveObject(partnersEmployee); ④
```

① Инициализация контекста объектов.

② Получение сервиса для работы с контрагентами.

③ Получение организации, в которую добавляется сотрудник.

④ Добавление и сохранение сотрудника.

IPartnersService.AddNewUnit — метод (PartnersCompany)

Добавляет подразделение в заданную организацию контрагента.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public PartnersCompany AddNewUnit(PartnersCompany parentUnit)
```

Параметры

parentUnit

Тип: PartnersCompany

Организация контрагента, в которую будет добавлено новое подразделение

Возвращаемое значение

Тип: PartnersCompany

Созданная организация/подразделение

Заметки

Если parentUnit не задан, организация будет добавлена в корневой каталог организаций.

IPartnersService.DeleteEmployee — метод (PartnersEmployee)

Удаляет сотрудника контрагента

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteEmployee(PartnersEmployee employee)
```

Параметры

employee

Тип: PartnersEmployee

Сотрудник контрагента

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр employee.

Примеры

```
①

IPartnersService partnersService = objectContext.GetService<IPartnersService>(); ②

PartnersEmployee partnersEmployee = partnersService.Get(new Guid("00000000-0000-0000-
0000-000000000000")); ③

partnersService.DeleteEmployee(partnersEmployee); ④
```

① Инициализация контекста объектов.

② Получаем сервис для работы с контрагентами.

③ Получаем сотрудника контрагента.

④ Удаляем сотрудника.

IPartnersService.FindCompanyByNameOnServer — метод (PartnersCompany, String)

Осуществляет поиск подразделения с заданным названием.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
PartnersCompany FindCompanyByNameOnServer(PartnersCompany parentUnit, string unitName)
```

Параметры

parentUnit

Тип: PartnersCompany

Организация, в которой выполняется поиск дочернего подразделения/организации

unitName

Тип: System.String

Название искомого подразделения

Возвращаемое значение

Тип: PartnersCompany

Организация с указанным названием

Заметки

Если parentUnit не задан, то будет осуществлен поиск в корневом каталоге подразделений.

IPartnersService.FindUnitGroups — метод (PartnersCompany)

Возвращает список всех групп, в которые входит заданное подразделение контрагента.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
List<PartnersGroup> FindUnitGroups(PartnersCompany company)
```

Параметры

company

Тип: PartnersCompany

Организация/подразделение контрагента

Возвращаемое значение

Тип: System.Collections.Generic.List<PartnersGroup>

Список групп, в которые входит подразделение

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `company`.

IPowerOfAttorneyMachineReadableProvider — интерфейс

Определяет методы формирования и чтения машиночитаемой доверенности.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface IPowerOfAttorneyMachineReadableProvider
```

Свойства

Имя	Описание
`PowerOfAttorneyMachineReadableEncoding`	Кодировка файла МЧД

Методы

Имя	Описание
`string CreatePowerOfAttorneyMachineReadable(PowerOfAttorneyData);`	Формирует содержимое машиночитаемой доверенности на основе данных доверенности *Параметры:* • `powerOfAttorneyData` — данные для формирования МЧД Возвращаемое значение: содержимое МЧД

Имя	Описание
`string CreatePowerOfAttorneyMachineReadable(PowerOfAttorneyData, PowerOfAttorneyHandlingFlags);`	Формирует содержимое машиночитаемой доверенности на основе данных доверенности *Параметры:* • `powerOfAttorneyData` — Данные для формирования МЧД • `flags` — Управляющие флаги Возвращаемое значение: содержимое МЧД.
`CreatePowerOfAttorneyRevokeRequest(PowerOfAttorneyRevocationData, string);`	Формирует заявление на отзыв доверенности. *Параметры:* • `revocationData` — содержимое МЧД. • `powerOfAttorneyMachineReadableString` — заявление на отзыв. Возвращаемое значение: заявление на отзыв.
`Validate(string, out string)`	Проверяет данные МЧД на соответствие формату *Параметры:* • `powerOfAttorneyMachineReadableData` — Содержимое МЧД. • `error` — Информация об ошибках. Возвращаемое значение: результат проверки.

Имя	Описание
`GetPowerOfAttorneyMachineReadableInfo(string);`	Возвращает информацию о машиночитаемой доверенности из содержимого машиночитаемой доверенности. *Параметры:* • `powerOfAttorneyMachineReadableString` — Содержимое МЧД в строковом виде. Возвращаемое значение: Информация о машиночитаемой доверенности.
`GetPowerOfAttorneyRevocationInfo(string);`	Возвращает информацию об отзыве доверенности. *Параметры:* • `powerOfAttorneyRevocationString` — Содержимое заявления на отзыв доверенности. Возвращаемое значение: Информация об отзыве доверенности.

IPowerOfAttorneyService — интерфейс

Интерфейс предоставляет методы для работы с системной карточкой доверенности.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface IPowerOfAttorneyService
```

Методы

Имя	Описание
`CreatePowerOfAttorney(PowerOfAttorneyData, StaffEmployee, StaffEmployee, PowersPowerOfAttorneyFormat);`	Создаёт новую СКД для представителя — физического лица *Параметры:* • `powerOfAttorneyData` — Данные доверенности • `representative` — Представитель • `signer` — Подписант • `powerOfAttorneyFormat` — Формат доверенности Возвращаемое значение: карточка доверенности
`CreatePowerOfAttorney(PowerOfAttorneyData, StaffEmployee, StaffEmployee, PowersPowerOfAttorneyFormat, PowerOfAttorneyHandlingFlags)`	Создаёт новую СКД для представителя — физического лица *Параметры:* • `powerOfAttorneyData` — Данные доверенности • `representative` — Представитель • `signer` — Подписант • `powerOfAttorneyFormat` — Формат доверенности • `flags` — Параметры обработки Возвращаемое значение: карточка доверенности

Имя	Описание
`CreatePowerOfAttorney(PowerOfAttorneyData, StaffUnit, StaffEmployee, PowersPowerOfAttorneyFormat);`	Создаёт новую СКД для представителя — юридического лица *Параметры:* • `powerOfAttorneyData` — Данные доверенности • `representative` — Представитель • `signer` — Подписант • `powerOfAttorneyFormat` — Формат доверенности Возвращаемое значение: карточка доверенности
`CreatePowerOfAttorney(PowerOfAttorneyData, StaffUnit, StaffEmployee, PowersPowerOfAttorneyFormat, PowerOfAttorneyHandlingFlags);`	Создаёт новую СКД для представителя — юридического лица *Параметры:* • `powerOfAttorneyData` — Данные доверенности • `representative` — Представитель • `signer` — Подписант • `powerOfAttorneyFormat` — Формат доверенности • `flags` — Параметры обработки Возвращаемое значение: карточка доверенности

Имя	Описание
`CreatePowerOfAttorney(PowerOfAttorneyData, StaffUnit, StaffEmployee, StaffEmployee, PowersPowerOfAttorneyFormat, PowerOfAttorneyHandlingFlags);`	Создаёт новую СКД для представителя — юридического лица *Параметры:* • `powerOfAttorneyData` — Данные доверенности • `representative` — Представитель • `representativeManager` — Руководитель организации-представителя • `signer` — Подписант • `powerOfAttorneyFormat` — Формат доверенности • `flags` — Параметры обработки Возвращаемое значение: карточка доверенности
`RetrustPowerOfAttorney(PowerOfAttorneyData, StaffEmployee, StaffEmployee, PowerOfAttorney);`	Создаёт СКД в рамках передоверия для представителя — физического лица *Параметры:* • `powerOfAttorneyData` — Данные доверенности • `representative` — Представитель • `signer` — Подписант • `parentPowerOfAttorney` — Доверенность, на основании которой осуществляется передоверие Возвращаемое значение: карточка доверенности

Имя	Описание
`RetrustPowerOfAttorney(PowerOfAttorneyData, StaffEmployee, StaffEmployee, PowerOfAttorney, PowerOfAttorneyHandlingFlags);`	Создаёт СКД в рамках передоверия для представителя — физического лица *Параметры:* • `powerOfAttorneyData` — Данные доверенности • `representative` — Представитель • `signer` — Подписант • `parentPowerOfAttorney` — Доверенность, на основании которой осуществляется передоверие • `flags` — Параметры обработки Возвращаемое значение: карточка доверенности
`RetrustPowerOfAttorney(PowerOfAttorneyData, StaffUnit, StaffEmployee, PowerOfAttorney);`	Создаёт СКД в рамках передоверия для представителя — юридического лица *Параметры:* • `powerOfAttorneyData` — Данные доверенности • `representative` — Представитель • `signer` — Подписант • `parentPowerOfAttorney` — Доверенность, на основании которой осуществляется передоверие Возвращаемое значение: карточка доверенности

Имя	Описание
`RetrustPowerOfAttorney(PowerOfAttorneyData, StaffUnit, StaffEmployee, PowerOfAttorney, PowerOfAttorneyHandlingFlags);`	Создаёт СКД в рамках передоверия для представителя — юридического лица *Параметры:* • `powerOfAttorneyData` — Данные доверенности • `representative` — Представитель • `signer` — Подписант • `parentPowerOfAttorney` — Доверенность, на основании которой осуществляется передоверие • `flags` — Параметры обработки Возвращаемое значение: карточка доверенности
`RetrustPowerOfAttorney(PowerOfAttorneyData, StaffUnit, StaffEmployee, StaffEmployee, PowerOfAttorney, PowerOfAttorneyHandlingFlags);`	Создаёт СКД в рамках передоверия для представителя - юридического лица *Параметры:* • `powerOfAttorneyData` — Данные доверенности • `representative` — Представител • `representativeManager` — Руководитель организации-представителя • `signer` — Подписант • `parentPowerOfAttorney` — Доверенность, на основании которой осуществляется передоверие • `flags` — Параметры обработки Возвращаемое значение: карточка доверенности

Имя	Описание
`ImportPowerOfAttorney(byte[], byte[], StaffEmployee, StaffEmployee, PowersPowerOfAttorneyFormat, PowerOfAttorneySignatureFormat, Func<string, ObjectContext, PowersCode>, PowerOfAttorneyHandlingFlags);`	Создаёт новую СКД для представителя — физического лица из файла МЧД *Параметры:* • `machineReadableData` — Данные МЧД с электронной подписью • `signatureData` — Данные электронной подписи для формата CADES • `representative` — Представитель • `signer` — Подписант • `powerOfAttorneyFormat` — Формат доверенности • `signatureFormat` — Формат подписи • `powersCodeResolve` — Функция получения кода полномочия Справочника полномочий по коду полномочия из МЧД • `flags` — Параметры обработки Возвращаемое значение: Карточка доверенности Примечание: при формате подписи XMLSIG, `signatureData` не передаётся

Имя	Описание
`ImportPowerOfAttorney(byte[], byte[], StaffEmployee, StaffEmployee, PowerOfAttorney, PowerOfAttorneySignatureFormat, Func<string, ObjectContext, PowersCode>, PowerOfAttorneyHandlingFlags);`	Создаёт новую СКД для представителя — физического лица из файла МЧД, выданной в порядке передоверия *Параметры:* • `machineReadableData` — Данные МЧД с электронной подписью • `signatureData` — Данные электронной подписи для формата CADES • `representative` — Представитель • `signer` — Подписант • `parentPowerOfAttorney` — Доверенность, на основании которой осуществляется передоверие • `signatureFormat` — Формат подписи • `powersCodeResolve` — Функция получения кода полномочия Справочника полномочий по коду полномочия из МЧД • `flags` — Параметры обработки Возвращаемое значение: Карточка доверенности Примечание: при формате подписи XMLSIG, `signatureData` не передаётся

Имя	Описание
`ImportPowerOfAttorney(byte[], byte[], StaffUnit, StaffEmployee, PowersPowerOfAttorneyFormat, PowerOfAttorneySignatureFormat, Func<string, ObjectContext, PowersCode>, PowerOfAttorneyHandlingFlags);`	Создаёт новую СКД для представителя — юридического лица из файла МЧД *Параметры:* • `machineReadableData` — Данные МЧД с электронной подписью • `signatureData` — Данные электронной подписи для формата CADES • `representative` — Представитель • `signer` — Подписант • `powerOfAttorneyFormat` — Формат доверенности • `signatureFormat` — Формат подписи • `powersCodeResolve` — Функция получения кода полномочия Справочника полномочий по коду полномочия из МЧД • `flags` — Параметры обработки Возвращаемое значение: Карточка доверенности Примечание: при формате подписи XMLSIG, `signatureData` не передаётся

Имя	Описание
`PowerOfAttorney ImportPowerOfAttorney(byte[], byte[], StaffUnit, StaffEmployee, StaffEmployee, PowersPowerOfAttorneyFormat, PowerOfAttorneySignatureFormat, Func<string, ObjectContext, PowersCode>, PowerOfAttorneyHandlingFlags);`	Создаёт новую СКД для представителя - юридического лица из файла МЧД *Параметры:* • `machineReadableData` — Данные МЧД с электронной подписью • `signatureData` — Данные электронной подписи для формата CADES • `representative` — Представитель • `representativeManager` — Руководитель организации-представителя • `signer` — Подписант • `powerOfAttorneyFormat` — Формат доверенности • `signatureFormat` — Формат подписи • `powersCodeResolve` — Функция получения кода полномочия Справочника полномочий по коду полномочия из МЧД • `flags` — Параметры обработки Возвращаемое значение: Карточка доверенности Примечание: при формате подписи XMLSIG, `signatureData` не передаётся

Имя	Описание
`ImportPowerOfAttorney(byte[], byte[], StaffUnit, StaffEmployee, PowerOfAttorney, PowerOfAttorneySignatureFormat, Func<string, ObjectContext, PowersCode>, PowerOfAttorneyHandlingFlags);`	Создаёт новую СКД для представителя — юридического лица из файла МЧД, выданной в порядке передоверия *Параметры:* • `machineReadableData` — Данные МЧД с электронной подписью • `signatureData` — Данные электронной подписи для формата CADES • `representative` — Представитель • `signer` — Подписант • `parentPowerOfAttorney` — Доверенность, на основании которой осуществляется передоверие • `signatureFormat` — Формат подписи • `powersCodeResolve` — Функция получения кода полночия Справочника полномочий по коду полночия из МЧД • `flags` — Параметры обработки Возвращаемое значение: Карточка доверенности Примечание: при формате подписи XMLSIG, `signatureData` не передаётся

Имя	Описание
`PowerOfAttorney ImportPowerOfAttorney(byte[], byte[], StaffUnit, StaffEmployee, StaffEmployee, PowerOfAttorney, PowerOfAttorneySignatureFormat, Func<string, ObjectContext, PowersCode>, PowerOfAttorneyHandlingFlags);`	Создаёт новую СКД для представителя - юридического лица из файла МЧД *Параметры:* • `machineReadableData` — Данные МЧД с электронной подписью • `signatureData` — Данные электронной подписи для формата CADES • `representative` — Представитель • `representativeManager` — Руководитель организации-представителя • `signer` — Подписант • `parentPowerOfAttorney` — Доверенность, на основании которой осуществляется передоверие • `signatureFormat` — Формат подписи • `powersCodeResolve` — Функция получения кода полномочия Справочника полномочий по коду полномочия из МЧД • `flags` — Параметры обработки Возвращаемое значение: Карточка доверенности Примечание: при формате подписи XMLSIG, `signatureData` не передаётся

Имя	Описание
`SignPowerOfAttorney(PowerOfAttorney, X509Certificate2, PowerOfAttorneySignatureFormat signatureFormat);`	Подписывает СКД *Параметры:* • `powerOfAttorney` — СКД • `certificate` — Сертификат подписи • `signatureFormat` — Формат хранения подписи Примечание: СКД должна быть в статусе "Подготовка"; текущий пользователь должен быть указан "Подписантом" в СКД; статус доверенности будет изменён на "Действует".
`SignPowerOfAttorney(PowerOfAttorney, X509Certificate2, PowerOfAttorneySignatureFormat, PowerOfAttorneyHandlingFlags);`	Подписывает СКД *Параметры:* • `powerOfAttorney` — СКД • `certificate` — Сертификат подписи • `signatureFormat` — Формат хранения подписи • `flags` — Параметры обработки Примечание: СКД должна быть в статусе "Подготовка". Текущий пользователь должен быть указан "Подписантом" в СКД. Статус доверенности будет изменён на "Действует".

Имя	Описание
`AttachSignature(PowerOfAttorney, byte[]);`	Загружает в СКД подпись в формате CADES *Параметры:* • `powerOfAttorney` — СКД • `signature` — Данные подписи Примечание: СКД должна быть в статусе "Подготовка"; текущий пользователь должен быть указан "Подписантом" в СКД; статус доверенности будет изменён на "Действует".
`AttachSignature(PowerOfAttorney, byte[], PowerOfAttorneyHandlingFlags);`	Загружает в СКД подпись в формате CADES *Параметры:* • `powerOfAttorney` — СКД • `signature` — Данные подписи • `flags` — Параметры обработки Примечание: СКД должна быть в статусе "Подготовка"; текущий пользователь должен быть указан "Подписантом" в СКД; статус доверенности будет изменён на "Действует".
`RequestRevocation(PowerOfAttorney, PowerOfAttorneyRevocationData);`	Создаёт запрос на отзыв доверенности *Параметры:* • `powerOfAttorney` — СКД • `revocationData` — Данные заявления на отзыв Примечание: СКД должна быть в статусе "Действует"

Имя	Описание
`SignRequestRevocation(PowerOfAttorney, X509Certificate2);`	Подписывает заявление на отзыв доверенности *Параметры:* • `powerOfAttorney` — СКД • `certificate` — Сертификат подписи Примечание: Заявление на отзыв должно быть сформировано
`SignRequestRevocation(PowerOfAttorney, X509Certificate2, PowerOfAttorneyHandlingFlags);`	Подписывает заявление на отзыв доверенности. Заявление на отзыв должно быть сформировано. *Параметры:* • `powerOfAttorney` — СКД • `certificate` — Сертификат подписи • `flags` — Параметры обработки
`AttachRequestRevocationSignature(PowerOfAttorney, byte[]);`	Загружает сформированную подпись отзыва доверенности в СКД. При добавлении подписи проверяется что в её сертификате у владельца СНИЛС соответствует тому, что в МЧД. Проверка выполняется по доверителю и представителю, `PowerOfAttorneyMachineReadableInfo.RepresentativesSnils` и `PowerOfAttorneyMachineReadableInfo.PrincipalsSnils`. Если СНИЛС из сертификата не соответствует указанному для доверителя или представителя, будет возвращена ошибка. *Параметры:* • `powerOfAttorney` — СКД • `signature` — Данные подписи

Имя	Описание
`AttachRequestRevocationSignature(PowerOfAttorney, byte[], PowerOfAttorneyHandlingFlags);`	Загружает сформированную подпись отзыва доверенности в СКД. *Параметры:* • `powerOfAttorney` — СКД • `signature` — Данные подписи • `flags` — Параметры обработки
`MarkAsRevoked(PowerOfAttorney, bool);`	Помечает доверенность как отозванную без вызова `RequestRevocation` с передачей и подписью заявления на отзыв *Параметры:* • `powerOfAttorney` — СКД • `withChildsPowerOfAttorney` — Включить отзыв дочерних доверенностей

Имя	Описание
`Revoke(PowerOfAttorney, bool withChildsPowerOfAttorney);`	Отзывает доверенность *Параметры:* • `powerOfAttorney` — СКД • `withChildsPowerOfAttorney` — Включить отзыв дочерних доверенностей Примечание: отзыв доверенности должен быть запрошен с помощью метода `RequestRevocation`. Заявление должно быть подписано `SignRequestRevocation` или прикреплена подпись с помощью `AttachRequestRevocationSignature`. Если включен отзыв дочерних доверенностей, то также будут отозваны дочерние доверенности, которые не были отозваны ранее. Если дочерняя доверенность была отозвана ранее, то она, а также все её дочерние доверенности не будут отозваны повторно.
`ExportMachineReadablePowerOfAttorney(PowerOfAttorney, string, bool);`	Выгружает содержимое МЧД в указанную папку *Параметры:* • `powerOfAttorney` — СКД • `folder` — Папка для сохранения • `withSignature` — Включает выгрузку МЧД с подписью Примечание: при выгрузке МЧД с подписью в формате `XMLDSIG` будет выгружен архив с одним файлом подписанной МЧД

Имя	Описание
`ExportRevocationPowerOfAttorney(PowerOfAttorney, string, bool);`	Выгружает содержимое заявления на отзыв доверенности в указанную папку *Параметры:* • `powerOfAttorney` — СКД • `folder` — Папка для сохранения • `withSignature` — Включает выгрузку МЧД с подписью
`string GetPowerOfAttorneyDisplayString(PowerOfAttorney);`	Возвращает отображаемое название СКД *Параметры:* • `powerOfAttorney` — СКД Возвращаемое значение: отображаемое название
`string GetMachineReadablePowerOfAttorneyFileData(PowerOfAttorney, out string);`	Возвращает данные машиночитаемой доверенности *Параметры:* • `powerOfAttorney` — СКД • `fileName` — Идентификатор файла машиночитаемой доверенности Возвращаемое значение: данные машиночитаемой доверенности
`GetRevocationPowerOfAttorneyFileData(PowerOfAttorney, out string);`	Возвращает данные отзыва доверенности *Параметры:* • `powerOfAttorney` — СКД • `fileName` — Название файла отзыва доверенности Возвращаемое значение: Данные отзыва доверенности

Имя	Описание
`GetMachineReadablePowerOfAttorneyContent(PowerOfAttorney);`	Возвращает содержимое машиночитаемой доверенности *Параметры:* • `powerOfAttorney` — СКД Возвращаемое значение: содержимое машиночитаемой доверенности
`IEnumerable<PowerOfAttorneyDigest> SearchPowerOfAttorney(StaffEmployee, Guid?, Guid?, Func<PowerOfAttorney, bool>);`	Выполняет поиск доверенностей по условиям *Параметры:* • `employee` — Сотрудник, для которого ищутся доверенности • `powerOfAttorneyFormat` — Формат доверенности • `representativePowerCode` — Код требуемого полномочия по доверенности • `filter` — Дополнительные условия фильтрации найденных доверенностей Возвращаемое значение: информация о найденных доверенностях
`IEnumerable<PowerOfAttorney> GetParentalPowerOfAttorneys(PowerOfAttorney powerOfAttorney);`	Получает родительские доверенности от заданной. • `powerOfAttorney` — СКД

/dv6/programmer/dv6/BackOffice-ObjectModel-Powers/IPowerOfAttorneyMachineReadableProvider_IN/[IPowerOfAttorneyMachineReadableProvider — интерфейс]

/dv6/programmer/dv6/BackOffice-ObjectModel-Powers/IPowerOfAttorneyService_IN/[IPowerOfAttorneyService — интерфейс]

PowerOfAttorney — класс

Системная карточка доверенности.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PowerOfAttorney: BaseCard
```

Свойства

Имя	Описание
MainInfo	Основная информация.
SubsidiaryPowersOfAttorney	Системные карточки дочерних доверенностей.
RepresentativesPowers	Полномочия представителей.

Поля

Имя	Описание
MainInfoProperty	Определяет свойство "Основная информация".
SubsidiaryPowersOfAttorneyProperty	Определяет свойство "Системные карточки дочерних доверенностей".
RepresentativesPowersProperty	Определяет свойство "Полномочия представителей".

PowerOfAttorneyMainInfo — класс

Представляет секцию "Основная информация" системной карточки доверенности.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PowerOfAttorneyMainInfo : BaseCardSectionRow
```

Свойства

Имя	Описание
Status	Статус доверенности.
UserCard	Пользовательская карточка доверенности.
Signer	Подписант.
PowerOfAttorneyStartDate	Дата совершения доверенности.
PowerOfAttorneyEndDate	Срок действия доверенности.
PowerOfAttorneyNumber	Номер доверенности.
SignatureFormat	Формат подписи.
MachineReadablePowerOfAttorney	Машиночитаемая доверенность.
PowerOfAttorneyFormat	Формат МЧД.
Signature	Подпись.
Principal	Доверитель.
PrincipalINN	ИНН доверителя.
ParentalPowerOfAttorney	Родительская доверенность.
RevocationRequestDate	Дата и время поступления заявления на отзыв.
RevocationDate	Дата отзыва.
RegTransferStatus	Статус передачи в реестр.
Representative	Представитель.
RevocationApplication	Заявление на отзыв доверенности.
RevocationApplicationSignature	Подпись заявления на отзыв доверенности.

Поля

Имя	Описание
StatusProperty	Определяет свойство "Статус доверенности".

Имя	Описание
UserCard	Определяет свойство "Пользовательская карточка доверенности".
SignerProperty	Определяет свойство "Подписант".
PowerOfAttorneyStartDateProperty	Определяет свойство "Дата совершения доверенности".
PowerOfAttorneyEndDateProperty	Определяет свойство "Срок действия доверенности".
PowerOfAttorneyNumberProperty	Определяет свойство "Номер доверенности".
SignatureFormatProperty	Определяет свойство "Формат подписи".
MachineReadablePowerOfAttorneyProperty	Определяет свойство "Машиночитаемая доверенность".
PowerOfAttorneyFormat	Определяет свойство "Формат МЧД".
SignatureProperty	Определяет свойство "Подпись".
PrincipalProperty	Определяет свойство "Доверитель".
PrincipalINN	Определяет свойство "ИНН доверителя".
ParentalPowerOfAttorneyProperty	Определяет свойство "Родительская доверенность".
RevocationRequestDateProperty	Определяет свойство "Дата и время поступления заявления на отзыв".
RevocationDateProperty	Определяет свойство "Дата отзыва".
RegTransferStatusProperty	Определяет свойство "Статус передачи в реестр".
RepresentativeProperty	Определяет свойство "Представитель".
RevocationApplicationProperty	Определяет свойство "Заявление на отзыв доверенности"

Имя	Описание
`RevocationApplicationSignatureProperty`	Определяет свойство "Подпись заявления на отзыв доверенности"

PowerOfAttorneyFNSDOVEL502RevocationData — класс

Данные отзыва доверенности версии 5.02 для взаимодействия с ФНС.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PowerOfAttorneyFNSDOVEL502RevocationData : PowerOfAttorneyRevocationData
```

Свойства

Имя	Описание
`Document`	Состав и структура документа (Обязательный).
`RecipientID`	Идентификатор получателя файла заявления (Обязательный).
`FinalRecipientID`	Идентификатор конечного получателя файла заявления (Обязательный). В случае передачи файла от отправителя к конечному получателю при отсутствии налоговых органов, осуществляющих передачу на промежуточных этапах, значения `RecipientID` и `FinalRecipientID` совпадают.

Имя	Описание
SenderID	Идентификатор отправителя информации (Обязательный). *Значение:* • Для организаций — 19-разрядный код (ИНН и КПП организации). • Для физических лиц — 12-разрядный код (ИНН физического лица, при наличии. При отсутствии ИНН — последовательность из двенадцати нулей).

Классы

Имя	Описание
FIO	Фамилия, имя, отчество (ФИОТип). Таблица 4.16.
ForeignCompanyInfo	Сведения об иностранной организации (при отзыве основной доверенности) (ИО). Таблица 4.8.
IdentityCardInfo	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип). Таблица 4.15.
IndividualInfo1	Сведения о физическом лице (СведФЛТип). Таблица 4.12.
IndividualInfo	Сведения о физическом лице, в том числе индивидуальном предпринимателе (СвФизЛицТип). Таблица 4.14.
OrganizationInfo	Сведения об организации (СвОргТип). Таблица 4.13.

Имя	Описание
PowerOfAttorneyInfo	Сведения для доверенности, подтверждающей полномочия нескольких уполномоченных представителей (СвПдтвУП). Таблица 4.4.
PrincipalInfo	Сведения о доверителе (СвДоверит). Таблица 4.5.
PrincipalWithoutPowerOfAttorneyInfo	Сведения о лице, действующем от имени юридического лица без доверенности (СвРукОрг). Таблица 4.7.
RetrustPrincipalInfo	Сведения о доверителе при отзыве доверенности, совершенной (выданной) в порядке передоверия (СвДоверщП). Таблица 4.10.
RevocationDocument	Состав и структура документа (Документ). Таблица 4.2.
RevocationInfo	Сведения заявления об отзыве доверенности (СвЗаяв). Таблица 4.3.
RussianCompanyInfo	Сведения о российской организации (при отзыве основной доверенности) (НПЮЛ). Таблица 4.6.
SeparateSubdivisionManagerInfo	Сведения о руководителе обособленного подразделения (СвРукОП). Таблица 4.9.

Имя	Описание
SignerInfo	Сведения о физическом лице, подписывающем заявление об отзыве доверенности от имени доверителя (российской организации / иностранной организации) без доверенности, или сведения о физическом лице, подписывающем заявление об отзыве доверенности от своего имени, в том числе при передаче полномочий другому лицу в порядке передоверия (Подписант). Таблица 4.11.

FIO — класс

Фамилия, имя, отчество (ФИОТип). Таблица 4.16.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class FIO
```

Свойства

Имя	Описание
LastName	Фамилия (Обязательный)
FirstName	Имя (Обязательный)
MiddleName	Отчество

ForeignCompanyInfo — класс

Сведения об иностранной организации (при отзыве основной доверенности) (ИО). Таблица 4.8.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ForeignCompanyInfo
```

Свойства

Имя	Описание
Name	Наименование иностранной организации (Обязательный).
Inn	ИНН организации (Обязательный).
Kpp	КПП обособленного подразделения (Обязательный).
Manager	Сведения о руководителе обособленного подразделения (Обязательный).

IdentityCardInfo — класс

Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип). Таблица 4.15.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class IdentityCardInfo
```

Свойства

Имя	Описание
DocumentKindCode	Код вида документа (Обязательный).
DocumentSerialNumber	Серия и номер документа (Обязательный).
IssueDate	Дата выдачи документа (Обязательный).

Имя	Описание
Issuer	Наименование органа, выдавшего документ (Обязательный с условием). Обязательный, если вид документа — Паспорт гражданина Российской Федерации.
IssuerCode	Код подразделения органа, выдавшего документ.

IndividualInfo1 — класс

Сведения о физическом лице (СведФЛТип). Таблица 4.12.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class IndividualInfo1
```

Свойства

Имя	Описание
Inn	ИНН физического лица.
Snils	СНИЛС (Обязательный).
Ogrnip	ОГРНИП.
Citizenship	Гражданство. Принимает значение в соответствии с Общероссийским классификатором стран мира.
BirthDate	Дата рождения (Обязательный).
IdentityCard	Сведения о документе, удостоверяющем личность физического лица (Обязательный).

IndividualInfo — класс

Сведения о физическом лице, в том числе индивидуальном предпринимателе (СвФизЛицТип). Таблица 4.14.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class IndividualInfo
```

Свойства

Имя	Описание
Inn	ИНН физического лица.
Ogrnip	ОГРНИП (Обязательный с условием). Обязателен, если лицо является индивидуальным предпринимателем.
Snils	СНИЛС (Обязательный).
Citizenship	Гражданство. Принимает значение в соответствии с Общероссийским классификатором стран мира.
BirthDate	Дата рождения.
Fio	Фамилия, имя, отчество физического лица (Обязательный).
IdentityCard	Сведения о документе, удостоверяющем личность физического лица (Обязательный).

OrganizationInfo — класс

Сведения об организации (СвОргТип). Таблица 4.13.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class OrganizationInfo
```

Свойства

Имя	Описание
`Name`	Наименование организации / Наименование обособленного подразделения организации (Обязательный)
`Inn`	ИНН организации (Обязательный)
`Kpp`	КПП организации (Обязательный)
`Ogrn`	ОГРН

PowerOfAttorneyInfo — класс

Сведения для доверенности, подтверждающей полномочия нескольких уполномоченных представителей (СвПдтвУП). Таблица 4.4.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PowerOfAttorneyInfo
```

Свойства

Имя	Описание
`Organization`	Сведения об организации (Обязательный с условием). Обязательный, если в доверенности в качестве уполномоченного представителя указано юридическое лицо.

Имя	Описание
Individual	Сведения о физическом лице, в том числе индивидуальном предпринимателе (Обязательный). Указываются сведения об уполномоченном представителе - физическом лице, в отношении которого отзывается доверенность.

PrincipalInfo — класс

Сведения о доверителе (СвДоверит). Таблица 4.5.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrincipalInfo
```

Свойства

Имя	Описание
RussianCompany	Сведения о российской организации (Обязательный с условием). Обязательный при отзыве основной доверенности, если доверителем является российская организация.
ForeignCompany	Сведения об иностранной организации (Обязательный с условием). Обязательный при отзыве основной доверенности, если доверителем является иностранная организация.
Individual	Сведения о физическом лице (Обязательный с условием). Обязательный при отзыве основной доверенности, если доверителем является физическое лицо.

Имя	Описание
RetrustPrincipal	Сведения о доверителе при отзыве доверенности, совершенной (выданной) в порядке передоверия (Обязательный с условием). Обязательный, если производится отзыв доверенности, совершенной (выданной) в порядке передоверия полномочий.

PrincipalWithoutPowerOfAttorneyInfo — класс

Сведения о лице, действующем от имени юридического лица без доверенности (СвРукОрг). Таблица 4.7.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrincipalWithoutPowerOfAttorneyInfo
```

Свойства

Имя	Описание
Position	Должность (Обязательный).
Organization	Сведения об организации (Обязательный с условием). Обязательный, если лицом, действующим от имени юридического лица, является организация.

RetrustPrincipalInfo — класс

Сведения о доверителе при отзыве доверенности, совершенной (выданной) в порядке передоверия (СвДоверщП). Таблица 4.10.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RetrustPrincipalInfo
```

Свойства

Имя	Описание
Organization	Сведения об организации (Обязательный с условием). Обязательный, если доверитель в порядке передоверия действует от имени юридического лица.
Individual	Сведения по физическому лицу (Обязательный). Указываются сведения о физическом лице (в том числе, когда в доверенности в качестве доверителя в порядке передоверия указано юридическое лицо) или индивидуальном предпринимателе.

RevocationDocument — класс

Состав и структура документа (Документ). Таблица 4.2.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RevocationDocument
```

Свойства

Имя	Описание
TaxAuthorityCode	Код налогового органа, в который представляется доверенность, созданная в электронной форме (Обязательный)

Имя	Описание
`Revocation`	Сведения заявления об отзыве доверенности (Обязательный)
`Principal`	Сведения о доверителе (Обязательный). Сведения о лице, доверявшем представителю выступать от своего имени в отношениях, регулируемых законодательством Российской Федерации о налогах и сборах.
`Signer`	Сведения о физическом лице, подписывающем заявление об отзыве доверенности от имени доверителя (российской организации / иностранной организации) без доверенности, или сведения о физическом лице, подписывающем заявление об отзыве доверенности от своего имени, в том числе при передаче полномочий другому лицу в порядке передоверия (Обязательный).

RevocationInfo — класс

Сведения заявления об отзыве доверенности (СвЗаяв). Таблица 4.3.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RevocationInfo
```

Свойства

Имя	Описание
`PowerOfAttorneyNumber`	Единый регистрационный номер доверенности (Обязательный).

Имя	Описание
InternalPowerOfAttorneyNumber	Номер доверенности (Обязательный). Внутренний номер доверенности.
PowerOfAttorneyStartDate	Дата совершения (выдачи) доверенности (Обязательный).
RevocationDate	Дата отзыва доверенности (Обязательный).
IRevocationReason	Причина отзыва доверенности. При отзыве доверенности в отношении всех уполномоченных представителей не заполняется.

RussianCompanyInfo — класс

Сведения о российской организации (при отзыве основной доверенности) (НПЮЛ). Таблица 4.6.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RussianCompanyInfo
```

Свойства

Имя	Описание
IName	Наименование организации (Обязательный).
IInn	ИНН организации (Обязательный).
IKpp	КПП организации (Обязательный).
IOgrn	ОГРН (Обязательный).
PrincipalWithoutPowerOfAttorney	Сведения о лице, действующем от имени юридического лица без доверенности (Обязательный).

SeparateSubdivisionManagerInfo — класс

Сведения о руководителе обособленного подразделения (СвРукОП). Таблица 4.9.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class SeparateSubdivisionManagerInfo
```

Свойства

Имя	Описание
`Inn`	ИНН физического лица - руководителя обособленного подразделения.
`Gender`	Пол физического лица (Обязательный).
`BirthDate`	Дата рождения (Обязательный).
`BirthPlace`	Место рождения.
`CitizenshipType`	Признак наличия гражданства (Обязательный).
`Citizenship`	Гражданство (Обязательный с условием). Обязательный для иностранного гражданина. Принимает значение в соответствии с Общероссийским классификатором стран мира.

SignerInfo — класс

Сведения о физическом лице, подписывающем заявление об отзыве доверенности от имени доверителя (российской организации / иностранной организации) без доверенности, или сведения о физическом лице, подписывающем заявление об отзыве доверенности от своего имени, в том числе при передаче полномочий другому лицу в порядке передоверия (Подписант). Таблица 4.11.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class SignerInfo
```

Свойства

Имя	Описание
IPhone	Номер контактного телефона.
IFio	Фамилия, имя, отчество (при наличии) руководителя организации (обособленного подразделения) — доверителя, физического лица — доверителя (Обязательный).

PowerOfAttorneyFNSDOVEL503RevocationData — класс

Данные отзыва доверенности версии 5.03 для взаимодействия с ФНС.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyFNSDOVEL503RevocationData : PowerOfAttorneyRevocationData
```

Свойства

Имя	Описание
Document	Состав и структура документа (Обязательный).
RecipientID	Идентификатор получателя файла заявления (Обязательный).

Имя	Описание
FinalRecipientID	Идентификатор конечного получателя файла заявления (Обязательный). В случае передачи файла от отправителя к конечному получателю при отсутствии налоговых органов, осуществляющих передачу на промежуточных этапах, значения RecipientID и FinalRecipientID совпадают.
SenderID	Идентификатор отправителя информации (Обязательный). *Значение:* • Для организаций — 19-разрядный код (ИНН и КПП организации). • Для физических лиц — 12-разрядный код (ИНН физического лица, при наличии. При отсутствии ИНН — последовательность из двенадцати нулей).

Классы

Имя	Описание
FIO	Фамилия, имя, отчество (ФИОТип). Таблица 4.16.
ForeignCompanyInfo	Сведения об иностранной организации (при отзыве основной доверенности) (ИО). Таблица 4.8.
IdentityCardInfo	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип). Таблица 4.15.
IndividualInfo1	Сведения о физическом лице (СведФЛТип). Таблица 4.12.

Имя	Описание
IndividualInfo	Сведения о физическом лице, в том числе индивидуальном предпринимателе (СвФизЛицТип). Таблица 4.14.
OrganizationInfo	Сведения об организации (СвОргТип). Таблица 4.13.
PowerOfAttorneyInfo	Сведения для доверенности, подтверждающей полномочия нескольких уполномоченных представителей (СвПдтвУП). Таблица 4.4.
PrincipalInfo	Сведения о доверителе (СвДоверит). Таблица 4.5.
PrincipalWithoutPowerOfAttorneyInfo	Сведения о лице, действующем от имени юридического лица без доверенности (СвРукОрг). Таблица 4.7.
RetrustPrincipalInfo	Сведения о доверителе при отзыве доверенности, совершенной (выданной) в порядке передоверия (СвДоверщП). Таблица 4.10.
RevocationDocument	Состав и структура документа (Документ). Таблица 4.2.
RevocationInfo	Сведения заявления об отзыве доверенности (СвЗаяв). Таблица 4.3.
RussianCompanyInfo	Сведения о российской организации (при отзыве основной доверенности) (НПЮЛ). Таблица 4.6.
SeparateSubdivisionManagerInfo	Сведения о руководителе обособленного подразделения (СвРукОП). Таблица 4.9.

Имя	Описание
SignerInfo	Сведения о физическом лице, подписывающем заявление об отзыве доверенности от имени доверителя (российской организации / иностранной организации) без доверенности, или сведения о физическом лице, подписывающем заявление об отзыве доверенности от своего имени, в том числе при передаче полномочий другому лицу в порядке передоверия (Подписант). Таблица 4.11.

FIO — класс

Фамилия, имя, отчество (ФИОТип). Таблица 4.16.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class FIO
```

Свойства

Имя	Описание
LastName	Фамилия (Обязательный)
FirstName	Имя (Обязательный)
MiddleName	Отчество

ForeignCompanyInfo — класс

Сведения об иностранной организации (при отзыве основной доверенности) (ИО). Таблица 4.8.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ForeignCompanyInfo
```

Свойства

Имя	Описание
Name	Наименование иностранной организации (Обязательный).
Inn	ИНН организации (Обязательный).
Kpp	КПП обособленного подразделения (Обязательный).
Manager	Сведения о руководителе обособленного подразделения (Обязательный).

IdentityCardInfo — класс

Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип). Таблица 4.15.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class IdentityCardInfo
```

Свойства

Имя	Описание
DocumentKindCode	Код вида документа (Обязательный).
DocumentSerialNumber	Серия и номер документа (Обязательный).
IssueDate	Дата выдачи документа (Обязательный).

Имя	Описание
Issuer	Наименование органа, выдавшего документ (Обязательный с условием). Обязательный, если вид документа — Паспорт гражданина Российской Федерации.
IssuerCode	Код подразделения органа, выдавшего документ.

IndividualInfo1 — класс

Сведения о физическом лице (СведФЛТип). Таблица 4.12.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class IndividualInfo1
```

Свойства

Имя	Описание
Inn	ИНН физического лица.
RecordNumberOfFederalPopulationRegister	Номер записи федерального регистра сведений о населении
Snils	СНИЛС (Обязательный).
Ogrnip	ОГРНИП.
Citizenship	Гражданство. Принимает значение в соответствии с Общероссийским классификатором стран мира.
BirthDate	Дата рождения (Обязательный).
IdentityCard	Сведения о документе, удостоверяющем личность физического лица (Обязательный).

IndividualInfo — класс

Сведения о физическом лице, в том числе индивидуальном предпринимателе (СвФизЛицТип). Таблица 4.14.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class IndividualInfo
```

Свойства

Имя	Описание
`Inn`	ИНН физического лица.
`Ogrnip`	ОГРНИП (Обязательный с условием). Обязателен, если лицо является индивидуальным предпринимателем.
`Snils`	СНИЛС (Обязательный).
`Citizenship`	Гражданство. Принимает значение в соответствии с Общероссийским классификатором стран мира.
`BirthDate`	Дата рождения.
`Fio`	Фамилия, имя, отчество физического лица (Обязательный).
`IdentityCard`	Сведения о документе, удостоверяющем личность физического лица (Обязательный).

OrganizationInfo — класс

Сведения об организации (СвОргТип). Таблица 4.13.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class OrganizationInfo
```

Свойства

Имя	Описание
Name	Наименование организации / Наименование обособленного подразделения организации (Обязательный)
Inn	ИНН организации (Обязательный)
Kpp	КПП организации (Обязательный)
Ogrn	ОГРН

PowerOfAttorneyInfo — класс

Сведения для доверенности, подтверждающей полномочия нескольких уполномоченных представителей (СвПдтвУП). Таблица 4.4.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyInfo
```

Свойства

Имя	Описание
Organization	Сведения об организации (Обязательный с условием). Обязательный, если в доверенности в качестве уполномоченного представителя указано юридическое лицо.

Имя	Описание
Individual	Сведения о физическом лице, в том числе индивидуальном предпринимателе (Обязательный). Указываются сведения об уполномоченном представителе - физическом лице, в отношении которого отзывается доверенность.

PrincipalInfo — класс

Сведения о доверителе (СвДоверит). Таблица 4.5.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrincipalInfo
```

Свойства

Имя	Описание
RussianCompany	Сведения о российской организации (Обязательный с условием). Обязательный при отзыве основной доверенности, если доверителем является российская организация.
ForeignCompany	Сведения об иностранной организации (Обязательный с условием). Обязательный при отзыве основной доверенности, если доверителем является иностранная организация.
Individual	Сведения о физическом лице (Обязательный с условием). Обязательный при отзыве основной доверенности, если доверителем является физическое лицо.

Имя	Описание
RetrustPrincipal	Сведения о доверителе при отзыве доверенности, совершенной (выданной) в порядке передоверия (Обязательный с условием). Обязательный, если производится отзыв доверенности, совершенной (выданной) в порядке передоверия полномочий.

PrincipalWithoutPowerOfAttorneyInfo — класс

Сведения о лице, действующем от имени юридического лица без доверенности (СвРукОрг). Таблица 4.7.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PrincipalWithoutPowerOfAttorneyInfo
```

Свойства

Имя	Описание
Position	Должность (Обязательный).
Organization	Сведения об организации (Обязательный с условием). Обязательный, если лицом, действующим от имени юридического лица, является организация.

RetrustPrincipalInfo — класс

Сведения о доверителе при отзыве доверенности, совершенной (выданной) в порядке передоверия (СвДоверщП). Таблица 4.10.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RetrustPrincipalInfo
```

Свойства

Имя	Описание
Organization	Сведения об организации (Обязательный с условием). Обязательный, если доверитель в порядке передоверия действует от имени юридического лица.
Individual	Сведения по физическому лицу (Обязательный). Указываются сведения о физическом лице (в том числе, когда в доверенности в качестве доверителя в порядке передоверия указано юридическое лицо) или индивидуальном предпринимателе.

RevocationDocument — класс

Состав и структура документа (Документ). Таблица 4.2.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RevocationDocument
```

Свойства

Имя	Описание
TaxAuthorityCode	Код налогового органа, в который представляется доверенность, созданная в электронной форме (Обязательный)

Имя	Описание
`Revocation`	Сведения заявления об отзыве доверенности (Обязательный)
`Principal`	Сведения о доверителе (Обязательный). Сведения о лице, доверявшем представителю выступать от своего имени в отношениях, регулируемых законодательством Российской Федерации о налогах и сборах.
`Signer`	Сведения о физическом лице, подписывающем заявление об отзыве доверенности от имени доверителя (российской организации / иностранной организации) без доверенности, или сведения о физическом лице, подписывающем заявление об отзыве доверенности от своего имени, в том числе при передаче полномочий другому лицу в порядке передоверия (Обязательный).

RevocationInfo — класс

Сведения заявления об отзыве доверенности (СвЗаяв). Таблица 4.3.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class RevocationInfo
```

Свойства

Имя	Описание
`PowerOfAttorneyNumber`	Единый регистрационный номер доверенности (Обязательный).

Имя	Описание
InternalPowerOfAttorneyNumber	Номер доверенности (Обязательный). Внутренний номер доверенности.
PowerOfAttorneyStartDate	Дата совершения (выдачи) доверенности (Обязательный).
RevocationDate	Дата отзыва доверенности (Обязательный).
IRevocationReason	Причина отзыва доверенности. При отзыве доверенности в отношении всех уполномоченных представителей не заполняется.

RussianCompanyInfo — класс

Сведения о российской организации (при отзыве основной доверенности) (НПЮЛ). Таблица 4.6.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RussianCompanyInfo
```

Свойства

Имя	Описание
IName	Наименование организации (Обязательный).
IInn	ИНН организации (Обязательный).
IKpp	КПП организации (Обязательный).
IOgrn	ОГРН (Обязательный).
PrincipalWithoutPowerOfAttorney	Сведения о лице, действующем от имени юридического лица без доверенности (Обязательный).

SeparateSubdivisionManagerInfo — класс

Сведения о руководителе обособленного подразделения (СвРукОП). Таблица 4.9.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class SeparateSubdivisionManagerInfo
```

Свойства

Имя	Описание
`Inn`	ИНН физического лица - руководителя обособленного подразделения.
`Gender`	Пол физического лица (Обязательный).
`BirthDate`	Дата рождения (Обязательный).
`BirthPlace`	Место рождения.
`CitizenshipType`	Признак наличия гражданства (Обязательный).
`Citizenship`	Гражданство (Обязательный с условием). Обязательный для иностранного гражданина. Принимает значение в соответствии с Общероссийским классификатором стран мира.

SignerInfo — класс

Сведения о физическом лице, подписывающем заявление об отзыве доверенности от имени доверителя (российской организации / иностранной организации) без доверенности, или сведения о физическом лице, подписывающем заявление об отзыве доверенности от своего имени, в том числе при передаче полномочий другому лицу в порядке передоверия (Подписант). Таблица 4.11.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class SignerInfo
```

Свойства

Имя	Описание
IPhone	Номер контактного телефона.
IFio	Фамилия, имя, отчество (при наличии) руководителя организации (обособленного подразделения) — доверителя, физического лица — доверителя (Обязательный).

PowerOfAttorneyRegTransferStatuses — перечисление

Статус передачи доверенности в распределённый реестр ФНС.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum PowerOfAttorneyRegTransferStatuses
```

Члены

Имя члена	Описание
NotSubmitted = 0	Не передана.
SubmittedForRegistration = 1	Передана на регистрацию.
Registered = 2	Зарегистрирована.
RequestForRevocationSubmitted = 3	Отправлен запрос на отзыв.
Revoked = 4	Отозвана.
OperationError = 5	Операционная ошибка.

PowerOfAttorneyRepresentative — класс

Представляет секцию "Представитель" системной карточки доверенности.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PowerOfAttorneyRepresentative: BaseCardSectionRow
```

Свойства

Имя	Описание
Individual	Представитель физическое лицо (сотрудник из справочника сотрудников).
Organization	Представитель юридическое лицо (организация, индивидуальный предприниматель).

Поля

Имя	Описание
IndividualProperty	Определяет свойство "Представитель физическое лицо".
OrganizationProperty	Определяет свойство "Представитель юридическое лицо".

PowerOfAttorneyRepresentativesPowers — класс

Представляет секцию "Полномочия" системной карточки доверенности.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PowerOfAttorneyRepresentativesPowers : BaseCardSectionRow
```

Свойства

Имя	Описание
PowersCode	Код полномочий.
PowersDescription	Текстовое описание полномочий.

Поля

Имя	Описание
PowersCodeProperty	Определяет свойство "Код полномочий".
PowersDescriptionProperty	Определяет свойство "Текстовое описание полномочий".

PowerOfAttorneyRetrustType — перечисление

Признак возможности оформления передоверия.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum PowerOfAttorneyRetrustType
```

Члены

Имя члена	Описание
OneTime	Однократное передоверие.
None	Без права передоверия.
Followed	С последующим передоверием.

PowerOfAttorneyRevocationApplicantInfo — класс

Информация о лице, отзывающем доверенность.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class PowerOfAttorneyRevocationApplicantInfo
```

Свойства

Имя	Описание
ApplicantType	Тип заявителя
Name	Название организации (ИП) заявителя
Inn	ИНН заявителя
Kpp	КПП заявителя
Ogrn	ОГРН заявителя
Snils	СНИЛС физического лица-заявителя
LastName	Фамилия физического лица-заявителя
FirstName	Имя физического лица-заявителя
MiddleName	Отчество физического лица-заявителя
Phone	Контактный телефон

PowerOfAttorneyRevocationApplicantType — перечисление

Тип заявителя отзыва доверенности.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum PowerOfAttorneyRevocationApplicantType
```

Члены

Имя члена	Описание
Organization = 0	Организация
SoleProprietor = 1	Индивидуальный предприниматель
Individual = 2	Физическое лицо

PowerOfAttorneyRevocationData — класс

Данные отзыва доверенности.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyRevocationData
```

Свойства

Имя	Описание
RevocationType	Тип заявления на отзыв
RevocationReason	Причина отзыв
ApplicantInfo	Информация о заявителе отзыва

PowerOfAttorneyRevocationInfo — класс

Содержит информацию об отзыве доверенности.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowerOfAttorneyRevocationInfo
```

Свойства

Имя	Описание
PowerOfAttorneyRevocationFileID	Идентификатор файла отзыва доверенности (имя файла без расширения)

PowerOfAttorneyRevocationType — перечисление

Тип заявления на отзыв.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum PowerOfAttorneyRevocationType
```

Члены

Имя члена	Описание
Representative = 0	Заявление представителя
Principal = 1	Заявление доверителя

PowerOfAttorneySignatureFormat — перечисление

Формат подписи.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum PowerOfAttorneySignatureFormat
```

Члены

Имя члена	Описание
XMLDSIG = 0	Формат XMLDSIG.
CADES = 1	Формат CADES.

PowerOfAttorneyStatus — перечисление

Статус доверенности.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum PowerOfAttorneyStatus
```

Члены

Имя члена	Описание
`Preparation = 0`	Подготовка.
`Valid = 1`	Действует.
`Revoked = 2`	Отозвана.

PowerOfAttorneySubsidiaryPowersOfAttorney — класс

Представляет секции "Системные карточки дочерних доверенностей".

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class PowerOfAttorneySubsidiaryPowersOfAttorney : BaseCardSectionRow
```

Методы

Имя	Описание
`PowerOfAttorneySubsidiaryPowersOfAttorney()`	Инициализирует новый экземпляр класса `PowerOfAttorneySubsidiaryPowersOfAttorney`
`PowerOfAttorneySubsidiaryPowersOfAttorney()`	Инициализирует новый экземпляр класса `PowerOfAttorneySubsidiaryPowersOfAttorney`
`PowerOfAttorneySubsidiaryPowersOfAttorney(data)`	Инициализирует новый экземпляр класса `PowerOfAttorneySubsidiaryPowersOfAttorney` с указанными данными

Свойства

Имя	Описание
PowerOfAttorney	Системная карточка дочерней доверенности.

Поля

Имя	Описание
PowerOfAttorneyProperty	Определяет свойство "Системная карточка дочерней доверенности".

Powers — класс

Представляет полномочия для справочника полномочий МЧД.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class Powers : BaseDictionaryCard
```

Свойства

Имя	Описание
Groups	Группы.
PowerOfAttorneyFormats	Форматы доверенности.

Поля

Имя	Описание
GroupsProperty	Определяет свойство "Группы".
PowerOfAttorneyFormatsProperty	Определяет свойство "Форматы доверенности".

PowersCode — класс

Представляет коды полномочий для справочника полномочий МЧД.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowersCode : ObjectBase
```

Свойства

Имя	Описание
Name	Имя кода полномочий.
Description	Текстовое описание кода полномочий.
Code	Код полномочий.
Mnemonics	Мнемоника полномочия в реестре полномочий.
CreationDate	Дата создания кода.
UpdateDate	Дата обновления кода.
NotAvailable	Код недоступен.

Поля

Имя	Описание
NameProperty	Определяет свойство "Имя кода полномочий".
DescriptionProperty	Определяет свойство "Описание кода полномочий".
CodeProperty	Определяет свойство "Код полномочий".
MnemonicsProperty	Определяет свойство "Мнемоника полномочия".
CreationDateProperty	Определяет свойство "Дата создания кода".
UpdateDateProperty	Определяет свойство "Дата обновления кода".
NotAvailableProperty	Определяет свойство "Код недоступен".

PowersGroup — класс

Представляет группы полномочий для справочника полномочий МЧД.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PowersGroup : ObjectBase
```

Свойства

Имя	Описание
`Name`	Имя кода полномочий.
`Code`	Код полномочий.
`Author`	Автор.
`NsiId`	Идентификатор составителя в ЕС НСИ
`Codes`	Коды.

Поля

Имя	Описание
`NameProperty`	Определяет свойство "Имя кода полномочий".
`CodeProperty`	Определяет свойство "Код полномочий".
`AuthorProperty`	Определяет свойство "Автор".
`NsiIdDate`	Определяет свойство "Идентификатор составителя в ЕС НСИ".
`CodesProperty`	Определяет свойство "Коды".

PowersPowerOfAttorneyFormat — класс

Представляет типы доверенности карточки справочник полномочий МЧД.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PowersPowerOfAttorneyFormat : ObjectBase
```

Свойства

Имя	Описание
Name	Имя.
ProviderTypeName	Название типа провайдера.

Поля

Имя	Описание
NameProperty	Определяет свойство "Имя".
ProviderTypeNameProperty	Определяет свойство "Название типа провайдера".

IReferenceListService — интерфейс

Сервис для работы со списками ссылок на карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IReferenceListService
```

Методы

Имя	Описание
CreateReference(ReferenceList, LinksLinkType, FoldersFolder)	Добавляет ссылку на папку в *список ссылок*.
CreateReference(ReferenceList, LinksLinkType, String)	Добавляет ссылку на URL-адрес в *список ссылок*.

Имя	Описание
CreateReference(ReferenceList, LinksLinkType, BaseCard, String, Boolean)	Добавляет ссылку на карточку в *список ссылок*.
CreateReference(ReferenceList, LinksLinkType, Guid, Guid, Boolean)	Добавляет ссылку на карточку с заданным идентификатором в *список ссылок*.
CreateReferenceFromReference(ReferenceList, ReferenceListReference)	Создаёт ссылку на основе существующей ссылки.
CreateReferenceList	Создаёт пустой *список ссылок на карточки*.
GetReferenceDescription(ReferenceListReference)	Возвращает описание для объекта, на который установлена ссылка.
RemoveReference(BaseCard, ReferenceList, ReferenceListReference)	Удаляет ссылку из *списка ссылок на карточки*.
TryGetReferenceListFromCard(Guid, Boolean, ReferenceList)	Получает *список ссылок на карточки* из карточки.

Примеры

Ниже приведён пример добавления ссылки на папку в список ссылок на карточки.

```
①

IReferenceListService referenceListService = objectContext.GetService
<IReferenceListService>();
ILinkService linkService = objectContext.GetService<ILinkService>(); ②

Guid documentId = new Guid("00000000-0000-0000-0000-000000000000"); ③

ReferenceList referenceList;
referenceListService.TryGetReferenceListFromCard(documentId, true, out referenceList); ④

FoldersFolder foldersFolder = objectContext.GetObject<FoldersFolder>(new Guid("00000000-
0000-0000-0000-000000000001")); ⑤

LinksLinkType linksLinkType = linkService.FindLink("Ссылки"); ⑥

referenceListService.CreateReference(referenceList, linksLinkType, foldersFolder); ⑦
```

```
objectContext.AcceptChanges(); ⑧
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение документа, в который добавляется ссылка.

④ Получение списка ссылок на карточки из документа.

⑤ Получение папки, на которую создаётся ссылка.

⑥ Получение типа ссылки.

⑦ Добавление ссылки в список ссылок.

⑧ Сохранение изменений.

IReferenceListService.CreateReference — метод (ReferenceList, LinksLinkType, FoldersFolder)

Добавляет ссылку на папку в *список ссылок*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
ReferenceListReference CreateReference(ReferenceList referenceList, LinksLinkType
linkType, FoldersFolder foldersFolder)
```

Параметры

referenceList

Тип: ReferenceList

Список ссылок на карточки, в который добавляется ссылка

linkType

Тип: LinksLinkType

Тип ссылки

foldersFolder

Тип: xFoldersFolder

Папка, на которую создаётся ссылка

Возвращаемое значение

Тип: ReferenceListReference

Ссылка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр referenceList или foldersFolder.

IReferenceListService.CreateReference — метод (ReferenceList, LinksLinkType, String)

Добавляет ссылку на URL-адрес в *список ссылок*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
ReferenceListReference CreateReference(ReferenceList referenceList, LinksLinkType
linkType, string url)
```

Параметры

referenceList

Тип: ReferenceList

Список ссылок на карточки, в который добавляется ссылка

linkType

Тип: LinksLinkType

Тип ссылки

url

Тип: System.String

Добавляемый адрес

Возвращаемое значение

Тип: `ReferenceListReference`

Ссылка

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `referenceList` или `url`.

IReferenceListService.CreateReference — метод (ReferenceList, LinksLinkType, BaseCard, String, Boolean)

Добавляет ссылку на карточку в *список ссылок*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
ReferenceListReference CreateReference(ReferenceList referenceList, LinksLinkType
linkType, BaseCard cardObject, string cardDescription, bool hardlink)
```

Параметры

referenceList

Тип: `ReferenceList`

Список ссылок на карточки, в который добавляется ссылка

linkType

Тип: `LinksLinkType`

Тип ссылки

cardObject

Тип: `BaseCard`

Карточка, на которую создаётся ссылка

cardDescription

Тип: System.String

Описание ссылки

hardlink

Тип: System.Boolean

Если true, то жесткая ссылка

Возвращаемое значение

Тип: ReferenceListReference

Ссылка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр referenceList или cardObject.

IReferenceListService.CreateReference — метод (ReferenceList, LinksLinkType, Guid, Guid, Boolean)

Добавляет ссылку на карточку с заданным идентификатором в *список ссылок*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
ReferenceListReference CreateReference(ReferenceList referenceList, LinksLinkType
linkType, Guid cardId, Guid cardTypeId, bool hardlink)
```

Параметры

referenceList

Тип: ReferenceList

Список ссылок на карточки, в который добавляется ссылка

linkType

Тип: `LinksLinkType`

Тип ссылки

cardId

Тип: `System.Guid`

Идентификатор карточки, на которую создаётся ссылка

cardTypeId

Тип: `System.Guid`

Идентификатор типа карточки

hardlink

Тип: `System.Boolean`

Если `true`, то жесткая ссылка

Возвращаемое значение

Тип: `ReferenceListReference`

Ссылка

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `referenceList` или `cardId`.

IReferenceListService.CreateReferenceFromReference — метод (ReferenceList, ReferenceListReference)

Создаёт ссылку на основе существующей ссылки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
ReferenceListReference CreateReferenceFromReference(ReferenceList referenceList,
```

```
ReferenceListReference reference)
```

Параметры

referenceList

Тип: ReferenceList

Список ссылок на карточки, в который добавляется ссылка

reference

Тип: ReferenceListReference

Ссылка, на основе которой создаётся другая ссылка

Возвращаемое значение

Тип: ReferenceListReference

Ссылка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр referenceList или reference.

Заметки

Метод создаёт копию ссылки reference в списке ссылок referenceList.

IReferenceListService.CreateReferenceList — метод

Создаёт пустой *список ссылок на карточки*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
ReferenceList CreateReferenceList()
```

Возвращаемое значение

Тип: ReferenceList

Список ссылок на карточки

IReferenceListService.GetReferenceDescription — метод (ReferenceListReference)

Возвращает описание для объекта, на который установлена ссылка.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string GetReferenceDescription(ReferenceListReference reference)
```

Параметры

reference

Тип: ReferenceListReference

Ссылка

Возвращаемое значение

Тип: System.String

Описание объекта

Заметки

Метод возвращает:

- Для ссылки на карточку: название типа карточки;
- Для ссылки на папку: название папки;
- Для url-адреса: сформированную HTML-ссылку.

Примеры

```
①
IReferenceListService referenceListService = objectContext.GetService
<IReferenceListService>();

ReferenceListReference reference = objectContext.GetObject<ReferenceListReference>(new
Guid("00000000-0000-0000-0000-000000000000")); ②

string description = referenceListService.GetReferenceDescription(reference);  ③
```

① Инициализация контекста объектов.

② Получение ссылки.

③ Получение описания объекта.

IReferenceListService.RemoveReference — метод (BaseCard, ReferenceList, ReferenceListReference)

Удаляет ссылку из *списка ссылок на карточки*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool RemoveReference(BaseCard parentCard, ReferenceList referenceList,
ReferenceListReference reference)
```

Параметры

parentCard

Тип: BaseCard

Карточка со списком ссылок

referenceList

Тип: ReferenceList

Список ссылок на карточки, из которого удаляется ссылка

reference

Тип: ReferenceListReference

Удаляемая ссылка

Возвращаемое значение

Тип: System.Boolean

Если `true` — ссылка была удалена, иначе — `false`

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр referenceList или reference.

Заметки

Также удаляется обратная ссылка.

IReferenceListService.TryGetReferenceListFromCard — метод (Guid, Boolean, ReferenceList)

Получает *список ссылок на карточки* из карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool TryGetReferenceListFromCard(Guid cardId, bool createNew, out ReferenceList
referenceList)
```

Параметры

cardId

Тип: System.Guid

Идентификатор карточки, из которой возвращает *список ссылок на карточки*

createNew

Тип: System.Boolean

При true, если *список ссылок на карточки* отсутствует в карточках, будет создан новый

referenceList

Тип: ReferenceList

Список ссылок на карточки

Возвращаемое значение

Тип: System.Boolean

Возвращает true, если есть *список ссылок на карточки*, иначе — `false`

Примеры

```
①
IReferenceListService referenceListService = objectContext.GetService
<IReferenceListService>();

Guid cardId = new Guid("DED125DE-4097-E511-9417-90E6BA57B9F8");
ReferenceList referenceList; ②

bool result = referenceListService.TryGetReferenceListFromCard(cardId, true, out
referenceList); ③
```

① Инициализация контекста объектов.

② Карточка, из которой будет получен список ссылок на карточки.

③ Получаем список ссылок.

ICompressService — интерфейс

Сервис для работы с архивами

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface ICompressService
```

Методы

Имя	Описание
`CompressFolder (string, string, CompressionFormat)`	Сжимает содержимое папки.

CompressFolder — метод (string, string, CompressionFormat)

Сжимает содержимое папки

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void CompressFolder(string sourceDirectoryName, string destinationArchiveFileName,
CompressionFormat compressionFormat);
```

Параметры

sourceDirectoryName

Тип: System.String

Путь к архивируемой папке.

destinationArchiveFileName

Тип: System.String

Имя файла архива.

compressionFormat

Тип: CompressionFormat

Формат сжатия.

IRoleModelService — интерфейс

Сервис для работы с *Конструктором ролей*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IRoleModelService
```

Свойства

Имя	Описание
RolesDictionary	Возвращает ссылку на объектную модель конструктора ролей.

Методы

Имя	Описание
AddCondition(RoleModelRoleConditionGroup)	Создаёт условие в переданной группе условий для роли.
AddConditionGroup(RoleModelRoleConditionGroup)	Создаёт группу условий в переданной корневой группе.
AddCustomOperation(String, String)	Создаёт новую пользовательскую операцию.
AddCustomParameter(String, RoleModelCustomParameterType)	Создаёт новый пользовательский параметр.
AddLink(RoleModelCardKindRoleSetting, StatesOperation, RoleModelRole, StatesState, RoleModelOperationStatus)	Добавляет в настройки соответствие между операцией и состоянием.
AddRootConditionGroup(RoleModelRole)	Создаёт корневую группу условий.
CreateCardField(String)	Создаёт модель поля карточки в рамках ролевой модели.
CreateDayWorkStatus(RoleModelConditionValueDayWorkStatus, Guid)	Создаёт значение статуса дня для использования в условие.
CreateKindRoleSettings(KindsCardKind)	Добавляет указанный вид карточки в ролевую модель.
CreateRole(RoleModelCardKindRoleSetting)	Создаёт роль для вида карточки.
CreateTimeWorkStatus(RoleModelConditionValueTimeWorkStatus, Guid)	Создаёт значение со статусом времени для использования в условие.
DeleteRole(RoleModelRole)	Удаление роли.
FindKindCommonRoleSettings	Предоставляет доступ к контейнеру общих ролей.
GetConditionsByCardType(Guid)	Возвращает коллекцию условий ролевой модели для указанного типа карточки.
GetConditionValueDescription(RoleModelRoleCondition)	Создаёт строку описания из заданного условия ролевой модели.

Имя	Описание
`GetDateValueDescription(Object)`	Создаёт строку описания для объекта содержащего дату (обычная `System.DateTime`, либо дата ролевой модели — `DocsVision.BackOffice.ObjectModel.RoleModelConditionDayWorkStatus`).
`GetDayOfWeekDescription(DayOfWeek)`	Создаёт стандартную строку описания для дня недели.
`GetDayStatusDescription(RoleModelConditionValueDayWorkStatus)`	Создаёт стандартную строку описания для типа рабочего дня (выходной, рабочий).
`GetKindRoleSettings(KindsCardKind)`	Возвращает настройки ролевой модели для вида карточки.
`GetLinks(BaseCard)`	Возвращает соответствие между операциями и состояниями для переданной карточки.
`GetLinks(Guid)`	Возвращает соответствие между операциями и состояниями для карточки с указанным идентификатором.
`GetOperationStatus(KindsCardKind, IEnumerable<RoleModelRole>, StatesState, StatesOperation)`	Определяет статус (запрещена, разрешена, не разрешена) заданной операции для списка ролей. Приоритет имеет статус "Запрещена".
`GetOrCreatePreferredCardKindSetting(KindsCardKind)`	Создаёт или возвращает, в зависимости от существования, настройки ролевой модели для вида карточки.
`GetPreferredCardKindSetting(KindsCardKind)`	Возвращает настройки ролевой модели для вида карточки.
`GetRoles(KindsCardKind)`	Возвращает коллекцию ролей, применимых для заданного вида карточки.

Имя	Описание
GetTimeStatusDescription(RoleModelConditionValueTimeWorkStatus)	Возвращает локализованное название для заданного статуса рабочего времени.
GetTimeValueDescription(Object)	Возвращает сформированное строковое представление времени.
MakeRoleCommon(RoleModelRole)	Переводит роль в разряд общих, доступных для всех видов карточек.
MakeRoleNotCommon(RoleModelRole, KindsCardKind)	Привязывает роль к заданному виду карточек.
RemoveCardKindRoleSetting(KindsCardKind)	Удаляет настройки ролевой модели у переданного вида карточек.
ValidateRoles(KindsCardKind, IList<String>)	Получает список ошибок ролевой модели для заданного вида карточки.

Примеры

В примере создаётся роль, которая может быть использована для ограничения времени доступа к документам в выходные дни.

```
①

IRoleModelService roleModelService = objectContext.GetService<IRoleModelService>();

KindsCardKind kindsCardKind = objectContext.GetObject<KindsCardKind>(new Guid("8ACE1220-
A452-455D-8EEB-9EDF9DC6E327")); ②

RoleModelCardKindRoleSetting roleModelCardKindRoleSetting = roleModelService
.GetOrCreatePreferredCardKindSetting(kindsCardKind);
RoleModelRole roleModelRole = roleModelService.CreateRole(roleModelCardKindRoleSetting);
roleModelRole.Name = "Работа по выходным"; ③

roleModelRole.ConditionGroup.Operation = RoleModelRoleConditionGroupOperation.Or; ④

RoleModelRoleCondition condition = roleModelService.AddCondition(roleModelRole
.ConditionGroup);
condition.Parameter = RoleModelRoleConditionParameter.Today;
condition.Operation = RoleModelRoleConditionOperation.Equals;
condition.Value = System.DayOfWeek.Saturday; ⑤

condition = roleModelService.AddCondition(roleModelRole.ConditionGroup);
```

```
condition.Parameter = RoleModelRoleConditionParameter.Today;
condition.Operation = RoleModelRoleConditionOperation.Equals;
condition.Value = System.DayOfWeek.Sunday; ⑥

objectContext.AcceptChanges(); ⑦

roleModelService.MakeRoleCommon(roleModelRole); ⑧
```

① Инициализация контекста объектов и получение сервиса.

② Получение вида карточки "Документ".

③ Создание настроек для вида карточки.

④ Логическая операция между условиями — ИЛИ.

⑤ Создание первого условия (сегодня = суббота).

⑥ Создание второго условия (сегодня = воскресенье).

⑦ Сохранение изменений.

⑧ Отметим роль признаком "Общая", что позволит использовать её во всех карточках.

IScriptingService — интерфейс

Сервис для работы с конструктором скриптов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IScriptingService
```

Свойства

Имя	Описание
Scripts	Предоставляет доступ к коллекции описаний и текстов скриптов.

Методы

Имя	Описание
`CreateScript(KindsCardKind, String, ScriptingLanguage)`	Создаёт скрипт с указанным кодом.
`RemoveScript(ScriptingScript)`	Удаляет указанный скрипт.

IServerExtensionProxyService — интерфейс

Предоставляет доступ к некоторым методам серверных расширений, специфичным для карточек BackOffice

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface IServerExtensionProxyService
```

Методы

Имя	Описание
`GetAccessInfo(Guid, Guid, Guid)`	Получает список доступных операции для указанного состояния карточки. Результат представлен в формате XML.
`GetExecutedRolesData(Guid, Guid)`	Получает список ролей у которых есть разрешение на работу с указанной карточкой. Вид и состояние будут получены из самой карточки. Для проверки условий роли, используется указанный в параметрах сотрудник. Результат представлен в формате XML.
`GetExecutedRolesData(Guid, Guid, Guid)`	Получает список ролей у которых есть разрешение на работу с указанной карточкой, при заданного виде и состоянии. Для проверки условий роли, используется текущий сотрудник. Результат представлен в формате XML.

Имя	Описание
`GetTasksTree(IEnumerable<Guid>, IEnumerable<Guid>)`	Возвращает коллекцию узлов дерева заданий. Поиск осуществляется для заданных списков идентификаторов — сначала заданий, потом групп заданий.
`LogFindMessages(Guid, DateTime, DateTime)`	Осуществляет серверный поиск сообщений журнала по заданным параметрам.
`LogWriteMessages(LogItems)`	Записывает переданное сообщение в журнал
`ResetRolesCache(Guid)`	Сбрасывает серверный кэш ролевой модели. Также сбрасывает кэш связанных карточек.
`StopExecutionRelatedTasks(Guid, Boolean)`	Останавливает связанные задания. При возможности (задания завершены) завершает все задания поддерева задания, начиная с родительского. В принудительном режиме может завершить заблокированные задания.

IServersService — интерфейс

Описывает сервис для работы со *Справочником серверов*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface IServersService
```

Методы

Имя	Описание
`AddServer`	Создаёт пустую запись сервера.
`AddServer(Guid, String, String, String, String, Boolean, ServerConnectionInfo)`	Создаёт запись сервера с заполненными данными.

Имя	Описание
`CopyServer(ServersServer)`	Создаёт копию записи сервера.
`CreateServerCard(ServersServer)`	Создаёт карточку сервера из записи в справочнике.
`DeleteServer(ServersServer)`	Удаляет заданный сервер.
`GetServerCardKind`	Возвращает вид карточки сервера.
`ValidateServer(ServersServer, String)`	Проверяет запись сервера на наличие ошибок. Проверяется отсутствие других серверов с аналогичным именем, а также наличие других главных серверов, если заданный объявлен главным.

ISettingsCardService — интерфейс

Сервис для работы с системными настройками.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface ISettingsCardService
```

Методы

Имя	Описание
`GetSetting<T>(String, String, String)`	Возвращает системную настройку с заданным именем для указанного расширения.
`GetSubgroupSettingsRows(String, String, String)`	Возвращает коллекцию строк подгруппы системных настроек.
`SetSetting(String, String, String, Object, FieldType)`	Задаёт системную настройку.

ISettingsService — интерфейс

Определяет вспомогательные методы создания и получения объектов содержащихся в значениях настроек расширений справочника видов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface ISettingsService
```

Методы

Имя	Описание
CreateKindPropertyObject<T>(KindsCardExtendedSettingGroupCollection)	Создаёт группу в коллекции групп настроек.
GetKindPropertyObject<T>(KindsCardKind, String)	Возвращает настройки расширения указанного вида карточки.
GetPropertyObject<T>(String, String)	Возвращает настройки указанного расширения.

ISignatureLabelService — интерфейс

Описывает сервис для работы со справочником меток подписей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface ISignatureLabelService
```

Методы

Имя	Описание
AddSignatureLabel(String)	Инициализирует метку подписи и добавляет её в справочник.

Имя	Описание
AddSignatureLabelLocalizedName(SignatureLabel, Int32, String)	Создаёт метку подписи с локализованным названием. В справочник не добавляется.
RemoveSignatureLabel(SignatureLabel)	Удаляет метку подписи из справочника.

ISignatureService — интерфейс

Предоставляет сервис подписи.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface ISignatureService
```

Методы

Имя	Описание
CheckSignatureResult CheckSignature(byte[], Stream, X509Certificate2, bool, bool, DateTime?);	Проверяет подпись.

CheckSignatureResult — метод (byte[], Stream, X509Certificate2, bool, bool, DateTime?)

Проверяет подпись.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
CheckSignatureResult CheckSignature(byte[] signature, Stream contentStream,
X509Certificate2 certificate, bool detached, bool isHash, DateTime? signatureDate)
```

Параметры

signature

Тип: System.Byte[]

Закодированный в сообщение CMS/PKCS 7 поток подписываемых данных

contentStream

Тип: `System.IO.Stream`

Поток подписываемых данных

certificate

Тип: `System.Security.Cryptography.X509Certificates.X509Certificate2`

Сертификат подписи

detached

Тип: `System.Boolean`

Устанавливается значение `true`, если подпись является отсоединённой

isHash

Тип: `System.Boolean`

Устанавливается значение `true`, если подпись является простой

signatureDate

Тип: `System.DateTime`

Дата создания подписи

Возвращаемое значение

Тип: `CheckSignatureResult`

Возвращает результат проверки подписи

IStaffService — интерфейс

Сервис для работы со *справочником сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface IStaffService
```

Свойства

Имя	Описание
Impersonated	Возвращает признак того, что при вызове методов сервиса будет использоваться принудительно установленная учетная запись сотрудника.

Методы

Имя	Описание
ActualizeEmployeesState	Актуализирует состояние сотрудника в соответствии с параметрами, заданными в *Справочнике сотрудников*.
AddDeputy(StaffEmployee)	Добавляет указанному сотруднику заместителя.
AddEmployeesToGroup(StaffGroup, IEnumerable<StaffEmployee>)	Добавляет указанных сотрудников в группу сотрудников.
AddEmployeeToGroup(StaffGroup, StaffEmployee)	Добавляет указанного сотрудника в группу сотрудников.
AddGroupFolder(StaffGroup, Guid, Boolean)	Добавляет папку группы к указанной группе сотрудников, с возможностью отображения в Windows-клиенте.
AddNewAddress(StaffUnit, StaffAddresseAddressType)	Добавляет новый адрес для указанного подразделения.
AddNewADMapping	Создаёт экземпляр класса StaffADsMapping для определения соответствие между атрибутом Active Directory и полем *справочника сотрудников*.
AddNewEmployee(StaffUnit)	Добавляет нового сотрудника в указанное подразделение.

Имя	Описание
AddNewEmployeeFormatItem(StaffUnit)	Добавляет новый элемент к формату отображения данных сотрудника подразделения.
AddNewEmployeePicture(StaffEmployee)	Добавляет сотруднику новую фотографию.
AddNewGroup(StaffGroup)	Данный метод позволяет добавить новою подгруппу в существующую группу сотрудников.
AddNewNameCase(StaffEmployee, StaffNameCaseNameCase)	Добавляет новый падеж имени у заданного сотрудника.
AddNewPosition	Добавляет новую должность в *справочник сотрудников*.
AddNewRole	Создаёт новую роль в *справочнике сотрудников*.
AddNewUnit(StaffUnit)	Добавляет вложенное подразделение в указанную организацию.
AddNewViewField(ObjectBase, GridViewFieldCollectionType)	Добавляет отображаемое поле для подразделения, либо сотрудника.
AddRoleEmployee(StaffRole, StaffEmployee)	Добавляет сотрудника в роль.
AddRoleFolder(StaffRole, Guid, Boolean)	Добавляет новую папку в список папок роли.
AddRoleGroup(StaffRole, StaffGroup)	Добавляет группу сотрудников в роль.
AddRoleRole(StaffRole, StaffRole)	Добавляет в роль дочернюю роль.
AddRoleUnit(StaffRole, StaffUnit)	Добавляет в роль подразделение сотрудников.
AddUnitTreeEmployeesToGroup(StaffGroup, StaffUnit)	Добавляет всех сотрудников подразделения в указанную группу *справочника сотрудников*.
ApplyDisplayFormatToEmployees(StaffUnit, Boolean)	Немедленно применяет формат отображения данных о сотрудниках подразделения.

Имя	Описание
CanMoveGroup(StaffGroup, StaffGroup)	Определяет возможность перемещения группы.
CanMoveUnit(StaffUnit, StaffUnit)	Определяет возможность перемещения подразделения.
CanMoveViewField(IGridViewField, GridViewFieldCollectionType, Boolean)	Определяет возможность перемещения отображаемого поля.
ChangeEmployeeFolder(StaffEmployee, FoldersFolder)	Изменяет личную папку сотрудника.
CheckSignatureAndEncryptionSetting	Проверяет настройки криптографии.
CheckUnitADPath(String)	Проверяет узел Active Directory.
CopyGroup(StaffGroup, StaffGroup)	Копирует группу *справочника сотрудников* в указанную группу.
CreateDelegateFolder(FoldersFolder, FoldersFolder)	Создаёт папку-делегат.
CreateEmployeeFolder(StaffEmployee)	Создаёт персональную папку сотрудника.
CreateFolder(StaffUnit, String)	Создаёт папку с указанным именем в корневой папке подразделения.
CreateUnitFolder(StaffUnit)	Создаёт корневую папку для указанного подразделения.
DeleteADMapping(StaffADsMapping)	Удаляет заданное соответствие полей Active Directory.
DeleteDeputy(StaffEmployee, StaffDeputy)	Удаляет заместителя сотрудника.
DeleteEmployee(StaffEmployee)	Удаляет сотрудника из *справочника сотрудников*.
DeleteEmployeeFormatItem(StaffUnit, StaffEmployeesFormat)	Удаляет заданный формат отображения сотрудника.
DeleteEmployeePicture(StaffEmployee, StaffPicture)	Удаляет указанную фотографию сотрудника.
DeleteGroup(StaffGroup)	Удаляет указанную группу из *справочника сотрудников*.

Имя	Описание
DeletePosition(StaffPosition)	Удаляет должность из *справочника сотрудников*.
DeleteRole(StaffRole)	Удаляет роль из *справочника сотрудников*.
DeleteRoleElement(StaffRole, StaffContain)	Удаляет указанной элемент из роли в *справочнике сотрудников*.
DeleteUnit(StaffUnit)	Удаляет подразделение из *справочника сотрудников*.
DeleteViewField(IGridViewField, GridViewFieldCollectionType)	Удаляет указанное отображаемое поле.
FindCompanyByNameOnServer(StaffUnit, String)	Выполняет поиск подразделения с заданным названием.
FindDeputiesWhereEmployeeIsDeputy(StaffEmployee)	Возвращает список замещений сотрудников, в которых указанный сотрудник является заместителем.
FindEmployeeGroups(StaffEmployee)	Возвращает группы, в которых числится указанный сотрудник.
FindEmployeeGroupsByRole(StaffEmployee, StaffGroupRole)	Возвращает рабочие группы, в которых указанный сотрудник имеет определённую роль.
FindEmpoyeeByAccountName(String)	Выполняет поиск сотрудника по названию учетной записи.
FindEmployeeRoles(StaffEmployee)	Возвращает список ролей, в которых участвует сотрудник.
FindGroupByAccountName(String)	Возвращает группу с определённым названием учетной записи.
FindGroupByName(StaffGroup, String)	Возвращает группу с указанным названием.
FindRoleByAccountName(String)	Возвращает роль, привязанную к указанной учетной записи.

Имя	Описание
FindRoleWithSameName(StaffRole, String)	Осуществляет поиск роли по названию с исключением указанной роли из результатов.
FindUnitByActiveDirectoryId(String)	Получает подразделение с указанным идентификатором контейнера в Active Directory.
FindViewField(ViewCardFieldsGroup, IGridViewField)	Осуществляет поиск отображаемого поля.
Get(Guid)	Возвращает сотрудника с указанным идентификатором.
GetAllEmployeeViewFields	Возвращает отображаемые поля сотрудников.
GetAllUnitViewFields	Возвращает отображаемые поля подразделений.
GetCurrentEmployee	Возвращает текущего сотрудника.
GetDepartment(Guid)	Возвращает подразделение с заданным идентификатором.
GetDeputyAccessRightsSetting	Возвращает режим доступа для заместителя из карточки "Системные настройки".
GetEmployeeAllFormatFields	Возвращает все отображаемые поля сотрудника.
GetEmployeeCardFieldValue(StaffEmployee, Guid, String)	Возвращает значение поля с указанным псевдонимом из карточки сотрудника.
GetEmployeeCardKind(StaffEmployee)	Получает для сотрудника его вид карточки.
GetEmployeeDefaultViewFields	Возвращает базовый список отображаемых полей сотрудника.

Имя	Описание
GetEmployeeDisplayString(StaffUnit, StaffEmployee)	Возвращает отображаемую строку сотрудника, сформированная в соответствии с настройками подразделения.
GetEmployeeFieldDisplayName(Guid, String)	Возвращает отображаемое название для указанного поля.
GetEmployeeManager(StaffEmployee)	Возвращает руководителя из карточки подразделения.
GetEmployeesCount(StaffUnit, Boolean, Boolean)	Возвращает количество сотрудников в заданном подразделении с учетом иерархии.
GetEmployeesDisplayFormat(StaffUnit)	Возвращает формат отображения сотрудников в заданном подразделении.
GetGroup(Guid)	Возвращает группу с заданным идентификатором.
GetGroupEmployeeRoleDescription	Возвращает описание к указанной роли сотрудника в группе.
GetGroupEmployees(StaffGroup)	Возвращает сотрудников группы.
GetGroupEmployees(StaffGroup, Boolean, Boolean)	Возвращает сотрудников группы.
GetGroupInheritedViewFields(StaffGroup)	Возвращает унаследованные от родительской группы отображаемые поля.
GetGroupItems(StaffGroup, Boolean)	Возвращает элементы заданной группы.
GetImpersonatedEmployee	Возвращает сотрудника, являющегося истинным владельцем сессии, независимо от назначения, выполненного методом SetCurrentEmployee.
GetInheritedCalendarId(StaffEmployee)	Возвращает идентификатор унаследованного календаря.

Имя	Описание
GetInheritedTemplateFolder(StaffUnit)	Возвращает унаследованную шаблонную папку.
GetRoleEmployees(StaffRole)	Возвращает список сотрудников с указанной ролью.
GetRoleEmployees(StaffRole, Boolean, Boolean)	Возвращает список сотрудников с указанной ролью.
GetUnitADpath(StaffUnit)	Возвращает путь AD для подразделения.
GetUnitCardFieldValue(StaffUnit, Guid, String)	Возвращает значение поля с указанным псевдонимом из карточки подразделения.
GetUnitCardKind(StaffUnit)	Получает для указанного подразделения его вид карточки.
GetUnitDefaultViewFields	Возвращает базовый список отображаемых полей подразделения.
GetUnitEmployeeCardKind(StaffUnit)	Возвращает вид карточки сотрудника для указанного подразделения.
GetUnitEmployees(StaffUnit, Boolean, Boolean)	Возвращает список сотрудников указанного подразделения.
GetUnitEmployees(StaffUnit, Boolean, Boolean, Boolean)	Возвращает список сотрудников указанного подразделения.
GetUnitEmployees(StaffUnit, Boolean, Boolean, Boolean, Boolean)	Возвращает список сотрудников указанного подразделения.
GetUnitFieldDisplayName(Guid, String)	Возвращает отображаемое название указанного поля подразделения.
GetUnitInheritedViewFields(StaffUnit, Boolean)	Возвращает коллекцию унаследованных отображаемых полей подразделения.
GetUnits(StaffUnit, Boolean)	Возвращает список дочерних подразделений.

Имя	Описание
`GetViewFields(ObjectBase, GridViewFieldCollectionType)`	Возвращает отображаемые поля для заданного объекта и типа объекта в справочнике.
`GetViewFieldsAsObject(ObjectBase, GridViewFieldCollectionType)`	Возвращает отображаемые поля для заданного объекта и типа объекта в справочнике, без приведения к типу.
`IsEmployeeInGroup(StaffEmployee, StaffGroup, Boolean)`	Определяет принадлежность сотрудника к указанной группе.
`LoadEmployeeFromAD(StaffEmployee)`	Загружает данные сотрудника из Active Directory.
`LoadEmployeeFromAD(StaffEmployee, String)`	Выполняет загрузку данных из AD в *Справочник сотрудников*. Позволяет указать имя учетной записи, независимо от значения данного поля у сотрудника в *Справочнике сотрудников*.
`MoveEmployee(StaffEmployee, StaffUnit, Boolean)`	Переносит сотрудника в указанное подразделение.
`MoveEmployees(IEnumerable<StaffEmployee>, StaffGroup, StaffGroup)`	Переносит несколько сотрудников в заданную группу.
`MoveEmployees(IEnumerable<StaffEmployee>, StaffUnit, Boolean)`	Переносит несколько сотрудников в заданное подразделение.
`MoveGroup(StaffGroup, StaffGroup)`	Перемещает группу в указанную группу.
`MoveUnit(StaffUnit, StaffUnit)`	Перемещает подразделение в указанное подразделение.
`MoveViewField(IGridViewField, GridViewFieldCollectionType, Boolean)`	Перемещает отображаемое поле в порядке отображаемых полей.
`OpenOrCreateDepartmentCard(StaffUnit, Boolean)`	Создаёт или возвращает карточку подразделения.
`OpenOrCreateEmployeeCard(StaffEmployee, Boolean)`	Создаёт или возвращает карточку сотрудника.

Имя	Описание
OpenOrCreateEmployeeCard(StaffEmployee, Boolean, Boolean)	Создаёт или возвращает карточку сотрудника.
OtherADMappingWithSameFieldNameExists(StaffADsMapping, String)	Проверяет наличие коллекции соответствий полей и атрибутов AD с указанным именем атрибута.
OtherEmployeeWithSameAccountExists(StaffEmployee, String)	Определяет наличие сотрудников с указанной учетной записью.
OtherPositionWithSameNameExists(StaffPosition, String)	Определяет наличие должности с указанным названием.
PrepareAddToWorkGroupMailNotifications(StaffGroup, IEnumerable<StaffGroupItem>, Dictionary<StaffGroupFolder, String>)	Подготавливает содержимое почтового сообщения для отсылки сотрудникам при добавлении в *рабочую группу*.
PrepareRemoveFromWorkGroupMailNotifications(StaffGroup, IEnumerable<StaffGroupItem>)	Подготавливает содержимое почтового сообщения для отсылки участникам при исключении из *рабочей группы*.
PrepareRemoveWorkGroupMailNotifications(StaffGroup)	Подготавливает содержимое почтового сообщения для отсылки участникам при удалении *рабочей группы*.
PropagateNotAvailable(StaffUnit)	Распространяет признак "Не показывать при выборе" с родительского подразделения на дочерние подразделения и сотрудников.
RemoveGroupEmployee(StaffGroup, StaffEmployee)	Удаляет сотрудника из группы *справочника сотрудников*.
RemoveGroupItems(StaffGroup, IEnumerable<StaffGroupItem>)	Удаляет элементы (группы, сотрудники) из группы *справочника сотрудников*.
SetCurrentEmployee(StaffEmployee)	Переопределяет текущего сотрудника, возвращаемого методом GetCurrentEmployee.

Имя	Описание
SetFoldersRights(FoldersFolder, String)	Предоставляет полные права доступа к папке сотруднику с указанной учетной записью.
SynchronizeGroupWithActiveDirectory(StaffGroup, IADSynchronizationSettings, ILongProcessManager)	Синхронизирует данные группы *справочника сотрудников* с Active Directory.
SynchronizeRoleWithActiveDirectory(StaffRole, IADSynchronizationSettings, ILongProcessManager)	Синхронизирует данные роли *справочника сотрудников* с Active Directory.
SynchronizeUnitWithActiveDirectory(StaffUnit, IADSynchronizationSettings, ILongProcessManager)	Синхронизирует данные подразделения *справочника сотрудников* с Active Directory.
SynchronizeWithActiveDirectory(IADSynchronizationSettings, ILongProcessManager)	Синхронизирует данные групп, ролей и подразделений *справочника сотрудников* с Active Directory.
TransferEmployee(StaffEmployee, StaffUnit, Boolean)	Переносит сотрудника в указанное подразделение.
ValidateUser	Проверяет вхождение текущего пользователя в доменную группу DocsVision Administrators.

IStaffService.Impersonated — свойство

Возвращает признак того, что при вызове методов сервиса будет использоваться принудительно установленная учетная запись сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool Impersonated { get; }
```

Значение свойства

Тип: System.Boolean

`true` — если владелец-сотрудник контекста был установлен принудительно, иначе — `false`

Заметки

Установка владельца контекста производится с использованием метода `IStaffService.SetCurrentEmployee`.

IStaffService.ActualizeEmployeesState — метод

Актуализирует состояние сотрудника в соответствии с параметрами, заданными в *Справочнике сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void ActualizeEmployeesState()
```

Заметки

Данный метод выполняет блокировку или разблокировку сотрудника, основываясь на значении периода его отсутствия, определённом в *Справочнике сотрудников*. Проверка осуществляется для всех сотрудников, присутствующих в *Справочнике сотрудников*. Данный метод используется в бизнес-процессе "Процесс актуализации состояния сотрудников".

Примеры

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

staffService.ActualizeEmployeesState(); ③
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Осуществление актуализации.

IStaffService.AddDeputy — метод (StaffEmployee)

Добавляет указанному сотруднику заместителя.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffDeputy AddDeputy(StaffEmployee employee)
```

Параметры

employee

Тип: StaffEmployee

Сотрудник, которому будет добавлен заместитель

Возвращаемое значение

Тип: StaffDeputy

Заместитель

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр employee.

Примеры

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffEmployee employee = staffService.Get(new Guid("00000000-0000-0000-0000-
000000000000")); ③

StaffDeputy deputy = staffService.AddDeputy(employee); ④

deputy.Employee = staffService.Get(new Guid("00000000-0000-0000-0000-000000000001")); ⑤
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение сотрудника, которому добавляется заместитель.

④ Добавление заместителя с минимальным набором прав.

⑤ Определение сотрудника, являющегося заместителем.

IStaffService.AddEmployeesToGroup — метод (StaffGroup, IEnumerable<StaffEmployee>)

Добавляет указанных сотрудников в группу сотрудников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
List<StaffGroupItem> AddEmployeesToGroup(StaffGroup group, IEnumerable<StaffEmployee>
employees)
```

Параметры

group

Тип: StaffGroup

Группа сотрудников, в которую добавляются сотрудники

employees

Тип: System.Collections.Generic.IEnumerable<StaffEmployee>

Коллекция сотрудников StaffEmployee

Возвращаемое значение

Тип: System.Collections.Generic.List<StaffGroupItem>

Список сотрудников, представленных в группе. Тип StaffGroupItem

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр group или employees.

Примеры

Ниже приведён пример копирования всех сотрудников одной группы в другую с использованием метода AddEmployeesToGroup

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffGroup groupSource = staffService.GetGroup(new Guid("00000000-0000-0000-0000-
000000000000")); ③

StaffGroup groupRecipient = staffService.GetGroup(new Guid("00000000-0000-0000-0000-
000000000001")); ④

IEnumerable<StaffEmployee> employees = staffService.GetGroupEmployees(groupSource); ⑤

staffService.AddEmployeesToGroup(groupRecipient, employees); ⑥
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получения сервиса.

③ Получение группы, из которой выбираются сотрудники для копирования.

④ Получение группы, в которую осуществляется копирование.

⑤ Получение всех сотрудников из первой группы.

⑥ Добавление сотрудников в группу-получатель.

IStaffService.AddEmployeeToGroup — метод (StaffGroup, StaffEmployee)

Добавляет указанного сотрудника в группу сотрудников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffGroupItem AddEmployeeToGroup(StaffGroup group, StaffEmployee employee)
```

Параметры

group

Тип: StaffGroup

Группа, в которую включается сотрудника

employee

Тип: StaffEmployee

Добавляемый сотрудник

Возвращаемое значение

Тип: StaffGroupItem

Сотрудник, включённый в группу

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр group или employee.

Примеры

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffGroup group = staffService.GetGroup(new Guid("00000000-0000-0000-0000-
000000000000")); ③

StaffEmployee employee = staffService.Get(new Guid("00000000-0000-0000-0000-
000000000001")); ④

staffService.AddEmployeeToGroup(group, employee); ⑤
objectContext.SaveObject(group);
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение группы, в которую добавляется сотрудник.

④ Получение сотрудника.

⑤ Добавление сотрудника.

IStaffService.AddGroupFolder — метод (StaffGroup, Guid, Boolean)

Добавляет папку в папки группы указанной группе сотрудников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffGroupFolder AddGroupFolder(StaffGroup staffGroup, Guid folderId, bool showInTab)
```

Параметры

staffGroup

Тип: StaffGroup

Группа, которой добавляется папка группы

folderId

Тип: System.Guid

Идентификатор добавляемой папки

showInTab

Тип: System.Boolean

`true` — папка будет отображаться в области *моих папок* пользователя, иначе — `false`

Возвращаемое значение

Тип: StaffGroupFolder

Папка группы

Примеры

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffGroup group = staffService.GetGroup(new Guid("00000000-0000-0000-0000-
000000000000")); ③

staffService.AddGroupFolder(group, new Guid("00000000-0000-0000-0000-000000000001"),
true); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение группы.

④ Добавление к группе существующей папки с идентификатором `00000000-0000-0000-0000-000000000001`.

IStaffService.AddNewAddress — метод (StaffUnit, StaffAddresseAddressType)

Добавляет новый адрес для указанного подразделения.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
StaffAddresse AddNewAddress(StaffUnit unit, StaffAddresseAddressType type)
```

Параметры

unit

Тип: `StaffUnit`

Подразделение, для которого добавляется адрес

type

Тип: `StaffAddresseAddressType`

Подразделение, для которого добавляется адрес

Возвращаемое значение

Тип: `StaffAddresse`

Адрес подразделения

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `unit`.

Заметки

При повторном вводе адреса, он будет перезаписан.

Примеры

Ниже представлен пример добавления почтового адреса

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffUnit unit = staffService.GetDepartment(new Guid("00000000-0000-0000-0000-
000000000000")); ③

StaffAddresse addresse = staffService.AddNewAddress(unit, StaffAddresseAddressType
.PostAddress); ④
addresse.Address = "Большой Смоленский пр, 2а";
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение подразделения, которому добавляется адрес.

④ Добавление адреса.

IStaffService.AddNewADMapping — метод

Создаёт экземпляр класса `StaffADsMapping` для определения соответствие между атрибутом Active Directory и полем *справочника сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
StaffADsMapping AddNewADMapping()
```

Возвращаемое значение

Тип: `StaffADsMapping`

Экземпляр класса `StaffADsMapping`

IStaffService.AddNewEmployee — метод (StaffUnit)

Добавляет нового сотрудника в указанное подразделение.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffEmployee AddNewEmployee(StaffUnit unit)
```

Параметры

unit

Тип: StaffUnit

Подразделение, в которое будет добавлен сотрудник

Возвращаемое значение

Тип: StaffEmployee

Добавленный сотрудник

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр unit.

Примеры

Ниже приведён пример добавления нового сотрудника с минимальными данными

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffUnit unit = staffService.GetDepartment(new Guid("00000000-0000-0000-0000-
000000000000")); ③

StaffEmployee employee = staffService.AddNewEmployee(unit);
employee.LastName = "Иванов";
employee.FirstName = "Иван"; ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение подразделения, в которое добавляется сотрудник.

④ Добавление сотрудника.

IStaffService.AddNewEmployeeFormatItem — метод (StaffUnit)

Добавляет новый элемент к формату отображения данных сотрудника подразделения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffEmployeesFormat AddNewEmployeeFormatItem(StaffUnit unit)
```

Параметры

unit

Тип: StaffUnit

Подразделение, для которого добавляется формат

Возвращаемое значение

Тип: StaffEmployeesFormat

Элемент формата отображения данных сотрудника

IStaffService.AddNewEmployeePicture — метод (StaffEmployee)

Добавляет сотруднику новую фотографию.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffPicture AddNewEmployeePicture(StaffEmployee employee)
```

Параметры

employee

Тип: StaffEmployee

Сотрудник

Возвращаемое значение

Тип: `StaffPicture`

Фотография сотрудника

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `employee`.

Примеры

Ниже приведён пример добавления фотографии текущему сотруднику. Изображение загружается с диска.

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffEmployee employee = staffService.GetCurrentEmployee(); ③

StaffPicture picture = staffService.AddNewEmployeePicture(employee); ④
picture.Image = new Bitmap(@"Z:\EmployeePicture.jpg");
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение текущего сотрудника.

④ Добавление фотографии.

IStaffService.AddNewGroup — метод (StaffGroup)

Данный метод позволяет добавить новою подгруппу в существующую группу сотрудников.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
StaffGroup AddNewGroup(StaffGroup parentGroup)
```

Параметры

parentGroup

Тип: StaffGroup

Группа пользователей, в которую осуществляется добавление новой группы

Возвращаемое значение

Тип: StaffGroup

Объектная модель новой группы пользователей

Заметки

Если parentGroup не задан, группа будет добавлена в корневой узел групп, в *Справочнике сотрудников*.

Примеры

Ниже приведён пример простого добавления нового подразделения в корневой узел, в *Справочник сотрудников*

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffUnit staffUnit = staffService.AddNewUnit(null);
staffUnit.Name = "Подразделение компании"; ③

objectContext.SaveObject(staffUnit); ④
```

① Инициализация контекста объектов.

② Получение сервиса для работы со Справочником сотрудников.

③ Добавление подразделения.

④ Сохранение изменений.

IStaffService.AddNewUnit — метод (StaffUnit)

Добавляет вложенное подразделение в указанную организацию.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
StaffUnit AddNewUnit(StaffUnit parentUnit)
```

Параметры

parentUnit

Тип: `StaffUnit`

Организация, в которую добавляется подразделение

Возвращаемое значение

Тип: `StaffUnit`

Добавленное подразделение

Заметки

Если параметр parentUnit не задан, организация будет добавлена в корень *Справочника сотрудников*.

Примеры

Ниже приведён пример добавления новой организации в *Справочник сотрудников*

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffUnit unit = staffService.AddNewUnit(null);
unit.Type = StaffUnitType.Organization;
unit.Name = "ООО \"Василёк\""; ③
objectContext.SaveObject(unit);
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Добавление новой организации (тип `StaffUnitType.Organization`).

IStaffService.AddNewPosition — метод

Добавляет новую должность в *справочник сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffPosition AddNewPosition()
```

Возвращаемое значение

Тип: StaffPosition

Должность

IStaffService.AddNewRole — метод

Создаёт новую роль в *справочнике сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffRole AddNewRole()
```

Возвращаемое значение

Тип: StaffRole

Роль

IStaffService.AddRoleEmployee — метод (StaffRole, StaffEmployee)

Добавляет сотрудника в роль.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffContain AddRoleEmployee(StaffRole role, StaffEmployee employee)
```

Параметры

role

Тип: StaffRole

Роль

employee

Тип: StaffEmployee

Сотрудник, добавляемый в роль

Возвращаемое значение

Тип: StaffContain

Контейнер роли

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр role или employee.

Примеры

Далее приведён пример добавления роли указанному сотруднику

```
①

IStaffService staffService = objectContext.GetService<IStaffService>();

StaffRole role = objectContext.GetObject<StaffRole>(new Guid("00000000-0000-0000-0000-
000000000000"));
StaffEmployee employee = objectContext.GetObject<StaffEmployee>(new Guid("00000000-0000-
0000-0000-000000000001"));

StaffContain result = staffService.AddRoleEmployee(role, employee);
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

IStaffService.AddRoleFolder — метод (StaffRole, Guid, Boolean)

Добавляет новую папку в список папок роли.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffRoleFolder AddRoleFolder(StaffRole staffRole, Guid folderId, bool showInTab)
```

Параметры

staffRole

Тип: StaffRole

Роль

folderId

Тип: System.Guid

Идентификатор папки

showInTab

Тип: System.Boolean

true, если папка должна отображается в области моих папок, иначе — false

Возвращаемое значение

Тип: StaffRoleFolder

Папка роли

IStaffService.AddRoleGroup — метод (StaffRole, StaffGroup)

Добавляет группу сотрудников в роль.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffContain AddRoleGroup(StaffRole role, StaffGroup group)
```

Параметры

role

Тип: StaffRole

Роль

group

Тип: StaffGroup

Группа

Возвращаемое значение

Тип: StaffContain

Контейнер роли

IStaffService.AddRoleRole — метод (StaffRole, StaffRole)

Добавляет в роль дочернюю роль.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffContain AddRoleRole(StaffRole role, StaffRole childRole)
```

Параметры

role

Тип: StaffRole

Роль

childRole

Тип: StaffGroup

Роль, добавляемая как подчиненная

Возвращаемое значение

Тип: StaffContain

Контейнер роли

IStaffService.AddRoleUnit — метод (StaffRole, StaffUnit)

Добавляет в роль подразделение сотрудников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffContain AddRoleUnit(StaffRole role, StaffUnit unit)
```

Параметры

role

Тип: StaffRole

Роль, в которую добавляется подразделение

unit

Тип: StaffUnit

Подразделение

Возвращаемое значение

Тип: StaffContain

Контейнер роли

IStaffService.AddUnitTreeEmployeesToGroup — метод (StaffGroup, StaffUnit)

Добавляет всех сотрудников подразделения в указанную группу *справочника сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
List<StaffGroupItem> AddUnitTreeEmployeesToGroup(StaffGroup group, StaffUnit unit)
```

Параметры

group

Тип: StaffGroup

Группа, в которую добавляются сотрудники

unit

Тип: StaffUnit

Подразделение, сотрудники которого добавляются в группу group

Возвращаемое значение

Тип: System.Collections.Generic.List<StaffGroupItem>

Список добавленных элементов типа StaffGroupItem

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр group или unit.

IStaffService.ApplyDisplayFormatToEmployees — метод (StaffUnit, Boolean)

Немедленно применяет формат отображения данных о сотрудниках подразделения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ApplyDisplayFormatToEmployees(StaffUnit unit, bool applyToChildUnits)
```

Параметры

unit

Тип: StaffUnit

Подразделение, к сотрудникам которого должны быть применены изменения

applyToChildUnits

Тип: System.Boolean

true — также применить к подчиненным подразделениям

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр unit.

IStaffService.FindCompanyByNameOnServer — метод (StaffUnit, String)

Выполняет поиск подразделения с заданным названием.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffUnit FindCompanyByNameOnServer(StaffUnit parentUnit, string unitName)
```

Параметры

parentUnit

Тип: StaffUnit

Подразделение, в котором выполняется поиск подразделения

unitName

Тип: System.String

Название искомого подразделения

Возвращаемое значение

Тип: StaffUnit

Искомое подразделение

Заметки

Значение unitName должно полностью соответствовать названию подразделения.

Поиск среди дочерних (к `parentUnit`) подразделений не производится.

При `parentUnit` равным null, поиск будет выполнен на корневом уровне дерева подразделений.

IStaffService.FindDeputiesWhereEmployeeIsDeputy — метод (StaffEmployee)

Возвращает список замещений сотрудников, в которых указанный сотрудник является заместителем.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
List<StaffDeputy> FindDeputiesWhereEmployeeIsDeputy(StaffEmployee employee);
```

Параметры

employee

Тип: `StaffEmployee`

Сотрудник, для которого определяется список замещений

Возвращаемое значение

Тип: `System.Collections.Generic.List<StaffDeputy>`

Список замещений, представленный коллекцией объектов `StaffDeputy`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `employee`.

Заметки

Заместители сотрудников добавляются в *Справочнике сотрудников*.

Примеры

Ниже приведён пример использования сервиса для отбора прав у текущего сотрудника, позволяющих ему, как заместителю другого сотрудника, выполнять подписание документов ЭП.

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

List<StaffDeputy> deputy = staffService.FindDeputiesWhereEmployeeIsDeputy(staffService
.GetCurrentEmployee()); ③

foreach (StaffDeputy item in deputy) ④
{
 item.Signature = false;
}
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение списка замещений в который участвует текущий сотрудник.

④ Сброс флага, разрешающего подписание документов, для всех замещений.

IStaffService.FindEmpoyeeByAccountName — метод (String)

Выполняет поиск сотрудника по названию учетной записи.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffEmployee FindEmpoyeeByAccountName(string accountName)
```

Параметры

accountName

Тип: System.String

Имя учетной записи искомого сотрудника

Возвращаемое значение

Тип: StaffEmployee

Сотрудник

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр accountName.

Примеры

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffEmployee employee = staffService.FindEmpoyeeByAccountName(@"domain\ivanov_ii"); ③
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Поиск сотрудника по названию учетной записи.

IStaffService.ChangeEmployeeFolder — метод (StaffEmployee, FoldersFolder)

Изменяет личную папку сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ChangeEmployeeFolder(StaffEmployee employee, FoldersFolder folder)
```

Параметры

employee

Тип: StaffEmployee

Сотрудник, у которого изменяется личная папка

folder

Тип: xFoldersFolder

Новая личная папка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр employee.

IStaffService.CopyGroup — метод (StaffGroup, StaffGroup)

Копирует группу *справочника сотрудников* в указанную группу.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffGroup CopyGroup(StaffGroup group, StaffGroup parentGroup)
```

Параметры

group

Тип: StaffGroup

Копируемая группа

parentGroup

Тип: StaffGroup

Группа сотрудников, в которую осуществляется копирование

Возвращаемое значение

Тип: StaffGroup

Копия группы group

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр group.

Заметки

Если parentGroup равен null, то группа group будет скопирована в корень дерева

групп сотрудников.

IStaffService.CreateDelegateFolder — метод (FoldersFolder, FoldersFolder)

Создаёт папку-делегат.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
FoldersFolder CreateDelegateFolder(FoldersFolder personalFolder, FoldersFolder
templateFolder);
```

Параметры

personalFolder

Тип: xFoldersFolder

Папка, в которую будет добавлена папка-делегат

templateFolder

Тип: xFoldersFolder

Папка, для которой папка-делегат

Возвращаемое значение

Тип: xFoldersFolder

Папка делегат

Примеры

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

FoldersFolder personalFolder = objectContext.GetObject<FoldersFolder>(new Guid("00000000-
0000-0000-0000-000000000000"));
FoldersFolder templateFolder = objectContext.GetObject<FoldersFolder>(new Guid("00000000-
0000-0000-0000-000000000001")); ③

FoldersFolder delegateFolder = staffService.CreateDelegateFolder(personalFolder,
templateFolder); ④
```

```
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение папки, в которой создаётся папка-делегат, а также папки, на которую создаётся делегат.

④ Создание папки-делегата.

IStaffService.CreateEmployeeFolder — метод (StaffEmployee)

Создаёт персональную папку сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void CreateEmployeeFolder(StaffEmployee employee)
```

Параметры

employee

Тип: StaffEmployee

Сотрудника, для которого создаётся персональная папка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр employee.

Заметки

Обязательным требованием для создания персональной папки пользователя является: наличие корневой папки у организации, в которой числится сотрудник.

IStaffService.CreateFolder — метод (StaffUnit, String)

Создаёт папку с указанным именем в корневой папке подразделения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
FoldersFolder CreateFolder(StaffUnit parentUnit, string folderName)
```

Параметры

parentUnit

Тип: StaffUnit

Подразделение, в корневую папку которого добавляется папка

folderName

Тип: System.String

Название добавляемой папки

Возвращаемое значение

Тип: xFoldersFolder

Созданная папка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр parentUnit, либо folderName не имеет значения. Также ошибка может быть возвращена в случае отсутствия у parentUnit корневой папки.

Заметки

В свойствах подразделения должна быть указана корневая папка.

IStaffService.CreateUnitFolder — метод (StaffUnit)

Создаёт корневую папку для указанного подразделения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
FoldersFolder CreateUnitFolder(StaffUnit unit)
```

Параметры

unit

Тип: StaffUnit

Подразделение, для которого требуется создать корневую папку

Возвращаемое значение

Тип: xFoldersFolder

Созданная папка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр unit.

Заметки

Название созданной папки будет совпадать с названием подразделения, для которого она будет создана.

IStaffService.DeleteDeputy — метод (StaffEmployee, StaffDeputy)

Удаляет заместителя сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteDeputy(StaffEmployee employee, StaffDeputy deputy)
```

Параметры

employee

Тип: StaffEmployee

Сотрудник с заместителем

deputy

Тип: StaffDeputy

Заместитель, который должен быть удалён

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр employee или deputy.

Примеры

Ниже приведён пример удаления заместителя у текущего сотрудника.

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffEmployee employee = staffService.GetCurrentEmployee();
StaffEmployee deputyEmployee = objectContext.GetObject<StaffEmployee>(new Guid("21E55387-
6A4A-4B86-AC0F-1A454DE1F3DD")); ③

StaffDeputy deputy = employee.Deputies.First(t => t.Employee = deputyEmployee); ④

staffService.DeleteDeputy(employee, deputy); ⑤
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение текущего сотрудника и сотрудника, выполняющего роль заместителя.

④ Получение удаляемого заместителя сотрудника.

⑤ Удаление сотрудника.

IStaffService.DeleteEmployee — метод (StaffEmployee)

Удаляет сотрудника из *справочника сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteEmployee(StaffEmployee employee)
```

Параметры

employee

Тип: StaffEmployee

Сотрудника, для которого создаётся персональная папка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр employee.

Примеры

Пример удаления уволенных сотрудников из группы.

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffGroup group = staffService.GetGroup(new Guid("00000000-0000-0000-0000-
000000000000")); ③

foreach (StaffEmployee employee in group.Employees)
{
 if (employee.Status = StaffEmployeeStatus.Discharged` || employee.Status =
StaffEmployeeStatus.DischargedNoRestoration) ④
 {
  staffService.DeleteEmployee(employee);
 }
}
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение группы, из которой удаляются сотрудники.

④ Если сотрудник уволен (в т.ч. без возможности восстановления), удаляем.

IStaffService.DeleteEmployeePicture — метод (StaffEmployee, StaffPicture)

Удаляет указанную фотографию сотрудника.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void DeleteEmployeePicture(StaffEmployee employee, StaffPicture picture)
```

Параметры

employee

Тип: `StaffEmployee`

Сотрудник, фотография которого удаляется

picture

Тип: `StaffPicture`

Удаляемое изображения

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `employee` или `picture`.

IStaffService.DeleteGroup — метод (StaffGroup)

Удаляет указанную группу из *справочника сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void DeleteGroup(StaffGroup group)
```

Параметры

group

Тип: StaffGroup

Удаляемая группа

Примеры

Пример очистки списка групп, путем удаления пустой дочерней группы.

```
①

IStaffService staffService = objectContext.GetService<IStaffService>();
StaffGroup staffGroup = staffService.GetGroup(new Guid("00000000-0000-0000-0000-
000000000000"));

foreach (var group in staffGroup.Groups)
{
 if (group.Groups.Count = 0 && group.GroupItems.Count = 0)
 {
  staffGroup.DeleteGroup(group);
 }
}

objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

IStaffService.DeletePosition — метод (StaffPosition)

Удаляет должность из *справочника сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeletePosition(StaffPosition position)
```

Параметры

position

Тип: `StaffPosition`

Удаляемая должность

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `position`.

IStaffService.DeleteRole — метод (StaffRole)

Удаляет роль из *справочника сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void DeleteRole(StaffRole role)
```

Параметры

role

Тип: `StaffRole`

Удаляемая роль

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `role`.

IStaffService.DeleteRoleElement — метод (StaffRole, StaffContain)

Удаляет указанной элемент из роли в *справочнике сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void DeleteRoleElement(StaffRole role, StaffContain staffContain)
```

Параметры

role

Тип: StaffRole

Роль, из которой удаляется элемент

staffContain

Тип: StaffContain

Удаляемый элемент роли

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр role или staffContain.

IStaffService.DeleteUnit — метод (StaffUnit)

Удаляет подразделение из *справочника сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeleteUnit(StaffUnit unit)
```

Параметры

unit

Тип: StaffUnit

Удаляемое подразделение

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр unit.

Заметки

При удалении подразделения будут удалены все подчиненные подразделения, а также сотрудники, которые в них числятся.

IStaffService.FindEmployeeGroups — метод (StaffEmployee)

Возвращает группы, в которых числится указанный сотрудник.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
List<StaffGroup> FindEmployeeGroups(StaffEmployee employee)
```

Параметры

employee

Тип: StaffEmployee

Сотрудник, для которого определяется список групп

Возвращаемое значение

Тип: System.Collections.Generic.List<StaffGroup>

Список групп типа StaffGroup

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр employee.

IStaffService.FindEmployeeGroupsByRole — метод (StaffEmployee, StaffGroupRole)

Возвращает рабочие группы, в которых указанный сотрудник имеет определённую роль.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
List<StaffGroup> FindEmployeeGroupsByRole(StaffEmployee employee, StaffGroupRole role)
```

Параметры

employee

Тип: StaffEmployee

Сотрудник

role

Тип: StaffGroupRole

Роль сотрудника в рабочей группе

Возвращаемое значение

Тип: System.Collections.Generic.List<StaffGroup>

Список групп типа StaffGroup

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр employee.

IStaffService.FindEmployeeRoles — метод (StaffEmployee)

Возвращает список ролей, в которых участвует сотрудник.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
List<StaffRole> FindEmployeeRoles(StaffEmployee employee)
```

Параметры

employee

Тип: StaffEmployee

Сотрудник

Возвращаемое значение

Тип: System.Collections.Generic.List<StaffRole>

Список ролей типа StaffRole

IStaffService.FindGroupByAccountName — метод (String)

Возвращает группу с определённым названием учетной записи.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffGroup FindGroupByAccountName(string accountName)
```

Параметры

accountName

Тип: System.String

Название учетной записи

Возвращаемое значение

Тип: StaffGroup

Группа с искомым названием учетной записи

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если передан пустой accountName.

Примеры

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffGroup group = staffService.FindGroupByAccountName(@"WorkDomain.com\ADGroupName"); ③
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Поиск группы с указанной учетной записью.

IStaffService.FindGroupByName — метод (StaffGroup, String)

Возвращает группу с указанным названием.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffGroup FindGroupByName(StaffGroup parentGroup, string groupName)
```

Параметры

parentGroup

Тип: StaffGroup

Группа, в которой выполняется поиск. Для поиска по верхнему уровню, нужно передать null

groupName

Тип: System.String

Название группы

Возвращаемое значение

Тип: StaffGroup

Группа сотрудников с конкретным именем

IStaffService.FindRoleByAccountName — метод (String)

Возвращает роль, привязанную к указанной учетной записи.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffRole FindRoleByAccountName(string accountName)
```

Параметры

accountName

Тип: System.String

Название учетной записи

Возвращаемое значение

Тип: StaffRole

Найденная роль

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если передан пустой accountName.

IStaffService.FindRoleWithSameName — метод (StaffRole, String)

Осуществляет поиск роли по названию с исключением указанной роли из результатов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffRole FindRoleWithSameName(StaffRole role, string roleName)
```

Параметры

role

Тип: StaffRole

Роль, которая должна быть исключена из результатов

roleName

Тип: System.String

Название искомой роли

Возвращаемое значение

Тип: StaffRole

Роль с названием roleName

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр role, либо передан пустой параметр roleName.

Заметки

Метод может быть полезен при поиске возможных дубликатов роли role.

IStaffService.FindUnitByActiveDirectoryId — метод (String)

Получает подразделение с указанным идентификатором контейнера в Active Directory.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffUnit FindUnitByActiveDirectoryId(string adId)
```

Параметры

adId

Тип: System.String

Идентификатор подразделения в Active Directory

Возвращаемое значение

Тип: StaffUnit

Подразделение в *справочнике сотрудников*, синхронизируемое с подразделением в Active Directory

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если передан пустой adId.

IStaffService.Get — метод (Guid)

Возвращает сотрудника с указанным идентификатором.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffEmployee Get(Guid employeeId)
```

Параметры

employeeId

Тип: System.Guid

Идентификатор сотрудника

Возвращаемое значение

Тип: StaffEmployee

Сотрудник

IStaffService.GetCurrentEmployee — метод

Возвращает текущего сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffEmployee GetCurrentEmployee()
```

Возвращаемое значение

Тип: StaffEmployee

Текущий сотрудник

Заметки

Данный метод возвращает текущего сотрудника, который может не совпадать с владельцем сессии — был переопределен методом SetCurrentEmployee. Если необходимо получить истинного текущего сотрудника, используйте метод GetImpersonatedEmployee.

IStaffService.GetDepartment — метод (Guid)

Возвращает подразделение с заданным идентификатором.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffUnit GetDepartment(Guid unitId)
```

Параметры

unitId

Тип: System.Guid

Идентификатор подразделения

Возвращаемое значение

Тип: StaffUnit

Подразделение с идентификатором unitId

IStaffService.GetEmployeeManager — метод (StaffEmployee)

Возвращает руководителя из карточки подразделения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffEmployee GetEmployeeManager(StaffEmployee employee)
```

Параметры

employee

Тип: StaffEmployee

Сотрудник, руководитель которого запрашивается.

Возвращаемое значение:

StaffEmployee

Тип: StaffEmployee

Руководитель сотрудника из карточки подразделения.

IStaffService.GetGroupEmployees — метод (StaffGroup)

Возвращает сотрудников группы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<StaffEmployee> GetGroupEmployees(StaffGroup group)
```

Параметры

group

Тип: StaffGroup

Группа сотрудников

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<StaffEmployee>

Список сотрудников типа StaffEmployee

Примеры

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffGroup group = staffService.FindGroupByName(null, "Бухгалтерия"); ③

IEnumerable<StaffEmployee> employees = staffService.GetGroupEmployees(group); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение группы.

④ Получение сотрудников, входящих в группу.

IStaffService.GetGroupEmployees — метод (StaffGroup, Boolean, Boolean)

Возвращает сотрудников группы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<StaffEmployee> GetGroupEmployees(StaffGroup group, bool useNotAvailable, bool
useNotSearchable)
```

Параметры

group

Тип: StaffGroup

Группа сотрудников

useNotAvailable

Тип: System.Boolean

Если true, то из результата будут исключены сотрудники с флагом NotAvailable

useNotSearchable

Тип: System.Boolean

Если true, то из результата будут исключены сотрудники с флагом `NotSearchable`

Возвращаемое значение

Тип: `System.Collections.Generic.IEnumerable<StaffEmployee>`

Коллекция выбранных сотрудников типа `StaffEmployee`

IStaffService.GetGroupItems — метод (StaffGroup, Boolean)

Возвращает элементы заданной группы.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
IEnumerable<StaffGroupItem> GetGroupItems(StaffGroup group, bool selection)
```

Параметры

group

Тип: `StaffGroup`

Группа

selection

Тип: `System.Boolean`

`true` — возвращать только элементы, которые могут быть выбраны в карточке, иначе — `false`

Возвращаемое значение

Тип: `System.Collections.Generic.IEnumerable<StaffGroupItem>`

Все элементы типа `StaffGroupItem`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `group`.

IStaffService.GetImpersonatedEmployee — метод

Возвращает истинного текущего сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffEmployee GetImpersonatedEmployee()
```

Возвращаемое значение

Тип: StaffEmployee

Истинный текущий сотрудник

Заметки

Текущий сотрудник (при использовании метода) может быть переопределен методом SetCurrentEmployee, а данный метод позволяет получить истинного текущего сотрудника.

IStaffService.GetRoleEmployees — метод (StaffRole)

Возвращает список сотрудников с указанной ролью.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
System.Collections.Generic.IEnumerable<StaffEmployee> GetRoleEmployees(StaffRole role)
```

Параметры

role

Тип: StaffRole

Роль сотрудника, подразделения или группы

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<StaffEmployee>

Список сотрудников типа `StaffEmployee`

Заметки

Метод `GetRoleEmployees`, помимо сотрудников непосредственно имеющих роль `role`, включает сотрудников из групп, ролей и подразделений, у которых включена роль `role`.

Примеры

Пример получения списка сотрудников, имеющих роль с идентификатором `00000000-0000-0000-0000-000000000000`

```
①

StaffRole role = objectContext.GetObject<StaffRole>(new Guid("00000000-0000-0000-0000-
000000000000"));
IStaffService staffService = objectContext.GetService<IStaffService>();

var employees = staffService.GetRoleEmployees(role);
```

① Инициализация контекста объектов.

IStaffService.GetRoleEmployees — метод (StaffRole, Boolean, Boolean)

Возвращает список сотрудников с указанной ролью.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
IEnumerable<StaffEmployee> GetRoleEmployees(StaffRole role, bool useNotAvailable, bool
useNotSearchable)
```

Параметры

role

Тип: `StaffRole`

Роль

useNotAvailable

Тип: `System.Boolean`

Если `true`, то из результата будут исключены сотрудники с флагом `NotAvailable`

useNotSearchable

Тип: `System.Boolean`

Если `true`, то из результата будут исключены сотрудники с флагом `NotSearchable`

Возвращаемое значение

Тип: `System.Collections.Generic.IList<StaffEmployee>`

Список сотрудников подразделения

IStaffService.GetUnitEmployees — метод (StaffUnit, Boolean, Boolean)

Возвращает список сотрудников указанного подразделения.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
IList<StaffEmployee> GetUnitEmployees(StaffUnit unit,  bool hierarchy,  bool selection)
```

Параметры

unit

Тип: `StaffUnit`

Подразделение, из которого возвращается список сотрудников

hierarchy

Тип: `System.Boolean`

При `true` в итоговый список сотрудников будут включены все подчиненные сотрудники, при `false` только подчиненные непосредственно `unit`

selection

Тип: `System.Boolean`

При `true` будут возвращены сотрудники, у которых отсутствует признак "Не показывать при выборе"

Возвращаемое значение

Тип: System.Collections.Generic.IList<StaffEmployee>

Список сотрудников подразделения

Примеры

Пример получения всего списка сотрудников, доступных для выбора, независимо от иерархии подразделения

```
①

StaffUnit unit = objectContext.GetObject<StaffUnit>(new Guid("00000000-0000-0000-0000-
000000000000"));
IStaffService staffService = objectContext.GetService<IStaffService>();

var employees = staffService.GetUnitEmployees(unit, true, true);
```

① Инициализация контекста объектов.

IStaffService.GetUnitEmployees — метод (StaffUnit, Boolean, Boolean, Boolean)

Возвращает список сотрудников указанного подразделения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IList<StaffEmployee> GetUnitEmployees(StaffUnit parentUnit, bool selection, bool
useNotAvailable, bool useNotSearchable)
```

Параметры

unit

Тип: StaffUnit

Подразделение, из которого возвращается список сотрудников

selection

Тип: System.Boolean

При true будут учтены значения useNotAvailable и useNotSearchable

useNotAvailable

Тип: System.Boolean

Если true, то из результата будут исключены сотрудники с флагом NotAvailable

useNotSearchable

Тип: System.Boolean

Если true, то из результата будут исключены сотрудники с флагом NotSearchable

Возвращаемое значение

Тип: System.Collections.Generic.IList<StaffEmployee>

Список сотрудников подразделения

IStaffService.GetUnitEmployees — метод (StaffUnit, Boolean, Boolean, Boolean, Boolean)

Возвращает список сотрудников указанного подразделения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IList<StaffEmployee> GetUnitEmployees(StaffUnit unit, bool hierarchy, bool selection,
bool useNotAvailable, bool useNotSearchable)
```

Параметры

unit

Тип: StaffUnit

Подразделение, из которого возвращается список сотрудников

hierarchy

Тип: System.Boolean

При true в итоговый список сотрудников будут включены все подчиненные сотрудники, при false только подчиненные непосредственно unit

selection

Тип: System.Boolean

При true будут учтены значения useNotAvailable и useNotSearchable

useNotAvailable

Тип: System.Boolean

Если true, то из результата будут исключены сотрудники с флагом NotAvailable

useNotSearchable

Тип: System.Boolean

Если true, то из результата будут исключены сотрудники с флагом NotSearchable

Возвращаемое значение

Тип: System.Collections.Generic.IList<StaffEmployee>

Список сотрудников подразделения

IStaffService.GetUnits — метод (StaffUnit, Boolean)

Возвращает список дочерних подразделений.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IList<StaffUnit> GetUnits(StaffUnit parentUnit, bool selection)
```

Параметры

parentUnit

Тип: StaffUnit

Родительское подразделение

selection

Тип: System.Boolean

При `true` будут возвращены подразделения, которые могут быть выбраны в карточке, иначе — все подразделения

Возвращаемое значение

Тип: `System.Collections.Generic.IList<StaffUnit>`

Список подразделений типа `StaffUnit`

IStaffService.IsEmployeeInGroup — метод (StaffEmployee, StaffGroup, Boolean)

Определяет принадлежность сотрудника к указанной группе.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
bool IsEmployeeInGroup(StaffEmployee employee, StaffGroup group, bool
searchInChildGroups)
```

Параметры

employee

Тип: `StaffEmployee`

Сотрудник

group

Тип: `StaffGroup`

Группа, к которой осуществляется проверка принадлежности

searchInChildGroups

Тип: `System.Boolean`

`true` — также проверять подчиненные группы

Возвращаемое значение

Тип: `System.Boolean`

`true` — пользователь входит в группу, иначе — `false`

IStaffService.LoadEmployeeFromAD — метод (StaffEmployee)

Загружает данные сотрудника из Active Directory.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
List<string> LoadEmployeeFromAD(StaffEmployee employee)
```

Параметры

employee

Тип: StaffEmployee

Сотрудник, для которого загружаются данные

Возвращаемое значение

Тип: System.Collections.Generic.List<string>

Список обновленных атрибутов

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр employee.

Примеры

Ниже приведён пример загрузки данных текущего сотрудника из Active Directory в *справочник сотрудников*.

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

staffService.LoadEmployeeFromAD(staffService.GetCurrentEmployee()); ③
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Загрузка данных сотрудника из AD.

IStaffService.OpenOrCreateEmployeeCard — метод (StaffEmployee, Boolean, Boolean)

Создаёт или возвращает карточку сотрудника.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
EmployeeCard OpenOrCreateEmployeeCard(StaffEmployee employee, out bool createNew, bool
attachNewCardToEmployee)
```

Параметры

employee

Тип: `StaffEmployee`

Сотрудник, для которого возвращается карточка

createNew

Тип: `System.Boolean`

Признак создания новой карточки: `true` — если была создана новая карточка, иначе — `false`

attachNewCardToEmployee

Тип: `System.Boolean`

Флаг прикрепления карточки сотрудника к сотруднику: `true` — прикреплять, иначе — `false`

Возвращаемое значение

Тип: `EmployeeCard`

Карточка сотрудника

Заметки

Метод `OpenOrCreateEmployeeCard` возвращает существующую карточку сотрудника, если она отсутствует, то будет создана новая. При создании новой карточки флаг `createNew` устанавливается в значение **true**. По умолчанию, если создаётся

новая карточка, она не прикрепляется к записи сотрудника — `StaffEmployee.CardEmployee` остается в null. Чтобы изменить данное поведение, флаг `attachNewCardToEmployee` должен быть выставлен в значение **true**.

IStaffService.OtherEmployeeWithSameAccountExists — метод (StaffEmployee, String)

Определяет наличие сотрудников с указанной учетной записью.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
bool OtherEmployeeWithSameAccountExists(StaffEmployee employee, string name)
```

Параметры

employee

Тип: `StaffEmployee`

Сотрудник, который не должен учитываться при проверке

name

Тип: `StaffEmployee`

Проверяемая учетная запись

Возвращаемое значение

Тип: `System.Boolean`

`true` — есть сотрудники с указанной учетной записью

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `employee`.

Заметки

Данный метод может быть использован при добавлении нового сотрудника, чтобы предупредить дублирование информации.

IStaffService.OtherPositionWithSameNameExists — метод (StaffPosition, String)

Определяет наличие должности с указанным названием.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool OtherPositionWithSameNameExists(StaffPosition position, string name)
```

Параметры

position

Тип: StaffPosition

Должность, которая должна быть исключена из результатов поиска

name

Тип: System.String

Искомое название

Возвращаемое значение

Тип: System.Boolean

true, если найдены должности (кроме position) с названием name

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр position или name.

IStaffService.PropagateNotAvailable — метод (StaffUnit)

Распространяет признак "Не показывать при выборе" с родительского подразделения на дочерние подразделения и сотрудников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void PropagateNotAvailable(StaffUnit staffUnit)
```

Параметры

staffUnit

Тип: StaffUnit

Подразделение *Справочника сотрудников*, для дочерних элементов которого устанавливается признак "Не показывать при выборе"

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр staffUnit.

Примеры

Ниже приведён пример распространения признака "Не показывать при выборе" с подразделения с идентификатором 00000000-0000-0000-0000-000000000000, на вложенные подразделения и сотрудников

```
①

StaffUnit unit = objectContext.GetObject<StaffUnit>(new Guid("00000000-0000-0000-0000-
000000000000"));
IStaffService staffService = objectContext.GetService<IStaffService>();

staffService.PropagateNotAvailable(unit);
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

IStaffService.RemoveGroupEmployee — метод (StaffGroup, StaffEmployee)

Удаляет сотрудника из группы *справочника сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool RemoveGroupEmployee(StaffGroup group, StaffEmployee employee)
```

Параметры

group

Тип: StaffGroup

Группа, из которой удаляется сотрудник

employee

Тип: StaffEmployee

Сотрудник, удаляемый из группы

Возвращаемое значение

Тип: System.Boolean

true — если удаляемый элемент был найден и удалён, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр group или employee.

IStaffService.RemoveGroupItems — метод (StaffGroup, IEnumerable<StaffGroupItem>)

Удаляет элементы (группы, сотрудники) из группы *справочника сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void RemoveGroupItems(StaffGroup group, IEnumerable<StaffGroupItem> items)
```

Параметры

group

Тип: StaffGroup

Группа из которой должны быть удалены элементы

items

Тип: `System.Collections.Generic.IEnumerable<StaffGroupItem>`

Список удаляемых элементов типа `StaffGroupItem`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `group` или `items`.

IStaffService.SetCurrentEmployee — метод (StaffEmployee)

Переопределяет текущего сотрудника, возвращаемого методом `GetCurrentEmployee`.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void SetCurrentEmployee(StaffEmployee employee)
```

Параметры

employee

Тип: `StaffEmployee`

Сотрудник

Исключения

Исключение	Условие
`System.InvalidOperationException`	Ошибка будет возвращена, если текущий владелец сессии не входит в группу **DocsVision Administrators** или *DocsVision Power Users*.

Примеры

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffEmployee staffEmployee = staffService.Get(new Guid("00000000-0000-0000-0000-
000000000000")); ③

staffService.SetCurrentEmployee(staffEmployee); ④
```

① Инициализация контекста объектов.

② Получение сервиса для работы со Справочником сотрудников.

③ Получение сотрудника, которым будет подменен текущий сотрудник.

④ Подмена сотрудника.

IStaffService.SetFoldersRights — метод (FoldersFolder, String)

Предоставляет полные права доступа к папке сотруднику с указанной учетной записью.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void SetFoldersRights(FoldersFolder folder, string accountName)
```

Параметры

folder

Тип: xFoldersFolder

Папка

accountName

Тип: System.String

Учетная запись сотрудника, которому выдаются права

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр folder.

IStaffService.SynchronizeGroupWithActiveDirectory — метод (StaffGroup, IADSynchronizationSettings, ILongProcessManager)

Синхронизирует данные группы *справочника сотрудников* с Active Directory.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void SynchronizeGroupWithActiveDirectory(StaffGroup group, IADSynchronizationSettings settings, ILongProcessManager syncManager)
```

Параметры

group

Тип: StaffGroup

Синхронизируемая группа

settings

Тип: IADSynchronizationSettings

Параметры синхронизации

syncManager

Тип: ILongProcessManager

Менеджер длительных процессов

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр group, settings или syncManager.

Заметки

Пример инициализации менеджера длительных процессов см. в примере.

IStaffService.SynchronizeRoleWithActiveDirectory — метод (StaffRole, IADSynchronizationSettings, ILongProcessManager)

Синхронизирует данные роли *справочника сотрудников* с Active Directory.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void SynchronizeRoleWithActiveDirectory(StaffRole role, IADSynchronizationSettings
settings, ILongProcessManager syncManager)
```

Параметры

role

Тип: StaffRole

Синхронизируемая роль

settings

Тип: IADSynchronizationSettings

Параметры синхронизации

syncManager

Тип: ILongProcessManager

Менеджер длительных процессов

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр role, settings или syncManager.

Заметки

Пример инициализации менеджера длительных процессов см. в примере.

IStaffService.SynchronizeUnitWithActiveDirectory — метод ([Тип параметра перегрузки])

Синхронизирует данные подразделения *справочника сотрудников* с Active Directory.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void SynchronizeUnitWithActiveDirectory(StaffUnit unit, IADSynchronizationSettings
settings, ILongProcessManager syncManager)
```

Параметры

unit

Тип: StaffUnit

Синхронизируемое подразделение

settings

Тип: IADSynchronizationSettings

Параметры синхронизации

syncManager

Тип: ILongProcessManager

Менеджер длительных процессов

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр unit, settings или syncManager.

Заметки

Пример инициализации менеджера длительных процессов см. в примере.

IStaffService.SynchronizeWithActiveDirectory — метод (IADSynchronizationSettings, ILongProcessManager)

Синхронизирует данные групп, ролей и подразделений *справочника сотрудников* с Active Directory.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void SynchronizeWithActiveDirectory(IADSynchronizationSettings settings,
ILongProcessManager syncManager)
```

Параметры

settings

Тип: IADSynchronizationSettings

Параметры синхронизации Справочника сотрудников и Active Directory

syncManager

Тип: ILongProcessManager

Менеджер длительных процессов

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр settings или syncManager.

Заметки

Синхронизация элементов *справочника сотрудников* возможна при наличии настроенной связи соответствующего элемента с контейнером в Active Directory.

Примеры

Пример синхронизации доступен по ссылке.

IStaffService.TransferEmployee — метод (StaffEmployee, StaffUnit, Boolean)

Переносит сотрудника в указанное подразделение.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StaffEmployee TransferEmployee(StaffEmployee employee, StaffUnit unit, bool moveFolders)
```

Параметры

employee

Тип: StaffEmployee

Сотрудник, предполагаемый к переносу

unit

Тип: StaffUnit

Подразделение в которое выполняется перенос сотрудника

moveFolders

Тип: System.Boolean

При значении true будет выполнен перенос персональной папки сотрудника в папку нового подразделения

Возвращаемое значение

Тип: StaffEmployee

Ссылка на запись перемещенного сотрудника

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр employee или unit.

Заметки

При переносе создаётся фактическая копия записи сотрудника в новом подразделении, старая запись отмечается признаком "Переведен".

Примеры

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffEmployee employee = staffService.Get(new Guid("00000000-0000-0000-0000-
000000000000"));
StaffUnit unit = objectContext.GetObject<StaffUnit>(new Guid("00000000-0000-0000-0000-
000000000001")); ③

StaffEmployee transferedEmployee = staffService.TransferEmployee(employee, unit, false);
④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение сотрудника и подразделения, в которое осуществляется перенос.

④ Перенос сотрудника.

IStaffService.ValidateUser — метод

Проверяет вхождение текущего пользователя в доменную группу **DocsVision Administrators**.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
bool ValidateUser()
```

Возвращаемое значение

Тип: `System.Boolean`

`true` — пользователь является администратором, иначе — `false`

IStateService — интерфейс

Сервис для взаимодействия с *конструктором состояний*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IStateService
```

Свойства

Имя	Описание
StatesDictionary	Возвращает ссылку на конструктор состояний.

Методы

Имя	Описание
AddBranch(StatesCardKindStateSetting, StatesState, StatesState, StatesOperation)	Создаёт новый переход в автомате состояний вида карточки.
AddOperation(StatesCardKindStateSetting, Boolean)	Добавляет операцию в настройки автомата состояний вида карточки.
AddOperation(StatesCardKindStateSetting, String, Boolean)	Добавляет операцию в настройки автомата состояний вида карточки.
AddOperationLocalizedDescription(StatesOperation, Int32, String)	Добавляет локализованное описание операции.
AddOperationLocalizedName(StatesOperation, Int32, String)	Добавляет локализованное название операции.
AddState(StatesCardKindStateSetting)	Добавляет новое состояние в автомат состояний вида карточки.
AddStateLocalizedName(StatesState, Int32, String)	Добавляет локализованное название состояния.
AreOperationsAllowed(IEnumerable<BuiltInOperation>, BaseCard)	Проверяет возможность выполнения встроенных операций, применительно к определённой карточке.
AreOperationsAllowed(IEnumerable<StatesOperation>, BaseCard)	Проверяет возможность выполнения операций (*конструктора состояний*), применительно к определённой карточке.

Имя	Описание
AreOperationsAllowedFull(IEnumerable<BuiltInOperation>, BaseCard)	Проверяет возможность выполнения встроенных операций, применительно к определённой карточке, с учетом прав сотрудника.
AreOperationsAllowedFull(IEnumerable<StatesOperation>, BaseCard)	Проверяет возможность выполнения операций (*конструктора состояний*), применительно к определённой карточке, с учетом прав сотрудника.
CanDeleteOperation(StatesOperation)	Проверяет возможность удаления операции из *конструктора состояний*.
ChangeState(BaseCard, StatesState)	Изменяет состояние карточки.
ChangeState(BaseCard, StatesStateMachineBranch)	Изменяет состояние карточки.
ChangeState(BaseCard, StatesState, Boolean, String)	Изменяет состояние карточки.
ChangeState(BaseCard, StatesStateMachineBranch, Boolean, String)	Изменяет состояние карточки.
ChangeState(BaseCard, StatesStateMachineBranch, IEnumerable<KindsCardProcess>, String)	Изменяет состояние карточки.
CreateStateMachineLayout(StatesCardKindStateSetting)	Создаёт разметку автомата состояний.
FindBranchByBuiltIn(BuiltInBranch, StatesState)	Возвращает переход автомата состояний, соответствующий определённому начальному состоянию и указанному встроенному переходу.
FindLineBranchesByStartState(StatesState)	Возвращает все допустимые переходы состояний карточки, имеющей указанное начальное состояние.
FindOperationByBuiltIn(BuiltInOperation, BaseCard)	Возвращает операцию *конструктора состояний*, соответствующую указанной встроенной операции.

Имя	Описание
FindOperationByBuiltIn(BuiltInOperation, KindsCardKind)	Возвращает операцию *конструктора состояний*, соответствующую указанной встроенной операции.
FindScriptBranchesByStartState(StatesState)	Возвращает переходы автомата состояний, в которых смена состояния осуществляется по скрипту.
FindStateByBuiltIn(BuiltInState, BaseCard)	Возвращает состояние карточки, представленное в *Конструкторе состояний* и соответствующее переданному встроенному состоянию.
GetKindStateSetting(KindsCardKind)	Возвращает настройки вида карточки, заданные в *Конструкторе состояний*.
GetOperations(KindsCardKind)	Возвращает список операций (редактирования и перехода), определённых в *Конструкторе состояний* для указанного вида карточки.
GetOrCreatePreferredCardKindSetting(KindsCardKind)	Возвращает, а при необходимости создаёт, настройки автомата состояний вида карточки.
GetStateMachineBranches(KindsCardKind)	Возвращает список доступных, для указанного вида карточки, переходов автомата состояний.
GetStateMachineLayout(KindsCardKind)	Возвращает разметку автомата состояний для вида карточки.
GetStates(KindsCardKind)	Возвращает список возможных состояний карточки.
IsOperationAllowed(BuiltInOperation, BaseCard)	Определяет возможность выполнения встроенной операции, применительно к определённой карточке.

Имя	Описание
IsOperationAllowed(StatesOperation, BaseCard)	Определяет возможность выполнения операции (*конструктора состояний*), применительно к определённой карточке.
IsOperationAllowedFull(BuiltInOperation, BaseCard)	Определяет возможность выполнения встроенной операции, применительно к определённой карточке, с учетом прав доступа.
IsOperationAllowedFull(StatesOperation, BaseCard)	Определяет возможность выполнения операции (*конструктора состояний*), применительно к определённой карточке, с учетом прав доступа.
RemoveKindStateSetting(KindsCardKind)	Удаляет настройки автомата состояний вида карточки из *конструктора состояний*.

IStateService.StatesDictionary — свойство

Возвращает объектную модель *конструктора состояний*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesDictionary StatesDictionary { get; }
```

Значение свойства

Тип: StatesDictionary

Конструктор состояний

IStateService.AddBranch — метод (StatesCardKindStateSetting, StatesState, StatesState, StatesOperation)

Создаёт новый переход в автомате состояний вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesStateMachineBranch AddBranch(StatesCardKindStateSetting kindStateSettings,
StatesState startState, StatesState endState, StatesOperation operation)
```

Параметры

kindStateSettings

Тип: StatesCardKindStateSetting

Настройки автомата состояний для вида карточки

startState

Тип: StatesState

Начальное состояние перехода

endState

Тип: StatesState

Конечное состояние перехода

operation

Тип: StatesOperation

Операция перехода

Возвращаемое значение

Тип: StatesStateMachineBranch

Переход автомата состояний

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр kindStateSettings, startState или operation.

IStateService.AddOperation — метод (StatesCardKindStateSetting, Boolean)

Добавляет операцию в настройки автомата состояний вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesOperation AddOperation(StatesCardKindStateSetting kindStateSettings, bool
editModeOnly)
```

Параметры

kindStateSettings

Тип: StatesCardKindStateSetting

Настройки автомата состояний вида карточки в *конструкторе состояний*

editModeOnly

Тип: System.Boolean

Если true, то операция доступна только в режиме редактирования

Возвращаемое значение

Тип: StatesOperation

Добавленное состояние

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр kindStateSettings.

Заметки

Название добавляемой операции будет сформировано на основе названия операции по умолчанию — "Операция N"

IStateService.AddOperation — метод (StatesCardKindStateSetting, String, Boolean)

Добавляет операцию в настройки автомата состояний вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesOperation AddOperation(StatesCardKindStateSetting kindStateSettings, string name,
bool editModeOnly)
```

Параметры

kindStateSettings

Тип: StatesCardKindStateSetting

Настройки автомата состояний вида карточки в *конструкторе состояний*

name

Тип: System.String

Название добавляемой операции

editModeOnly

Тип: System.Boolean

Если true, то операция доступна только в режиме редактирования

Возвращаемое значение

Тип: StatesOperation

Добавленное состояние

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр kindStateSettings.

IStateService.AreOperationsAllowed — метод (IEnumerable<BuiltInOperation>, BaseCard)

Проверяет возможность выполнения встроенных операций, применительно к определённой карточке.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IDictionary<BuiltInOperation, bool> AreOperationsAllowed(IEnumerable<BuiltInOperation>
builtInOperations, BaseCard baseCard)
```

Параметры

builtInOperations

Тип: System.Collections.Generic.IEnumerable<BuiltInOperation>

Список проверяемых встроенных операций

baseCard

Тип: BaseCard

Карточка, для которой выполняется проверка

Возвращаемое значение

Тип: System.Collections.Generic.IDictionary<BuiltInOperation, Boolean>

Коллекция пар: встроенная операция и признак того, что она может быть выполнена (если значение — true)

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр builtInOperations или baseCard.

Заметки

Список операций проверяется исключительно на основе данных *конструктора состояние*; проверка возможности выполнения операции по ролевой модели не выполняется.

IStateService.AreOperationsAllowed — метод (IEnumerable<StatesOperation>, BaseCard)

Проверяет возможность выполнения операций (*конструктора состояний*), применительно к определённой карточке.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IDictionary<StatesOperation, bool> AreOperationsAllowed(IEnumerable<StatesOperation>
statesOperations, BaseCard baseCard)
```

Параметры

statesOperations

Тип: System.Collections.Generic.IEnumerable<StatesOperation>

Список проверяемых операций

baseCard

Тип: BaseCard

Карточка, для которой выполняется проверка

Возвращаемое значение

Тип: System.Collections.Generic.IDictionary<StatesOperation, Boolean>

Коллекция пар: операция и признак того, что она может быть выполнена (если значение — `true`)

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр `statesOperations` или `baseCard`.

Заметки

Список операций проверяется исключительно на основе данных *конструктора состояние*; проверка возможности выполнения операции по ролевой модели не выполняется.

IStateService.AreOperationsAllowedFull — метод (IEnumerable<BuiltInOperation>, BaseCard)

Проверяет возможность выполнения встроенных операций, применительно к определённой карточке, с учетом прав сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IDictionary<BuiltInOperation, bool> AreOperationsAllowed(IEnumerable<BuiltInOperation>
builtInOperations, BaseCard baseCard)
```

Параметры

builtInOperations

Тип: System.Collections.Generic.IEnumerable<BuiltInOperation>

Список проверяемых встроенных операций

baseCard

Тип: BaseCard

Карточка, для которой выполняется проверка

Возвращаемое значение

Тип: System.Collections.Generic.IDictionary<BuiltInOperation, Boolean>

Коллекция пар: встроенная операция и признак того, что она может быть выполнена (если значение — true)

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр builtInOperations или baseCard.

IStateService.AddOperationLocalizedDescription — метод (StatesOperation, Int32, String)

Добавляет локализованное описание операции *конструктора состояний*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesOperationDescription AddOperationLocalizedDescription(StatesOperation operation,
int localeId, string name)
```

Параметры

operation

Тип: StatesOperation

Операция, к которой добавляется локализованное описание

localeId

Тип: System.Int32

Номер локали

name

Тип: System.String

Локализованное описание

Возвращаемое значение

Тип: StatesOperationDescription

Локализованное описание операции

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр operation, либо передан пустой name.
System.ArgumentOutOfRangeException	Ошибка возвращается в случае, если передан отрицательный localeId.
System.PlatformNotSupportedException	Ошибка возвращается в случае, если локаль localeId не поддерживается.

IStateService.AddState — метод (StatesCardKindStateSetting)

Добавляет новое состояние в автомат состояний вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesState AddState(StatesCardKindStateSetting kindStateSettings)
```

Параметры

kindStateSettings

Тип: StatesCardKindStateSetting

Настройки автомата состояний вида карточки

Возвращаемое значение

Тип: StatesState

Добавленное состояний

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр kindStateSettings.

IStateService.AreOperationsAllowedFull — метод (IEnumerable<StatesOperation>, BaseCard)

Проверяет возможность выполнения операций (*конструктора состояний*), применительно к определённой карточке, с учетом прав сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IDictionary<StatesOperation, bool> AreOperationsAllowedFull(IEnumerable<StatesOperation>
statesOperations, BaseCard baseCard)
```

Параметры

statesOperations

Тип: System.Collections.Generic.IEnumerable<StatesOperation>

Список проверяемых встроенных операций

baseCard

Тип: BaseCard

Карточка, для которой выполняется проверка

Возвращаемое значение

Тип: System.Collections.Generic.IDictionary<StatesOperation, Boolean>

Коллекция пар: операция и признак того, что она может быть выполнена (если значение — true)

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр statesOperations или baseCard.

IStateService.ChangeState — метод (BaseCard, StatesState)

Изменяет состояние карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ChangeState(BaseCard card, StatesState endState)
```

Параметры

card

Тип: BaseCard

Карточка, для которой выполняется смена состояния

endState

Тип: StatesState

Новое состояние карточки

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр card или endState.

Примеры

```
①

IStateService stateService = objectContext.GetService<IStateService>(); ②

Task card = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

stateService.ChangeState(card, stateService.FindStateByBuiltIn(Task.CompletedState,
card)); ④
```

① Инициализация контекста объектов.

② Получение сервиса для работы с состояниями.

③ Получение задания.

④ Смена состояния на состояние "Завершено".

IStateService.ChangeState — метод (BaseCard, StatesStateMachineBranch)

Изменяет состояние карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ChangeState(BaseCard card, StatesStateMachineBranch branch)
```

Параметры

card

Тип: BaseCard

Карточка, для которой выполняется смена состояния

branch

Тип: StatesStateMachineBranch

Переход автомата состояний

Примеры

Далее приведён пример завершения исполнения карточки, следуя переходу состояний — "Начато-Завершено"

```
①

IStateService stateService = objectContext.GetService<IStateService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

stateService.ChangeState(task, stateService.FindBranchByBuiltIn(Task
.NotStartedToCompletedByComplete, task.SystemInfo.State)); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Задание, у которого изменяется состояние.

④ Смена состояния.

IStateService.ChangeState — метод (BaseCard, StatesState, Boolean, String)

Изменяет состояние карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ChangeState(BaseCard card, StatesState endState, bool startBusinessProcess, out
string startBusinessProcessErrors)
```

Параметры

card

Тип: BaseCard

Карточка, для которой выполняется смена состояния

endState

Тип: StatesState

Конечное состояние карточки

startBusinessProcess

Тип: System.Boolean

Если true, то будут запущены бизнес-процессы, связанные с изменением состояния карточки

startBusinessProcessErrors

Тип: System.String

Журнал ошибок запуска бизнес-процессов

IStateService.ChangeState — метод (BaseCard, StatesStateMachineBranch, Boolean, String)

Изменяет состояние карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ChangeState(BaseCard card, StatesStateMachineBranch branch, bool
startBusinessProcess, out string startBusinessProcessErrors);
```

Параметры

card

Тип: BaseCard

Карточка, для которой выполняется смена состояния

branch

Тип: StatesStateMachineBranch

Переход автомата состояний

startBusinessProcess

Тип: System.Boolean

Если true, то будут запущены бизнес-процессы, связанные с изменением состояния карточки

startBusinessProcessErrors

Тип: System.String

Журнал ошибок запуска бизнес-процессов

IStateService.ChangeState — метод (BaseCard, StatesStateMachineBranch, IEnumerable<KindsCardProcess>, String)

Изменяет состояние карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ChangeState(BaseCard card, StatesStateMachineBranch branch, IEnumerable
<KindsCardProcess> checkedProcessesToStart, out string startBusinessProcessErrors)
```

Параметры

card

Тип: BaseCard

Карточка, для которой выполняется смена состояния

branch

Тип: StatesStateMachineBranch

Переход автомата состояний

checkedProcessesToStart

Тип: System.Collections.Generic.IEnumerable<KindsCardProcess>

Список бизнес-процессов, которые должны быть запущена

startBusinessProcessErrors

Тип: System.String

Журнал ошибок запуска бизнес-процессов

IStateService.FindBranchByBuiltIn — метод (BuiltInBranch, StatesState)

Возвращает переход автомата состояний, соответствующий определённому начальному состоянию и указанному встроенному переходу.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesStateMachineBranch FindBranchByBuiltIn(BuiltInBranch builtInBranch, StatesState
state)
```

Параметры

builtInBranch

Тип: BackOffice-ObjectModel-BuiltIn:BuiltInBranch_CL.adoc[BuiltInBranch]

Встроенный переход

state

Тип: StatesState

Начальное состояние

Возвращаемое значение

Тип: StatesStateMachineBranch

Переход состояний, представленный в *Конструкторе состояний*

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр builtInBranch или state.

Примеры

Ниже приведён пример использования метода FindBranchByBuiltIn для получения перехода из состояния Отложено в Отклонено, для карточки задания вида *На исполнение*

```
①

IStateService stateService = objectContext.GetService<IStateService>();
KindsCardKind cardKind = objectContext.GetObject<KindsCardKind>(new Guid("AB801854-70AF-
4B6C-AB48-1B59B5D11AA9")); ②

StatesState state = stateService.FindStateByBuiltIn(Task.DeferredState, cardKind); ③

StatesStateMachineBranch branch = stateService.FindBranchByBuiltIn(Task
.DeferredToRejectedByReject, state); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение состояния "Отложено".

④ Получение перехода из "Отложено" в "Отклонено".

IStateService.FindLineBranchesByStartState — метод (StatesState)

Возвращает все допустимые переходы состояний карточки, имеющей указанное начальное состояние.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<StatesStateMachineBranch> FindLineBranchesByStartState(StatesState state)
```

Параметры

state

Тип: StatesState

Начальное состояние

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<StatesStateMachineBranch>

Коллекция объектов типа StatesStateMachineBranch

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр state.

Примеры

Ниже приведён пример использования метода FindLineBranchesByStartState при получении всех доступных переходов из состояния Делегировано для карточки задания вида На исполнение

```
①

IStateService stateService = objectContext.GetService<IStateService>();
KindsCardKind cardKind = objectContext.GetObject<KindsCardKind>(new Guid("AB801854-70AF-
4B6C-AB48-1B59B5D11AA9")); ②

StatesState state = stateService.FindStateByBuiltIn(Task.DelegatedState, cardKind); ③

IEnumerable<StatesStateMachineBranch> branches = stateService
.FindLineBranchesByStartState(state); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение состояния "Делегировано".

④ Доступные переходы из состояния "Делегирования".

IStateService.FindOperationByBuiltIn — метод (BuiltInOperation, BaseCard)

Возвращает операцию *конструктора состояний*, соответствующую указанной встроенной операции.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesOperation FindOperationByBuiltIn(BuiltInOperation builtInOperation, BaseCard
baseCard)
```

Параметры

builtInOperation

Тип: BackOffice-ObjectModel-BuiltIn:BuiltInOperation_CL.adoc[BuiltInOperation]

Встроенная операция

baseCard

Тип: BaseCard

Карточка, относительно типа которой определяется переход

Возвращаемое значение

Тип: StatesOperation

Операция *конструктора состояний*

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр builtInOperation.

IStateService.FindOperationByBuiltIn — метод (BuiltInOperation, KindsCardKind)

Возвращает операцию (редактирования или перехода), определённую в *Конструкторе состояний* и соответствующую заданной встроенной операции.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesOperation FindOperationByBuiltIn(BuiltInOperation builtInOperation, KindsCardKind
kind)
```

Параметры

builtInOperation

Тип: BackOffice-ObjectModel-BuiltIn:BuiltInOperation_CL.adoc[BuiltInOperation]

Встроенная операция

kind

Тип: KindsCardKind

Вид карточки

Возвращаемое значение

Тип: StatesOperation

Операция *конструктора состояний*

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр builtInOperation.

IStateService.FindStateByBuiltIn — метод (BuiltInState, BaseCard)

Возвращает состояние карточки, представленное в *Конструкторе состояний* и соответствующее переданному встроенному состоянию.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesState FindStateByBuiltIn(BuiltInState builtInState, BaseCard baseCard)
```

Параметры

builtInState

Тип: BackOffice-ObjectModel-BuiltIn:BuiltInState_CL.adoc[BuiltInState]

Встроенное состояние

baseCard

Тип: BaseCard

Карточка, для которой выполняется

Возвращаемое значение

Тип: StatesState

Состояние, определённое в *Конструкторе состояний*

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр builtInState.

Заметки

Параметр baseCard используется для получения вида карточки, из параметров которого может быть найдено соответствие с переданным состоянием builtInState.

Примеры

```
①

IStateService stateService = objectContext.GetService<IStateService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

StatesState state = stateService.FindStateByBuiltIn(Task.RejectedState, task); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение задания.

④ Состояние карточки, соответствующее builtInState.

IStateService.GetKindStateSetting — метод (KindsCardKind)

Возвращает настройки вида карточки, заданные в *Конструкторе состояний*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesCardKindStateSetting GetKindStateSetting(KindsCardKind cardKind)
```

Параметры

cardKind

Тип: `KindsCardKind`

Вид карточки

Возвращаемое значение

Тип: `StatesCardKindStateSetting`

Параметры *Конструктора состояний* для вида карточки

Примеры

Далее приведён пример использования метода `GetKindStateSetting` при получении начального состояния карточки задание вида "На исполнение".

```
①

IStateService stateService = objectContext.GetService<IStateService>();
KindsCardKind cardKind = objectContext.GetObject<KindsCardKind>(new Guid("AB801854-70AF-
4B6C-AB48-1B59B5D11AA9")); ②

StatesCardKindStateSetting stateSetting = stateService.GetKindStateSetting(cardKind); ③

StatesState startState = stateSetting.FirstState; ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение настроек вида.

④ Получение начального состояния.

IStateService.GetOperations — метод (KindsCardKind)

Возвращает список операций (редактирования и перехода), определённых в *Конструкторе состояний* для указанного вида карточки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
IList<StatesOperation> GetOperations(KindsCardKind cardKind);
```

Параметры

cardKind

Тип: KindsCardKind

Вид карточки

Возвращаемое значение

Тип: System.Collections.Generic.IList<StatesOperation>

Список операций в виде списка объектов типа StatesOperation

Заметки

Если для данного вида задействовано наследование, то будет возвращен список операций от родительского вида карточки.

IStateService.GetOrCreatePreferredCardKindSetting — метод (KindsCardKind)

Возвращает, а при необходимости создаёт, настройки автомата состояний вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesCardKindStateSetting GetOrCreatePreferredCardKindSetting(KindsCardKind cardKind)
```

Параметры

cardKind

Тип: KindsCardKind

Вид карточки

Возвращаемое значение

Тип: StatesCardKindStateSetting

Настройки автомата состояний

Заметки

В том случае, если вид `cardKind` наследует свои настройки от родительского вида, настройки автомата состояний будут сформированы для него (для родительского вида с собственными настройками).

IStateService.GetStateMachineBranches — метод (KindsCardKind)

Возвращает список доступных, для указанного вида карточки, переходов автомата состояний.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
IList<StatesStateMachineBranch> GetStateMachineBranches(KindsCardKind cardKind)
```

Параметры

cardKind

Тип: `KindsCardKind`

Вид карточки, для которого должны быть получены переходы автомата состояний

Возвращаемое значение

Тип: `System.Collections.Generic.IList<StatesStateMachineBranch>`

Список переходов автомата состояний, типа `StatesStateMachineBranch`

IStateService.GetStates — метод (KindsCardKind)

Возвращает список возможных состояний карточки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
IList<StatesState> GetStates(KindsCardKind cardKind)
```

Параметры

cardKind

Тип: KindsCardKind

Вид карточки

Возвращаемое значение

Тип: System.Collections.Generic.IList<StatesState>

Список операций в виде списка объектов типа StatesOperation

Заметки

Если для данного вида задействовано наследование, то будет возвращен список состояний от родительского вида карточки.

IStateService.IsOperationAllowed — метод (BuiltInOperation, BaseCard)

Определяет возможность выполнения встроенной операции, применительно к определённой карточке.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool IsOperationAllowed(BuiltInOperation builtInOperation, BaseCard baseCard)
```

Параметры

builtInOperation

Тип: BackOffice-ObjectModel-BuiltIn:BuiltInOperation_CL.adoc[BuiltInOperation]

Встроенная операция

baseCard

Тип: BaseCard

Карточка, для которой выполняется проверка

Возвращаемое значение

Тип: System.Boolean

`true` — операция допустима, иначе — `false`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `builtInOperation` или `baseCard`.

Заметки

Доступность операции проверяется исключительно на основе данных *конструктора состояние*.

Примеры

В примере демонстрируется использование метода `IsOperationAllowed` для определения возможности делегирования задания другому сотруднику

```
①

IStateService stateService = objectContext.GetService<IStateService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

bool isValid = stateService.IsOperationAllowed(Task.DelegateOperation, task); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение задания, для которого определяется возможность делегирования.

④ Проверка возможности совершения операции делегирования.

IStateService.IsOperationAllowed — метод (StatesOperation, BaseCard)

Определяет возможность выполнения операции (*конструктора состояний*), применительно к определённой карточке.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
bool IsOperationAllowed(StatesOperation operation, BaseCard baseCard)
```

Параметры

operation

Тип: `StatesOperation`

Операция из *конструктора состояний*

baseCard

Тип: `BaseCard`

Карточка, для которой выполняется проверка

Возвращаемое значение

Тип: `System.Boolean`

`true` — операция допустима, иначе — `false`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `operation` или `baseCard`.

IStateService.IsOperationAllowedFull — метод (BuiltInOperation, BaseCard)

Данный метод определяет возможность выполнения указанной операции с заданной карточкой, с учетом прав доступа.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
bool IsOperationAllowedFull(BuiltInOperation builtInOperation, BaseCard baseCard)
```

Параметры

builtInOperation

Тип: BackOffice-ObjectModel-BuiltIn:BuiltInOperation_CL.adoc[BuiltInOperation]

Встроенная операция

baseCard

Тип: BaseCard

Карточка, для которой выполняется проверка

Возвращаемое значение

Тип: System.Boolean

`true` — операция допустима, иначе — `false`

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр `builtInOperation` или `baseCard`.

Примеры

В примере демонстрируется использование метода `IsOperationAllowedFull` для определения возможности отклонения задания текущим сотрудником

```
①

IStateService stateService = objectContext.GetService<IStateService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

bool isValid = stateService.IsOperationAllowedFull(Task.RejectOperation, task); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение задания, для которого определяется возможность делегирования.

④ Проверка возможности отклонения сотрудником.

IStateService.IsOperationAllowedFull — метод (StatesOperation, BaseCard)

Определяет возможность выполнения операции (*конструктора состояний*), применительно к определённой карточке, с учетом прав доступа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool IsOperationAllowedFull(StatesOperation operation, BaseCard baseCard)
```

Параметры

operation

Тип: StatesOperation

Операция из *конструктора состояний*

baseCard

Тип: BaseCard

Карточка, для которой выполняется проверка

Возвращаемое значение

Тип: System.Boolean

true — операция допустима, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр operation или baseCard.

ISurveyService — интерфейс

Сервис для работы со списком опросов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface ISurveyService
```

Методы

Имя	Описание
`AddAnswer(SurveyListSurvey, StaffEmployee, StatesState, DateTime)`	Добавляет ответ на вопрос опроса.
`AddAnswerIssue(SurveyListSurveyAnswer, SurveyListSurveyQuestion)`	Добавляет элемент вопрос-ответ.
`AddIssueBaseUniversalItemValue(SurveyListSurveyAnswerIssue, BaseUniversalItem)`	Создаёт значение ответа в виде записи конструктора справочников.
`AddIssueBooleanValue(SurveyListSurveyAnswerIssue, Boolean)`	Создаёт значение ответа в виде логического значения.
`AddIssueCardReferenceValue(SurveyListSurveyAnswerIssue, Guid)`	Создаёт значение ответа в виде ссылки на карточку.
`AddIssueCategoryValue(SurveyListSurveyAnswerIssue, CategoriesCategory)`	Создаёт значение ответа в виде категории.
`AddIssueDateTimeValue(SurveyListSurveyAnswerIssue, DateTime)`	Создаёт значение ответа в виде дата/время.
`AddIssueDepartmentValue(SurveyListSurveyAnswerIssue, StaffUnit)`	Создаёт значение ответа в виде организации.
`AddIssueDoubleValue(SurveyListSurveyAnswerIssue, Double)`	Создаёт значение ответа в виде значения с плавающей запятой.
`AddIssueEmployeesValue(SurveyListSurveyAnswerIssue, IEnumerable<Guid>)`	Создаёт коллекцию значений ответов в виде сотрудников.
`AddIssueEmployeeValue(SurveyListSurveyAnswerIssue, StaffEmployee)`	Создаёт значение ответа в виде сотрудника.
`AddIssueEnumValue(SurveyListSurveyAnswerIssue, SurveyListSurveyQuestionEnumValue)`	Создаёт значение ответа в виде перечисления.
`AddIssueFolderValue(SurveyListSurveyAnswerIssue, FoldersFolder)`	Создаёт значение ответа в виде папки.

Имя	Описание
`AddIssueGroupValue(SurveyListSurveyAnswerIssue, StaffGroup)`	Создаёт значение ответа в виде группы сотрудников.
`AddIssueImageValue(SurveyListSurveyAnswerIssue, Guid)`	Создаёт значение ответа в виде изображения.
`AddIssueIntegerValue(SurveyListSurveyAnswerIssue, Int32)`	Создаёт значение ответа в виде целого числа.
`AddIssueNumeratorValue(SurveyListSurveyAnswerIssue, String, Guid)`	Создаёт значение ответа в виде значения нумератора.
`AddIssueOperationValue(SurveyListSurveyAnswerIssue, StatesOperation)`	Создаёт значение ответа в виде операции.
`AddIssuePartnerCompanyValue(SurveyListSurveyAnswerIssue, PartnersCompany)`	Создаёт значение ответа в виде организации контрагента.
`AddIssuePartnerEmployeeValue(SurveyListSurveyAnswerIssue, PartnersEmployee)`	Создаёт значение ответа в виде сотрудника контрагента.
`AddIssueRoleModelRoleValue(SurveyListSurveyAnswerIssue, RoleModelRole)`	Создаёт значение ответа в виде роли ролевой модели.
`AddIssueStateValue(SurveyListSurveyAnswerIssue, StatesState)`	Создаёт значение ответа в виде состояния.
`AddIssueStringValue(SurveyListSurveyAnswerIssue, String)`	Создаёт значение ответа в виде строки.
`AddIssueVariantValue(SurveyListSurveyAnswerIssue, Object)`	Создаёт значение ответа в виде неявного значения (тип `System.Object`).
`AddQuestion(SurveyListSurvey, String, String, SurveyListSurveyQuestionDataType)`	Добавляет вопрос к опросу.
`AddQuestionEnumValue(SurveyListSurveyQuestion, Int32, String)`	Добавляет варианты ответов к вопросу.
`AddSignature(SurveyListSurvey, SurveyListSurveyAnswer, X509Certificate2)`	Подписывает ЭП ответ на опрос.
`AddSurvey(SurveyList, String)`	Создаёт опрос в списке опросов.

Имя	Описание
`CreateSignatureList`	Создаёт карточку списка подписей. Объект автоматически сохраняется в контексте.
`CreateSurveyList`	Создаёт карточку списка опросов.
`FindAnswerIssue(SurveyListSurveyAnswer, SurveyListSurveyQuestion)`	Осуществляет поиск в ответах указанного значения ответа.
`FindOrCreateAnswerIssue(SurveyListSurveyAnswer, SurveyListSurveyQuestion)`	Осуществляет поиск в ответах указанного значения ответа. В случае отсутствия — будет создан новый элемент вопрос-ответ.
`FindOrCreateQuestion(SurveyListSurvey, String, String, SurveyListSurveyQuestionDataType)`	Осуществляет поиск вопроса в опросе. В случае отсутствия — будет создан новый.
`FindOrCreateSurvey(SurveyList, String)`	Осуществляет поиск опроса в списке опросов. В случае отсутствия — будет создан новый.
`FindQuestion(SurveyListSurvey, String)`	Осуществляет поиск вопроса в опросе.
`FindSurvey(SurveyList, String)`	Осуществляет поиск опроса в списке опросов.
`GetAnswerHash(SurveyListSurveyAnswer)`	Вычисляет хэш `SHA512` для коллекции вопрос-ответ, содержащейся в ответе на опрос.
`GetIssueValue(SurveyListSurveyQuestionDataType, IEnumerable<SurveyListSurveyAnswerIssueValue>)`	Получает коллекцию значений из коллекции ответов.
`GetIssueValue(SurveyListSurveyQuestionDataType, SurveyListSurveyAnswerIssueValue)`	Получает значение из ответа.
`VerifySignature(SurveyListSurveyAnswer, X509Certificate2)`	Проверяет действительность установленной на ответ ЭП.

Примеры

Создание опроса является довольно простой задачей. Основной момент здесь —

это реализация соответствующей разметки в карточке, к которой привязан опрос. Ниже приведён пример создания простого опроса

```
①

ISurveyService surveyService = objectContext.GetService<ISurveyService>(); ②

SurveyList surveyList = surveyService.CreateSurveyList(); ③

SurveyListSurvey surveyListSurvey = surveyService.AddSurvey(surveyList, "Утренние
опросы"); ④
surveyListSurvey.OneTime = true; ⑤
surveyListSurvey.Sign = true; ⑥
surveyListSurvey.SignatureList = surveyService.CreateSignatureList();

SurveyListSurveyQuestion surveyListSurveyQuestion = surveyService.AddQuestion
(surveyListSurvey, "Доброе", "Скажите, является ли утро добрым?",
SurveyListSurveyQuestionDataType.Boolean); ⑦

objectContext.AcceptChanges(); ⑧
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Создание списка.

④ Добавление опроса в список опросов.

⑤ Отвечать единожды.

⑥ Подписывать.

⑦ Добавляем вопрос, тип данных `boolean`.

⑧ Сохранение.

ITaskGroupService — интерфейс

Сервис для работы со группами заданий.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface ITaskGroupService
```

Методы

Имя	Описание
AddDelegatePreset(TaskGroupPresets, StaffEmployee)	Добавляет исполнителя в индивидуальные настройки исполнителя группы заданий.
AddDelegatePreset(TaskGroupPresets, StaffGroup)	Добавляет группу сотрудников в число исполнителей в индивидуальные настройки исполнителя группы заданий.
AddDelegatePreset(TaskGroupPresets, StaffRole)	Добавляет роль в число исполнителей в индивидуальные настройки исполнителя группы заданий.
AddDelegatePreset(TaskGroupPresets, StaffUnit)	Добавляет подразделение в число исполнителей в индивидуальные настройки исполнителя группы заданий.
AddDelegatePresetSearchWord(TaskGroupPresets, Guid)	Добавляет исполнителя с индивидуальными настройками. Исполнитель задаётся в виде поискового слова.
AddSelectedPerformer(TaskGroup, Guid)	Добавляет поисковое слово в число выбранных исполнителей группы заданий.
AddSelectedPerformer(TaskGroup, StaffEmployee)	Добавляет сотрудника в список выбранных исполнителей группы заданий.
AddSelectedPerformer(TaskGroup, StaffGroup)	Добавляет группу в число выбранных исполнителей группы заданий.
AddSelectedPerformer(TaskGroup, StaffRole)	Добавляет роль в число выбранных исполнителей группы заданий.
AddSelectedPerformer(TaskGroup, StaffUnit)	Добавляет подразделение в число выбранных исполнителей группы заданий.

Имя	Описание
CreateChildTaskGroup(Task, KindsCardKind, BaseCard, TaskList)	Создаёт новую подчиненную группу заданий.
CreateTaskGroup(KindsCardKind)	Создаёт группу заданий указанного вида.
GetKindSettings(KindsCardKind)	Возвращает настройки вида группы заданий.
GetTaskGroupCompletionMailNotificationInfo (TaskGroup, String)	Получает почтовое сообщение информирующее о завершении группы заданий.
RecallTaskGroup(TaskGroup)	Отзывает группу заданий.
ReorderSelectedPerformers(ObjectCollection <TaskGroupSelectedPerformer>)	Повторно выстраивает порядок исполнения группы задания исполнителями.
SendToPerformance(TaskGroup, KindsCardKind)	Отправляет группу заданий на исполнение.
SendToPerformance(TaskGroup, KindsCardKind, KindsCardKind)	Отправляет группу заданий на исполнение.
ValidateBeforeSendToPerformance(TaskGroup, String)	Определяет готовность группы заданий к отправке на исполнение.
ValidateInactivePerformers(TaskGroup, String)	Проверяет группу заданий на отсутствие активных исполнителей.

Примеры

Ниже приведён пример использования сервиса при отправке на исполненные группы заданий.

```
①

ITaskGroupService taskGroupService = objectContext.GetService<ITaskGroupService>();
IStaffService staffService = objectContext.GetService<IStaffService>();
IStateService stateService = objectContext.GetService<IStateService>(); ②

QueryObject query = new QueryObject(RefKinds.CardKinds.Name, "Группа заданий УД");
KindsCardKind cardKind = objectContext.FindObject<KindsCardKind>(query); ③
```

```
TaskGroup taskGroup = taskGroupService.CreateTaskGroup(cardKind);
taskGroup.MainInfo.ExecutionType = TaskGroupExecutionType.Parallel;
taskGroup.MainInfo.Author = staffService.GetCurrentEmployee();
taskGroup.Description = "Подготовить финансовый отчёт";
taskGroup.MainInfo.Name = "Подготовить финансовый отчёт"; ④
objectContext.AcceptChanges();

query = new QueryObject(RefStaff.Units.Name, "Бухгалтерия");
StaffUnit unit = objectContext.FindObject<StaffUnit>(query);
taskGroupService.AddSelectedPerformer(taskGroup, unit); ⑤
objectContext.AcceptChanges();

Guid defaultTaskCardKindId = taskGroupService.GetKindSettings(taskGroup.SystemInfo
.CardKind).DefaultTaskCardKind; ⑥

taskGroupService.SendToPerformance(taskGroup, objectContext.GetObject<KindsCardKind
>(defaultTaskCardKindId));
stateService.ChangeState(taskGroup, stateService.FindStateByBuiltIn(TaskGroup
.PerformanceState, taskGroup)); ⑦
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение вида группы заданий.

④ Создание группы заданий с параллельным исполнением.

⑤ Добавление подразделения в число исполнителей.

⑥ Получение вида задания, создаваемого по умолчанию, из настроек вида группы заданий.

⑦ Отправка группы заданий на исполнение.

ITaskGroupService.AddDelegatePreset — метод (TaskGroupPresets, StaffEmployee)

Добавляет исполнителя в индивидуальные настройки исполнителя группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskGroupPresetsDelegate AddDelegatePreset(TaskGroupPresets presets, StaffEmployee
```

```
employee)
```

Параметры

presets

Тип: TaskGroupPresets

Индивидуальные настройки исполнителя

Возвращаемое значение

Тип: TaskGroupPresetsDelegate

Исполнитель

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр presets.

ITaskGroupService.AddDelegatePreset — метод (TaskGroupPresets, StaffGroup)

Добавляет группу сотрудников в число исполнителей в индивидуальные настройки исполнителя группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskGroupPresetsDelegate AddDelegatePreset(TaskGroupPresets presets, StaffGroup group)
```

Параметры

presets

Тип: TaskGroupPresets

Группа заданий

group

Тип: StaffGroup

Группа сотрудников

Возвращаемое значение

Тип: TaskGroupPresetsDelegate

Индивидуальные настройки исполнителя группы заданий

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр presets.

ITaskGroupService.AddDelegatePreset — метод (TaskGroupPresets, StaffRole)

Добавляет роль в число исполнителей в индивидуальные настройки исполнителя группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskGroupPresetsDelegate AddDelegatePreset(TaskGroupPresets presets, StaffRole role)
```

Параметры

presets

Тип: TaskGroupPresets

Группа заданий

role

Тип: StaffRole

Роль

Возвращаемое значение

Тип: TaskGroupPresetsDelegate

Индивидуальные настройки исполнителя группы заданий

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр presets.

ITaskGroupService.AddDelegatePreset — метод (TaskGroupPresets, StaffUnit)

Добавляет подразделение в число исполнителей в индивидуальные настройки исполнителя группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskGroupPresetsDelegate AddDelegatePreset(TaskGroupPresets presets, StaffUnit unit)
```

Параметры

presets

Тип: TaskGroupPresets

Группа заданий

unit

Тип: StaffUnit

Подразделение

Возвращаемое значение

Тип: TaskGroupPresetsDelegate

Индивидуальные настройки исполнителя группы заданий

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр presets.

ITaskGroupService.AddDelegatePresetSearchWord — метод (TaskGroupPresets, Guid)

Добавляет поисковое слово в число исполнителей в индивидуальные настройки

исполнителя группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskGroupPresetsDelegate AddDelegatePresetSearchWord(TaskGroupPresets presets, Guid
searchWordId)
```

Параметры

presets

Тип: TaskGroupPresets

Группа заданий

searchWordId

Тип: System.Guid

Идентификатор поискового слова

Возвращаемое значение

Тип: TaskGroupPresetsDelegate

Индивидуальные настройки исполнителя группы заданий

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр presets.

ITaskGroupService.AddSelectedPerformer — метод (TaskGroup, StaffEmployee)

Добавляет сотрудника в список выбранных исполнителей группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskGroupSelectedPerformer AddSelectedPerformer(TaskGroup routeStage, StaffEmployee
```

```
employee)
```

Параметры

routeStage

Тип: TaskGroup

Группа заданий, в список выбранных исполнителей которой добавляется сотрудник

employee

Тип: StaffEmployee

Добавляемый сотрудник

Возвращаемое значение

Тип: TaskGroupSelectedPerformer

Выбранный исполнитель

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр routeStage или employee.

Примеры

Ниже приведён пример добавления текущего сотрудника в список выбранных исполнителей группы заданий.

```
①

ITaskGroupService taskGroupService = objectContext.GetService<ITaskGroupService>();
IStaffService staffService = objectContext.GetService<IStaffService>(); ②

TaskGroup taskGroup = objectContext.GetObject<TaskGroup>(new Guid("00000000-0000-0000-
0000-000000000000")); ③

TaskGroupSelectedPerformer selectedPerformer = taskGroupService.AddSelectedPerformer
(taskGroup, staffService.GetCurrentEmployee()); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение группы заданий.

④ Добавление исполнителя.

ITaskGroupService.AddSelectedPerformer — метод (TaskGroup, Guid)

Добавляет поисковое слово в число выбранных исполнителей группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskGroupSelectedPerformer AddSelectedPerformer(TaskGroup routeStage, Guid searchWordId)
```

Параметры

routeStage

Тип: TaskGroup

Группа заданий, в список выбранных исполнителей которой добавляется сотрудник

searchWordId

Тип: System.Guid

Идентификатор поискового слова

Возвращаемое значение

Тип: TaskGroupSelectedPerformer

Выбранный исполнитель

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр routeStage.

Заметки

Идентификаторы основных поисковых слов предоставляет класс `PerformerSearchWordsManager`.

ITaskGroupService.AddSelectedPerformer — метод (TaskGroup, StaffGroup)

Добавляет группу в число выбранных исполнителей группы заданий.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
TaskGroupSelectedPerformer AddSelectedPerformer(TaskGroup routeStage, StaffGroup group)
```

Параметры

routeStage

Тип: `TaskGroup`

Группа заданий, в список выбранных исполнителей которой добавляется сотрудник

group

Тип: `StaffGroup`

Группа сотрудников

Возвращаемое значение

Тип: `TaskGroupSelectedPerformer`

Выбранный исполнитель

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `routeStage` или `group`.

ITaskGroupService.AddSelectedPerformer — метод (TaskGroup, StaffRole)

Добавляет роль в число выбранных исполнителей группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskGroupSelectedPerformer AddSelectedPerformer(TaskGroup routeStage, StaffRole role)
```

Параметры

routeStage

Тип: TaskGroup

Группа заданий, в список выбранных исполнителей которой добавляется сотрудник

role

Тип: StaffRole

Добавляемая роль

Возвращаемое значение

Тип: TaskGroupSelectedPerformer

Выбранный исполнитель

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр routeStage или role.

ITaskGroupService.AddSelectedPerformer — метод (TaskGroup, StaffUnit)

Добавляет подразделение в число выбранных исполнителей группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskGroupSelectedPerformer AddSelectedPerformer(TaskGroup routeStage, StaffUnit unit)
```

Параметры

routeStage

Тип: `TaskGroup`

Группа заданий, в список выбранных исполнителей которой добавляется сотрудник

unit

Тип: `StaffUnit`

Добавляемое подразделение

Возвращаемое значение

Тип: `TaskGroupSelectedPerformer`

Выбранный исполнитель

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `routeStage` или `unit`.

ITaskGroupService.CreateChildTaskGroup — метод (Task, KindsCardKind, BaseCard, TaskList)

Создаёт новую подчиненную группу заданий.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
TaskGroup CreateChildTaskGroup(Task parentTask, KindsCardKind kind, BaseCard parentCard,
TaskList taskList)
```

Параметры

parentTask

Тип: `Task`

Задание, в которое может быть добавлена группа. Может быть передан `null`, тогда группа заданий будет добавлена в `taskList`

kind

Тип: KindsCardKind

Вид группы заданий

parentCard

Тип: KindsCardKind

Карточка, в которую будет добавлена группы заданий

taskList

Тип: TaskList

Список подписей, к которому будет прикреплена новая группа заданий

Возвращаемое значение

Тип: TaskGroup

Созданная группа заданий

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр kind, parentCard или taskList.

Примеры

Ниже приведён пример создания группы заданий в существующем документе

```
①

ITaskGroupService taskGroupService = objectContext.GetService<ITaskGroupService>(); ②

KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("6D76D0A7-5434-40F2-912E-6370D33C3151")); ③

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-000000000000")); ④

taskGroupService.CreateChildTaskGroup(null, kind, document, document.MainInfo.Tasks);
objectContext.AcceptChanges(); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса для работы с группой заданий.

③ Получение вида группы заданий (например, "Группа заданий УД").

④ Получение документа, в который будет добавлена группа заданий.

⑤ Добавление группы заданий.

ITaskGroupService.CreateTaskGroup — метод (KindsCardKind)

Создаёт группу заданий указанного вида.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
TaskGroup CreateTaskGroup(KindsCardKind kind)
```

Параметры

kind

Тип: `KindsCardKind`

Вид группы заданий

Возвращаемое значение

Тип: `TaskGroup`

Группа заданий

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `kind`.

Примеры

Ниже приведён пример создания группы вида "Группа заданий УД".

```
①
```

```
ITaskGroupService taskGroupService = objectContext.GetService<ITaskGroupService>();
IStaffService staffService = objectContext.GetService<IStaffService>(); ②

QueryObject query = new QueryObject(RefKinds.CardKinds.Name, "Группа заданий УД");
KindsCardKind cardKind = objectContext.FindObject<KindsCardKind>(query); ③

TaskGroup taskGroup = taskGroupService.CreateTaskGroup(cardKind);
taskGroup.MainInfo.Author = staffService.GetCurrentEmployee();
taskGroup.Description = "Подготовить финансовый отчёт";
taskGroup.MainInfo.Name = "Подготовить финансовый отчёт"; ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение вида группы заданий.

④ Создание группы заданий.

ITaskGroupService.GetKindSettings — метод (KindsCardKind)

Возвращает настройки вида группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskGroupSetting GetKindSettings(KindsCardKind kind)
```

Параметры

kind

Тип: KindsCardKind

Вид группы заданий, для которого получаем настройки

Возвращаемое значение

Тип: TaskGroupSetting

Настройки вида

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр kind.

Заметки

В данном случае имеются в виду настройки, которые были установлены в *справочнике видов карточек* у вида группы заданий.

ITaskGroupService.RecallTaskGroup — метод (TaskGroup)

Отзывает группу заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string RecallTaskGroup(TaskGroup taskGroup)
```

Параметры

taskGroup

Тип: TaskGroup

Отзываемая группа заданий

Возвращаемое значение

Тип: System.String

Сообщения всех заданий об отзыве

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр taskGroup.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для отзыва группы заданий.

Примеры

```
①

ITaskGroupService taskGroupService = objectContext.GetService<ITaskGroupService>();
IStateService stateService = objectContext.GetService<IStateService>(); ②

TaskGroup taskGroup = objectContext.GetObject<TaskGroup>(new Guid("00000000-0000-0000-
0000-000000000000")); ③

taskGroupService.RecallTaskGroup(taskGroup);
stateService.ChangeState(taskGroup, stateService.FindStateByBuiltIn(TaskGroup
.RecalledState, taskGroup)); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение отзываемой группы заданий.

④ Отзыв группы заданий.

ITaskGroupService.ReorderSelectedPerformers — метод (ObjectCollection<TaskGroupSelectedPerformer>)

Повторно выстраивает порядок исполнения группы задания исполнителями.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ReorderSelectedPerformers(ObjectCollection<TaskGroupSelectedPerformer>
selectedPerformers)
```

Параметры

selectedPerformers

Тип: ObjectCollection<TaskGroupSelectedPerformer>

Коллекция исполнителей группы заданий — TaskGroupSelectedPerformer

Заметки

Данный метод просто заново проставляет значение Order у исполнителя группы заданий.

ITaskGroupService.SendToPerformance — метод (TaskGroup, KindsCardKind)

Отправляет группу заданий на исполнение.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void SendToPerformance(TaskGroup taskGroup, KindsCardKind taskKind)
```

Параметры

taskGroup

Тип: TaskGroup

Отправляемая на исполнение группа заданий

taskKind

Тип: KindsCardKind

Вид заданий, отправляемых исполнителям

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр taskGroup или taskKind.

ITaskGroupService.SendToPerformance — метод (TaskGroup, KindsCardKind, KindsCardKind)

Отправляет группу заданий на исполнение.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void SendToPerformance(TaskGroup taskGroup, KindsCardKind taskKind, KindsCardKind
taskKindResponsiblePerformer)
```

Параметры

taskGroup

Тип: TaskGroup

Отправляемая на исполнение группа заданий

taskKind

Тип: KindsCardKind

Вид заданий, отправляемых исполнителям

taskKindResponsiblePerformer

Тип: KindsCardKind

Вид задания, отправляемого ответственному исполнителю

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр taskGroup, taskKind или taskKindResponsiblePerfomer.

ITaskGroupService.ValidateBeforeSendToPerformance — метод (TaskGroup, String)

Определяет готовность группы заданий к отправке на исполнение.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool ValidateBeforeSendToPerformance(TaskGroup assignment, out string errorMessage)
```

Параметры

assignment

Тип: TaskGroup

Группа заданий

errorMessage

Тип: System.String

Список ошибок, мешающих отправке

Возвращаемое значение

Тип: System.Boolean

true — группа заданий может быть отправлена, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр assignment.

Заметки

В ходе проверки определяется заполнение следующих данных: автор; название; контроллер или ответственный исполнитель, если группа заданий на контроле; исполнители. Также проверяется вхождение индивидуальных сроков исполнения заданий исполнителями в общий срок исполнения группы заданий.

ITaskGroupService.ValidateInactivePerformers — метод (TaskGroup, String)

Проверяет группу заданий на отсутствие активных исполнителей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool ValidateInactivePerformers(TaskGroup taskGroup, out string errorMessage)
```

Параметры

taskGroup

Тип: TaskGroup

Группа заданий

errorMessage

Тип: System.String

Сообщение об ошибке

Возвращаемое значение

Тип: `System.Boolean`

`true` — в списке исполнителей есть активные исполнители, иначе — `false`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `taskGroup`.

Заметки

Активные исполнители — исполнители, у которых состояние находится в значении "Активен".

ITaskListService — интерфейс

Сервис `ITaskListService` предоставляет методы для работы со списками заданий.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface ITaskListService
```

Методы

Имя	Описание
`AddTask(TaskList, Task, BaseCard)`	Добавляет существующее задание в предоставленный список заданий.
`AddTaskGroup(TaskList, TaskGroup, BaseCard)`	Добавляет группу заданий в предоставленный список заданий.
`CopyTask(TaskList, Task, BaseCard)`	Создаёт копию указанного задания в списке заданий.

Имя	Описание
`CreateTask(TaskList, Task, Boolean, StaffEmployee)`	Добавляет существующее задание в предоставленный список заданий, без внесения изменений в историю карточки.
`CreateTaskGroup(TaskList, TaskGroup)`	Добавляет существующую группу заданий в предоставленный список заданий, без внесения изменений в историю карточки.
`CreateTaskList()`	Инициализирует новый пустой список заданий.
`GetTasksTree(TaskList)`	Возвращает список узлов дерева заданий, построенного из списка заданий.
`GetTasksTree(IEnumerable<Task>, IEnumerable<TaskGroup>)`	Возвращает список узлов дерева заданий, построенного из списка заданий.
`GetTasksTree(TaskList, IEnumerable<Guid>)`	Возвращает список узлов дерева заданий, построенного из списка заданий.
`GetTasksTree(IEnumerable<Task>, IEnumerable<TaskGroup>, IEnumerable<Guid>)`	Возвращает список узлов дерева заданий, построенного из списка заданий.
`RemoveTask(TaskList, Task, BaseCard)`	Удаляет задание из переданного списка заданий.
`RemoveTaskGroup(TaskList, TaskGroup, BaseCard)`	Удаляет группу заданий из списка заданий.

Примеры

Приведенный далее пример демонстрирует использование сервиса ITaskListService при создании списка заданий в карточке *Документ*.

```
①

ITaskListService taskListService = objectContext.GetService<ITaskListService>();
ITaskService taskService = objectContext.GetService<ITaskService>();
```

```
IStaffService staffService = objectContext.GetService<IStaffService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("AB801854-70AF-4B6C-
AB48-1B59B5D11AA9")); ④

Task task = taskService.CreateTask(kind);
task.MainInfo.Name = "Задание документа";
task.MainInfo.Author = staffService.GetCurrentEmployee(); ⑤

TaskList taskList = taskListService.CreateTaskList(); ⑥

taskListService.AddTask(taskList, task, document); ⑦
document.MainInfo.Tasks = taskList;
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение необходимых сервисов.

③ Получение документа, к которому добавляется список заданий.

④ Получение вида задания (например, "На исполнение").

⑤ Создание нового задания.

⑥ Создание нового списка заданий.

⑦ Добавление задания в список заданий.

ITaskListService.AddTask — метод (TaskList, Task, BaseCard)

Добавляет существующее задание в предоставленный список заданий

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskListTask AddTask(TaskList taskList, Task task, BaseCard parentCard)
```

Параметры

taskList

Тип: TaskList

Список заданий

task

Тип: `Task`

Добавляемое задание

parentCard

Тип: `BaseCard`

Карточка, содержащая список заданий, в который осуществляется добавление

Возвращаемое значение

Тип: `TaskListTask`

Задание в списке заданий

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `taskList`, `task` или `parentCard`.

Заметки

Данный метод записывает событие добавления подчиненного задания в журнал карточки parentCard.

ITaskListService.AddTaskGroup — метод (TaskList, TaskGroup, BaseCard)

Добавляет группу заданий в предоставленный список заданий.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
TaskListTaskGroup AddTaskGroup(TaskList taskList, TaskGroup taskGroup, BaseCard
parentCard)
```

Параметры

taskList

Тип: `TaskList`

Список заданий, в который добавляется группа заданий

taskGroup

Тип: `TaskGroup`

Добавляемая группа заданий

parentCard

Тип: `BaseCard`

Карточка, содержащая список заданий, в который добавляется группа заданий

Возвращаемое значение

Тип: `TaskListTaskGroup`

Группа карточек

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `taskList`, `taskGroup` или `parentCard`.

Заметки

Дополнительно записывает событие добавления группы карточек в историю работы с карточкой.

ITaskListService.CopyTask — метод (TaskList, Task, BaseCard)

Создаёт копию указанного задания в списке заданий.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
TaskListTask CopyTask(TaskList taskList, Task task, BaseCard parentCard)
```

Параметры

taskList

Тип: TaskList

Список заданий, в который осуществляется копирование

task

Тип: Task

Копируемое задание

parentCard

Тип: BaseCard

Карточка, содержащая список заданий

Возвращаемое значение

Тип: TaskListTask

Полученная копия задания

ITaskListService.CreateTask — метод (TaskList, Task, Boolean, StaffEmployee)

Добавляет существующее задание в предоставленный список заданий, без внесения изменений в историю карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskListTask CreateTask(TaskList taskList, Task task, bool onControl, StaffEmployee
controller)
```

Параметры

taskList

Тип: TaskList

Список заданий, в который осуществляется добавление задания

task

Тип: `Task`

Добавляемое задание

onControl

Тип: `System.Boolean`

Признак постановки задания на контроль. Если `true` — контроль установлен, иначе — `false`

controller

Тип: `StaffEmployee`

Контролёр задания, если `onControl = true`

Возвращаемое значение

Тип: `TaskListTask`

Добавленное задание

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `taskList` или `task`.

Заметки

В отличие от метода `AddTask`, метод `CreateTask` не осуществляет записи события в историю карточки. Кроме этого, данный метод позволяет указать признак наличия контроля для задания дополнительно к тому, что указано в самом задании.

Примеры

Далее приведён пример использования метода `CreateTask` при добавлении нового задания с признаком "На контроле" в существующий документ

```
①
```

```
IStaffService staffService = objectContext.GetService<IStaffService>();
ITaskService taskService = objectContext.GetService<ITaskService>();
ITaskListService taskListService = objectContext.GetService<ITaskListService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

TaskList taskList = document.MainInfo.Tasks; ④

KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("AB801854-70AF-4B6C-
AB48-1B59B5D11AA9")); ⑤

Task task = taskService.CreateTask(kind);
task.MainInfo.Name = "Задание документа";
task.MainInfo.Author = staffService.GetCurrentEmployee();
task.MainInfo.OnControl = true;
task.MainInfo.Controller = task.MainInfo.Author;

taskListService.CreateTask(taskList, task,task.MainInfo.OnControl, task.MainInfo
.Controller); ⑥
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение документа.

④ Получение существующего списка заданий из документа.

⑤ Получение вида задания.

⑥ Добавление задания в список заданий.

ITaskListService.CreateTaskGroup — метод (TaskList, TaskGroup)

Добавляет существующую группу заданий в предоставленный список заданий, без внесения изменений в историю карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskListTaskGroup CreateTaskGroup(TaskList taskList, TaskGroup taskGroup)
```

Параметры

taskList

Тип: TaskList

Список заданий, в который будет добавлена группа заданий

taskGroup

Тип: TaskGroup

Добавляемая группа заданий

Возвращаемое значение

Тип: TaskListTaskGroup

Добавленная группа заданий

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр taskList или taskGroup.

Заметки

Данный метод, в отличие от метода AddTaskGroup, не добавляет соответствующую запись в историю карточки.

Примеры

```
①

IStaffService staffService = objectContext.GetService<IStaffService>();
ITaskGroupService taskGroupService = objectContext.GetService<ITaskGroupService>();
ITaskListService taskListService = objectContext.GetService<ITaskListService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000"));
TaskList taskList = document.MainInfo.Tasks; ③

KindsCardKind kind = objectContext.GetObject<KindsCardKind>(new Guid("6D76D0A7-5434-40F2-
912E-6370D33C3151")); ④

TaskGroup taskGroup = taskGroupService.CreateTaskGroup(kind);
taskGroup.MainInfo.Name = "Группа заданий документа";
```

```
taskGroup.MainInfo.Author = staffService.GetCurrentEmployee(); ⑤

taskListService.CreateTaskGroup(taskList, taskGroup); ⑥
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение настройки.

③ Получение документа со списком заданий.

④ Получение вида карточки для группы заданий (в примере — Группа заданий УД).

⑤ Создание новой группы заданий.

⑥ Добавление группы в список заданий.

ITaskListService.CreateTaskList — метод

Инициализирует новый пустой список заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskList CreateTaskList()
```

Возвращаемое значение

Тип: TaskList

Пустой список заданий

Заметки

Список заданий используется, например, в карточке *Документа* (свойство DocumentMainInfo.Tasks).

ITaskListService.GetTasksTree — метод (TaskList)

Возвращает список узлов дерева заданий, построенного из списка заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<TaskTreeInfo> GetTasksTree(TaskList taskList)
```

Параметры

taskList

Тип: TaskList

Список заданий

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<TaskTreeInfo>

Коллекция информации по узлам TaskTreeInfo дерева заданий

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр taskList.

Примеры

Далее приведён пример получения дерева заданий из списка задания с использованием метода GetTasksTree

```
①

ITaskListService taskListService = objectContext.GetService<ITaskListService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

IEnumerable<TaskTreeInfo> tree = taskListService.GetTasksTree(document.MainInfo.Tasks);
④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа, содержащего список заданий.

④ Получение дерева заданий из списка заданий документа.

ITaskListService.RemoveTask — метод (TaskList, Task, BaseCard)

Удаляет задание из переданного списка заданий.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void RemoveTask(TaskList taskList, Task task, BaseCard parentCard)
```

Параметры

taskList

Тип: `TaskList`

Список заданий

task

Тип: `Task`

Удаляемое задание

parentCard

Тип: `BaseCard`

Карточка, в которой содержится `taskList`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `taskList`, `task` или `parentCard`.
`System.Exception`	Ошибка возвращается при наличии установленной на `taskList` блокировки.

Примеры

Ниже приведён пример, в котором производится удаление первого задания в списке заданий документа

```
①

ITaskListService taskListService = objectContext.GetService<ITaskListService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

TaskList taskList = document.MainInfo.Tasks; ④
Task task = taskList.Tasks.First().Task;

taskListService.RemoveTask(taskList, task, document); ⑤
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение документа с заполненным списком заданий.

④ Получение списка заданий из документа и первого задания из списка.

⑤ Удаление задания.

ITaskListService.RemoveTaskGroup — метод (TaskList, TaskGroup, BaseCard)

Удаляет группу заданий из списка заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void RemoveTaskGroup(TaskList taskList, TaskGroup taskGroup, BaseCard parentCard)
```

Параметры

taskList

Тип: TaskList

Список заданий

taskGroup

Тип: TaskGroup

Удаляемая группа заданий

parentCard

Тип: `BaseCard`

Карточка, из которой удаляется группа заданий

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `taskList`, `taskGroup` или `parentCard`.

ITaskService — интерфейс

Сервис `ITaskService` предоставляет методы создания и управления заданиями.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface ITaskService
```

Методы

Имя	Описание
`AcceptTask(Task)`	приёмка указанного задания.
`AddChildCopyField(Task)`	Добавляет в настройки задания параметры копирования поля из родительского задания в дочернее.
`AddChildKindSettings(KindsCardKind, ObjectCollection<TaskPresetChildKindSetting>, Boolean)`	Добавляет в переданную коллекцию настроек вид, доступный для выбора при создании подчинённого задания.
`AddComment(Task, StaffEmployee, String)`	Добавляет комментарий в указанное задание.
`AddComment(Task, StaffEmployee, String, DateTime)`	Добавляет комментарий в указанное задание.

Имя	Описание
AddDelegatePreset(Task, System.Guid)	Добавляет поисковое слово в список делегатов, выбираемых из задания, при делегировании задания вручную.
AddDelegatePreset(Task, StaffEmployee)	Добавляет сотрудника в список делегатов, выбираемых из задания, при делегировании задания вручную.
AddDelegatePreset(Task, StaffUnit)	Добавляет подразделение в список делегатов, выбираемых из задания, при делегировании задания вручную.
AddDelegatePreset(Task, StaffRole)	Добавляет роль в список делегатов, выбираемых из задания, при делегировании задания вручную.
AddDelegatePreset(Task, StaffGroup)	Добавляет группу сотрудников в список делегатов, выбираемых из задания, при делегировании задания вручную.
AddLinkOnParentCard(Task, TaskSetting, BaseCard)	Добавляет в задание ссылку на родительскую карточку.
AddPerformer(Task, StaffEmployee)	Добавляет сотрудника в список назначенных исполнителей задания.
AddSelectedPerformer(TaskMainInfo, Guid)	Добавляет выбранного исполнителя задания в формате поискового слова.
AddSelectedPerformer(TaskMainInfo, StaffEmployee)	Добавляет указанного сотрудника в список выбранных исполнителей задания.
AddSelectedPerformer(TaskMainInfo, StaffGroup)	Добавляет выбранного исполнителя задания в формате группы сотрудников.
AddSelectedPerformer(TaskMainInfo, StaffRole)	Добавляет выбранного исполнителя задания в формате роли.
AddSelectedPerformer(TaskMainInfo, StaffUnit)	Добавляет выбранного исполнителя задания в формате подразделения.

Имя	Описание
AddSignature(Task, X509Certificate2, StatesOperation, Guid, IEnumerable<CardFieldSetting>)	Данный метод добавляет новую подпись в список подписей карточки *Задание*, и устанавливает ЭП на указанные поля карточки.
AddSignature(Task, X509Certificate2, StatesOperation, Guid, IEnumerable<CardFieldSetting>, IEnumerable<Document>)	Данный метод добавляет новую подпись в список подписей карточки *Задание*, и устанавливает ЭП на указанные поля карточки, а также на указанные документы.
AddSignatureToCardReconcileLog(Task, X509Certificate2)	Подписывает ЭП карточку хода согласования.
CancelDelegate(Task)	Отменяет делегирование задания.
CanCompleteTask(Task, TaskSetting, String)	Проверяет возможность завершения задания.
CanCompleteTaskByCompletionParameter(Task, StatesStateMachineBranch)	Проверяет возможность завершения задания.
ChangeTask(Task)	Добавляет в историю карточки запись об изменении задания.
ChangeTaskController(Task, StaffEmployee, DateTime)	Заменяет контролёра задания.
ChangeTaskCurrentPerformer(Task, StaffEmployee)	Заменяет текущего исполнителя задания.
ChangeTaskExecutionDate(Task, DateTime, DateTime, String)	Устанавливает новые сроки исполнения задания.
CompleteTask(Task, String)	Завершает указанное задание.
CompleteTask(Task, String, StaffEmployee)	Завершает задание от имени указанного сотрудника.
CopyDataFromParentTask(Task, TaskSetting, Task)	Копирует значения и ссылки из родительского задания в дочернее.
CopyResultsToParentTask(Task, Task, TaskCopyResultsOptions)	Переносит результаты выполнения дочернего задания в родительское.
CopyTask(Task)	Создаёт копию задания.

Имя	Описание
CreateChildTask(Task, KindsCardKind, BaseCard, TaskList)	Создаёт дочернее задание.
CreateSignatureList	Создаёт пустой список подписей.
CreateTask	Создаёт пустое задание.
CreateTask(KindsCardKind)	Создаёт новое задание указанного вида.
CreateTaskFromTemplate(Guid)	Создаёт задание по шаблону.
DeferTask(Task)	Откладывает задание.
Delegate(Task, IEnumerable<StaffEmployee>, StaffEmployee, Boolean, Boolean, String)	Делегирует задание.
Delegate(Task, IEnumerable<StaffEmployee>, StaffEmployee, Boolean, Boolean, String, StaffEmployee)	Делегирует задание.
DoNotTakeToWorkTask(Task)	Отменяет взятие задания в работу.
GetCompletionParameters(Task)	Возвращает параметры завершения задания.
GetKindSettings(KindsCardKind)	Возвращает настройки вида задания.
GetLastDelegate(Task)	Возвращает последнюю запись списка делегирования.
GetLastDelegate(Task, Boolean)	Возвращает последнюю запись списка делегирования.
GetSelectedPerformers(Task)	Возвращает выбранных исполнителей задания.
GetTaskAcceptanceMailNotificationInfo(Task, String)	Возвращает письмо, подготовленное для отправки, с информацией о передаче задания на приёмку.
GetTaskBuiltInState(Task)	Получает для задания встроенное состояние.
GetTaskCompletionMailNotificationInfo(Task, String)	Возвращает письмо, подготовленное для отправки, с информацией о завершении задания.

Имя	Описание
GetTaskRejectionMailNotificationInfo(Task, String)	Возвращает письмо, подготовленное для отправки, с информацией об отклонении задания.
HasCompletionParameterValue(Task, Guid, String)	Проверяет наличие значения у параметра завершения задания.
InitializeDefaults(Task)	Инициализация параметров нового задания.
IsCurrentUserCurrentPerformer(Task)	Возвращает признак того, что текущий сотрудник входит в список текущих исполнителей задания.
IsCurrentUserDeputyOfCurrentPerformer(Task)	Определяет, является ли текущий сотрудник заместителем одного из текущих исполнителей.
IsTaskDelegated(Task)	Возвращает признак того, что задание делегировано.
IsUserCurrentPerformer(Task, StaffEmployee)	Определяет входит ли указанный сотрудник в список текущих исполнителей.
RecallTask(Task)	Отзывает задание.
RejectTask(Task, StaffEmployee, String)	Отклоняет задание.
RemoveComment(TaskComment)	Удаляет комментарий к заданию.
ResolveDelegatePerformer(Task)	Получает список текущих исполнителей задания, за которых текущий сотрудник может делегировать задание.
ResolveDelegatePerformer(Task, StaffEmployee)	Получает список текущих исполнителей задания, за которых указанный сотрудник может делегировать задание.
SetTaskCompletionResult(Task, TaskCompletionParameter)	Устанавливает результат завершения задания.

Имя	Описание
SignatureInfo GetPartSignatureInfo(BaseCardSignaturePart signaturePart)	Возвращает информацию о части подписи документа.
StartTask(Task)	Отправляет задание на исполнение.
StopExecutionRelatedTask(Task, Boolean)	Прекращает исполнение подчиненных заданий.
StopExecutionTask(Task)	Прекращает исполнение задания.
TakeTaskToRework(Task)	Отправляет задание на доработку.
TakeToWork(Task)	Взятие задание в работу.
UpdateComment(TaskComment, String)	Обновляет содержимое комментария.
ValidateForBegin(Task, String)	Определяет готовность задания к отправке на исполнение.
ValidateInactivePerformers(Task, String)	Определяет наличие в задании активных исполнителей.
VerifySignature(Task, BaseCardSignature, X509Certificate2, Boolean, IEnumerable<CardFieldSetting>)	Выполняет проверку подлинности ЭП, установленной на полях карточки Задание, а также на Документах, связанных к заданием.
ITaskService.SignatureInfo GetPartSignatureInfo(BaseCardSignaturePart signaturePart)	Метод для получения информации о подписи

Примеры

В приведенном ниже примере демонстрируется использование сервиса для работы с заданиями: создание нового задания и отправка его на исполнение.

```
①

IBaseCardService baseCardService = objectContext.GetService<IBaseCardService>();
ITaskService taskService = objectContext.GetService<ITaskService>();
IStaffService staffService = objectContext.GetService<IStaffService>();
IStateService stateService = objectContext.GetService<IStateService>(); ②

KindsCardKind cardKind = objectContext.GetObject<KindsCardKind>(new Guid("AB801854-70AF-
```

```
4B6C-AB48-1B59B5D11AA9")); ③

Task task = taskService.CreateTask(cardKind);
taskService.InitializeDefaults(task);
task.MainInfo.Name = "Пример задания"; ④

taskService.AddSelectedPerformer(task.MainInfo, staffService.Get(new Guid("00000000-0000-
0000-0000-000000000000")));
objectContext.SaveObject(task); ⑤

CardData cardData = userSession.CardManager.GetCardData(objectContext.GetObjectRef(task
).Id);
string digest = baseCardService.GenerateDigest(task, cardData, "Задание на исполнение");
task.Description = digest; ⑥

taskService.StartTask(task);
objectContext.SaveObject(task); ⑦

StatesStateMachineBranch state = stateService.FindBranchByBuiltIn(Task
.InitializationToStartedByStart, task.SystemInfo.State); ⑧
stateService.ChangeState(task, state);
```

① Инициализация контекста объектов.

② Получаем сервисы.

③ Получаем вид карточки "На исполнение".

④ Инициализируем новое задание.

⑤ Добавляем исполнителя.

⑥ Формируем дайджест.

⑦ Отправляем задание.

⑧ Изменяем состояние задания.

ITaskService.AcceptTask — метод (Task)

приёмка указанного задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void AcceptTask(Task task)
```

Параметры

task

Тип: Task

Объектная модель задания, для которого выполняется смена состояния

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.
System.MethodAccessException	Ошибка возвращается в случае, если изменение статуса задания неразрешено.

Заметки

При вызове метода, непосредственная смена состояния задания не производится. Для изменения состояния карточки необходимо дополнительно использовать метод IStateService.ChangeState.

ITaskService.AddChildCopyField — метод (Task)

Добавляет в настройки задания параметры копирования поля из родительского задания в дочернее.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskPresetChildCopyField AddChildCopyField(Task task)
```

Параметры

task

Тип: Task

Задание, для которого выполняется настройка

Возвращаемое значение

Тип: TaskPresetChildCopyField

Объектная модель копируемого поля

Заметки

Данный метод позволяет добавить настройку копирования полей из родительского задания в подчиненное для отдельной карточки *Задания*. Указанные поля (дополнительно к тем, что определённы в *Справочнике видов карточек*) будут скопированы в подчиненное задание при создании из родительской карточки.

Примеры

В приведенном примере демонстрируется использование метода AddChildCopyField при добавлении полей, копируемых в подчиненное задание.

```
①

IMetadataProvider metadataProvider = objectContext.GetService<IMetadataProvider>();
ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

TaskPresetChildCopyField taskPresetChildCopyField = taskService.AddChildCopyField(task);
④

DocsVision.Platform.Data.Metadata.CardModel.SectionField sectionField = metadataProvider
.GetField(new Guid("F5C843C0-5CE1-4727-81BF-0C764A43243B"));
DocsVision.Platform.Data.Metadata.CardModel.CardSection section = sectionField.GetParent
<DocsVision.Platform.Data.Metadata.CardModel.CardSection>(); ⑤

taskPresetChildCopyField.SectionName = section.Name;
taskPresetChildCopyField.FieldId = sectionField.Id;
taskPresetChildCopyField.FieldName = sectionField.DisplayName.GetLocalization(System
.Globalization.CultureInfo.CurrentCulture);
taskPresetChildCopyField.FieldAlias = sectionField.Name; ⑥

Stack<Guid> sections = new Stack<Guid>(); ⑦
while (section != null)
{
 sections.Push(section.Id);
 section = section.GetParent<DocsVision.Platform.Data.Metadata.CardModel.CardSection>();
}
taskPresetChildCopyField.SectionPath = CardFieldSetting.BuildSectionPath(sections);
objectContext.SaveObject<Task>(task);
```

① Инициализация контекста объектов.

② Получение необходимых сервисов.

③ Получение задания.

④ Добавление настройки копирования поля.

⑤ Получение поля "Название" из секции "Задание".

⑥ Определение значений копируемого поля.

⑦ Формирование пути к секции.

ITaskService.AddChildKindSettings — метод (KindsCardKind, ObjectCollection<TaskPresetChildKindSetting>, Boolean)

Добавляет в переданную коллекцию настроек вид, доступный для выбора при создании подчинённого задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskPresetChildKindSetting AddChildKindSettings(KindsCardKind kindsCardKind,
ObjectCollection<TaskPresetChildKindSetting> kindSettings, bool creatable)
```

Параметры

kindsCardKind

Тип: KindsCardKind

Вид карточки

kindSettings

Тип: ObjectCollection<TaskPresetChildKindSetting>

Коллекция настроек видов (типа TaskPresetChildKindSetting) для подчиненного задания

creatable

Тип: System.Boolean

true — данный вид может быть создан из родительского задания, иначе — false

Возвращаемое значение

Тип: TaskPresetChildKindSetting

Параметры создания вида подчиненного задания

ITaskService.AddDelegatePreset — метод (Task, System.Guid)

Добавляет поисковое слово в список делегатов, выбираемых из задания, при делегировании задания вручную.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskPresetDelegate AddDelegatePreset(Task task, Guid searchWord)
```

Параметры

task

Тип: Task

Задание, в которое добавляется настройка выбора делегата

searchWord

Тип: System.Guid

Идентификатор добавляемого поискового слова

Возвращаемое значение

Тип: TaskPresetDelegate

Настройки делегирования

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

ITaskService.AddDelegatePreset — метод (Task, StaffEmployee)

Добавляет сотрудника в список делегатов, выбираемых из задания, при

делегировании задания вручную.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskPresetDelegate AddDelegatePreset(Task task, StaffEmployee employee)
```

Параметры

task

Тип: Task

Задание, в которое добавляется настройка выбора делегата

employee

Тип: StaffEmployee

Сотрудник для делегирования задания

Возвращаемое значение

Тип: TaskPresetDelegate

Настройки делегирования

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

ITaskService.AddDelegatePreset — метод (Task, StaffUnit)

Добавляет подразделение в список делегатов, выбираемых из задания, при делегировании задания вручную.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskPresetDelegate AddDelegatePreset(Task task, StaffUnit unit)
```

Параметры

task

Тип: Task

Задание, в которое добавляется настройка выбора делегата

unit

Тип: StaffUnit

Подразделение для делегирования задания

Возвращаемое значение

Тип: TaskPresetDelegate

Настройки делегирования

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

ITaskService.AddDelegatePreset — метод (Task, StaffRole)

Добавляет роль в список делегатов, выбираемых из задания, при делегировании задания вручную.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskPresetDelegate AddDelegatePreset(Task task, StaffRole role)
```

Параметры

task

Тип: Task

Задание, в которое добавляется настройка выбора делегата

role

Тип: `StaffRole`

Роль для делегирования задания

Возвращаемое значение

Тип: `TaskPresetDelegate`

Настройки делегирования

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `task`.

ITaskService.AddDelegatePreset — метод (Task, StaffGroup)

Добавляет роль в список делегатов, выбираемых из задания, при делегировании задания вручную.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
TaskPresetDelegate AddDelegatePreset(Task task, StaffGroup group)
```

Параметры

task

Тип: `Task`

Задание, в которое добавляется настройка выбора делегата

group

Тип: `StaffGroup`

Группа для делегирования задания

Возвращаемое значение

Тип: TaskPresetDelegate

Настройки делегирования

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

ITaskService.AddComment — метод (Task, StaffEmployee, String)

Добавляет комментарий в указанное задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskComment AddComment(Task task, StaffEmployee author, string text)
```

Параметры

task

Тип: Task

Задание, в которое добавляется комментарий

author

Тип: StaffEmployee

Сотрудник, который осуществляет добавление

text

Тип: System.String

Содержимое комментария

Возвращаемое значение

Тип: TaskComment

Сущность комментария

Заметки

Для добавленного комментария указывается текущая дата создания.

ITaskService.AddComment — метод (Task, StaffEmployee, String, DateTime)

Добавляет комментарий в указанное задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskComment AddComment(Task task, StaffEmployee author, string text, DateTime
creationDate)
```

Параметры

task

Тип: Task

Задание, в которое добавляется комментарий

author

Тип: StaffEmployee

Сотрудник, который осуществляет добавление

text

Тип: System.String

Содержимое комментария

creationDate

Тип: System.DateTime

Дата добавления комментария

Возвращаемое значение

Тип: TaskComment

Сущность комментария

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task, author или text.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав на добавление комментария.

ITaskService.AddLinkOnParentCard — метод (Task, TaskSetting, BaseCard)

Добавляет в задание ссылку на родительскую карточку.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void AddLinkOnParentCard(Task task, TaskSetting taskSetting, BaseCard card)
```

Параметры

task

Тип: Task

Задание, в список ссылок которого будет добавлена ссылка на карточку

taskSetting

Тип: TaskSetting

Настройки вида задания

card

Тип: BaseCard

Карточка, на которую добавляется ссылка

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task, taskSetting или card.

Заметки

При добавлении в задание ссылки на карточку card, в самой карточке ссылка на задание не формируется.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000"));
BaseCard card = objectContext.GetObject<BaseCard>(new Guid("00000000-0000-0000-0000-
000000000001")); ③

TaskSetting taskSetting = taskService.GetKindSettings(task.SystemInfo.CardKind); ④

taskService.AddLinkOnParentCard(task, taskSetting, card); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение карточек задание и его родительской карточки.

④ Получение настроек вида задания.

⑤ Добавление ссылки.

ITaskService.AddPerformer — метод (Task, StaffEmployee)

Добавляет сотрудника в список назначенных исполнителей задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void AddPerformer(Task task, StaffEmployee performer)
```

Параметры

task

Тип: `Task`

Задание, в которое назначается исполнитель

performer

Тип: `StaffEmployee`

Исполнитель, непосредственно взявший задание в исполнение

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `task` или `performer`.

Примеры

Ниже приведён простой пример использования метода `AddPerformer` для назначения исполнителя задания.

```
①

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000"));
StaffEmployee performer = objectContext.GetObject<StaffEmployee>(new Guid("00000000-0000-
0000-0000-000000000001")); ②

ITaskService taskService = objectContext.GetService<ITaskService>(); ③

taskService.AddPerformer(task, performer); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение задания и сотрудника-исполнителя по идентификаторам.

③ Получение сервиса для работы с заданиями.

④ Назначение исполнителя в задание.

ITaskService.AddSelectedPerformer — метод (TaskMainInfo, Guid)

Добавляет выбранного исполнителя задания в формате поискового слова.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskSelectedPerformer AddSelectedPerformer(TaskMainInfo mainInfo, Guid searchWordId)
```

Параметры

mainInfo

Тип: TaskMainInfo

Основная информация карточки

searchWordId

Тип: System.Guid

Идентификатор поискового слова

Возвращаемое значение

Тип: TaskSelectedPerformer

Сущность выбранного исполнителя

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр mainInfo или searchWordId.

Заметки

Идентификаторы поисковых слов могут быть получены из PerformerSearchWordsManager.

ITaskService.AddSelectedPerformer — метод (TaskMainInfo, StaffEmployee)

Добавляет указанного сотрудника в список выбранных исполнителей задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskSelectedPerformer AddSelectedPerformer(TaskMainInfo mainInfo, StaffEmployee employee)
```

Параметры

mainInfo

Тип: TaskMainInfo

Основная информация карточки

employee

Тип: StaffEmployee

Выбранный исполнитель

Возвращаемое значение

Тип: TaskSelectedPerformer

Сущность выбранного исполнителя

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр mainInfo или employee.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000"));
StaffEmployee employee = objectContext.GetObject<StaffEmployee>(new Guid("00000000-0000-
0000-0000-000000000001")); ③

taskService.AddSelectedPerformer(task.MainInfo, employee); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение задания и исполнителя для него.

④ Добавление выбранного исполнителя.

ITaskService.AddSelectedPerformer — метод (TaskMainInfo, StaffGroup)

Добавляет выбранного исполнителя задания в формате группы сотрудников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskSelectedPerformer AddSelectedPerformer(TaskMainInfo mainInfo, StaffGroup group)
```

Параметры

mainInfo

Тип: TaskMainInfo

Основная информация карточки

group

Тип: StaffGroup

Группа сотрудников

Возвращаемое значение

Тип: TaskSelectedPerformer

Сущность выбранного исполнителя

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр mainInfo или group.

ITaskService.AddSelectedPerformer — метод (TaskMainInfo, StaffRole)

Добавляет выбранного исполнителя задания в формате роли.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskSelectedPerformer AddSelectedPerformer(TaskMainInfo mainInfo, StaffRole role)
```

Параметры

mainInfo

Тип: TaskMainInfo

Основная информация карточки

role

Тип: StaffRole

Роль

Возвращаемое значение

Тип: TaskSelectedPerformer

Сущность выбранного исполнителя

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр mainInfo или role.

ITaskService.AddSelectedPerformer — метод (TaskMainInfo, StaffUnit)

Добавляет выбранного исполнителя задания в формате подразделения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskSelectedPerformer AddSelectedPerformer(TaskMainInfo mainInfo, StaffUnit unit)
```

Параметры

mainInfo

Тип: TaskMainInfo

Основная информация карточки

role

Тип: StaffUnit

Подразделение

Возвращаемое значение

Тип: TaskSelectedPerformer

Сущность выбранного исполнителя

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр mainInfo или unit.

ITaskService.AddSignature — метод (Task, X509Certificate2, StatesOperation, Guid, IEnumerable<CardFieldSetting>)

Данный метод добавляет новую подпись в список подписей карточки *Задание*, и устанавливает ЭП на указанные поля карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardSignature AddSignature(Task task, X509Certificate2 certificate, StatesOperation
operation, Guid signatureType, IEnumerable<CardFieldSetting> fields)
```

Параметры

task

Тип: Task

Задание, содержащее подписываемые данные

certificate

Тип: `System.Security.Cryptography.X509Certificates.X509Certificate2`

Сертификат сотрудника, которым будет выполнено подписания

operation

Тип: `StatesOperation`

Подписываемая операция

signatureType

Тип: `System.Guid`

Тип подписи, который может быть получен из `TaskOperationSignatureSetting.Name`

fields

Тип: `System.Collections.Generic.IEnumerable<CardFieldSetting>`

Коллекция подписываемых полей типа `CardFieldSetting`, определённых для карточки в Справочнике видов карточек

Возвращаемое значение

Тип: `BaseCardSignature`

Подпись

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `task` и `operation`.

Заметки

Настройки (подписываемые поля; операция, при которой будет произведено подписание; и т.д.), используемые при подписании, задаются в *Справочнике видов карточек*.

ITaskService.AddSignature — метод (Task, X509Certificate2, StatesOperation, Guid, IEnumerable<CardFieldSetting>, IEnumerable<Document>)

Данный метод добавляет новую подпись в список подписей карточки *Задание*, и

устанавливает ЭП на указанные поля карточки, а также на указанные документы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardSignature AddSignature(Task task, X509Certificate2 certificate, StatesOperation
operation, Guid signatureType, IEnumerable<CardFieldSetting> fields, IEnumerable<
Document> documents)
```

Параметры

task

Тип: Task

Задание, содержащее подписываемые данные

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат сотрудника, которым будет выполнено подписания

operation

Тип: StatesOperation

Подписываемая операция

signatureType

Тип: System.Guid

Тип подписи, который может быть получен из TaskOperationSignatureSetting.Name

fields

Тип: System.Collections.Generic.IEnumerable<CardFieldSetting>

Коллекция подписываемых полей типа CardFieldSetting, определённых для карточки в Справочнике видов карточек

documents

Тип: System.Collections.Generic.IEnumerable<Document>

Коллекция подписываемых документов типа Document

Возвращаемое значение

Тип: BaseCardSignature

Подпись

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task и operation.

Заметки

Настройки (подписываемые поля; операция, при которой будет произведено подписание; и т.д.), используемые при подписании, задаются в *Справочнике видов карточек*.

ITaskService.AddSignatureToCardReconcileLog — метод (Task, X509Certificate2)

Подписывает при помощи ЭП карточку хода согласования.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void AddSignatureToCardReconcileLog(Task task, X509Certificate2 certificate)
```

Параметры

task

Тип: Task

Задание, для подписания журнала

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат сотрудника, которым будет выполнено подписания

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

ITaskService.CancelDelegate — метод (Task)

Отменяет делегирование задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
StatesState CancelDelegate(Task task)
```

Параметры

task

Тип: Task

Делегированное задание

Возвращаемое значение

Тип: StatesState

Состояние задания

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.
System.MethodAccessException	Ошибка возвращается, если для отмены недостаточно прав.

ITaskService.CanCompleteTask — метод (Task, TaskSetting, String)

Проверяет возможность завершения задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanCompleteTask(Task task, TaskSetting taskSetting, out string errorMessage)
```

Параметры

task

Тип: Task

Задание

taskSetting

Тип: TaskSetting

Задание

errorMessage

Тип: System.String

Сообщение о причине недостаточности прав

Возвращаемое значение

Тип: System.Boolean

true — задание может быть завершено, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или taskSetting.

ITaskService.CanCompleteTaskByCompletionParameter — метод (Task, StatesStateMachineBranch)

Проверяет возможность завершения задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanCompleteTaskByCompletionParameter(Task task, out StatesStateMachineBranch branch)
```

Параметры

task

Тип: Task

Задание

branch

Тип: StatesStateMachineBranch

Переход автомата состояний в конечное состояние

Возвращаемое значение

Тип: System.Boolean

true — задание может быть завершено, иначе — false

ITaskService.ChangeTask — метод (Task)

Добавляет в историю карточки запись об изменении задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ChangeTask(Task task)
```

Параметры

task

Тип: Task

Задание

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

Заметки

В историю будет добавлена запись: "Задание … изменено".

ITaskService.ChangeTaskController — метод (Task, StaffEmployee, DateTime)

Заменяет контролёра задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ChangeTaskController(Task task, StaffEmployee employee, DateTime? controlDate)
```

Параметры

task

Тип: Task

Задание

employee

Тип: StaffEmployee

Новый контролёр

controlDate

Тип: System.DateTime

Новый дата контроля, если требуется изменить

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или employee.

Исключение	Условие
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для изменения контролёра.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000"));
StaffEmployee employee = objectContext.GetObject<StaffEmployee>(new Guid("00000000-0000-
0000-0000-000000000001")); ③

taskService.ChangeTaskController(task, employee, null); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение задания и нового контролёра.

④ Замена контролёра.

ITaskService.ChangeTaskCurrentPerformer — метод (Task, StaffEmployee)

Заменяет текущего исполнителя задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ChangeTaskCurrentPerformer(Task task, StaffEmployee employee)
```

Параметры

task

Тип: Task

Задание

employee

Тип: StaffEmployee

Новый текущий исполнитель

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или employee.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для выполнения операции.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000"));
StaffEmployee employee = objectContext.GetObject<StaffEmployee>(new Guid("00000000-0000-
0000-0000-000000000001")); ③

taskService.ChangeTaskCurrentPerformer(task, employee); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение задания и нового исполнителя.

④ Замена текущего исполнителя.

ITaskService.ChangeTaskExecutionDate — метод (Task, DateTime, DateTime, String)

Устанавливает новые сроки исполнения задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void ChangeTaskExecutionDate(Task task, DateTime? newStartDate, DateTime? newEndDate,
string comment)
```

Параметры

task

Тип: `Task`

Задание, для которого устанавливаются новые сроки

newStartDate

Тип: `System.DateTime`

Новая плановая дата начала

newStartDate

Тип: `System.DateTime`

Новая плановая дата завершения

comment

Тип: `System.DateTime`

Комментарий к операции изменения сроков для записи в историю карточки

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `task`, либо параметры `newStartDate` и `newEndDate` не имеют значения, либо `newStartDate` и `newEndDate` меньше минимальной даты.
`System.MethodAccessException`	Ошибка возвращается, если недостаточно прав на изменение сроков или на запись в историю карточки.
`System.ArgumentOutOfRangeException`	Ошибка возвращается, если дата `newEndDate` меньше `newStartDate`.

Заметки

Сроки могут быть изменены для отправленного задания.

Примеры

```
ITaskService taskService = objectContext.GetService<ITaskService>(); ①

Task task = objectContext.GetObject<Task>(new Guid("89C2EB62-8B6D-E511-9415-
90E6BA57B9F8"));
taskService.ChangeTaskExecutionDate(task, DateTime.Parse("10.10.2015"), DateTime.Parse
("14.10.2015"),"Сотрудник отправлен в командировку.");
objectContext.AcceptChanges();
```

① Инициализация контекста объектов

ITaskService.CompleteTask — метод (Task, String)

Завершает указанное задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BuiltInState CompleteTask(Task task, string comments)
```

Параметры

task

Тип: Task

Задание

comments

Тип: System.String

Комментарий к завершению задания

Возвращаемое значение

Тип: BackOffice-ObjectModel-BuiltIn:BuiltInState_CL.adoc[BuiltInState]

Состояние завершения

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

Заметки

Данный метод завершает задание от имени владельца сессии; дополнительно требуется вызвать IStateService.ChangeState для изменения фактического состояния карточки.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

BuiltInState state = taskService.CompleteTask(task, "Все документы получены"); ④

IStateService stateService = objectContext.GetService<IStateService>();
stateService.ChangeState(task, stateService.FindStateByBuiltIn(state, task)); ⑤
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение задания.

④ Завершение задания.

⑤ Изменение состояния карточки.

ITaskService.CompleteTask — метод (Task, String, StaffEmployee)

Завершает задание от имени указанного сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BuiltInState CompleteTask(Task task, string comments, StaffEmployee completedUser)
```

Параметры

task

Тип: Task

Задание

comments

Тип: System.String

Комментарий к завершению задания

completedUser

Тип: StaffEmployee

Сотрудник, завершающий задание

Возвращаемое значение

Тип: BackOffice-ObjectModel-BuiltIn:BuiltInState_CL.adoc[BuiltInState]

Состояние завершения

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или completedUser.
System.MethodAccessException	Ошибка возвращается, если у владельца сессии недостаточно прав для завершения задания.

Заметки

Данный метод завершает задание с указанием completedUser в качестве сотрудника, завершившего задание. Операция выполняется в контексте владельца сессии, поэтому владелец сессии должен иметь право на завершении данного задания. Для изменения фактического состояния карточки необходимо использовать метод IStateService.ChangeState.

ITaskService.CopyDataFromParentTask — метод (Task, TaskSetting, Task)

Копирует значения и ссылки из родительского задания в дочернее.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void CopyDataFromParentTask(Task task, TaskSetting taskSetting, Task parentTask)
```

Параметры

task

Тип: Task

Задание (подчиненное), в которое переносятся значения

taskSetting

Тип: TaskSetting

Настройки вида задания, для определения параметров копирования значений

parentTask

Тип: Task

Задание (родительское), из которого выполняется копирование значений

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task, taskSetting или parentTask.

Заметки

При создании дочернего здания методом CreateChildTask, вызывать метод CopyDataFromParentTask не требуется — выполняется автоматически.

Примеры

В приведенном ниже примере демонстрируется использование метода

`CopyDataFromParentTask` при копировании данных из основного задания в дочернее. В основном задании данные были изменены, к примеру, из объектной модели, поэтому изменения в подчиненное не попали. Настройки (*справочника видов карточек*) подчиненного вида задания подразумевают получение определённых данных из родительского.

```
①

ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Task parentTask = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000"));
Task childTask = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000001")); ③

TaskSetting childTaskKindSetting = taskService.GetKindSettings(childTask.SystemInfo
.CardKind); ④

taskService.CopyDataFromParentTask(childTask, childTaskKindSetting, parentTask); ⑤
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение основного и подчиненного заданий.

④ Получение настроек вида подчиненного задания, в которое должно быть выполнено копирование значений.

⑤ Копирование данных.

ITaskService.CopyResultsToParentTask — метод (Task, Task, TaskCopyResultsOptions)

Переносит результаты выполнения дочернего задания в родительское.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void CopyResultsToParentTask(Task task, Task parentTask, TaskCopyResultsOptions options)
```

Параметры

task

Тип: `Task`

Подчиненное задание, из которого должны быть скопированы результаты завершения

parentTask

Тип: `Task`

Задание (основное), в которое будут перенесены результаты выполнения подчиненного задания

options

Тип: `TaskCopyResultsOptions`

Параметры переноса результатов

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `task`, `parentTask` или `options`. Также ошибка возвращает, если задание `task` не завершено.
`System.MethodAccessException`	Ошибка возвращается, если недостаточно прав на перенос результатов.
`System.Exception`	Ошибка возвращается, если карточка основного задания заблокирована, либо заблокирована карточка списка ссылок.

Заметки

Параметр `options` формируется самостоятельно.

ITaskService.CopyTask — метод (Task)

Создаёт копию задания.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
Task CopyTask(Task task)
```

Параметры

task

Тип: Task

Копируемое задание

Возвращаемое значение

Тип: Task

Копия задания

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

Заметки

У копии задание устанавливает состояние "Инициализация". Комментарии исходного задания в копию не переносятся.

ITaskService.CreateChildTask — метод (Task, KindsCardKind, BaseCard, TaskList)

Создаёт дочернее задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
Task CreateChildTask(Task parentTask, KindsCardKind kind, BaseCard parentCard, TaskList
parentCardTaskList)
```

Параметры

parentTask

Тип: Task

Основное задание

kind

Тип: KindsCardKind

Вид создаваемого задания

parentCard

Тип: BaseCard

Родительская карточка

parentCardTaskList

Тип: TaskList

Список подчинённых заданий из родительской карточки

Возвращаемое значение

Тип: Task

Подчиненное задание

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр kind, parentCard или parentCardTaskList.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Task parentTask = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

Task childTask = taskService.CreateChildTask(parentTask, parentTask.SystemInfo.CardKind,
parentTask, parentTask.MainInfo.ChildTaskList); ④
```

```
childTask.MainInfo.Name = "Подчиненное задание";
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение основного задания.

④ Создание дочернего задания.

ITaskService.CreateTask — метод

Создаёт пустое задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
Task CreateTask()
```

Возвращаемое значение

Тип: Task

Задание

Заметки

Метод CreateTask создаёт задание коревого вида. Чтобы создать задание определённого вида, используйте другую перегрузку метода CreateTask.

ITaskService.CreateTask — метод (KindsCardKind)

Создаёт новое задание указанного вида.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
Task CreateTask(KindsCardKind kind)
```

Параметры

kind

Тип: KindsCardKind

Вид создаваемого задания

Возвращаемое значение

Тип: Task

Созданное задание

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр kind.

Примеры

Ниже приведён пример создания нового задания вида "На исполнение".

```
①

ITaskService taskService = objectContext.GetService<ITaskService>();
IStaffService staffService = objectContext.GetService<IStaffService>(); ②

QueryObject query = new QueryObject(RefKinds.CardKinds.Name, "На исполнение");
KindsCardKind cardKind = objectContext.FindObject<KindsCardKind>(query); ③

Task newTask = taskService.CreateTask(cardKind);
newTask.MainInfo.Name = newTask.Description = "Новое задание"; ④

taskService.InitializeDefaults(newTask); ⑤
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение вида На исполнение.

④ Создание задания.

⑤ Инициализация основных параметров задания.

ITaskService.CreateTaskFromTemplate — метод (Guid)

Создаёт задание по шаблону.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
Task CreateTaskFromTemplate(Guid templateId)
```

Параметры

templateId

Тип: System.Guid

Идентификатор задания-шаблона

Возвращаемое значение

Тип: Task

Созданное задание

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр templateId.
System.InvalidOperationException	Ошибка возвращается в случае, если шаблон не найден, либо карточка не является шаблоном или не является шаблоном задания.

ITaskService.CreateSignatureList — метод

Создаёт пустой список подписей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
SignatureList CreateSignatureList()
```

Возвращаемое значение

Тип: SignatureList

Список подписей

ITaskService.DeferTask — метод (Task)

Откладывает задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void DeferTask(Task task)
```

Параметры

task

Тип: Task

Задание

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав, чтобы отложить задание.

ITaskService.Delegate — метод (Task, IEnumerable<StaffEmployee>, StaffEmployee, Boolean, Boolean, String)

Делегирует задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool Delegate(Task task, IEnumerable<StaffEmployee> delegates, StaffEmployee noticeable,
bool restrictDelegation, bool returnAfterDelegation, string comment)
```

Параметры

task

Тип: Task

Задание

delegates

Тип: System.Collections.Generic.IEnumerable<StaffEmployee>

Список сотрудников, которым делегируется задание

noticeable

Тип: StaffEmployee

Уведомляемый сотрудник, если при делегировании невозможно однозначно разрешить, кто является исполнителем задания

restrictDelegation

Тип: System.Boolean

Устанавливает флаг "Запретить делегирование"

returnAfterDelegation

Тип: System.Boolean

Устанавливает флаг "Возврат с делегирования"

comment

Тип: System.String

Комментарий при делегировании

Возвращаемое значение

Тип: System.Boolean

true — делегирование выполнено успешно, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или delegates.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для делегирования задания.
System.InvalidOperationException	Ошибка возвращается в случае, если запрещено делегировать задание вручную.

Заметки

Сотрудник, переданный в параметре noticeable, будет указан в качестве исполнителя задания, делегировавшего задание, если невозможно однозначно определить сотрудника, выполнившего делегирование, что может произойти, к примеру, при делегировании задания сотрудником, являющимся заместителем для нескольких исполнителей задания.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

StaffEmployee delegEmployee = objectContext.GetObject<StaffEmployee>(new Guid("00000000-
0000-0000-0000-000000000001"));
StaffEmployee noticEmployee = objectContext.GetObject<StaffEmployee>(new Guid("00000000-
0000-0000-0000-000000000002"));
IEnumerable<StaffEmployee> delegates = new StaffEmployee[] { delegateEmployee }; ④

taskService.Delegate(task, delegates, noticEmployee, true, true, "Передано компетентному
сотруднику");
objectContext.AcceptChanges(); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение делегируемого задания.

④ Получение сотрудника, которому выполняется делегирование, и уведомляемого сотрудника.

⑤ Делегирование задания.

ITaskService.Delegate — метод (Task, IEnumerable<StaffEmployee>, StaffEmployee, Boolean, Boolean, String, StaffEmployee)

Делегирует задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool Delegate(Task task, IEnumerable<StaffEmployee> delegates, StaffEmployee noticeable,
bool restrictDelegation, bool returnAfterDelegation, string comment, StaffEmployee
delegatedUser)
```

Параметры

task

Тип: Task

Задание

delegates

Тип: System.Collections.Generic.IEnumerable<StaffEmployee>

Список сотрудников, которым делегируется задание

noticeable

Тип: StaffEmployee

Уведомляемый сотрудник, если при делегировании невозможно однозначно разрешить, кто является исполнителем задания

restrictDelegation

Тип: System.Boolean

Устанавливает флаг "Запретить делегирование"

returnAfterDelegation

Тип: System.Boolean

Устанавливает флаг "Возврат с делегирования"

comment

Тип: `System.String`

Комментарий при делегировании

delegatedUser

Тип: `StaffEmployee`

Сотрудник, выполнивший делегирование (для дерева исполнения; в истории отображается владелец сессии)

Возвращаемое значение

Тип: `System.Boolean`

`true` — делегирование выполнено успешно, иначе — `false`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `task`, delegates или `delegatedUser`.
`System.MethodAccessException`	Ошибка возвращается, если недостаточно прав для делегирования задания.
`System.InvalidOperationException`	Ошибка возвращается в случае, если запрещено делегировать задание вручную.

ITaskService.DoNotTakeToWorkTask — метод (Task)

Отменяет взятие задания в работу.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
BuiltInState DoNotTakeToWorkTask(Task task)
```

Параметры

task

Тип: Task

Задание для отмены взятия

Возвращаемое значение

Тип: BackOffice-ObjectModel-BuiltIn:BuiltInState_CL.adoc[BuiltInState]

Новое состояние карточки

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для отмены взятия.

Заметки

Возможность отмены взятия в работу задания должна быть предусмотрена соответствующими настройками в конструкторах.

ITaskService.GetCompletionParameters — метод (Task)

Возвращает параметры завершения задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<TaskCompletionParameter> GetCompletionParameters(Task task)
```

Параметры

task

Тип: Task

Задание

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<TaskCompletionParameter>

Коллекция параметров завершения типа TaskCompletionParameter

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

ITaskService.GetKindSettings — метод (KindsCardKind)

Возвращает настройки вида задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskSetting GetKindSettings(KindsCardKind kind)
```

Параметры

kind

Тип: KindsCardKind

Вид задания

Возвращаемое значение

Тип: TaskSetting

Настройки вида

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр kind.

ITaskService.GetLastDelegate — метод (Task)

Возвращает последнюю запись списка делегирования.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskDelegate GetLastDelegate(Task task)
```

Параметры

task

Тип: Task

Задание

Возвращаемое значение

Тип: TaskDelegate

Элемент списка делегирования

Примеры

Приведенный далее пример демонстрирует использование метода GetLastDelegate при получении сотрудника (первого из списка), которому выполнено делегирование.

```
①

ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

TaskDelegate taskDelegate = taskService.GetLastDelegate(task);
StaffEmployee employee = taskDelegate.DelegatedTo.FirstOrDefault().Employee; ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение задания.

④ Получение сотрудника, которому выполнено делегирование.

ITaskService.GetLastDelegate — метод (Task, Boolean)

Возвращает последнюю запись списка делегирования.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskDelegate GetLastDelegate(Task task, bool returned)
```

Параметры

task

Тип: Task

Задание

returned

Тип: System.Boolean

При true возвращает только возвращенные с делегирования, иначе — все делегированные

Возвращаемое значение

Тип: TaskDelegate

Элемент списка делегирования

ITaskService.GetSelectedPerformers — метод (Task)

Возвращает выбранных исполнителей задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<StaffEmployee> GetSelectedPerformers(Task task)
```

Параметры

task

Тип: Task

Задание

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<StaffEmployee>

Список исполнителей StaffEmployee

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

ITaskService.GetTaskAcceptanceMailNotificationInfo — метод (Task, String)

Возвращает письмо, подготовленное для отправки, с информацией о передаче задания на приёмку.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
MailNotificationInfo GetTaskAcceptanceMailNotificationInfo(Task task, string taskCardUrl)
```

Параметры

task

Тип: Task

Задание

taskCardUrl

Тип: Task

Ссылка на карточку для возможности открытия карточки по url

Возвращаемое значение

Тип: MailNotificationInfo

Сущность сообщения

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или taskCardUrl.
System.PlatformNotSupportedException	Ошибка возвращается в случае, если задание не находится в состоянии "На приёмке".

ITaskService.GetTaskBuiltInState — метод (Task)

Получает для задания встроенное состояние.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public BuiltInState GetTaskBuiltInState(Task task)
```

Параметры

task

Тип: Task

Задание

Возвращаемое значение

Тип: BackOffice-ObjectModel-BuiltIn:BuiltInState_CL.adoc[BuiltInState]

Встроенное состояние

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

ITaskService.GetTaskCompletionMailNotificationInfo — метод (Task, String)

Возвращает письмо, подготовленное для отправки, с информацией о завершении задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
MailNotificationInfo GetTaskCompletionMailNotificationInfo(Task task, string taskCardUrl)
```

Параметры

task

Тип: Task

Задание

taskCardUrl

Тип: Task

Ссылка на карточку для возможности открытия карточки по url

Возвращаемое значение

Тип: MailNotificationInfo

Сущность сообщения

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или taskCardUrl. Также ошибка может быть возвращена, если не определён автор, либо его почтовый адрес, или сотрудник, завершивший задание.
System.PlatformNotSupportedException	Ошибка возвращается в случае, если задание не находится в состоянии "Завершено".

ITaskService.GetTaskRejectionMailNotificationInfo — метод (Task, String)

Возвращает письмо, подготовленное для отправки, с информацией об отклонении задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
MailNotificationInfo GetTaskRejectionMailNotificationInfo(Task task, string taskCardUrl)
```

Параметры

task

Тип: Task

Задание

taskCardUrl

Тип: Task

Ссылка на карточку для возможности открытия карточки по url

Возвращаемое значение

Тип: MailNotificationInfo

Сущность сообщения

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или taskCardUrl. Также ошибка может быть возвращена, если не определён автор, либо его почтовый адрес, или сотрудник, отклонивший задание.
System.PlatformNotSupportedException	Ошибка возвращается в случае, если задание не находится в состоянии "Отклонено".

ITaskService.HasCompletionParameterValue — метод (Task, Guid, String)

Проверяет наличие значения у параметра завершения задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool HasCompletionParameterValue(Task task, Guid sectionId, string fieldAlias)
```

Параметры

task

Тип: Task

Задание

sectionId

Тип: System.Guid

Идентификатор секции

fieldAlias

Тип: System.String

Псевдоним поля

Возвращаемое значение

Тип: `System.Boolean`

Если имеет значение — `true`, иначе — `false`

ITaskService.InitializeDefaults — метод (Task)

Инициализация параметров нового задания.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void InitializeDefaults(Task task)
```

Параметры

task

Тип: `Task`

Задание

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `task`.

Заметки

При вызове метода `InitializeDefaults`, владелец сессии устанавливается в качестве автора задания, а настройки заполняются в соответствии с настройками вида карточки. Данный метод рекомендуется вызывать после создания задания методом `CreateTask`.

ITaskService.IsCurrentUserCurrentPerformer — метод (Task)

Возвращает признак того, что текущий сотрудник входит в список текущих исполнителей задания.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool IsCurrentUserCurrentPerformer(Task task)
```

Параметры

task

Тип: Task

Задание

Возвращаемое значение

Тип: System.Boolean

Возвращает true, если текущий сотрудник сходит в список текущих исполнителей задания, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

ITaskService.IsCurrentUserDeputyOfCurrentPerformer — метод (Task)

Определяет, является ли текущий сотрудник заместителем одного из текущих исполнителей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool IsCurrentUserDeputyOfCurrentPerformer(Task task)
```

Параметры

task

Тип: Task

Задание

Возвращаемое значение

Тип: `System.Boolean`

true, если текущий сотрудник является заместителем одного из текущих исполнителей, иначе — `false`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `task`.

ITaskService.IsTaskDelegated — метод (Task)

Возвращает признак того, что задание делегировано.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
bool IsTaskDelegated(Task task)
```

Параметры

task

Тип: `Task`

Задание

Возвращаемое значение

Тип: `System.Boolean`

`true` — задание делегировано, иначе — `false`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `task`.

ITaskService.IsUserCurrentPerformer — метод (Task, StaffEmployee)

Определяет, входит ли указанный сотрудник в список текущих исполнителей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool IsUserCurrentPerformer(Task task, StaffEmployee user)
```

Параметры

task

Тип: Task

Задание

user

Тип: StaffEmployee

Сотрудник

Возвращаемое значение

Тип: System.Boolean

true — сотрудник входит в число текущих исполнителей, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или user.

ITaskService.RecallTask — метод (Task)

Отзывает задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void RecallTask(Task task)
```

Параметры

task

Тип: Task

Отзываемое задание

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для отзыва задания.

Примеры

Ниже приведён пример отзыва задания. Для фактической смены состояния карточки необходимо использовать метод IStateService.ChangeState.

```
①

ITaskService taskService = objectContext.GetService<ITaskService>();
IStateService stateService = objectContext.GetService<IStateService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

taskService.RecallTask(task);
stateService.ChangeState(task, stateService.FindStateByBuiltIn(Task.RecalledState,
task)); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение задания.

④ Отзыв задания.

ITaskService.RejectTask — метод (Task, StaffEmployee, String)

Отклоняет задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void RejectTask(Task task, StaffEmployee employee, string comment)
```

Параметры

task

Тип: Task

Отклоняемое задание

employee

Тип: StaffEmployee

Сотрудник, отклонивший задание

comment

Тип: System.String

Комментарий

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или employee.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для отзыва задания.

Примеры

Ниже приведён пример отклонения задания. Для фактической смены состояния карточки необходимо использовать метод IStateService.ChangeState.

```
①
```

```
ITaskService taskService = objectContext.GetService<ITaskService>();
IStateService stateService = objectContext.GetService<IStateService>();
IStaffService staffService = objectContext.GetService<IStaffService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000"));

taskService.RejectTask(task, staffService.GetCurrentEmployee(), "Уже выполнено");
stateService.ChangeState(task, stateService.FindStateByBuiltIn(Task.RejectedState,
task)); ③
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Отклонение задания.

ITaskService.RemoveComment — метод (TaskComment)

Удаляет комментарий к заданию.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void RemoveComment(TaskComment comment)
```

Параметры

comment

Тип: TaskComment

Комментарий

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр comment.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для удаления комментария.

ITaskService.ResolveDelegatePerformer — метод (Task)

Получает список текущих исполнителей задания, за которых текущий сотрудник может делегировать задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<StaffEmployee> ResolveDelegatePerformer(Task task)
```

Параметры

task

Тип: Task

Задание

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<StaffEmployee>

Список сотрудников типа StaffEmployee

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

Заметки

Если текущий сотрудник входит в число текущих исполнителей задания, то будет возвращен только он. При получении списка определяется принадлежность сотрудника к заместителям исполнителей задания. Проверка прав на делегирование не выполняется.

ITaskService.ResolveDelegatePerformer — метод (Task, StaffEmployee)

Получает список текущих исполнителей задания, за которых указанный сотрудник может делегировать задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<StaffEmployee> ResolveDelegatePerformer(Task task, StaffEmployee
delegatedEmployee)
```

Параметры

task

Тип: Task

Задание

delegatedEmployee

Тип: StaffEmployee

Сотрудник, для которого выполняется проверка

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<StaffEmployee>

Список сотрудников типа StaffEmployee

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или delegatedEmployee.

Заметки

Если текущий сотрудник входит в число текущих исполнителей задания, то будет возвращен только он. При получении списка определяется принадлежность сотрудника к заместителям исполнителей задания. Проверка прав на делегирование не выполняется.

ITaskService.SetTaskCompletionResult — метод (Task, TaskCompletionParameter)

Устанавливает результат завершения задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void SetTaskCompletionResult(Task task, TaskCompletionParameter completionParameter)
```

Параметры

task

Тип: Task

Задание, для которого устанавливается результат завершения

completionParameter

Тип: TaskCompletionParameter

Параметры завершения задания

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task или completionParameter.

ITaskService.GetPartSignatureInfo — метод (BaseCardSignaturePart signaturePart)

Возвращает информацию о части подписи документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
SignatureInfo GetPartSignatureInfo(Task task, BaseCardSignaturePart signaturePart,
X509Certificate2 certificate);
```

Параметры

task

Тип: Task

Задание, из которого будет получена информация о подписи

signaturePart

Тип: BaseCardSignaturePart

Часть подписи

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат подписи

Возвращаемое значение

Тип: SignatureInfo

Информация о подписи

ITaskService.StartTask — метод (Task)

Отправляет задание на исполнение.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void StartTask(Task task)
```

Параметры

task

Тип: Task

Отправляемое на исполнение задание

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для отправки задания.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>();
IStateService stateService = objectContext.GetService<IStateService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

taskService.StartTask(task);
stateService.ChangeState(task, stateService.FindStateByBuiltIn(Task.StartedState, task));
④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение задания.

④ Отправка задания на исполнение.

ITaskService.StopExecutionRelatedTask — метод (Task, Boolean)

Прекращает исполнение подчиненных заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskStopExecutionInfo StopExecutionRelatedTask(Task task, bool force)
```

Параметры

task

Тип: Task

Задание

force

Тип: System.Boolean

Если true, то при наличии заблокированных заданий операция прекращения исполнения будет продолжена, иначе — прервана

Возвращаемое значение

Тип: TaskStopExecutionInfo

Содержимое ошибки, возникшей при прекращении заданий

Заметки

Заблокированные задания не завершаются.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

TaskStopExecutionInfo taskStopExecutionInfo = taskService.StopExecutionRelatedTask(task,
true); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение задания.

④ Прекращение исполнения подчиненных заданий.

ITaskService.StopExecutionTask — метод (Task)

Прекращает исполнение задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void StopExecutionTask(Task task)
```

Параметры

task

Тип: Task

Задание, которое будет прекращено

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для прекращения задания.

Заметки

Прекращение задания, в отличие от завершения, не является штатной операцией, и необходима для принудительного завершения задания. Исполнение подчиненных заданий не прекращается.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>();
IStateService stateService = objectContext.GetService<IStateService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

taskService.StopExecutionTask(task);
stateService.ChangeState(task, stateService.FindStateByBuiltIn(Task.CompletedState,
task)); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение задания.

④ Прекращение исполнения задания.

ITaskService.TakeTaskToRework — метод (Task)

Отправляет задание на доработку.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void TakeTaskToRework(Task task)
```

Параметры

task

Тип: Task

Возвращаемое задание

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для отправки задания на доработку.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>();
IStateService stateService = objectContext.GetService<IStateService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

taskService.TakeTaskToRework(task);
stateService.ChangeState(task, stateService.FindStateByBuiltIn(Task.NotAcceptedState,
task)); ④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение задания.

④ Возвращение задания на доработку.

ITaskService.TakeToWork — метод (Task)

Взятие задание в работу.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void TakeToWork(Task task)
```

Параметры

task

Тип: Task

Задание, которое берется в работу

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для взятия задания в работу.

Примеры

```
①

ITaskService taskService = objectContext.GetService<ITaskService>();
IStateService stateService = objectContext.GetService<IStateService>(); ②

Task task = objectContext.GetObject<Task>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

taskService.TakeToWork(task);
stateService.ChangeState(task, stateService.FindStateByBuiltIn(Task.InWorkState, task));
④
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение задания.

④ Взятие задания в работу.

ITaskService.UpdateComment — метод (TaskComment, String)

Обновляет содержимое комментария

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void UpdateComment(TaskComment comment, string text)
```

Параметры

comment

Тип: TaskComment

Комментария к заданию

text

Тип: System.String

Новое содержимое комментария

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр comment или text.
System.MethodAccessException	Ошибка возвращается, если недостаточно прав для изменения комментария.

ITaskService.ValidateForBegin — метод (Task, String)

Определяет готовность задания к отправке на исполнение.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool ValidateForBegin(Task task, out string errorMessage)
```

Параметры

task

Тип: Task

Задание

errorMessage

Тип: System.String

Сообщение с ошибкой

Возвращаемое значение

Тип: System.Boolean

true — ошибок не обнаружено, иначе — false

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

Заметки

Для успешного прохождения проверки, должны быть заполнены автор и название задания, а также выбран исполнитель/исполнители задания.

ITaskService.ValidateInactivePerformers — метод (Task, String)

Определяет наличие в задании активных исполнителей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool ValidateInactivePerformers(Task task, out string errorMessage)
```

Параметры

task

Тип: Task

Задание

errorMessage

Тип: System.String

Сообщение с ошибкой

Возвращаемое значение

Тип: System.Boolean

false — все исполнители задания неактивны (и не имеют активных заместителей), иначе — true

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр task.

Заметки

В ходе проверки определяется наличие активных (находятся в состоянии "Активен") исполнителей задания, если таковых не найдено (к примеру, все исполнители в отпуске), то проверяется наличие у них активных заместителей.

ITaskService.VerifySignature — метод (Task, BaseCardSignature, X509Certificate2, Boolean, IEnumerable<CardFieldSetting>)

Выполняет верификацию ЭП, установленной на полях карточки Задание, а также на Документах, связанных с заданием.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
BaseCardSignatureVerification VerifySignature(Task task, BaseCardSignature signature,
X509Certificate2 certificate, bool signDocument, IEnumerable<CardFieldSetting> fields)
```

Параметры

task

Тип: Task

Задание, содержащий подписанные данные

signature

Тип: BaseCardSignature

Проверяемая подпись

certificate

Тип: System.Security.Cryptography.X509Certificates.X509Certificate2

Сертификат ЭП сотрудника, чей сертификат использовался при подписании

signDocument

Тип: System.Boolean

Параметр не проверяется

fields

Тип: System.Collections.Generic.IEnumerable<CardFieldSetting>

Коллекция полей типа CardFieldSetting, у которых должна быть проверена подпись. Список подписываемых полей задаётся в Справочнике видов карточек

Возвращаемое значение

Тип: BaseCardSignatureVerification

Результат проверки ЭП

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `task` или `signature`.

ITspService — интерфейс

Описывает функциональность сервиса для работы со службой штампов времени по протоколу TSP.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface ITspService
```

Методы

Имя	Описание
`DateTime GetTimestampExpireDate(byte[] timestamp)`	Получает дату и время истечения срока действия штампа времени.
`GetSignatureTimestamp(Stream)`	Получает штамп времени для подписи.
`GetTimestampDateTime(Byte[])`	Получает дату и время формирования штампа времени.
`Verify(Byte[])`	Проверяет действительность штампа времени. Если штамп времени недействителен, будет создано исключение.

Заметки

Для работы с сервисом (возможно только из скрипта карточки), должны быть выполнены настройки, приведенные в /dv6/backoffice/6.1/admin/system-settings/#signature-cypher[документации по администрированию] модуля *Базовые объекты*.

IUserProfileCardService — интерфейс

Описывает сервис для работы с карточкой настроек пользователя.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public interface IUserProfileCardService
```

Методы

Имя	Описание
GetCertificate(String)	Получает сертификат пользователя с указанным именем учетной записи.
GetSetting(Int32)	Получает из профиля пользователя настройку указанного типа.
GetSetting(Int32, Object)	Получает из профиля пользователя настройку указанного типа, в случае её отсутствия, возвращает значение второго параметра.
GetSetting(Int32, Object, Guid)	Получает из профиля пользователя настройку указанного типа. Параметр objectId конкретизирует настройку, в случае присутствии нескольких однотипных настроек.
SetCertificate(String, X509Certificate2)	Задаёт сертификат указанному пользователю.
SetPersonalFolder(String, FoldersFolder)	Задаёт значение личной папки пользователя в карточке.
SetSetting(Int32, Object)	Сохраняет в профиль настройку заданного типа.
SetSetting(Int32, Guid, Object)	Сохраняет в профиль настройку с заданным идентификатором.

Примеры

приведён пример простого использования методов сервиса — запись и чтение настроек

```
①

int settingType = 999;
Guid objectID = Guid.NewGuid(); ②

iUserProfileCardService.SetSetting(settingType, objectID, "Белый"); ③

string value = (iUserProfileCardService.GetSetting(settingType, string.Empty, objectID)
as string); ④
Console.WriteLine(value);
```

① Инициализация контекста объектов.

② Задаем тип свойства и идентификатор.

③ Задаем значение свойства.

④ Получаем значение свойства, если свойство отсутствует, что в контексте примера конечно невозможно, то вернется пуста строка.

Ещё один пример использования сервиса — создание и установка личной папки пользователя

```
①

IStateService stateService = objectContext.GetService<IStateService>();
IUserProfileCardService iUserProfileCardService = objectContext.GetService
<IUserProfileCardService>(); ②

StaffEmployee staffEmployee = staffService.Get(new Guid("00000000-0000-0000-0000-
000000000000")); ③

FoldersFolder foldersFolder = objectContext.GetObject<FoldersFolder>(new Guid("FFFFFFFF-
FFFF-0003-FFFF-000000000000")); ④

FoldersFolder newfoldersFolder = new FoldersFolder(); ⑤
newfoldersFolder.Name = "OtherFolder";

foldersFolder.Folders.Add(newfoldersFolder); ⑥
objectContext.AcceptChanges();

iUserProfileCardService.SetPersonalFolder(staffEmployee.AccountName, newfoldersFolder);
⑦
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение сервисов по работе со справочником сотрудников и профилями пользователей.

③ Получение пользователя с идентификатором `00000000-0000-0000-0000-000000000000`.

④ Получение корневой папки "Папки".

⑤ Создание новой папки с названием "OtherFolder".

⑥ Добавление к корневой созданной папки и сохранение контекста.

⑦ Назначение персональной папки.

IVersionedFileCardService — интерфейс

Сервис для работы с карточкой версионного файла.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public interface IVersionedFileCardService
```

Методы

Имя	Описание
`CanCheckIn(VersionedFileCard)`	Определяет возможность снятия блокировки с версионного файла.
`CanCheckOut(VersionedFileCard)`	Определяет возможность блокировки версионного файла.
`CanCheckOut(VersionedFileCard, Boolean)`	Определяет возможность блокировки версионного файла.
`CheckFileChanged(VersionedFileCard)`	Определяет наличие изменений в файле.
`CheckIn(VersionedFileCard, Boolean, Int32, String, Boolean, String, Guid)`	Снимает блокировку с карточки файла с версиями.
`CheckOut(VersionedFileCard)`	Выгружает файл на диск с установкой постоянной блокировки.

Имя	Описание
CheckVersionInCache(Guid)	Определяет наличие в кэше сервиса определённой версии файла.
CreateCard(FileData)	Создаёт версионный файл на основе обычного файла.
CreateCard(String)	Создаёт версионный файл из файла, расположенного на диске.
Download(VersionedFileCard)	Сохраняет последнюю версию версионного файла на диск.
Download(VersionedFileCard, Guid)	Сохраняет указанную версию версионного файла на диск.
Download(VersionedFileCard, Guid, String)	Сохраняет указанную версию версионного файла на диск.
FindLockedVersionFileName(Guid)	Возвращает путь выгрузки заблокированного файла.
FindVersionFileNames(Guid)	Возвращает пути ко всем экземплярам выгруженной версии файла.
GetLockOwner(VersionedFileCard)	Возвращает имя учетной записи сотрудника, заблокировавшего версионный файл.
GetLockStatus(VersionedFileCard)	Возвращает статус блокировки для карточки файл с версиями.
GetUsedFiles	Возвращает расположение всех выгруженных файлов.
GetVersionNumber(VersionedFileCard, Guid)	Возвращает версии файла в текстовом виде.
HasVersion(VersionedFileCard, Guid)	Определяет наличие у версионного файла указанной версии.
MakeLockedInstanceForceChanged(VersionedFileCard)	Принудительно устанавливает флаг наличия изменений в файле.
OpenCard(Guid)	Возвращает карточку файла с версиями.

Имя	Описание
PurgeCache(VersionedFileCard)	Удаляет из кэша сервиса версионный файл.
PurgeLockCache	Удаляет из кэша сервиса информацию о блокировках версионных файлов.
RenameFile(VersionedFileCard, String)	Переименовывает файл *карточки файла с версиями*.
UndoCheckOut(VersionedFileCard)	Отменяет блокировку версионного файла.
UpdateFileCache(VersionedFileCard, String)	Обновляет кэш сервиса.
Upload(VersionedFileCard, String, Guid)	Загружает новую версию версионного файла.
Upload(VersionedFileCard, String, Guid, Int32)	Загружает новую версию версионного файла.

События

Имя	Описание
CheckedInFile	Событие происходит после снятия блокировки с файла.

Примеры

Ниже приведён пример простого использования сервиса — создаётся карточка файла с версиями, в которую загружается локальный файл, и поднимается номер версии

```
IVersionedFileCardService iVersionedFileCardService = objectContext.GetService
<IVersionedFileCardService>(); ①

VersionedFileCard versionedFileCard = iVersionedFileCardService.CreateCard(
@"..\test.txt"); ②

iVersionedFileCardService.CheckOut(versionedFileCard); ③

iVersionedFileCardService.MakeLockedInstanceForceChanged(versionedFileCard); ④

iVersionedFileCardService.CheckIn(versionedFileCard, false, 1); ⑤
```

① Инициализация контекста объектов и получение сервиса.

② Создание карточки файла с версиями (версия 1.0), загрузка в карточку файла из локальной файловой системы.

③ Блокировка файла.

④ Принудительно сообщаем сервису, что изменения были.

⑤ Разблокировка и увеличением номера версии до 1.1.

IVersionedFileCardService.CanCheckIn — метод (VersionedFileCard)

Определяет возможность снятия блокировки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanCheckIn(VersionedFileCard versionedFileCard)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Версионный файл

Возвращаемое значение

Тип: System.Boolean

true — блокировка может быть снята, иначе — false

IVersionedFileCardService.CanCheckOut — метод (VersionedFileCard)

Определяет возможность блокировки версионного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CanCheckOut(VersionedFileCard versionedFileCard)
```

Параметры

versionedFileCard

Тип: `VersionedFileCard`

Версионный файл

Возвращаемое значение

Тип: `System.Boolean`

`true` — файл может быть заблокирован, иначе — `false`

IVersionedFileCardService.CanCheckOut — метод (VersionedFileCard, Boolean)

Определяет возможность блокировки версионного файла.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
bool CanCheckOut(DocsVision.Platform.ObjectManager.SystemCards.VersionedFileCard
versionedFileCard, bool force)
```

Параметры

versionedFileCard

Тип: `VersionedFileCard`

Версионный файл

force

Тип: `System.Boolean`

При `true` разрешается блокировать файлы, у которых уже установлена блокировка от имени текущего пользователя

Возвращаемое значение

Тип: `System.Boolean`

`true` — файл может быть заблокирован, иначе — `false`

IVersionedFileCardService.CheckFileChanged — метод (VersionedFileCard)

Определяет наличие изменений в файле.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CheckFileChanged(DocsVision.Platform.ObjectManager.SystemCards.VersionedFileCard
versionedFileCard)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Версионный файл

Возвращаемое значение

Тип: System.Boolean

true — файл был изменён, иначе — false

Заметки

Файл считается изменённым, если присутствует разница (от 2 секунд) между фактической датой изменения выгруженного файла и датой его изменения в карточке. Также при проверке учитывается наличие флага ForceChanged (флаг принудительного оповещения о наличии изменений) у экземпляра версионного файла.

IVersionedFileCardService.CheckIn — метод (VersionedFileCard, Boolean, Int32, String, Boolean, String, Guid)

Снимает блокировку с карточки файла с версиями.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void CheckIn(VersionedFileCard versionedFileCard, bool keepCheckout, int level, string
comment, bool commentToVersion, string checkOutPath, System.Guid userId)
```

Параметры

versionedFileCard

Тип: `VersionedFileCard`

Карточка файла с версиями

keepCheckout

Тип: `System.Boolean`

Для сохранения блокировки должно быть значение `true`

level

Тип: `System.Int32`

Позиция числа, значение которого будет увеличено, в номере версии

comment

Тип: `System.String`

Комментарий к версии или к файлу

commentToVersion

Тип: `System.Boolean`

При значении true, `comment` — комментарий к версии, иначе — комментарий к файлу

checkOutPath

Тип: `System.String`

Путь к выгруженному файлу

userId

Тип: `System.Guid`

Идентификатор сотрудника, выполнившего блокировку файла

Исключения

Исключение	Условие
System.InvalidOperationException	Ошибка возникает, если файл не заблокирован, либо заблокирован, но не сотрудником с идентификатором userId, либо выгруженный файл checkOutPath не существует.

Примеры

Пример снятия блокировки с заблокированного и первого файла документа с повышением вспомогательного номера версии. Предполагается, что блокировка и изменения были произведены ранее.

```
①

IVersionedFileCardService versionedFileCardService = objectContext.GetService
<IVersionedFileCardService>();
Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ②

VersionedFileCard versionedFileCard = versionedFileCardService.OpenCard(document.Files[
0].FileId); ③

versionedFileCardService.CheckIn(versionedFileCard, false, 1); ④
```

① Инициализация контекста объектов.

② Получение файла с версией из документа.

③ Снятие блокировки с повышением версии.

IVersionedFileCardService.CheckedInFile — событие

Событие происходит после снятия блокировки с файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
event System.EventHandler<VersionedFileEventArgs> CheckedInFile
```

Заметки

Событие срабатывает при вызове метода `IVersionedFileCardService.CheckIn`, после снятие блокировки.

IVersionedFileCardService.CheckOut — метод (VersionedFileCard)

Выгружает файл на диск с установкой постоянной блокировки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
string CheckOut(VersionedFileCard versionedFileCard)
```

Параметры

versionedFileCard

Тип: `VersionedFileCard`

Выгружаемый файл с версиями

Возвращаемое значение

Тип: `System.String`

Расположение, выгруженного файла

Заметки

Будет выгружена текущая версия файла.

Примеры

```
①

IVersionedFileCardService versionedFileCardService = objectContext.GetService
<IVersionedFileCardService>(); ②

VersionedFileCard versionedFileCard = versionedFileCardService.OpenCard(new Guid
("00000000-0000-0000-0000-000000000000")); ③

string path = versionedFileCardService.CheckOut(versionedFileCard); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение существующей карточки файла с версиями.

④ Выгрузка файла.

IVersionedFileCardService.CheckVersionInCache — метод (Guid)

Определяет наличие в кэше сервиса определённой версии файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool CheckVersionInCache(Guid versionId)
```

Параметры

versionId

Тип: System.Guid

Идентификатор версии

Возвращаемое значение

Тип: System.Boolean

true — версия файла присутствует в кэше сервиса

IVersionedFileCardService.CreateCard — метод (FileData)

Создаёт версионный файл на основе обычного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
VersionedFileCard CreateCard(FileData fileData)
```

Параметры

fileData

Тип: FileData

Файл

Возвращаемое значение

Тип: VersionedFileCard

Карточка файла с версиями

IVersionedFileCardService.CreateCard — метод (String)

Создаёт карточку файла с версиями из файла, расположенного на файловой системе.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
VersionedFileCard CreateCard(string filePath)
```

Параметры

filePath

Тип: System.String

Расположение файла

Возвращаемое значение

Тип: VersionedFileCard

Карточка файла с версиями

Заметки

Метод создаёт версию 1.0.

Примеры

```
①

IVersionedFileCardService versionedFileCardService = objectContext.GetService
<IVersionedFileCardService>(); ②

VersionedFileCard versionedFileCard = versionedFileCardService.CreateCard
(@"Z:\Sample.docx"); ③
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Создания карточки файла с версиями.

IVersionedFileCardService.Download — метод (VersionedFileCard)

Сохраняет последнюю версию версионного файла на диск.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string Download(VersionedFileCard versionedFileCard)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Карточка файла с версиями

Возвращаемое значение

Тип: System.String

Расположение выгруженного файла

Заметки

Возвращает текущую версию файла. Блокировки не производится.

Примеры

```
①

IVersionedFileCardService versionedFileCardService = objectContext.GetService
<IVersionedFileCardService>(); ②

VersionedFileCard versionedFileCard = versionedFileCardService.OpenCard(new Guid
("00000000-0000-0000-0000-000000000000")); ③

string path = versionedFileCardService.Download(versionedFileCard); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение существующей карточки файла с версиями.

④ Выгрузка файла.

IVersionedFileCardService.Download — метод (VersionedFileCard, Guid)

Сохраняет указанную версию версионного файла на диск.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string Download(DocsVision.Platform.ObjectManager.SystemCards.VersionedFileCard
versionedFileCard, Guid versionId)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Карточка файла с версиями

versionId

Тип: System.Guid

Идентификатор выгружаемой версии файла

Возвращаемое значение

Тип: System.String

Расположение выгруженного файла

IVersionedFileCardService.Download — метод (VersionedFileCard, Guid, String)

Сохраняет указанную версию версионного файла на диск.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string Download(VersionedFileCard versionedFileCard, Guid versionId, string filePath)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Карточка файла с версиями

versionId

Тип: System.Guid

Идентификатор выгружаемой версии файла

filePath

Тип: System.String

Путь для выгрузки файла

Возвращаемое значение

Тип: System.String

Расположение выгруженного файла

IVersionedFileCardService.FindLockedVersionFileName — метод (Guid)

Возвращает путь выгрузки заблокированного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string FindLockedVersionFileName(Guid versionId)
```

Параметры

versionId

Тип: System.Guid

Идентификатор выгруженной версии файла

Возвращаемое значение

Тип: System.String

Путь, по которому был выгружен файл

IVersionedFileCardService.FindVersionFileNames — метод (Guid)

Возвращает пути ко всем экземплярам выгруженной версии файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<string> FindVersionFileNames(Guid versionId)
```

Параметры

versionId

Тип: System.Guid

Идентификатор выгруженной версии файла

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<String>

Коллекция путей, по которым была выгружена версия файла

IVersionedFileCardService.GetLockOwner — метод (VersionedFileCard)

Возвращает имя учетной записи сотрудника, заблокировавшего версионный файл.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string GetLockOwner(VersionedFileCard versionedFileCard)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Версионный файл

Возвращаемое значение

Тип: System.String

Учетная запись сотрудника

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр versionedFileCard.

IVersionedFileCardService.GetLockStatus — метод (VersionedFileCard)

Возвращает состояние блокировки для карточки файл с версиями.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
LockStatus GetLockStatus(VersionedFileCard versionedFileCard)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Версионная карточка

Возвращаемое значение

Тип: LockStatus

Состояние блокировки

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр versionedFileCard.

IVersionedFileCardService.GetUsedFiles — метод

Возвращает расположение всех выгруженных файлов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
IEnumerable<string> GetUsedFiles()
```

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<String>

Расположение файлов

IVersionedFileCardService.GetVersionNumber — метод (VersionedFileCard, Guid)

Возвращает версии файла в текстовом виде.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
string GetVersionNumber(VersionedFileCard versionedFileCard, Guid versionId)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Версионный файл

Возвращаемое значение

Тип: System.String

Тестовая метка версии

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр versionedFileCard или versionId.

Примеры

```
①

IVersionedFileCardService versionedFileCardService = objectContext.GetService
<IVersionedFileCardService>();

Document document = objectContext.GetObject<Document>(new Guid("94F4E425-917C-E511-9416-
90E6BA57B9F8"));
DocumentFile documentFile = document.Files.First();

VersionedFileCard fileCard =  versionedFileCardService.OpenCard(documentFile.FileId);
string version = versionedFileCardService.GetVersionNumber(fileCard, documentFile
.FileVersionId);
```

① Инициализация контекста объектов.

IVersionedFileCardService.HasVersion — метод (VersionedFileCard, Guid)

Определяет наличие у версионного файла указанной версии.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
bool HasVersion(VersionedFileCard versionedFileCard, Guid versionId)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Версионный файл

versionId

Тип: System.Guid

Версия, наличие которой проверяется

Возвращаемое значение

Тип: System.Boolean

Если true, то такая версия файла существует, иначе — false

IVersionedFileCardService.MakeLockedInstanceForceChanged — метод (VersionedFileCard)

Принудительно устанавливает флаг наличия изменений в файле.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void MakeLockedInstanceForceChanged(VersionedFileCard versionedFileCard)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Карточка версионного файла

Заметки

Флаг может быть установлен только у выгруженных файлов.

IVersionedFileCardService.OpenCard — метод (Guid)

Возвращает карточку файла с версиями.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
VersionedFileCard OpenCard(System.Guid cardId)
```

Параметры

cardId

Тип: System.Guid

Идентификатор карточки файла с версиями

Возвращаемое значение

Тип: `VersionedFileCard`

Карточка файла с версиями

Исключения

Исключение	Условие
`System.ArgumentOutOfRangeException`	Ошибка возвращается в случае, если файл с идентификатором *cardId* не существует, либо удалён.

IVersionedFileCardService.PurgeCache — метод (VersionedFileCard)

Удаляет из кэша сервиса версионный файл.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void PurgeCache(VersionedFileCard versionedFileCard)
```

Параметры

versionedFileCard

Тип: `VersionedFileCard`

Версионный файл

IVersionedFileCardService.PurgeLockCache — метод

Удаляет из кэша сервиса информацию о блокировках версионных файлов.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel.Services`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
void PurgeLockCache()
```

IVersionedFileCardService.RenameFile — метод (VersionedFileCard, String)

Переименовывает файл *карточки файла с версиями*.

- **Пространство имён: DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void RenameFile(VersionedFileCard versionedFileCard, string newFileName);
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Карточка файла с версиями

newFileName

Тип: System.String

Новое название файла. Имя файла должно содержать расширение.

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр versionedFileCard или newFileName.
System.ArgumentException	Ошибка возвращается, если newFileName содержит только символы-разделители, либо содержит недопустимые (для названия файла) символы.

Примеры

```
①

IVersionedFileCardService versionedFileCardService = objectContext.GetService
<IVersionedFileCardService>(); ②

VersionedFileCard fileCard = versionedFileCardService.OpenCard(new Guid("00000000-0000-
```

```
0000-0000-000000000000")); ③

versionedFileCardService.RenameFile(fileCard, "Новое название файла.docx"); ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение карточки файла с версиями.

④ Переименовываем файл.

Можно указывать новое название файла без расширения, а расширение получать из оригинального файла, например, так:

```
string newFileName = string.Format("Новое название файла{0}", Path.GetExtension(fileCard
.Name)); ①
```

① Формируем имя файла, получив расширение из оригинального файла.

IVersionedFileCardService.UndoCheckOut — метод (VersionedFileCard)

Отменяет блокировку версионного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void UndoCheckOut(VersionedFileCard versionedFileCard)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Карточка файла с версиями

Исключения

Исключение	Условие
System.InvalidOperationException	Ошибка возникает в случае, если файл не заблокирован

IVersionedFileCardService.UpdateFileCache — метод (VersionedFileCard, String)

Обновляет кэш сервиса.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void UpdateFileCache(VersionedFileCard versionedFileCard, string filePath)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Версионный файл

filePath

Тип: System.String

Расположение выгруженного файла

Заметки

Данный метод добавляет (или обновляет) в кэш сервиса информацию о выгруженном файле с признаком блокировки, установленной другим пользователем.

IVersionedFileCardService.Upload — метод (VersionedFileCard, String, Guid)

Загружает новую версию версионного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void Upload(VersionedFileCard versionedFileCard, string filePath, Guid userId)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Версионный файл

filePath

Тип: System.String

Расположение выгруженного файла

userId

Тип: System.Guid

Идентификатор сотрудника, сделавшего изменение

Исключения

Исключение	Условие
System.InvalidOperationException	Ошибка возвращается в случае, если указан несуществующий filePath.

IVersionedFileCardService.Upload — метод (VersionedFileCard, String, Guid, Int32)

Загружает новую версию версионного файла.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel.Services
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
void Upload(VersionedFileCard versionedFileCard, string filePath, Guid userId, int level)
```

Параметры

versionedFileCard

Тип: VersionedFileCard

Версионный файл

filePath

Тип: System.String

Расположение выгруженного файла

userId

Тип: System.Guid

Идентификатор сотрудника, сделавшего изменение

level

Тип: `System.Int32`

Уровень версии

Исключения

Исключение	Условие
`System.InvalidOperationException`	Ошибка возвращается в случае, если указан несуществующий `filePath`.

AccessInfo — класс

Результат проверки доступности операции для вида карточки и набора исполняемых ролей.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class AccessInfo
```

Конструкторы

Имя	Описание
`AccessInfo()`	Инициализирует новый экземпляр класса `AccessInfo`.

Свойства

Имя	Описание
`Operations`	Возвращает список всех доступных для карточки операций и признак разрешения запуска согласно ролевой модели.
`Roles`	Возвращает коллекцию ролей, у которых есть доступ к карточке.

Методы

Имя	Описание
`Load(XmlReader)`	Загружает параметры доступа из XML.
`Save(XmlWriter)`	Сохраняет параметры доступа в XML.

Примеры

Ниже приведён пример вывода в консоль информации по доступности операций документа с идентификатором `00000000-0000-0000-0000-000000000000`.

```
①

IServerExtensionProxyService iServerExtensionProxyService = objectContext.GetService
<IServerExtensionProxyService>(); ②

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ③

string accessInfoString = iServerExtensionProxyService.GetAccessInfo(objectContext
.GetObjectRef<Document>(document).Id,
 objectContext.GetObjectRef<KindsCardKind>(document.SystemInfo.CardKind).Id,
 objectContext.GetObjectRef<StatesState>(document.SystemInfo.State).Id); ④

AccessInfo accessInfo = new AccessInfo();
accessInfo.Load(new XmlTextReader(new StringReader(accessInfoString))); ⑤

foreach (var item in accessInfo.Operations) ⑥
{
 string operationName = objectContext.GetObject<StatesOperation>(item.OperationId
).LocalizedName;
 Console.WriteLine(string.Format("Operation {0} is {1}", operationName, item.Result)); ⑦
}
```

① Инициализация контекста объектов.

② Получение необходимых сервисов.

③ Получение документа.

④ Получение прав в формате XML, статус и вид карточки забираем из самой карточки.

⑤ Заполнение `AccessInfo` данными.

⑥ Обрабатываем только операции.

⑦ Получение локализованного названия операции, вывод в консоль вместе с признаком разрешения.

AccessInfo — конструктор

Инициализирует новый экземпляр класса `AccessInfo`.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public AccessInfo()
```

AccessInfo.Operations — свойство

Возвращает список всех доступных для карточки операций и признак разрешения запуска, согласно ролевой модели.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public IList<AccessInfoOperation> Operations { get; }
```

Значение свойства

Тип: `System.Collections.Generic.IList<AccessInfoOperation>`

Коллекция элементов типа `AccessInfoOperation` — определяет разрешено или не разрешено выполнение операции

AccessInfo.Roles — свойство

Возвращает коллекцию ролей, у которых есть доступ к карточке.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public IList<AccessInfoRole> Roles { get; }
```

Значение свойства

Тип: System.Collections.Generic.IList<AccessInfoRole>

Коллекция идентификаторов ролей в типе AccessInfoRole

AccessInfo.Load — метод (XmlReader)

Загружает параметры доступа из XML.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public void Load(XmlReader reader)
```

Параметры

reader

Тип: System.Xml.XmlReader

Данные можно получить методом IServerExtensionProxyService.GetAccessInfo

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр reader.

Заметки

Метод последовательно заполняет списки операций и ролей.

AccessInfo.Save — метод (XmlWriter)

Сохраняет параметры доступа в XML.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public void Save(XmlWriter writer)
```

Параметры

writer

Тип: System.Xml.XmlWriter

Объект для записи данных

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр writer.

Заметки

Метод сериализует списки операций и ролей в XmlWriter.

AccessInfoRole — класс

Предоставляет специфичную для ролей информацию.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class AccessInfoRole
```

Конструкторы

Имя	Описание
AccessInfoRole()	Инициализирует экземпляр класса AccessInfoRole.

Свойства

Имя	Описание
`RoleId`	Задаёт или возвращает идентификатор роли.

AccessInfoOperation — класс

Сопоставляет операции доступность или недоступность её выполнения.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class AccessInfoOperation
```

Конструкторы

Имя	Описание
`AccessInfoOperation()`	Инициализирует экземпляр класса AccessInfoOperation.

Свойства

Имя	Описание
`OperationId`	Задаёт или возвращает идентификатор операции.
`Result`	Задаёт или возвращает допустимость операции.

AccessInfoOperationResult — перечисление

Определяет режим доступа к операции.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum AccessInfoOperationResult
```

Члены

Имя члена	Описание
`None`	Не задан (не запрещен или разрешён).
`Allowed`	Операция разрешена.
`Denied`	Операция запрещена.

Заметки

`AccessInfoOperationResult` получает значение в результате загрузки информации по операциям методом `AccessInfoOperation.Load`, и может быть получен из свойства `AccessInfoOperation.Result`.

Примеры

Ниже приведён пример получения `AccessInfoOperationResult` из информации по первой операции документа

```
①

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ②

string accessInfoString = iServerExtensionProxyService.GetAccessInfo(objectContext
.GetObjectRef<Document>(document).Id,
 objectContext.GetObjectRef<KindsCardKind>(document.SystemInfo.CardKind).Id,
 objectContext.GetObjectRef<StatesState>(document.SystemInfo.State).Id); ③

AccessInfo accessInfo = new AccessInfo();
accessInfo.Load(new XmlTextReader(new StringReader(accessInfoString))); ④

AccessInfoOperationResult result = accessInfo.Operations.FirstOrDefault().Result; ⑤
```

① Инициализация контекста объектов.

② Получение документа.

③ Получение прав в формате XML, статус и вид карточки забираем из самой карточки.

④ Заполнение `AccessInfo` данными.

⑤ Получение результата по первой операции.

BaseCard — класс

Базовый класс для объектно-ориентированных карточек библиотеки *Базовые объекты*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class BaseCard : ObjectBase
```

Свойства

Имя	Описание
`Barcode`	Задаёт или возвращает штрих-код карточки.
`CardType`	Задаёт или возвращает тип карточки.
`ChangeDate`	Возвращает дату и время изменения карточки.
`CreateDate`	Возвращает дату и время создания карточки.
`Description`	Задаёт или возвращает описание карточки.
`Numbers`	Возвращает коллекцию номеров выданных карточке.
`Processes`	Возвращает коллекцию бизнес-процессов связанных с карточкой.
`SystemInfo`	Возвращает вид и состояние карточки.

Методы

Имя	Описание
`AddProperty(LayoutsProperty)`	Добавляет свойство карточки в хранилище свойств.
`GetProperties(LayoutsProperty)`	Получает свойство карточки из хранилища свойств.

Имя	Описание
GetSection(Guid)	Получает все строки указанной секции карточки.
RefreshProperty(LayoutsProperty)	Обновляет значение свойства карточки.
RemoveProperty(LayoutsProperty, BaseCardProperty)	Удаляет указанной свойство карточки из хранилища свойств.

Поля

Имя	Описание
BarcodeProperty	Определяет свойство "Штрих-код".
ChangeDateProperty	Представляет свойство "Дата изменения карточки".
CloseCardOperation	Представляет операцию закрытия карточки.
CreateCardOperation	Представляет операцию создания карточки.
CreateDateProperty	Представляет свойство "Дата создания".
DeleteOperation	Представляет операцию удаления.
DescriptionProperty	Представляет свойство "Описание".
EditOperation	Представляет операцию редактирования.
NumbersProperty	Представляет свойство "Коллекция выданных номеров".
OpenCardOperation	Представляет операцию открытия карточки.
ProcessesProperty	Представляет свойство "Коллекция бизнес-процессов".
ReadOperation	Представляет операцию чтения.
RootCardTypeId	Используется для получения типа карточки.

Имя	Описание
SaveCardOperation	Представляет операцию сохранения карточки.
StopProcessOperation	Представляет операцию остановка бизнес-процесса.
SystemInfoProperty	Представляет свойство "Системные свойства".
ViewHistoryOperation	Представляет операцию просмотра журнала.
ViewReferenceOperation	Представляет операцию просмотра списка ссылок.
ViewSignaturesOperation	Представляет операцию просмотра списка подписей

Заметки

Как правило при получении объекта используется специфический тип, например Document, а базовый класс используется при реализации объектной модели для собственного вида карточки.

Примеры

Ниже приведён пример обращения к базовому типу карточки с целью получения реального названия вида карточки

```
①

BaseCard card = objectContext.GetObject<BaseCard>(new Guid("00000000-0000-0000-0000-
000000000000")); ②

KindsCardKind kindsCardKind = card.SystemInfo.CardKind; ③

string kindName = kindsCardKind.Name; ④
```

① Инициализация контекста объектов.

② Получение карточки.

③ Получение вида карточки.

④ Получение названия вида.

BaseCard.Barcode — свойство

Задаёт или возвращает штрих-код карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public string Barcode { get; set; }
```

Значение свойства

Тип: System.String

Штрих кода

Заметки

Штрих-код настраивается в карточке *Системные настройки*, в разделе *Общие настройки*.

BaseCard.CardType — свойство

Задаёт или возвращает тип карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public Card CardType { get; set; }
```

Значение свойства

Тип: Card

Параметры типа карточки

BaseCard.ChangeDate — свойство

Возвращает дату и время изменения карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public DateTime ChangeDate { get; }
```

Значение свойства

Тип: System.DateTime

Дата и время изменения карточки

BaseCard.CreateDate — свойство

Возвращает дату и время создания карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public DateTime CreateDate { get; }
```

Значение свойства

Тип: System.DateTime

Дата и время создания карточки

BaseCard.Description — свойство

Задаёт или возвращает описание карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public string Description { get; set; }
```

Значение свойства

Тип: System.String

Описание карточки

BaseCard.Numbers — свойство

Возвращает коллекцию номеров выданных карточке.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public virtual ObjectCollection<BaseCardNumber> Numbers { get; }
```

Значение свойства

Тип: ObjectCollection<BaseCardNumber>

Коллекция номеров типа BaseCardNumber

BaseCard.Processes — свойство

Возвращает коллекцию бизнес-процессов связанных с карточкой.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public virtual ObjectCollection<BaseCardProcess> Processes { get; }
```

Значение свойства

Тип: ObjectCollection<BaseCardProcess>

Коллекция бизнес-процессов типа BaseCardProcess

BaseCard.SystemInfo — свойство

Возвращает вид и состояние карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public virtual BaseCardSystemInfo SystemInfo { get; }
```

Значение свойства

Тип: BaseCardSystemInfo

Системная информация

BaseCard.GetSection — метод (Guid)

Получает все строки указанной секции карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public IList GetSection(Guid sectionId)
```

Параметры

sectionId

Тип: System.Guid

Идентификатор секции

Возвращаемое значение

Тип: System.Collections.Generic.IList

Список строк секции

Исключения

Исключение	Условие
System.ArgumentOutOfRangeException	Ошибка возвращается в случае, если передан нулевой идентификатор секции.

Примеры

Ниже приведён пример скрипта, записывающего пустую строку в поле fieldName строчек определённой секции карточки Документ:

```
①
Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-000000000000"));
```

```
var rows = (ICollection<BaseCardSectionRow>)document.GetSection(new Guid("00000000-0000-0000-0000-000000000001")); ②

foreach (var item in rows)
{
 item["fieldName"] = string.Empty; ③
}
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение всех строчек секции с идентификатором `00000000-0000-0000-0000-000000000001`.

③ Присвоение пустого строкового значения полю `fieldName` каждой строчки секции.

BaseCardNumber — класс

Объектная модель секции "Номера" базовой карточки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class BaseCardNumber : BaseCardSectionRow
```

Свойства

Имя	Описание
`Number`	Задаёт или возвращает номер в текстовом виде.
`NumericPart`	Задаёт или возвращает идентификатор занятого номера в карточке нумератора.

Методы

Имя	Описание
`ToString`	Возвращает значение Number, если задано.

Поля

Имя	Описание
`NumberProperty`	Определяет свойство "Номер".
`NumericPartProperty`	Определяет свойство "Числовая часть".

BaseCardExternalPowerOfAttorney — класс

Раздел списка подписей для хранения информации о машиночитаемых доверенностях контрагентов

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class BaseCardExternalPowerOfAttorney : BaseCardSectionRow
```

Свойства

Имя	Описание
`Number`	Номер сторонней доверенности
`Status`	Статус сторонней доверенности: • `Unknown = 0` — Не определен • `Success = 1` — Успешный • `Warning = 2` — Предупреждение • `Error = 3` — Ошибка
`Details`	Дополнительная информация о сторонней доверенности)

BaseCardProcess — класс

Предоставляет ссылку и настройки бизнес-процесса.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class BaseCardProcess : BaseCardSectionRow
```

Свойства

Имя	Описание
ProcessId	Задаёт или возвращает идентификатор бизнес-процесса.
ProcessKind	Задаёт или возвращает настройки экземпляра бизнес-процесса.

Поля

Имя	Описание
ProcessIdProperty	Определяет свойство "Ссылка на процесс".
ProcessKindProperty	Определяет свойство "Вид процесса".

BaseCardProperty — класс

Предоставляет доступ к коллекции BaseCardPropertyField.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class BaseCardProperty
```

Свойства

Имя	Описание
Fields	Возвращает коллекцию, доступную только на чтение, описаний полей.

Имя	Описание
Item	Задаёт или возвращает элемент коллекции с указанным псевдонимом.

BaseCardProperty.Fields — свойство

Возвращает коллекцию, доступную только на чтение, описаний полей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ReadOnlyCollection<BaseCardPropertyField> Fields { get; }
```

Значение свойства

Тип: System.Collections.ObjectModel.ReadOnlyCollection<BaseCardPropertyField>

Коллекция описаний полей типа BaseCardPropertyField

BaseCardProperty.Item — свойство

Задаёт или возвращает элемент коллекции с указанным псевдонимом.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public object this[string alias] { get; set; }
```

Параметры

alias

Тип: System.String

Псевдоним поля

Значение свойства

Тип: System.Object

Значение поля

BaseCardPropertyField — класс

Представляет описание ссылочного поля.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class BaseCardPropertyField
```

Свойства

Имя	Описание
`Alias`	Возвращает псевдоним поля.
`LinkField`	Возвращает имя ссылочного поля.
`LinkParentField`	Возвращает псевдоним связанного родительского поля.
`LinkSection`	Возвращает идентификатор ссылочной секции.

BaseCardSectionRow — класс

Базовый класс для представления строки секции карточки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class BaseCardSectionRow : ObjectBase
```

Конструкторы

Имя	Описание
`BaseCardSectionRow`	Инициализирует новый экземпляр класса BaseCardSectionRow.

Свойства

Имя	Описание
`Item`	Возвращает или задаёт значение свойства с указанным названием.

Методы

Имя	Описание
`GetGuid(String)`	Возвращает идентификатор пользовательского свойства по его псевдониму.
`SetGuid(String, Guid)`	Задаёт идентификатор пользовательского свойства.

BaseCardSignature — класс

Содержит данные ЭП, установленной на карточку.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class BaseCardSignature : BaseCardSectionRow
```

Свойства

Имя	Описание
`SignatureType`	Задаёт или возвращает идентификатор типа подписи.
`Signer`	Задаёт или возвращает сотрудника непосредственно поставившего подпись.
`State`	Задаёт или возвращает подписываемый статус.
`TimeStamp`	Задаёт или возвращает время установки подписи.

Имя	Описание
`Certificate`	Задаёт или возвращает отпечаток сертификата.
`Operation`	Задаёт или возвращает подписываемую операцию.
`Imported`	Возвращает `true` в случае, если тип подписи эквивалентен значению `BaseCardSignature.ImportedSignatureType`.
`Label`	Задаёт или возвращает метку подписи (согласовано, не согласовано).
`PowerOfAttorneyID`	Идентификатор МЧД.
`SignedFromConfirmation`	Задаёт или возвращает текст подтверждения.
`SignedFrom`	Задаёт или возвращает сотрудника за которого была поставлена подпись, в случае подписания от имени другого сотрудника.
`Parts`	Возвращает коллекцию частей подписи.
`Version`	Получает или устанавливает версию подписи.
`ExternalPowerOfAttorney`	Сторонняя доверенность.
`SupportOrderedAttachments`	Отображает, поддерживает ли подпись упорядоченные приложения для простой подписи.
`SupportOrderedAttachmentsForAnySignature`	Отображает, поддерживает ли подпись упорядоченные приложения для любой подписи.

Поля

Имя	Описание
`SignatureTypeProperty`	Описывает свойство "Тип подписи".
`SignerProperty`	Описывает свойство "Подписант".

Имя	Описание
StateProperty	Описывает свойство "Состояние".
TimeStampProperty	Описывает свойство "Время нанесения".
CertificateProperty	Описывает свойство "Сертификат".
OperationProperty	Описывает свойство "Операция".
LabelProperty	Описывает свойство "Метка".
PowerOfAttorneyIDProperty	Описывает свойство "Идентификатор МЧД".
SignedFromProperty	Описывает свойство "Подписано от имени".
SignedFromConfirmationProperty	Описывает свойство "Текст подтверждения".
PartsProperty	Описывает свойство "Части подписи".
VersionProperty	Описывает версию подписи.
ExternalPowerOfAttorneyProperty	Описывает свойство "Сторонняя доверенность".
SupportOrderedAttachmentsProperty	Описывает свойство "Поддержка упорядоченных приложений для простой подписи".
SupportOrderedAttachmentsForAnySignatureProperty	Описывает свойство "Поддержка упорядоченных приложений для любой подписи".

BaseCardSignaturePart — класс

Представляет часть ЭП.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class BaseCardSignaturePart : BaseCardSectionRow
```

Свойства

Имя	Описание
`Description`	Задаёт или возвращает описание.
`File`	Задаёт или возвращает идентификатор подписанного файла.
`FileVersion`	Задаёт или возвращает идентификатор версии файла.
`Object`	Задаёт или возвращает идентификатор подписанного объекта.
`Signature`	Задаёт или возвращает идентификатор объекта в хранилище подписей.
`Type`	Задаёт или возвращает идентификатор типа части подписи.

Поля

Имя	Описание
`DescriptionProperty`	Определяет свойства "Описание".
`FileProperty`	Определяет свойства "Файл".
`FileVersionProperty`	Определяет свойства "Версия файла".
`ObjectProperty`	Определяет свойства "Идентификатор подписываемого объекта".
`SignatureProperty`	Определяет свойства "Идентификатор объекта в хранилище подписей".
`TypeProperty`	Определяет свойства "Идентификатор типа части".

BaseCardSystemInfo — класс

Содержит системную информацию карточки (секция "Системные свойства").

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class BaseCardSystemInfo : BaseCardSectionRow
```

Свойства

Имя	Описание
`CardKind`	Задаёт или возвращает вид карточки.
`State`	Задаёт или возвращает состояние карточки.

Поля

Имя	Описание
`CardKindProperty`	Определяет свойство "Вид".
`StateProperty`	Определяет свойство "Состояние".

BaseDictionaryCard — класс

Базовый класс для справочников библиотеки *Базовые объекты*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class BaseDictionaryCard : BaseCard
```

Свойства

Имя	Описание
`Processes`	Возвращает список бизнес-процессов.
`SystemInfo`	Возвращает системную информацию.

Заметки

Справочник представляет собой обычную карточку, но с некоторыми ограничениями. В частности, в классе переопределены свойства `Processes` и `SystemInfo` — возвращают null.

BackOffice — класс

Предоставляет статические методы для получения ссылок на основные справочники `BackOffice`.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public static class BackOffice
```

Свойства

Имя	Описание
`Categories`	Возвращает *Справочник категорий*.
`Kinds`	Возвращает *Справочник видов карточек*.
`Layouts`	Возвращает *Конструктор разметок*.
`Links`	Возвращает *Справочник ссылок*.
`NumerationRules`	Возвращает *Конструктор правил нумерации*.
`RoleModel`	Возвращает *Конструктор ролей*.
`Scripting`	Возвращает *Конструктор скриптов*.
`States`	Возвращает *Конструктор состояний*.

Методы

Имя	Описание
`Register(ObjectContext)`	Регистрирует стандартные (SystemCardsMapperFactory, BackOfficeMapperFactory) *фабрики преобразователей данных* в переданном контексте объектов.

Заметки

Для получения справочников должен быть проинициализирован статический

контекст объектов StaticObjectContext.

BaseUniversal — класс

Объектная модель *Конструктора справочников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class BaseUniversal : BaseDictionaryCard
```

Свойства

Имя	Описание
ItemTypes	Возвращает коллекцию узлов *Конструктора справочников*.

Поля

Имя	Описание
ItemTypesProperty	Представляет свойство "Узлы справочника".

BaseUniversal.ItemTypes — свойство

Возвращает коллекцию узлов *Конструктора справочников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<BaseUniversalItemType> ItemTypes { get; }
```

Значение свойства

Тип: ObjectCollection<BaseUniversalItemType>

Коллекция элементов типа BaseUniversalItemType

BaseUniversalItem — класс

Представляет строку в узле *Конструктора справочников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class BaseUniversalItem : ObjectBase
```

Свойства

Имя	Описание
Description	Задаёт или возвращает описание строки справочника.
ItemCard	Задаёт или возвращает *Карточку строки справочника*.
Name	Задаёт или возвращает название строки.
NotAvailable	Задаёт или возвращает признак того, что строка не используется при выборе из справочника.
Order	Задаёт или возвращает порядковый номер строки.
ParentItemType	Возвращает узел, которому принадлежит строка.

Поля

Имя	Описание
DescriptionProperty	Определяет свойство "Описание".
ItemCardProperty	Определяет свойство "Ссылка на карточку".
NameProperty	Определяет свойство "Имя".
NotAvailableProperty	Определяет свойство "Не используется".

Имя	Описание
OrderProperty	Определяет свойство "Порядковый номер".

BaseUniversalItem.ItemCard — свойство

Задаёт или возвращает *Карточку строки справочника*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public BaseUniversalItemCard ItemCard { get; set; }
```

Значение свойства

Тип: BaseUniversalItemCard

Карточка строки справочника

BaseUniversalItemCard — класс

Объектная модель *Карточки строки справочника*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class BaseUniversalItemCard : BaseCard
```

Свойства

Имя	Описание
MainInfo	Содержит данные строки *Конструктора справочников*.

Поля

Имя	Описание
MainInfoProperty	Определяет свойство "Основная информация".

BaseUniversalItemCard.MainInfo — свойство

Содержит данные строки *Конструктора справочников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public BaseUniversalItemCardMainInfo MainInfo { get; }
```

Значение свойства

Тип: BaseUniversalItemCardMainInfo

Данные строки *Конструктора справочников*

BaseUniversalItemCardMainInfo — класс

Основная информация *Карточки строки справочника*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class BaseUniversalItemCardMainInfo : BaseCardSectionRow
```

Свойства

Имя	Описание
Item	Задаёт или возвращает строку узла *Конструктора справочников*.

Поля

Имя	Описание
ItemProperty	Определяет свойство "Ссылка".

BaseUniversalItemCardMainInfo.Item — свойство

Задаёт или возвращает строку узла *Конструктора справочников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public BaseUniversalItem Item { get; set; }
```

Значение свойства

Тип: BaseUniversalItem

Строка *Конструктора справочников*

BaseUniversalItemType — класс

Представляет узел *Конструктора справочников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class BaseUniversalItemType : ObjectBase
```

Свойства

Имя	Описание
Description	Задаёт или возвращает описание к узлу.
ItemKind	Задаёт или возвращает вид карточки строки справочника.

Имя	Описание
ItemKindSpecified	Задаёт или возвращает признак того, что вид карточки строки справочника задан.
Items	Возвращает коллекцию строк из узла *Конструктора справочников*.
ItemTypes	Возвращает коллекцию подчиненных узлов справочника.
Name	Задаёт или возвращает название узла.
NotAvailable	Задаёт или возвращает признак того, что узел не используется.
ParentItemType	Возвращает родительский узел.
Security	Возвращает параметры безопасности.
UseOrder	Задаёт или возвращает признак того, что узел будет использовать порядковые номера для строк.
ViewFields	Задаёт или возвращает коллекцию отображаемых полей.

Поля

Имя	Описание
DescriptionProperty	Определяет свойство "Описание".
ItemKindProperty	Определяет свойство "Ссылка на вид".
ItemKindSpecifiedProperty	Определяет свойство "Вид записи задан".
ItemsProperty	Определяет свойство "Строка справочника".
ItemTypesProperty	Определяет свойство "Узлы справочника".
LockedProperty	Определяет свойство "Закрыт".
NameProperty	Определяет свойство "Название".

Имя	Описание
NotAvailableProperty	Определяет свойство "Не используется".
SecurityProperty	Определяет свойство "Параметры безопасности".
UseOrderProperty	Определяет свойство "Использовать порядковые номера".
UserSettingsProperty	Определяет свойство "Настройки пользователя".
ViewFieldsProperty	Определяет свойство "Отображаемые поля".

BaseUniversalItemType.Items — свойство

Возвращает коллекцию строк из узла *Конструктора справочников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public DocsVision.Platform.ObjectModel.ObjectCollection<BaseUniversalItem> Items { get; }
```

Значение свойства

Тип: ObjectCollection<BaseUniversalItem>

Коллекция элементов типа BaseUniversalItem

BaseUniversalItemTypeViewField — класс

Представляет отображаемое поле узла *Конструктора справочников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class BaseUniversalItemTypeViewField : ObjectBase
```

Свойства

Имя	Описание
FieldName	Задаёт или возвращает имя поля.
Order	Задаёт или возвращает порядковый номер поля.
SectionField	Задаёт или возвращает идентификатор поле секции.
ShortFieldName	Возвращает сокращенное имя поля.

Поля

Имя	Описание
FieldNameProperty	Определяет свойство "Имя поля".
OrderProperty	Определяет свойство "Порядковый номер".
SectionFieldProperty	Определяет свойство "Поле секции".

BuiltInBranch — класс

Представляет встроенный переход в автомате состояний.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class BuiltInBranch
```

Свойства

Имя	Описание
BranchType	Возвращает тип перехода.
EndState	Возвращает конечное состояние.
Id	Возвращает идентификатор перехода.
Operation	Возвращает встроенную операцию при которой происходит переход.

Имя	Описание
`Owner`	Возвращает идентификатор владельца переходо состояния (идентификатор типа карточки).
`StartState`	Возвращает начальное состояние перехода.

Методы

Имя	Описание
`Register(Guid, Guid, BuiltInState, BuiltInOperation, BuiltInState)`	Регистрирует встроенное состояние в объектной модели.
`BuiltInBranch Register(Guid, Guid, BuiltInState, BuiltInOperation, BuiltInState, StatesStateMachineBranchBranchType)`	Регистрирует встроенное состояние в объектной модели.
`ToString`	Возвращает специально сформированную строку, содержащую начальное и конечное состояние.

Заметки

В классе `BuiltInBranch` используется статический список для хранения зарегистрированных переходов состояний, и последующего обращения к ним. Количество зарегистрированных переходов состояний ограничено и составляет `65535`.

BuiltInOperation — класс

Предоставляет методы регистрации встроенных операций и получения их свойств.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class BuiltInOperation
```

Свойства

Имя	Описание
`CardEditModeOnly`	Определяет признак того, что операция доступна только в режиме редактирования.
`Id`	Возвращает идентификатор зарегистрированной операции.
`LocalizedName`	Возвращает локализованное название операции. Если название не задано, вернет пустую строку.
`Name`	Возвращает английское название операции, а если отсутствует, то будет возвращено первое локализованное название из списка.
`Owner`	Возвращает идентификатор владельца операции (идентификатор типа карточки).

Методы

Имя	Описание
`Register(Dictionary<Int32, String>, Guid, Guid)`	Регистрирует операцию в коллекции. CardEditModeOnly устанавливается в true.
`Register(Dictionary<Int32, String>, Guid, Guid, Boolean)`	Регистрирует операцию в коллекции и позволяет задать значение `CardEditModeOnly`.
`ToString`	Возвращает название или локализованное название операции.

Заметки

В классе `BuiltInOperation` используется статический список для хранения зарегистрированных операций, и последующего обращения к ним. Количество зарегистрированных операций ограничено и составляет `65535`.

BuiltInState — класс

Представляет встроенное состояние конструктора состояний.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class BuiltInState
```

Свойства

Имя	Описание
`First`	Возвращает признак того, что состояние является начальным для объекта.
`Id`	Возвращает идентификатор состояния.
`LocalizedName`	Возвращает локализованное название состояния.
`Name`	Возвращает название состояния.
`Owner`	Возвращает идентификатор владельца состояния (идентификатор типа карточки).

Методы

Имя	Описание
`Register(Dictionary<Int32, String>, Guid, Guid)`	Регистрирует состояние в коллекции.
`Register(Dictionary<Int32, String>, Guid, Boolean, Guid)`	Регистрирует состояние в коллекции. Позволяет обозначить состояние признаком начального состояния.
`ToString`	Возвращает название состояния, по возможности локализованное.

Заметки

В классе `BuiltInState` используется статический список для хранения зарегистрированных состояний, и последующего обращения к ним. Количество зарегистрированных состояний ограничено и составляет `65535`.

Calendar — класс

Объектная модель карточки "Бизнес-календарь".

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class Calendar : BaseCard
```

Свойства

Имя	Описание
`DefaultWorkTime`	Задаёт или возвращает коллекцию объектов представляющих стандартное рабочее время.
`MainInfo`	Возвращает основную информацию.
`SystemInfo`	Возвращает вид и статус карточки.
`Years`	Задаёт или возвращает коллекцию настроек бизнес-календаря.

Поля

Имя	Описание
`DefaultWorkTimeProperty`	Представляет свойство "Стандартное рабочее время".
`MainInfoProperty`	Представляет свойство "Основная информация".
`YearsProperty`	Представляет свойство "Года".

CalendarDay — класс

Определяет свойства рабочего дня в карточке *Бизнес-календарь*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class CalendarDay : BaseCardSectionRow
```

Свойства

Имя	Описание
DayNumber	Задаёт или возвращает номер дня из общего числа дней в году.
Type	Задаёт или возвращает тип дня.
WorkTime	Задаёт или возвращает список отрезков рабочего времени дня в карточке *Бизнес-календарь*.

Поля

Имя	Описание
DayNumberProperty	Определяет свойство "Номер дня".
TypeProperty	Определяет свойство "Тип дня".
WorkTimeProperty	Определяет свойство "Рабочее время".

CalendarDay.DayNumber — свойство

Задаёт или возвращает номер дня из общего числа дней в году.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public int DayNumber { get; set; }
```

Значение свойства

Тип: System.Int32

Порядковый номер дня в году

Заметки

Например, значение 162 определяет 162-й день из 365.

CalendarDay.Type — свойство

Задаёт или возвращает тип дня.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public CalendarDayType Type { get; set; }
```

Значение свойства

Тип: CalendarDayType

Тип дня

CalendarDay.WorkTime — свойство

Задаёт или возвращает список отрезков рабочего времени дня в карточке *Бизнес-календарь*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public DocsVision.Platform.ObjectModel.ObjectCollection<CalendarWorkTime> WorkTime { get;
set; }
```

Значение свойства

Тип: ObjectCollection<CalendarWorkTime>

Коллекция элементов типа CalendarWorkTime

CalendarDayType — перечисление

Определяет тип календарного дня в карточке *Бизнес-календарь*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum CalendarDayType
```

Члены

Имя члена	Описание
Working	День является рабочим.
NonWorking	День является нерабочим.
Holiday	День является праздничным.

CalendarDefaultWorkTime — класс

Данные стандартного рабочего времени для карточки *Бизнес-календарь*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class CalendarDefaultWorkTime : BaseCardSectionRow
```

Свойства

Имя	Описание
EndTime	Задаёт или возвращает конечное рабочее время.
StartTime	Задаёт или возвращает начальное рабочее время.

Поля

Имя	Описание
EndTimeProperty	Определяет свойство "Конечное время".

Имя	Описание
StartTimeProperty	Определяет свойство "Начальное время".

CalendarMainInfo — класс

Основная информация карточки *Бизнес-календарь*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class CalendarMainInfo : BaseCardSectionRow
```

Свойства

Имя	Описание
Name	Задаёт или возвращает название календаря.

Поля

Имя	Описание
NameProperty	Определяет свойство "Название".

CalendarYear — класс

Календарный год карточки *Бизнес-календарь*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class CalendarYear : BaseCardSectionRow
```

Свойства

Имя	Описание
Days	Задаёт или возвращает список дней года в карточке *Бизнес-календарь*.
Year	Задаёт или возвращает год в карточке *Бизнес-календарь*.

Поля

Имя	Описание
DaysProperty	Определяет свойство "Дни".
YearProperty	Определяет свойство "Год".

CalendarYear.Days — свойство

Задаёт или возвращает список дней года в карточке *Бизнес-календарь*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public DocsVision.Platform.ObjectModel.ObjectCollection<CalendarDay> Days { get; set; }
```

Значение свойства

Тип: ObjectCollection<CalendarDay>

Коллекция элементов типа CalendarDay

CalendarYear.Year — свойство

Задаёт или возвращает год в карточке *Бизнес-календарь*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public int Year { get; set; }
```

Значение свойства

Тип: System.Int32

Год

Заметки

Год обозначает обычным числом, например: 2015.

CalendarWorkTime — класс

Определяет отрезок рабочего времени в карточке *Бизнес-календарь*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class CalendarWorkTime : BaseCardSectionRow
```

Свойства

Имя	Описание
EndTime	Задаёт или возвращает конечное время в отрезке рабочего времени.
StartTime	Задаёт или возвращает начальное время в отрезке рабочего времени.

Поля

Имя	Описание
EndTimeProperty	Определяет свойство "Конечное время".
StartTimeProperty	Определяет свойство "Начальное время".

CalendarWorkTime.EndTime — свойство

Задаёт или возвращает конечное время в отрезке рабочего времени.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public DateTime EndTime { get; set; }
```

Значение свойства

Тип: System.DateTime

Конечное время

CalendarWorkTime.StartTime — свойство

Задаёт или возвращает начальное время в отрезке рабочего времени.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public DateTime StartTime { get; set; }
```

Значение свойства

Тип: System.DateTime

Начальное время

Categories — класс

Объектная модель *Справочника категорий*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class Categories : BaseDictionaryCard
```

Свойства

Имя	Описание
CategoryCollection	Возвращает список категорий *Справочника категорий*.
Main	Предоставляет идентификатор корневой папки в *Справочнике категорий*.

Поля

Имя	Описание
CategoriesProperty	Определяет свойство "Категории".
MainProperty	Определяет свойство "Основная".

Categories.CategoryCollection — свойство

Возвращает список категорий *Справочника категорий*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public DocsVision.Platform.ObjectModel.ObjectCollection<CategoriesCategory>
CategoryCollection { get; }
```

Значение свойства

Тип: ObjectCollection<CategoriesCategory>

Коллекция элементов типа CategoriesCategory

Categories.Main — свойство

Предоставляет идентификатор корневой папки в *Справочнике категорий*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public CategoriesMain Main { get; }
```

Значение свойства

Тип: CategoriesMain

Основная информация *Справочника категорий*

CategoriesCategory — класс

Представляет категорию *Справочника категорий*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class CategoriesCategory : ObjectBase
```

Свойства

Имя	Описание
Categories	Задаёт или возвращает коллекцию подкатегорий.
Comment	Задаёт или возвращает комментарий к категории.
Folder	Задаёт или возвращает параметры папки, отражающей категорию в структуре виртуальных папок.
LinkedCategories	Задаёт или возвращает список связанных категорий.
Name	Задаёт или возвращает название категории.
NotAvailable	Задаёт или возвращает признак того, что категория не используется.
Parent	Возвращает родительскую категорию.
Parent	Возвращает родительскую категорию.

Имя	Описание
Path	Возвращает путь к текущей категории, начиная с корневой.
SearchChildren	Определяет возможность поиска в дочерних категориях.
Security	Возвращает настройки безопасности.
View	Задаёт или возвращает представление, которое будет применено к виртуальной папке категории.

Поля

Имя	Описание
CategoriesProperty	Определяет свойство "Категории".
CommentProperty	Определяет свойство "Комментарий".
FolderProperty	Определяет свойство "Папка категории".
LinkedCategoriesProperty	Определяет свойство "Связанные категории".
NameProperty	Определяет свойство "Название категории".
NotAvailableProperty	Определяет свойство "Не используется".
SearchChildrenProperty	Определяет свойство "Искать в дочерних категориях".
SearchProperty	Определяет свойство "Поиск".
SecurityProperty	Определяет свойство "Безопасность".
SyncTagProperty	Определяет свойство "Поле синхронизации".
ViewProperty	Определяет свойство "Представление".

CategoriesLinkedCategory — класс

Представляет связанную категорию *Справочника категорий*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class CategoriesLinkedCategory : ObjectBase
```

Свойства

Имя	Описание
Category	Задаёт или возвращает связанную категорию.

Поля

Имя	Описание
CategoryIDProperty	Определяет свойство "Категория".

CategoriesLinkedCategory.Category — свойство

Задаёт или возвращает связанную категорию.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public CategoriesCategory Category { get; set; }
```

Значение свойства

Тип: CategoriesCategory

Связанная категория

CategoriesMain — класс

Основная информация *Справочника категорий*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class CategoriesMain : ObjectBase
```

Свойства

Имя	Описание
RootFolderRef	Задаёт или возвращает идентификатор корневой папки.

Поля

Имя	Описание
RootFolderRefProperty	Определяет свойство "ID корневой папки".

CategoryList — класс

Объектная модель карточки *Список категорий*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class CategoryList : BaseCard
```

Свойства

Имя	Описание
Categories	Задаёт или возвращает список категорий.

Поля

Имя	Описание
CategoriesProperty	Определяет свойство "Категории".

CategoryListCategory — класс

Представляет категорию в *Списке категорий*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class CategoryListCategory : BaseCardSectionRow
```

Свойства

Имя	Описание
`Category`	Задаёт или возвращает категорию.

Поля

Имя	Описание
`CategoryProperty`	Определяет свойство "Категория".

DeputyAccessRights — перечисление

Определяет режим доступа для заместителя в приложении *Делопроизводство*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum DeputyAccessRights
```

Члены

Имя члена	Описание
`None`	Не задан.
`All`	Полный доступ.
`Custom`	Особый.

Заметки

Имеет ключевое значение для карточек из решения *Делопроизводство*. В базовых объектах всегда установлен в `DeputyAccessRights.Custom`.

Document — класс

Объектная модель карточки Документ.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class Document : BaseCard
```

Свойства

Имя	Описание
`Files`	Возвращает коллекцию файлов карточки документа.
`MainInfo`	Предоставляет доступ к основной информации документа.

Поля

Имя	Описание
`CancelLockFileOperation`	Операция отмены блокировки файла.
`ChangeBarcodeOperation`	Операция изменения штрих-кода карточки.
`DeleteCategoryOperation`	Операция удаления категории.
`EditFileCommentOperation`	Операция редактирования комментария к файлу.
`ExportDocumentOperation`	Операция экспорта документа.
`ExtraFileAddOperation`	Операция добавления дополнительного файла.
`ExtraFileReadOperation`	Операция чтения дополнительного файла.
`ExtraFileRemoveOperation`	Операция удаления дополнительного файла.
`FilesProperty`	Представляет свойство "Файлы".

Имя	Описание
MainFileAddOperation	Операция добавления основного файла.
MainFileEditOperation	Операция редактирования основного файла.
MainFileLockOperation	Операция длительной блокировки основного файла.
MainFileReadOperation	Операция чтения основного файла.
MainFileRemoveOperation	Операция удаления основного файла.
MainFileVersionAddCommentOperation	Операция добавления комментария к версии основного файла.
MainFileVersionAddOperation	Операция создания новой версии основного файла.
MainFileVersionEditOperation	Операция сохранения изменений в текущую версию основного файла.
MainFileVersionMakeCurrentOperation	Операция смены текущей версии основного файла.
MainFileVersionRemoveOldOperation	Операция удаления старой версии основного файла.
MainInfoProperty	Представляет свойство "Основная информация".
PrintBarcodeOperation	Операция печати штрих-кода.
PrintDocumentOperation	Операция печати документа.
RemoveFileCommentOperation	Операция удаления комментария к файлу.
SendDocumentOperation	Операция отправки документа по электронной почте.
SetCategoryOperation	Операция назначения категории.
SignDocumentOperation	Операция подписания документа.
SynchronizationFromFileOperation	Операция синхронизации свойств из файла.

Имя	Описание
SynchronizationToFileOperation	Операция синхронизации свойств в файл.
UnlockFileOperation	Операция снятия блокировки файла.

Примеры

Ниже приведён пример создания документа и выделение ему регистрационного номера

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>();
IBaseCardService baseCardService = objectContext.GetService<IBaseCardService>();
INumerationRulesService iNumerationRulesService = objectContext.GetService
<INumerationRulesService>(); ②

KindsCardKind kindsCardKind = objectContext.GetObject<KindsCardKind>(new Guid("00000000-
0000-0000-0000-000000000000")); ③

Document document = documentService.CreateDocument(null, kindsCardKind);
document.MainInfo.Name = "Счет на оплату";
objectContext.SaveObject<Document>(document); ④

CardData cardData = userSession.CardManager.GetCardData(objectContext.GetObjectRef
(document).Id);
document.Description = baseCardService.GenerateDigest(document, cardData, document
.MainInfo.Name); ⑤

NumerationRulesRule numerationRulesRule = objectContext.GetObject<NumerationRulesRule
>(new Guid("00000000-0000-0000-0000-000000000001")); ⑥

BaseCardNumber number = iNumerationRulesService.CreateNumber(cardData, document,
numerationRulesRule);
document.MainInfo.SetGuid("RegNumber", objectContext.GetObjectRef(number).Id);
document.MainInfo["RegDate"] = DateTime.Now; ⑦

objectContext.SaveObject<Document>(document); ⑧
```

① Инициализация контекста объектов.

② Получение сервисов.

③ Получение вида документа.

④ Создание документа.

⑤ Генерация дайджеста.

⑥ Получение правила нумерации.

⑦ Выделение номера для документа и установка даты регистрации документа.

⑧ Сохранение изменений.

Document.Files — свойство

Возвращает коллекцию файлов карточки документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<DocumentFile> Files { get; }
```

Значение свойства

Тип: ObjectCollection<DocumentFile>

Коллекция файлов типа DocumentFile

Заметки

Данное свойство предоставляет доступ к свойствам текущей версии файла.

Document.MainInfo — свойство

Предоставляет доступ к основной информации документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public DocumentMainInfo MainInfo { get; }
```

Значение свойства

Тип: DocumentMainInfo

Объектная модель секции MainInfo

DocumentFile — класс

Объектная модель секции "Файлы" карточки "Документ".

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class DocumentFile : BaseCardSectionRow
```

Свойства

Имя	Описание
FileChangeDate	Возвращает дату изменения карточки.
FileCheckoutDate	Возвращает дату установки блокировки на файл.
FileCheckoutUser	Возвращает идентификатор пользователя, забравшего файл в единоличное владение.
FileId	Задаёт или возвращает идентификатор карточки файла с версиями.
FileName	Возвращает название файла.
FileSize	Возвращает размер файла.
FileType	Задаёт или возвращает тип файла документа.
FileVersionId	Возвращает идентификатор текущей версии файла.
FileVersionRowId	Возвращает идентификатор файла в версии.

Поля

Имя	Описание
FileIdProperty	Определяет свойство "Файл".
FileTypeProperty	Определяет свойство "Тип файла".

DocumentFile.FileType — свойство

Задаёт или возвращает тип файла документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public DocumentFileType FileType { get; set; }
```

Значение свойства

Тип: DocumentFileType

Тип документа

DocumentFile.FileId — свойство

Задаёт или возвращает идентификатор карточки файла с версиями.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public Guid FileId { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор карточки файла с версиями, в котором находится файл документа

DocumentFile.FileVersionId — свойство

Возвращает идентификатор текущей версии файла документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public Guid FileVersionId { get; }
```

Значение свойства

Тип: System.Guid

Идентификатор версии файла в карточки файла с версиями

DocumentFile.FileVersionRowId — свойство

Возвращает идентификатор файла в версии.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public Guid FileVersionRowId { get; }
```

Значение свойства

Тип: System.Guid

Идентификатор файла

Заметки

Под файлом понимается сущность "Файл" платформы Docsvision, с которой можно работать с помощью FileManager.

DocumentFileType — перечисление

Определяет тип файла документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum DocumentFileType
```

Члены

Имя члена	Описание
Main	Основной
Additional	Дополнительный

DocumentMainInfo — класс

Объектная модель секции "Основная информация" карточки "Документ".

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class DocumentMainInfo : BaseCardSectionRow
```

Свойства

Имя	Описание
Author	Задаёт или возвращает автора документа.
CategoryList	Задаёт или возвращает список категорий.
ClerkId	Задаёт или возвращает идентификатор делопроизводителя.
DeliveryDate	Задаёт или возвращает дату документа.
Name	Задаёт или возвращает название документа.
ReferenceList	Задаёт или возвращает список ссылок документа.
Registrar	Задаёт или возвращает регистратора документа.
SignatureList	Задаёт или возвращает список подписей.
Surveys	Задаёт или возвращает список опросов.

Имя	Описание
Tasks	Задаёт или возвращает список заданий, прикрепленных к документу.
VersioningType	Задаёт или возвращает тип версионирования файлов документа.

Поля

Имя	Описание
AuthorProperty	Представляет свойство "Автор".
CategoryListProperty	Представляет свойство "Категории".
ClerkIdProperty	Представляет свойство "Идентификатор делопроизводителя".
DeliveryDateProperty	Представляет свойство "Дата от".
NameProperty	Представляет свойство "Название".
ReferenceListProperty	Представляет свойство "Ссылки".
RegistrarProperty	Представляет свойство "Регистратор".
SignatureListProperty	Представляет свойство "Список подписей".
SurveysProperty	Представляет свойство "Опрос".
TaskListProperty	Представляет свойство "Задания".
VersioningTypeProperty	Представляет свойство "Тип версий".

DocumentMainInfo.Author — свойство

Задаёт или возвращает автора документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StaffEmployee Author { get; set; }
```

Значение свойства

Тип: `StaffEmployee`

Сотрудник организации

DocumentMainInfo.CategoryList — свойство

Задаёт или возвращает список категорий.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public CategoryList CategoryList { get; set; }
```

Значение свойства

Тип: `CategoryList`

Список категорий

DocumentMainInfo.Tasks — свойство

Задаёт или возвращает список заданий, прикрепленных к документу.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public TaskList Tasks { get; set; }
```

Значение свойства

Тип: `TaskList`

Список заданий

Заметки

Для управления списками заданий используется сервис `ITaskListService`.

DocumentMainInfo.ReferenceList — свойство

Задаёт или возвращает список ссылок документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ReferenceList ReferenceList { get; set; }
```

Значение свойства

Тип: ReferenceList

Список ссылок

DocumentMainInfo.SignatureList — свойство

Задаёт или возвращает список подписей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public SignatureList SignatureList { get; set; }
```

Значение свойства

Тип: SignatureList

Список подписей

DocumentMainInfo.VersioningType — свойство

Задаёт или возвращает тип версионирования файлов документа.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public DocumentVersioningType VersioningType { get; set; }
```

Значение свойства

Тип: DocumentVersioningType

Тип версионинга

DocumentVersioningType — перечисление

Определяет тип версий файла для карточки "Документ".

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum DocumentVersioningType
```

Члены

Имя члена	Описание
None	Нет.
Auto	Авто.
Manual	Ручной.
Select	Необходимо выбрать иное значение.

EmployeeCard — класс

Класс EmployeeCard представляет объектную модель карточки сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class EmployeeCard : BaseCard
```

Свойства

Имя	Описание
MainInfo	Возвращает основную информацию *Карточки сотрудника*.

Поля

Имя	Описание
LockOperation	Операция блокировки карточки.
MainInfoProperty	Определяет свойство "Основная информация".

EmployeeCard.MainInfo — свойство

Возвращает основную информацию *Карточки сотрудника*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public EmployeeCardMainInfo MainInfo { get; }
```

Значение свойства

Тип: EmployeeCardMainInfo

Секция "Основная информация"

EmployeeCardMainInfo — класс

Класс EmployeeCardMainInfo представляет объектную модель секции "Основная информация" *Карточки сотрудника*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class EmployeeCardMainInfo : BaseCardSectionRow
```

Kinds — класс

Класс Kinds представляет объектную модель карточки *Справочник видов карточек*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class Kinds : BaseDictionaryCard
```

Свойства

Имя	Описание
CardTypes	Возвращает типы карточек, зарегистрированные в *Справочнике видов карточек*.
Extensions	Возвращает коллекцию расширений *Справочника видов карточек*.

Поля

Имя	Описание
CardTypesProperty	Определяет свойство "Типы карточек".
ExtensionsProperty	Определяет свойство "Расширения".

Kinds.CardTypes — свойство

Возвращает типы карточек, зарегистрированные в *Справочнике видов карточек*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<KindsCardType> CardTypes { get; }
```

Значение свойства

Тип: ObjectCollection<KindsCardType>

Коллекция типов карточек `KindsCardType` из *Справочнике видов карточек*

Kinds.Extensions — свойство

Возвращает коллекцию расширений *Справочника видов карточек*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public ObjectCollection<KindsCardExtension> Extensions { get; }
```

Значение свойства

Тип: `ObjectCollection<KindsCardExtension>`

Коллекция элементов типа `KindsCardExtension`

KindsCardCreationSetting — класс

Представляет в *Справочнике видов карточек* настройки режима создания вида карточки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class KindsCardCreationSetting : BaseCardSectionRow
```

Свойства

Имя	Описание
`CreatorKind`	Задаёт или возвращает идентификатор вида карточки, из которого может быть создана данная карточка.
`Link`	Задаёт или возвращает тип ссылки, связывающей подчиненную карточку с родительской.

Имя	Описание
Location	Задаёт или возвращает расположения по умолчанию для создания карточки.
Mode	Задаёт или возвращает идентификатор режима создания карточки.
ModeName	Задаёт или возвращает название режима создания карточки.
ShowCommand	Задаёт или возвращает признак отображения на ленте карточки кнопки "Создание карточки".
ShowDialog	Задаёт или возвращает признак открытия окна для выбора папки, в которой будут создаваться карточки.
Template	Задаёт или возвращает идентификатор шаблона карточки для создаваемой карточки текущего вида.
UseCreatorKind	Задаёт или возвращает признак того, то в настройках создания карточки задан родительский вид карточки.
WithAllChildren	Задаёт или возвращает признак того, что в качестве родительского вида карточки может быть использован любой вид карточки.

Поля

Имя	Описание
CreatorKindProperty	Представляет свойство "Идентификатор вида, из которого может быть создана карточка".
FolderProperty	Представляет свойство "Папка создаваемой карточки".
LinkProperty	Представляет свойство "Ссылка".

Имя	Описание
LocationProperty	Представляет свойство "Размещение карточки".
ModeNameProperty	Представляет свойство "Название способа создания".
ModeProperty	Представляет свойство "Идентификатор способа создания".
OperationProperty	Представляет свойство "Операция".
ShowCommandProperty	Представляет свойство "Отображать команду".
ShowDialogProperty	Представляет свойство "Флаг, указывающий на отображение диалога".
TemplateProperty	Представляет свойство "Идентификатор шаблона карточки данного вида".
UseCreatorKindProperty	Представляет свойство "Флаг, определяющий использование настроек вида".
WithAllChildrenProperty	Представляет свойство "Со всеми подчинёнными".

KindsCardCreationSetting.CreatorKind — свойство

Задаёт или возвращает идентификатор вида карточки, из которого может быть создана данная карточка.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public KindsCardKind CreatorKind { get; set; }
```

Значение свойства

Тип: KindsCardKind

Вид карточки

KindsCardCreationSetting.Link — свойство

Задаёт или возвращает тип ссылки, связывающей подчиненную карточку с родительской.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public LinksLinkType Link { get; set; }
```

Значение свойства

Тип: LinksLinkType

Тип ссылки

KindsCardCreationSetting.Location — свойство

Задаёт или возвращает расположения по умолчанию для создания карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public KindsCardCreationSettingLocation Location { get; set; }
```

Значение свойства

Тип: KindsCardCreationSettingLocation

KindsCardCreationSetting.Mode — свойство

Задаёт или возвращает идентификатор режима создания карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public Guid Mode { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор режима

KindsCardCreationSetting.ModeName — свойство

Задаёт или возвращает название режима создания карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public string ModeName { get; set; }
```

Значение свойства

Тип: System.String

Название режима создания

KindsCardCreationSettingLocation — перечисление

Определяет способ размещения карточки определённого вида.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum KindsCardCreationSettingLocation
```

Члены

Имя члена	Описание
None	Значение не задано.
CurrentFolder	Текущая папка.

Имя члена	Описание
`InitialCardFolder`	Папка дочерней карточки.
`PersonalFolder`	Личная папка.
`Custom`	Произвольная папка.

KindsCardExtension — класс

Объектная модель расширения *Справочника видов карточек*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class KindsCardExtension : ObjectBase
```

Свойства

Имя	Описание
`Enabled`	Задаёт или возвращает признак того, что расширения включено.
`Name`	Задаёт или возвращает название расширения.
`TypeName`	Задаёт или возвращает полное имя класса, реализующего расширение.

Поля

Имя	Описание
`EnabledProperty`	Определяет свойство "Включено".
`NameProperty`	Определяет свойство "Название".
`TypeNameProperty`	Определяет свойство "Полное имя типа".

KindsCardExtendedSetting — класс

Параметры расширения *Справочника видов карточек*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class KindsCardExtendedSetting : ObjectBase
```

Свойства

Имя	Описание
Extension	Задаёт или возвращает параметры расширения.
Groups	Возвращает коллекцию групп настроек расширения *Справочника видов карточек*.

Поля

Имя	Описание
ExtensionProperty	Определяет свойство "Расширение".
GroupsProperty	Определяет свойство "Группы настроек".

KindsCardExtendedSetting.Extension — свойство

Задаёт или возвращает параметры расширения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public KindsCardExtension Extension { get; set; }
```

Значение свойства

Тип: KindsCardExtension

Параметры расширения

KindsCardExtendedSetting.Groups — свойство

Возвращает коллекцию групп настроек расширения *Справочника видов карточек*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public KindsCardExtendedSettingGroupCollection Groups { get; }
```

Значение свойства

Тип: KindsCardExtendedSettingGroupCollection

Коллекция групп настроек

KindsCardExtendedSettingCollection — класс

Представляет коллекцию настроек расширений видов карточек. Предоставляет метод получения настроек для расширения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class KindsCardExtendedSettingCollection : ObjectCollection
<KindsCardExtendedSetting>
```

Конструкторы

Имя	Описание
KindsCardExtendedSettingCollection()	Инициализирует экземпляр класса KindsCardExtendedSettingCollection.
KindsCardExtendedSettingCollection(ObjectCollectionInitializationData)	Инициализирует экземпляр класса KindsCardExtendedSettingCollection с помощью указанного значения.

Методы

Имя	Описание
`GetExtendedSetting(KindsCardExtension)`	Возвращает настройки для указанного расширения вида карточки.

KindsCardExtendedSettingGroup — класс

Группа настроек расширения *Справочника видов карточек*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class KindsCardExtendedSettingGroup : ObjectBase
```

Свойства

Имя	Описание
`Groups`	Возвращает коллекцию вложенных групп настроек.
`Name`	Задаёт или возвращает название группы настроек.
`ParentGroup`	Возвращает родительскую группу настроек.
`Settings`	Возвращает коллекцию настроек расширения.

Поля

Имя	Описание
`GroupsProperty`	Определяет свойство "Группы настроек".
`NameProperty`	Определяет свойство "Название".
`SettingsProperty`	Определяет свойство "Настройки".

KindsCardExtendedSettingGroupCollection — класс

Предоставляет коллекцию сгруппированных настроек расширения *Справочника*

видов карточек.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class KindsCardExtendedSettingGroupCollection : ObjectCollection
<KindsCardExtendedSettingGroup>
```

Конструкторы

Имя	Описание
KindsCardExtendedSettingGroupCollection()	Инициализирует новый экземпляр класса KindsCardExtendedSettingGroupCollection.
KindsCardExtendedSettingGroupCollection(ObjectCollectionInitializationData)	Инициализирует новый экземпляр класса KindsCardExtendedSettingGroupCollection с передачей метода получения данных.

Методы

Имя	Описание
FindGroup(String)	Возвращает первую группу настроек с указанным именем.
GetGroup(String)	Возвращает первую группу настроек с указанным именем. Если группа настроек отсутствует, то она будет создана.

KindsCardExtendedSettingGroupCollection.FindGroup — метод (String)

Возвращает первую группу настроек с указанным именем.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public KindsCardExtendedSettingGroup FindGroup(string name)
```

Параметры

name

Тип: System.String

Точное и полное название группы настроек

Возвращаемое значение

Тип: KindsCardExtendedSettingGroup

Группа настроек расширения

KindsCardExtendedSettingGroupCollection.GetGroup — метод (String)

Возвращает первую группу настроек с указанным именем. Если группа настроек отсутствует, то она будет создана.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public KindsCardExtendedSettingGroup FindGroup(string name)
```

Параметры

name

Тип: System.String

Точное и полное название группы настроек

Возвращаемое значение

Тип: KindsCardExtendedSettingGroup

Группа настроек расширения

KindsCardExtendedSettingGroupSetting — класс

Настройки из группы настроек расширения *Справочника видов карточек*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class KindsCardExtendedSettingGroupSetting : ObjectBase
```

Свойства

Имя	Описание
CardReference	Задаёт или возвращает ссылка на карточку.
ClearWhenCopy	Задаёт или возвращает признак необходимости очистки настроек при копировании вида.
DataType	Задаёт или возвращает тип свойства.
FileReference	Задаёт или возвращает ссылку на файл.
Image	Задаёт или возвращает изображение.
Name	Задаёт или возвращает название свойства.
Unitext	Задаёт или возвращает текст.
Value	Задаёт или возвращает значение свойства.

Поля

Имя	Описание
CardReferenceProperty	Определяет свойство "Ссылка на карточку".
ClearWhenCopyProperty	Определяет свойство "При копировании настроек вида очищать значение настройки".
DataTypeProperty	Определяет свойство "Тип свойства".

Имя	Описание
FileReferenceProperty	Определяет свойство "Ссылка на файл".
ImageProperty	Определяет свойство "Изображение".
NameProperty	Определяет свойство "Название".
UnitextProperty	Определяет свойство "Текст".
ValueProperty	Определяет свойство "Значение свойства".

KindsCardExtendedSettingGroupSettingCollection — класс

Коллекция настроек для группы расширения *Справочника видов карточек*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class KindsCardExtendedSettingGroupSettingCollection : ObjectCollection
<KindsCardExtendedSettingGroupSetting>
```

Конструкторы

Имя	Описание
KindsCardExtendedSettingGroupSettingCollection()	Инициализирует новый экземпляр класса KindsCardExtendedSettingGroupSettingCollection.
KindsCardExtendedSettingGroupSettingCollection(ObjectCollectionInitializationData)	Инициализирует новый экземпляр класса KindsCardExtendedSettingGroupSettingCollection с передачей метода получения данных.

Методы

Имя	Описание
GetSetting(String)	Возвращает настройки с указанным именем. Если настройки отсутствуют, то они будут созданы.
SetSetting(String, Object)	Устанавливает значение указанной настройки расширения.

KindsCardExtendedSettingGroupSettingCollection.GetSetting — метод (String)

Возвращает настройки с указанным именем. Если настройки отсутствуют, то они будут созданы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public object GetSetting(string name)
```

Параметры

name

Тип: System.String

Полное название свойства

Возвращаемое значение

Тип: System.Object

Значение свойства

KindsCardExtendedSettingGroupSettingCollection.SetSetting — метод (String, Object)

Устанавливает значение указанной настройки расширения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public KindsCardExtendedSettingGroupSetting SetSetting(string name, object value)
```

Параметры

name

Тип: System.String

Название свойства

value

Тип: System.Object

Значение свойства

Возвращаемое значение

Тип: KindsCardExtendedSettingGroupSetting

Измененное или созданное свойство

Заметки

Если свойство с указанным названием отсутствует, то оно будет создано.

KindsCardKind — класс

Представляет вид карточки объектной модели уровня бизнес-логики.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class KindsCardKind : ObjectBase
```

Свойства

Имя	Описание
CreationSettings	Возвращает настройки способа создания вида карточки.
Digest	Задаёт или возвращает XSLT инструкцию формирования дайджеста.
ExtendedSettings	Возвращает коллекцию настроек расширений вида карточки.

Имя	Описание
Kinds	Возвращает коллекцию подвидов данного вида карточки.
KindsCardType	Возвращает тип карточки, которому принадлежит данный вид.
Name	Задаёт или возвращает название вида карточки.
NotAvailable	Задаёт или возвращает признак недоступности выбора вида карточки из соответствующих форм.
ParentCardKind	Возвращает родительский вид карточки.
Processes	Возвращает коллекцию бизнес-процессов из настроек вида карточки.
Script	Задаёт или возвращает скрипт привязанный к виду карточки.
Security	Возвращает настройки безопасности вида карточки.
UseOwnExtendedSettings	Задаёт или возвращает признак использования собственных настроек расширений.
UseOwnLayouts	Задаёт или возвращает признак использования собственных разметок.
UseOwnSettings	Задаёт или возвращает признак использования собственные настроек вида, автомат состояний, ролевую модель и разметки.

Методы

Имя	Описание
ToString	Возвращает название вида карточки.

Поля

Имя	Описание
CreationSettingsProperty	Определяет свойство "Настройки способов создания карточки вида".
DigestProperty	Определяет свойство "XSLT для дайджеста".
ExtendedSettingsProperty	Определяет свойство "Настройки расширения".
KindsProperty	Определяет свойство "Виды карточек".
NameProperty	Определяет свойство "Название".
NotAvailableProperty	Определяет свойство "Не показывать при выборе".
ProcessesProperty	Определяет свойство "Бизнес-процессы".
ScriptProperty	Определяет свойство "Скрипт".
ScriptProtectProperty	Определяет свойство "Защита сценариев".
SecurityProperty	Определяет свойство "Безопасность".
UseOwnExtendedSettingsProperty	Определяет свойство "Использовать собственные настройки расширений".
UseOwnLayoutsProperty	Определяет свойство "Использовать собственные разметки".
UseOwnSettingsProperty	Определяет свойство "Использовать собственные настройки вида, автомат состояний, ролевую модель и разметки".

KindsCardKind.Digest — свойство

Задаёт или возвращает XSLT инструкцию формирования дайджеста.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public string Digest { get; set; }
```

Значение свойства

Тип: System.String

XSLT-инструкция по получению дайджеста

Заметки

XSLT-инструкция может быть представлена в виде "*Документ <xsl:value-of select=\"//MainInfo/@Name\"/>*"

KindsCardKind.CreationSettings — свойство

Возвращает настройки способа создания вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<KindsCardCreationSetting> CreationSettings { get; }
```

Значение свойства

Тип: ObjectCollection<KindsCardCreationSetting>

Коллекция настроек KindsCardCreationSetting вида карточки

KindsCardKind.ExtendedSettings — свойство

Возвращает коллекцию настроек расширений вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public KindsCardExtendedSettingCollection ExtendedSettings { get; }
```

Значение свойства

Тип: KindsCardExtendedSettingCollection

Коллекция настроек

KindsCardKind.Kinds — свойство

Возвращает коллекцию подвидов данного вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<KindsCardKind> Kinds { get; }
```

Значение свойства

Тип: ObjectCollection<KindsCardKind>

Коллекция видов карточек KindsCardKind

KindsCardKind.KindsCardType — свойство

Возвращает тип карточки, которому принадлежит данный вид.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public KindsCardType KindsCardType { get; }
```

Значение свойства

Тип: KindsCardType

Тип карточки

KindsCardKind.Name — свойство

Задаёт или возвращает название вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public string Name { get; set; }
```

Значение свойства

Тип: System.String

Название вида карточки

KindsCardKind.NotAvailable — свойство

Задаёт или возвращает признак недоступности выбора вида карточки из соответствующих форм.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public bool NotAvailable { get; set; }
```

Значение свойства

Тип: System.Boolean

true — вид карточки недоступен на выбор, иначе — false

KindsCardKind.ParentCardKind — свойство

Возвращает родительский вид карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public KindsCardKind ParentCardKind { get; }
```

Значение свойства

Тип: KindsCardKind

Родительский вид карточки

Заметки

Если вид карточки не имеет родительского вида, будет возвращен null.

KindsCardKind.Processes — свойство

Возвращает коллекцию бизнес-процессов из настроек вида карточки

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<KindsCardProcess> Processes { get; }
```

Значение свойства

Тип: ObjectCollection<KindsCardProcess>

Коллекция настроек бизнес-процессов KindsCardProcess, определённых для вида карточки

KindsCardKind.Script — свойство

Задаёт или возвращает скрипт привязанный к виду карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ScriptingScript Script { get; set; }
```

Значение свойства

Тип: ScriptingScript

Скрипт

KindsCardProcess — класс

Определяет параметры бизнес-процесса, запускаемого из карточки определённого вида.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class KindsCardProcess : ObjectBase
```

Свойства

Имя	Описание
AllowView	Задаёт или возвращает признак разрешения просмотра бизнес-процесса из карточки.
Branches	Возвращает коллекцию переходов состояний при которых происходит запуск бизнес-процесса.
Image	Задаёт или возвращает иконку.
Limit	Задаёт или возвращает максимальное количество экземпляров бизнес-процесса.
Name	Задаёт или возвращает название экземпляров бизнес-процесса.
NoError	Задаёт или возвращает признак скрытия возникающей ошибки при запуске бизнес-процесса.
Operations	Возвращает коллекцию операций при которых происходит запуск бизнес-процесса.
Policy	Задаёт или возвращает политику запуска бизнес-процесса.
TemplateId	Задаёт или возвращает идентификатор шаблона на основании которого будет формироваться экземпляр бизнес-процесса.

Имя	Описание
`Variables`	Возвращает коллекцию переменных бизнес-процесса.

Поля

Имя	Описание
`AllowViewProperty`	Определяет свойство "Позволять просматривать экземпляр".
`BranchesProperty`	Определяет свойство "Переходы состояний".
`ImageProperty`	Определяет свойство "Иконка".
`LimitProperty`	Определяет свойство "Количество экземпляров".
`NameProperty`	Определяет свойство "Название".
`NoErrorProperty`	Определяет свойство "Не показывать ошибки при старте".
`OperationProperty`	Определяет свойство "Операция".
`OperationsProperty`	Определяет свойство "Операции процесса".
`PolicyProperty`	Определяет свойство "Политика инициализации".
`TemplateIdProperty`	Определяет свойство "Идентификатор шаблона".
`VariablesProperty`	Определяет свойство "Настройки переменных".

KindsCardProcessOperation — класс

Определяет операцию, указанную в качестве инициирующей запуск бизнес-процесса из определённого вида карточки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class KindsCardProcessOperation : ObjectBase
```

Свойства

Имя	Описание
Operation	Задаёт или возвращает операцию карточки, после совершения которой следует вызов бизнес-процесса.

Поля

Имя	Описание
OperationProperty	Описывает свойство "Операции процесса".

KindsCardProcessOperation.Operation — свойство

Задаёт или возвращает операцию карточки, после совершения которой следует вызов бизнес-процесса.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StatesOperation Operation { get; set; }
```

Значение свойства

Тип: StatesOperation

Операция, связанная с изменением статуса

KindsCardProcessPolicy — перечисление

Определяет политику запуска бизнес-процесса для вида карточки.

Это перечисление имеет атрибут FlagsAttribute, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
[System.Flags]
public enum KindsCardProcessPolicy
```

Члены

Имя члена	Описание
None	Не задана. Значение 0x00.
Command	Запускается вручную из карточки. Значение 0x01.
StateBranch	При переходе состояния. Значение 0x02.
StateOperation	По операции. Значение 0x04.

KindsCardProcessVariableSync — перечисление

Тип синхронизации параметров.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public  enum KindsCardProcessVariableSync
```

Члены

Имя члена	Описание
ToProcessOneTime	Передавать значение из карточки в переменную при запуске процесса.
ToProcessMultiTimes	Передавать значение из карточки в переменную при каждом его изменении.

Имя члена	Описание
FromProcess	Передавать значение из процесса в карточку при каждом открытии.

KindsCardType — класс

Представляет объектную модель типа карточки, представленного в *Справочник видов карточек*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class KindsCardType : ObjectBase
```

Свойства

Имя	Описание
CardTypeId	Задаёт или возвращает идентификатор типа карточки.
ExtendedSettings	Возвращает настройки расширения из родительского вида карточки.
Kinds	Возвращает дочерние виды карточек, принадлежащие текущему типа карточки.
Processes	Возвращает бизнес-процессы, привязанные к типу карточки.
RootKind	Возвращает родительский вид карточки.

Поля

Имя	Описание
CardTypeIdProperty	Определяет свойство "Идентификатор типа карточки".
HelpTopicProperty	Определяет свойство "Тема справки".

Имя	Описание
HelpUrlProperty	Определяет свойство "URL файла справки".
KindsProperty	Определяет свойство "Виды карточек".

Примеры

```
①

IKindService kindService = objectContext.GetService<IKindService>(); ②

KindsCardType kindType = kindService.GetCardType(new Guid("B9F7BFD7-7429-455E-A3F1-94FFB569C794")); ③

KindsCardKind newKind = kindService.AddCardKind(kindType); ④

objectContext.SaveObject(newKind); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса для работы со Справочником видов карточек.

③ Получение типа карточки "Документ".

④ Добавление нового вида карточки.

⑤ Обязательное сохранение.

KindsCardType.Kinds — свойство

Возвращает дочерние виды карточек, принадлежащие текущему типа карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<KindsCardKind> Kinds { get; }
```

Значение свойства

Тип: ObjectCollection<KindsCardKind>

Возвращает коллекцию видов карточек типа KindsCardKind

LayoutsProperty — класс

Объектная модель секции "Свойство карточки" карточки "Конструктор разметок".

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class LayoutsProperty : ObjectBase
```

Свойства

Имя	Описание
`ColumnAttributes`	Возвращает коллекцию свойств столбцов для элемента управления "таблица".
`DefaultValue`	Задаёт или возвращает значение по умолчанию.
`EditOperation`	Задаёт или возвращает операцию конструктора состояний.
`FieldAlias`	Задаёт или возвращает псевдоним связанного поля.
`Item`	Задаёт или возвращает тип элемента (обычный, элемент панели инструментов).
`LayoutAttributes`	Возвращает коллекцию атрибутов разметки.
`LinkField`	Задаёт или возвращает псевдоним ссылочного поля.
`LinkProperty`	Задаёт или возвращает ссылку на свойство
`Name`	Задаёт или возвращает название свойства.
`PropertyType`	Задаёт или возвращает тип свойства.

Имя	Описание
SectionId	Задаёт или возвращает идентификатор секции — источник данных для свойства.
ToolTip	Задаёт или возвращает текст всплывающей подсказки.

Методы

Имя	Описание
ToString	Возвращает название свойства.

Поля

Имя	Описание
ColumnAttributesProperty	Определяет свойство "Атрибуты столбца".
DefaultValueProperty	Определяет свойство "Значение по умолчанию".
EditOperationProperty	Определяет свойство "Операция редактирования".
FieldAliasProperty	Определяет свойство "Псевдоним связанного поля".
ItemTypeProperty	Определяет свойство "Тип элемента".
LayoutAttributesProperty	Определяет свойство "Атрибуты разметки".
LinkFieldProperty	Определяет свойство "Псевдоним ссылочного поля".
LinkPropertyProperty	Определяет свойство "Ссылка на свойство".
NameProperty	Определяет свойство "Название свойства".
PropertyTypeProperty	Определяет свойство "Тип свойства".

Имя	Описание
SectionIdProperty	Определяет свойство "Идентификатор секции".
ToolTipProperty	Определяет свойство "Всплывающая подсказка".

Заметки

Свойства класса LayoutsProperty отражают некоторые свойства элементов управления разметки карточки.

LayoutsDesignTreeType — перечисление

Определяет тип узла дерева дизайнов.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum LayoutsDesignTreeType
```

Члены

Имя члена	Описание
Kind	Вид
Role	Роль
State	Состояние

Заметки

Дерево дизайнов — дерево условий, по которому определяется разметка карточки.

Примеры

В приведенном примере, определяется тип узла разметки, полученной из дерева разметок

```
ILayoutService layoutService = objectContext.GetService<ILayoutService>(); ①
```

```
KindsCardKind kindsCardKind = objectContext.GetObject<KindsCardKind>(new Guid("00000000-0000-0000-0000-000000000000")); ②

LayoutsDesignTree layoutsDesignTree = layoutService.GetDesignTree(kindsCardKind); ③

LayoutsDesignTreeType layoutsDesignTreeType = layoutsDesignTree.NodeType; ④
```

① Инициализация контекста объектов и получение сервиса.

② Получение вида карточки.

③ Получение дерева дизайнов для вида карточки.

④ Получение типа корневого узла (обычно `LayoutsDesignTreeType.Kind`).

LayoutsLayoutAttributeVisibility — перечисление

Определяет режим отображения атрибута разметки.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum LayoutsLayoutAttributeVisibility
```

Члены

Имя члена	Описание
`Visible`	Видимый.
`EmptySpace`	Пустая ячейка.
`Invisible`	Невидимый.

Заметки

Атрибуты разметки определяют дополнительные свойства элементов управления и представлены в секции "Свойства карточки" конструктора разметок.

Примеры

Ниже приведён пример установки видимости элемента разметки

```
ILayoutService layoutService = objectContext.GetService<ILayoutService>(); ①
```

```
KindsCardKind kindsCardKind = objectContext.GetObject<KindsCardKind>(new Guid("00000000-
0000-0000-0000-000000000000")); ②

LayoutsDesignTree layoutsDesignTree = layoutService.GetDesignTree(kindsCardKind);
LayoutsLayout layoutsLayout = layoutsDesignTree.NodeLayouts.FirstOrDefault().Layout; ③

LayoutsProperty layoutsProperty = layoutService.GetCustomProperty(kindsCardKind,
"ButtonGo", false); ④

LayoutsLayoutAttribute layoutsLayoutAttribute = layoutService.GetLayoutAttribute
(layoutsProperty, layoutsLayout); ⑤

layoutsLayoutAttribute.Visibility = LayoutsLayoutAttributeVisibility.Invisible;
objectContext.SaveObject(layoutsProperty); ⑥
```

① Инициализация контекста объектов и получение сервиса.

② Получение вида карточки.

③ Получение дерева дизайнов и представления из первого узла дерева.

④ Получение свойства карточки.

⑤ Получение атрибута разметки.

⑥ Установка видимости элемента.

LayoutsPropertyItem — перечисление

Определяет тип элемента управления в разметке.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum LayoutsPropertyItem
```

Члены

Имя члена	Описание
Layout	Обычный элемент управления.
Toolbar	Элемент панели инструментов

Заметки

Данное перечисление характеризует назначение элемента управления в свойстве `LayoutsProperty.Item`. Изменение типа разметки приведет к ошибке при открытии карточки.

LayoutsPropertyType — перечисление

Определяет тип свойства в конструкторе разметок.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum LayoutsPropertyType
```

Члены

Имя члена	Описание
`String`	Строка.
`Integer`	Целое число.
`NullableInteger`	Целое число.
`Float`	Число.
`Boolean`	Булевое значение.
`Date`	Дата.
`Time`	Время.
`DateTime`	Дата и время.
`Image`	Изображение.
`Reference`	Ссылки.
`Command`	Кнопка.
`Table`	Таблица.
`NoData`	Нет данных. Возможно элемент управления некорректно настроен.
`Numerator`	Нумератор.

Имя члена	Описание
EmployeeReference	Сотрудник.
PartnerReference	Контрагент.
DepartmentReference	Подразделение.
PartnerOrganizationReference	Подразделение контрагента.
CategoryViewer	Категории.
FolderReference	Папка.
SurveyGroup	Опрос.
TaskTreeView	Дерево исполнения.
BaseUniversalReference	Строка конструктора справочников.
CardLinksView	Дерево связей.
FilePreview	Предпросмотр файла.

Заметки

Элементы перечисления соответствуют типу содержимого доступных элементов управления конструктора разметок.

LinksLinkType — класс

Тип ссылки, зарегистрированной в *Справочнике ссылок*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class LinksLinkType : ObjectBase
```

Конструкторы

Имя	Описание
LinksLinkType()	Инициализирует новый экземпляр класса LinksLinkType.

Свойства

Имя	Описание
`DisplayName`	Задаёт или возвращает отображаемое название ссылки.
`DisplayString`	Возвращает отображаемое название ссылки. Если отображаемое название не указано, будет возвращен тип ссылки.
`LinkName`	Задаёт или возвращает название ссылки.
`NotAvailable`	Задаёт или возвращает признак доступности типа ссылки для выбора.
`OppositeLinkName`	Задаёт или возвращает название обратной ссылки.

Поля

Имя	Описание
`DisplayNameProperty`	Определяет свойство "Отображаемое название".
`LinkNameProperty`	Определяет свойство "Название ссылки".
`NotAvailableProperty`	Определяет свойство "Не показывать при выборе".
`OppositeLinkNameProperty`	Определяет свойство "Название обратной ссылки".

NumerationRules — класс

Представляет *конструктор правил нумерации*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class NumerationRules : BaseDictionaryCard
```

Свойства

Имя	Описание
Numerators	Возвращает коллекцию нумераторов из *конструктора правил нумерации*.
Rules	Возвращает список правил нумерации из *конструктора правил нумерации*.

Поля

Имя	Описание
NumeratorsProperty	Определяет свойство "Нумераторы".
RulesProperty	Определяет свойство "Правила".

NumerationRules.Numerators — свойство

Возвращает коллекцию нумераторов из *конструктора правил нумерации*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<NumerationRulesNumerator> Numerators { get; }
```

Значение свойства

Тип: ObjectCollection<PartnersCompany>

Коллекция элементов типа NumerationRulesNumerator

NumerationRules.Rules — свойство

Возвращает список правил нумерации из *конструктора правил нумерации*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<NumerationRulesRule> Rules { get; }
```

Значение свойства

Тип: ObjectCollection<NumerationRulesRule>

Коллекция элементов типа NumerationRulesRule

NumerationRulesNumerator — класс

Представляет нумератор в *конструкторе правил нумерации*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class NumerationRulesNumerator : ObjectBase
```

Конструкторы

Имя	Описание
NumerationRulesNumerator()	Инициализирует экземпляр класса NumerationRulesNumerator.

Свойства

Имя	Описание
NumeratorCardId	Задаёт или возвращает идентификатор карточки нумератора.
ZoneDate	Задаёт или возвращает дату первой зоны (отсчетная дата для переключения зоны).
ZoneDay	Задаёт или возвращает день обновления зоны.
ZoneInterval	Задаёт или возвращает периодичность обновления зоны.

Имя	Описание
ZoneType	Задаёт или возвращает тип обновления зоны.

Поля

Имя	Описание
NumeratorProperty	Определяет свойство "Нумератор".
ZoneDateProperty	Определяет свойство "Дата первой зоны".
ZoneDayProperty	Определяет свойство "День обновления зоны".
ZoneIntervalProperty	Определяет свойство "Интервал обновления зоны".
ZoneTypeProperty	Определяет свойство "Тип обновления зоны".

NumerationRulesNumeratorZoneType — перечисление

Определяет режим обновления зоны нумератора.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum NumerationRulesNumeratorZoneType
```

Члены

Имя члена	Описание
None	Нет обновления.
Daily	Ежедневно.
Weekly	Еженедельно.
Monthly	Ежемесячно.
Yearly	Ежегодно.

Заметки

Тип обновления определяет частоту переключение на следующую зону нумератора. Зона создаётся автоматически.

Примеры

Ниже рассматривается пример получения типа обновления зоны нумератора

```
①

NumerationRulesNumerator numerationRulesRule = objectContext.GetObject
<NumerationRulesNumerator>(new Guid("00000000-0000-0000-0000-000000000000")); ②

NumerationRulesNumeratorZoneType zoneType = numerationRulesRule.ZoneType; ③
```

① Инициализация контекста объектов.

② Получение правила нумерации.

③ Получение типа обновления.

NumerationRulesRule — класс

Представляет правило нумерации *конструктора правил нумерации*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class NumerationRulesRule : ObjectBase
```

Конструкторы

Имя	Описание
`NumerationRulesRule()`	Инициализирует экземпляр класса `NumerationRulesRule`.

Свойства

Имя	Описание
CardTypeId	Задаёт или возвращает тип карточки, который будет использовать данный нумератор.
Items	Возвращает коллекцию правила нумерации.
Name	Задаёт или возвращает название правила нумерации.

Поля

Имя	Описание
CardTypeIdProperty	Определяет свойство "Тип карточки".
ItemsProperty	Определяет свойство "Значения".
RuleNameProperty	Определяет свойство "Название правила".

NumerationRulesRule.Items — свойство

Возвращает коллекцию правил нумерации.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<NumerationRulesRuleItem> Items { get; }
```

Значение свойства

Тип: ObjectCollection<NumerationRulesRuleItem>

Коллекция элементов типа NumerationRulesRuleItem

NumerationRulesRuleItem — класс

Элемент правила нумерации в *конструкторе правил нумерации*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class NumerationRulesRuleItem : ObjectBase
```

Конструкторы

Имя	Описание
NumerationRulesRuleItem()	Инициализирует экземпляр класса NumerationRulesRuleItem.

Свойства

Имя	Описание
Conditions	Возвращает список условий для строки правила нумерации.
NumeratorRef	Задаёт или возвращает идентификатор нумератора, используемого элементом правила нумерации.
Order	Задаёт или возвращает порядковый номер строки в правиле нумерации.
Prefix	Задаёт или возвращает содержимое префикса в строке правила нумерации.
PrefixType	Задаёт или возвращает тип префикса в строке правила нумерации.
Suffix	Задаёт или возвращает содержимое суффикса в строке правила нумерации.
SuffixType	Задаёт или возвращает тип суффикса в строке правила нумерации.

Поля

Имя	Описание
ConditionsProperty	Определяет свойство "Условия".
NumeratorRefProperty	Определяет свойство "Нумератор".

Имя	Описание
OrderProperty	Определяет свойство "Порядковый номер".
PrefixProperty	Определяет свойство "Префикс".
PrefixTypeProperty	Определяет свойство "Тип префикса".
SuffixProperty	Определяет свойство "Суффикс".
SuffixTypeProperty	Определяет свойство "Тип суффикса".

NumerationRulesRuleItem.Conditions — свойство

Возвращает список условий для строки правила нумерации.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<NumerationRulesRuleItemCondition> Conditions { get; }
```

Значение свойства

Тип: ObjectCollection<NumerationRulesRuleItemCondition>

Коллекция элементов типа NumerationRulesRuleItemCondition

NumerationRulesRuleItem.NumeratorRef — свойство

Задаёт или возвращает идентификатор нумератора, используемого элементом правила нумерации.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public Guid NumeratorRef { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор нумератора

NumerationRulesRuleItem.PrefixType — свойство

Задаёт или возвращает тип префикса в строке правила нумерации.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public NumerationRulesRuleItemLexeme PrefixType { get; set; }
```

Значение свойства

Тип: NumerationRulesRuleItemLexeme

Тип префикса

NumerationRulesRuleItem.SuffixType — свойство

Задаёт или возвращает тип суффикса в строке правила нумерации.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public NumerationRulesRuleItemLexeme SuffixType { get; set; }
```

Значение свойства

Тип: NumerationRulesRuleItemLexeme

Тип суффикса

NumerationRulesRuleItemLexeme — перечисление

Тип префикса при создании номеров нумератора.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum NumerationRulesRuleItemLexeme
```

Члены

Имя члена	Описание
Constant	Жестко заданный префикс.
Xslt	Префикс формируется согласно указанному XSLT преобразованию.

Заметки

Префикс предшествует числовой части выданного номера нумератора.

NumerationRulesRuleItemCondition — класс

Условие в элементе правила нумерации *конструктора правил нумерации*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class NumerationRulesRuleItemCondition : ObjectBase
```

Конструкторы

Имя	Описание
NumerationRulesRuleItemCondition()	Инициализирует экземпляр класса NumerationRulesRuleItemCondition.

Свойства

Имя	Описание
FieldName	Задаёт или возвращает название поля.
KeyName	Возвращает отображаемое имя условия.
Operation	Задаёт или возвращает название операции, применяемой в условии.

Имя	Описание
`SectionId`	Задаёт или возвращает идентификатор секции, в которой находится поле `FieldName`.
`Value`	Задаёт или возвращает значение поля, проверяемое условием.

Поля

Имя	Описание
`FieldNameProperty`	Определяет свойство "Имя поля".
`OperationProperty`	Определяет свойство "Операция".
`SectionIdProperty`	Определяет свойство "Раздел".
`ValueProperty`	Определяет свойство "Значение".

PartnersAddresseAddressType — перечисление

Определяет тип адреса контрагента.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum PartnersAddresseAddressType
```

Члены

Имя члена	Описание
`ContactAddress`	Контактный адрес.
`PostAddress`	Почтовый адрес.
`LegalAddress`	Юридический адрес.

Partners — класс

Объектная модель *Справочника контрагентов*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class Partners : BaseDictionaryCard
```

Свойства

Имя	Описание
AllDepViewFields	Возвращает коллекцию всех отображаемых полей подразделений.
AllEmplViewFields	Возвращает коллекцию всех отображаемых полей сотрудников.
AllGrpViewFields	Возвращает коллекцию всех отображаемых полей группы.
Companies	Задаёт или возвращает список подразделений *Справочника контрагентов*.
Groups	Задаёт или возвращает список групп подразделений *Справочника контрагентов*.
OrgTypes	Задаёт или возвращает список типов юридических лиц *Справочника контрагентов*.
Positions	Задаёт или возвращает список должностей сотрудников контрагентов.
Titles	Задаёт или возвращает варианты обращений к сотрудникам контрагентов.
UserSettings	Возвращает пользовательские настройки для справочника, определяющие виды карточек и режимы открытия.

Поля

Имя	Описание
AllDepViewFieldsProperty	Определяет свойство "Отображаемые поля подразделений".
AllEmplViewFieldsProperty	Определяет свойство "Отображаемые поля сотрудников".
AllGrpViewFieldsProperty	Определяет свойство "Отображаемые поля группы".
CompaniesProperty	Определяет свойство "Подразделения".
GroupsProperty	Определяет свойство "Группы".
OrgTypesProperty	Определяет свойство "Типы юридических лиц".
PositionsProperty	Определяет свойство "Должности".
TitlesProperty	Определяет свойство "Обращения".
UserSettingsProperty	Определяет свойство "Пользовательские настройки".

Partners.Companies — свойство

Задаёт или возвращает список подразделений *Справочника контрагентов*

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<PartnersCompany> Companies { get; set; }
```

Значение свойства

Тип: ObjectCollection<PartnersCompany>

Коллекция элементов типа PartnersCompany

Partners.Groups — свойство

Задаёт или возвращает список групп подразделений *Справочника контрагентов*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<PartnersGroup> Groups { get; set; }
```

Значение свойства

Тип: ObjectCollection<PartnersGroup>

Коллекция элементов типа PartnersGroup

Partners.OrgTypes — свойство

Задаёт или возвращает список типов юридических лиц *Справочника контрагентов*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<PartnersOrgType> OrgTypes { get; set; }
```

Значение свойства

Тип: ObjectCollection<PartnersOrgType>

Коллекция элементов типа PartnersOrgType

Partners.Positions — свойство

Задаёт или возвращает список должностей сотрудников контрагентов

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<PartnersPosition> Positions { get; set; }
```

Значение свойства

Тип: ObjectCollection<PartnersPosition>

Коллекция элементов типа PartnersPosition

PartnersCompany — класс

Контрагент в *Справочнике контрагентов*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PartnersCompany : ObjectBase
```

Свойства

Имя	Описание
Account	Задаёт или возвращает номер расчетного счета контрагента.
Address	Возвращает адрес организации контрагента.
Addresses	Задаёт или возвращает список адресов организации контрагента.
BankAccounts	Задаёт или возвращает список дополнительных банковских реквизитов контрагента.
BankName	Задаёт или возвращает банковские реквизиты контрагента.
BIk	Задаёт или возвращает БИК банка.
CardDepartment	Задаёт или возвращает карточку подразделения контрагента.
ChiefAccountant	Задаёт или возвращает сотрудника, являющегося главным бухгалтером.
Codes	Задаёт или возвращает государственные коды.

Имя	Описание
Comments	Задаёт или возвращает комментарий.
Companies	Возвращает список подразделений контрагента.
ContactPerson	Задаёт или возвращает контактное лицо контрагента.
Contacts	Задаёт или возвращает контакты контрагента.
CorrespondentAccount	Задаёт или возвращает корреспондентский счет.
DepViewFields	Задаёт или возвращает отображаемые поля подчиненных подразделений.
Email	Задаёт или возвращает e-mail адрес.
EmployeeKind	Задаёт или возвращает вид карточки сотрудника.
EmployeeKindSpecified	Задаёт или возвращает признак того, что установлен особый вид карточки сотрудника.
Employees	Задаёт или возвращает список сотрудников контрагента.
EmployeesFormat	Задаёт или возвращает формат отображения сотрудников контрагента.
EmplViewFields	Задаёт или возвращает формат отображения данных о сотрудниках контрагента.
Fax	Задаёт или возвращает факс контрагента.
FullName	Задаёт или возвращает полное название организации.
INn	Задаёт или возвращает ИНН контрагента.

Имя	Описание
IsClient	Задаёт или возвращает признак клиента.
IsVendor	Задаёт или возвращает признак поставщика.
Kind	Задаёт или возвращает вид карточки контрагента.
KindSpecified	Задаёт или возвращает признак того, что установлен особый вид карточки контрагента.
KPp	Задаёт или возвращает КПП контрагента.
Manager	Задаёт или возвращает руководителя контрагента.
Name	Задаёт или возвращает название организации контрагента.
NotAvailable	Задаёт или возвращает признак того, что организация недоступна для выбора.
OKONh	Задаёт или возвращает ОКВЭД.
OKPO	Задаёт или возвращает ОКПО.
OrgType	Задаёт или возвращает тип юридического лица.
ParentCompany	Возвращает вышестоящую, по отношению к текущему подразделению, организацию.
Phone	Задаёт или возвращает телефон контрагента.
Properties	Задаёт или возвращает свойства контрагента.
SyncTag	Задаёт или возвращает поле синхронизации.

Имя	Описание
TabSections	Задаёт или возвращает разделы свойств.
Telex	Задаёт или возвращает телекс контрагента.
Type	Задаёт или возвращает тип организация/контрагент.
URl	Задаёт или возвращает адрес сайта компании.

Поля

Имя	Описание
AccountProperty	Определяет свойство "Расчетный счет".
AddressesProperty	Определяет свойство "Адреса".
BankAccountsProperty	Определяет свойство "Банковские реквизиты".
BankNameProperty	Определяет свойство "Название банка".
BIKProperty	Определяет свойство "БИК".
ChiefAccountantProperty	Определяет свойство "Главный бухгалтер".
CodesProperty	Определяет свойство "Коды".
CommentsProperty	Определяет свойство "Дополнительная информация".
CompaniesProperty	Определяет свойство "Подразделения".
ContactPersonProperty	Определяет свойство "Контактное лицо".
ContactsProperty	Определяет свойство "Контакты".
CorrespondentAccountProperty	Определяет свойство "Корреспондентский счет".

Имя	Описание
DepartmentProperty	Определяет свойство "Ссылка на карточку CardPartnersDepartment".
DepViewFieldsProperty	Определяет свойство "Отображаемые поля подчиненных подразделений".
EmailProperty	Определяет свойство "E-mail".
EmployeeKindProperty	Определяет свойство "Вид карточки сотрудника".
EmployeeKindSpecifiedProperty	Определяет свойство "Вид карточек сотрудника задан".
EmployeesFormatProperty	Определяет свойство "Формат отображения сотрудников".
EmployeesProperty	Определяет свойство "Сотрудники".
EmplViewFieldsProperty	Определяет свойство "Отображаемые поля сотрудников подразделения".
FaxProperty	Определяет свойство "Факс".
FullNameProperty	Определяет свойство "Полное название".
INNProperty	Определяет свойство "ИНН".
IsClientProperty	Определяет свойство "Клиент".
IsVendorProperty	Определяет свойство "Поставщик".
KindProperty	Определяет свойство "Вид".
KindSpecifiedProperty	Определяет свойство "Вид карточки подразделения задан".
KPPProperty	Определяет свойство "КПП".
ManagerProperty	Определяет свойство "Руководитель".
NameProperty	Определяет свойство "Название".
NotAvailableProperty	Определяет свойство "Не показывать при выборе".
OKONHProperty	Определяет свойство "ОКВЭД".

Имя	Описание
OKPOProperty	Определяет свойство "ОКПО".
OrgTypeProperty	Определяет свойство "Тип юридического лица".
PhoneProperty	Определяет свойство "Телефон".
PropertiesProperty	Определяет свойство "Свойства".
SyncTagProperty	Определяет свойство "Поле синхронизации".
TabSectionsProperty	Определяет свойство "Разделы свойств".
TelexProperty	Определяет свойство "Телекс".
TypeProperty	Определяет свойство "Тип подразделения".
URLProperty	Определяет свойство "Сайт компании".

PartnersCompany.Employees — свойство

Задаёт или возвращает список сотрудников контрагента.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<PartnersEmployee> Employees { get; set; }
```

Значение свойства

Тип: ObjectCollection<PartnersEmployee>

Предоставляет коллекцию сотрудников типа PartnersEmployee

PartnersCompany.Manager — свойство

Задаёт или возвращает руководителя контрагента

- **Пространство имён:** DocsVision.BackOffice.ObjectModel

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public PartnersEmployee Manager { get; set; }
```

Значение свойства

Тип: PartnersEmployee

Руководитель

PartnersCompany.OrgType — свойство

Задаёт или возвращает тип юридического лица.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public PartnersOrgType OrgType { get; set; }
```

Значение свойства

Тип: PartnersOrgType

Тип юридического лица

PartnersCompanyType — перечисление

Определяет тип подразделения контрагента в *Справочнике контрагентов*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum PartnersCompanyType
```

Члены

Имя члена	Описание
Organization	Организация.
Department	Подразделение.

PartnersEmployee — класс

Представляет сотрудника контрагента.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PartnersEmployee : ObjectBase
```

Свойства

Имя	Описание
AdditionalPhone	Задаёт или возвращает номер дополнительного телефона.
Address	Задаёт или возвращает адрес сотрудника.
BirthDate	Задаёт или возвращает день рождения.
CardEmployeeKind	Задаёт или возвращает вид карточки сотрудника.
CardEmployeeKindSpecified	Задаёт или возвращает признак того, что задан вид карточки.
ChProperties	Задаёт или возвращает свойства для сотрудника.
City	Задаёт или возвращает город сотрудника .
Comments	Задаёт или возвращает комментарий.
Country	Задаёт или возвращает страну сотрудника.

Имя	Описание
DepartmentName	Возвращает название подразделения, в котором зарегистрирован сотрудник.
DisplayString	Задаёт или возвращает отображаемое имя сотрудника. Отображаемое имя составлено на основе формата, определённого в *Справочнике контрагентов*.
Email	Задаёт или возвращает адрес электронный почты.
EmplName	Возвращает полное ФИО сотрудника.
Fax	Задаёт или возвращает факс.
FirstName	Задаёт или возвращает имя сотрудника.
FullName	Возвращает полное ФИО сотрудника.
Gender	Задаёт или возвращает пол сотрудника контрагента.
LastName	Задаёт или возвращает фамилию сотрудника контрагента.
MiddleName	Задаёт или возвращает отчество сотрудника контрагента.
MobilePhone	Задаёт или возвращает мобильный телефон сотрудника.
NameCases	Задаёт или возвращает падежи имён.
NotAvailable	Задаёт или возвращает признак того, что сотрудник недоступен для выбора.
Phone	Задаёт или возвращает номер телефона сотрудника.
Position	Задаёт или возвращает должность сотрудника.
PositionName	Задаёт или возвращает отображаемую должность сотрудника.

Имя	Описание
ShortName	Задаёт или возвращает сокращенное ФИО.
ShortPositionName	Задаёт или возвращает сокращенное название должности сотрудника.
SyncTag	Задаёт или возвращает поле синхронизации.
Title	Задаёт или возвращает обращение к сотруднику.
Unit	Возвращает контрагента сотрудника.
ZipCode	Задаёт или возвращает индекс подразделения.

Поля

Имя	Описание
AdditionalPhoneProperty	Определяет свойство "Дополнительный телефон".
AddressProperty	Определяет свойство "Адрес".
BirthDateProperty	Определяет свойство "Дата рождения".
CardEmployeeKindProperty	Определяет свойство "Вид".
CardEmployeeKindSpecifiedProperty	Определяет свойство "Вид карточки сотрудника задан".
CardEmployeeProperty	Определяет свойство "Ссылка на карточку CardPartnersEmployee".
ChPropertiesProperty	Определяет свойство "Свойства для сотрудников".
CityProperty	Определяет свойство "Город".
CommentsProperty	Определяет свойство "Дополнительная информация".
CountryProperty	Определяет свойство "Страна".

Имя	Описание
DisplayStringProperty	Определяет свойство "Строка отображения".
EmailProperty	Определяет свойство "E-mail".
FaxProperty	Определяет свойство "Факс".
FirstNameProperty	Определяет свойство "Имя".
GenderProperty	Определяет свойство "Пол".
LastNameProperty	Определяет свойство "Фамилия".
MiddleNameProperty	Определяет свойство "Отчество".
MobilePhoneProperty	Определяет свойство "Мобильный телефон".
NameCasesProperty	Определяет свойство "Падежи имени".
NotAvailableProperty	Определяет свойство "Не показывать при выборе".
PhoneProperty	Определяет свойство "Рабочий телефон".
PositionProperty	Определяет свойство "Должность".
SyncTagProperty	Определяет свойство "Поле синхронизации".
TitleProperty	Определяет свойство "Обращение".
ZipCodeProperty	Определяет свойство "Индекс".

PartnersEmployee.Unit — свойство

Возвращает контрагента сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public PartnersCompany Unit { get; }
```

Значение свойства

Тип: PartnersCompany

Контрагент

PartnersGroup — класс

Группа подразделений контрагента в *Справочнике контрагентов*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PartnersGroup : ObjectBase
```

Свойства

Имя	Описание
Comments	Задаёт или возвращает комментарий к группе.
Group	Возвращает список подразделений группы.
Groups	Возвращает список вложенных групп.
GrpViewFields	Задаёт или возвращает коллекцию отображаемых полей группы.
Name	Задаёт или возвращает название группы.
ParentGroup	Возвращает название родительской группы.

Поля

Имя	Описание
CommentsProperty	Определяет свойство "Комментарий".
GroupProperty	Определяет свойство "Группа".
GroupsProperty	Определяет свойство "Группы".

Имя	Описание
GrpViewFieldsProperty	Определяет свойство "Отображаемые поля группы".
NameProperty	Определяет свойство "Название".

PartnersGroupGroup — класс

Представляет подразделение контрагента в группе *Справочника контрагентов*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PartnersGroupGroup : ObjectBase
```

Свойства

Имя	Описание
Department	Задаёт или возвращает подразделение контрагента.

Поля

Имя	Описание
DepartmentProperty	Определяет свойство "Подразделение".

PartnersOrgType — класс

Тип юридического лица в *Справочнике контрагентов*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class PartnersOrgType : ObjectBase
```

Свойства

Имя	Описание
`Name`	Задаёт или возвращает название типа юридического лица.

Поля

Имя	Описание
`NameProperty`	Определяет свойство "Название".

PartnersPosition — класс

Представляет должность сотрудника контрагента в *Справочнике контрагентов*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class PartnersPosition : ObjectBase
```

Свойства

Имя	Описание
`Accusative`	Задаёт или возвращает название в винительном падеже.
`AlternativeName`	Задаёт или возвращает краткое название должности.
`Comments`	Задаёт или возвращает комментарий к должности.
`Dative`	Задаёт или возвращает название в дательном падеже.
`Genitive`	Задаёт или возвращает название в родительном падеже.
`Instrumental`	Задаёт или возвращает название в творительном падеже.

Имя	Описание
`Name`	Задаёт или возвращает полное название должности.
`Prepositional`	Задаёт или возвращает название в предложном падеже.

Поля

Имя	Описание
`AccusativeProperty`	Определяет свойство "Винительный".
`AlternativeNameProperty`	Определяет свойство "Название по-английски".
`CommentsProperty`	Определяет свойство "Комментарии".
`DativeProperty`	Определяет свойство "Дательный".
`GenitiveProperty`	Определяет свойство "Родительный падеж".
`InstrumentalProperty`	Определяет свойство "Творительный".
`NameProperty`	Определяет свойство "Название".
`PrepositionalProperty`	Определяет свойство "Предложный".

PartnersPropertyShowType — перечисление

Определяет режим отображения свойства *Справочника контрагентов*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum PartnersPropertyShowType
```

Члены

Имя члена	Описание
`PropAndLabel`	Свойство и метку.

Имя члена	Описание
PropertyOnly	Только свойство.
LabelOnly	Только метку.

/dv6/programmer/dv6/BackOffice-ObjectModel-Powers/PowerOfAttorneyMainInfo_CL/[PowerOfAttorneyMainInfo — класс]

ReferenceList — класс

Объектная модель карточки *Список ссылок на карточки*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ReferenceList : BaseCard
```

Свойства

Имя	Описание
References	Возвращает коллекцию ссылок на карточки.

Поля

Имя	Описание
ReferencesProperty	Определяет свойство "Ссылки".

ReferenceListReference — класс

Ссылка в *Списке ссылок на карточки*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class ReferenceListReference : BaseCardSectionRow
```

Свойства

Имя	Описание
Author	Задаёт или возвращает сотрудника, которым добавлена ссылка.
Card	Задаёт или возвращает карточку, на которую указывает ссылка.
CardDescription	Задаёт или возвращает описание карточки.
CardHardlink	Задаёт или возвращает жесткую ссылку на карточку.
CardType	Задаёт или возвращает тип карточки.
CreationDate	Задаёт или возвращает дату добавления ссылки.
Folder	Задаёт или возвращает папку, на которую установлена ссылка.
LinkDescription	Задаёт или возвращает описание ссылки.
Type	Задаёт или возвращает тип ссылки.
URl	Задаёт или возвращает адрес, на который установлена ссылка.

Поля

Имя	Описание
AuthorProperty	Определяет свойство "Кем добавлена".
CardHardlinkProperty	Определяет свойство "Жесткая ссылка на карточку".
CardProperty	Определяет свойство "Карточка".
CardTypeProperty	Определяет свойство "Тип карточки".
CreationDateProperty	Определяет свойство "Дата создания".
FolderProperty	Определяет свойство "Папка".
LinkDescriptionProperty	Определяет свойство "Описание".

Имя	Описание
TypeProperty	Определяет свойство "Тип ссылки".
URLProperty	Определяет свойство "URL".

RoleModel — класс

Объектная модель карточки "Конструктор ролей".

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RoleModel : BaseDictionaryCard
```

Свойства

Имя	Описание
CardKindRoleSettings	Задаёт или возвращает коллекцию настроек прав в рамках ролевой модели.
CustomOperations	Задаёт или возвращает коллекцию зарегистрированных пользовательских операций.
CustomParameters	Задаёт или возвращает коллекцию зарегистрированных пользовательских параметров.

Поля

Имя	Описание
CardKindRoleSettingsProperty	Определяет свойство "Настройки прав".
CustomOperationsProperty	Определяет свойство "Пользовательские операции".
CustomParametersProperty	Определяет свойство "Пользовательские параметры".

RoleModelRole — класс

Представляет роль в ролевой модели.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RoleModelRole : ObjectBase
```

Свойства

Имя	Описание
ConditionGroup	Задаёт или возвращает группу условий роли.
IsCommon	Возвращает признак того, что роль доступна для всех видов карточек.
Name	Задаёт или возвращает название роли.

Методы

Имя	Описание
AddConditionGroup(RoleModelRoleConditionGroup)	Добавляет группу условий к роли.
Validate(Boolean, List<String>)	Осуществляет проверку группы условий роли.

Поля

Имя	Описание
ConditionGroupProperty	Определяет свойство "Группы условий".
NameProperty	Определяет свойство "Наименование".

RoleModelRole.ConditionGroup — свойство

Задаёт или возвращает группу условий роли.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public RoleModelRoleConditionGroup ConditionGroup { get; set; }
```

Значение свойства

Тип: RoleModelRoleConditionGroup

Группа условий

RoleModelRole.AddConditionGroup — метод (RoleModelRoleConditionGroup)

Добавляет группу условий к роли.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public void AddConditionGroup(RoleModelRoleConditionGroup conditionGroup)
```

Параметры

conditionGroup

Тип: RoleModelRoleConditionGroup

Группа условий

RoleModelRole.Validate — метод (Boolean, List<String>)

Осуществляет проверку группы условий роли.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public bool Validate(bool isRoleCommon, out List<string> reasons)
```

Параметры

isRoleCommon

Тип: System.Boolean

Роль является общей

reasons

Тип: System.Collections.Generic.List<String>

Результат проверки

Возвращаемое значение

Тип: System.Boolean

true — ошибок не обнаружено, иначе — false

Заметки

Метод проверяет на ошибки условия заданные в роли.

RoleModelRoleConditionGroup — класс

Предоставляет группу условий для ролевой модели.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class RoleModelRoleConditionGroup : ObjectBase, ICloneable
```

Свойства

Имя	Описание
ConditionGroups	Задаёт или возвращает дочернюю группу условий.
Conditions	Задаёт или возвращает список условий.
Operation	Задаёт или возвращает тип операции, применяемой на результаты выполнения условий.

Имя	Описание
Parent	Возвращает вышестоящую группу условий.
Role	Возвращает роль, которой принадлежит группа условий.

Методы

Имя	Описание
Clone	Создаёт копию группы условий.
Extract(IEnumerable)	Извлекает из существующей группы условий условия, согласно переданному списку. Извлеченные условия удаляются из существующей группы условий.
GetOperationDescription	Возвращает локализованное название для свойства Operation.
GetOperationDescription(RoleModelRoleConditionGroupOperation)	Возвращает локализованное название указанной операции.
Ungroup	Извлекает группы и условия из текущей группы условий в родительскую, при условии, что таковая имеется. Возвращает родительскую группу.

Поля

Имя	Описание
ConditionGroupsProperty	Определяет свойство "Группы условий".
ConditionsProperty	Определяет свойство "Условия".
OperationProperty	Определяет свойство "Операция группы условий".

RoleModelConditionValueDayOfWeek — перечисление

Определяет дни недели для условий, создаваемых для ролевой модели.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum RoleModelConditionValueDayOfWeek
```

Члены

Имя члена	Описание
Monday	Понедельник
Tuesday	Вторник
Wednesday	Среда
Thursday	Четверг
Friday	Пятница
Saturday	Суббота
Sunday	Воскресенье

RoleModelConditionValueDayWorkStatus — перечисление

Определяет статусы дня недели для условий, создаваемых для ролевой модели.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum RoleModelConditionValueDayWorkStatus
```

Члены

Имя члена	Описание
Workday	Рабочий день.
Holiday	Выходной.

RoleModelConditionValueTimeWorkStatus — перечисление

Определяет статус времени для условий, создаваемых для ролевой модели.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum RoleModelConditionValueTimeWorkStatus
```

Члены

Имя члена	Описание
WorkTime	Рабочее время.
FreeTime	Свободное время.

RoleModelRoleConditionParameter — перечисление

Определяет предустановленный параметр для условий в ролевой модели.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum RoleModelRoleConditionParameter
```

Члены

Имя члена	Описание
I	Текущий сотрудник. Значение 0x00.
Manager	Руководитель. Значение 0x01.
Subordinates	Подчиненные. Значение 0x02.
SubordinatesAll	Все подчиненные. Значение 0x03.
SubordinatesTempDeputy	Временнозамещаемые. Значение 0x04.
SubordinatesPermDeputy	Постояннозамещаемые. Значение 0x05.
Deputy	Заместитель. Значение 0x06.

Имя члена	Описание
`Deputized`	Замещаемый. Значение `0x07`.
`FirstActiveDeputy`	Я — первый активный заместитель. Значение `0x08`.
`FirstActivePermanentDeputy`	Я — первый активный постоянный заместитель. Значение `0x09`.
`FirstActiveTempDeputy`	Я — первый активный временный заместитель. Значение `0x0A`.
`FirstActiveDeputyByPerf`	Я — первый активный заместитель исполнения. Значение `0x0B`.
`FirstActiveDeputyOfTheResponsiblePerformer`	Я — первый активный заместитель ответственного исполнения. Значение `0x0C`.
`FirstActiveDeputyBySignature`	Я — первый активный заместитель подписи. Значение `0x0D`.
`InactiveMeIsFirstActive`	Я — временный заместитель в период неактивности замещаемого. Значение `0x0E`.
`MeIsFirstPermanent`	Я — постоянный заместитель. Значение `0x0F`.
`DeputyForSign`	Я — заместитель подписи. Значение `0x10`.
`Today`	Сегодня. Значение `0x60`.
`Now`	Сейчас. Значение `0x61`.
`Field`	Поле. Значение `0x62`.
`Everyone`	Все. Значение `0x64`.

ScriptingScript — класс

Класс ScriptingScript представляется скрипт из *Конструктора скриптов*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class ScriptingScript : ObjectBase, ICloneable
```

Свойства

Имя	Описание
Assemblies	Возвращает дополнительные сборки, используемые в скрипте.
Code	Задаёт или возвращает исходный код скрипта.
Language	Задаёт или возвращает язык программирования, на котором написан скрипт.

Методы

Имя	Описание
Clone	Создаёт копию текущего скрипта.

Поля

Имя	Описание
AssembliesProperty	Представляет свойство "Используемые сборки".
CodeProperty	Представляет свойство "Код скрипта".
LanguageProperty	Представляет свойство "Язык скрипта".

SignatureLabel — класс

Метка подписи из Справочника меток подписей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class SignatureLabel : BaseCardSectionRow
```

Свойства

Имя	Описание
`CanDelete`	Задаёт или возвращает признак разрешения удаления метки.
`Label`	Задаёт или возвращает название метки.
`LabelNames`	Задаёт или возвращает локализованные значения метки.
`LocalizedName`	Возвращает локализованное значение метки.

Методы

Имя	Описание
`ToString`	Предоставляет локализованное название метки.

Поля

Имя	Описание
`CanDeleteProperty`	Определяет свойство "Разрешено удалять".
`LabelNamesProperty`	Определяет свойство "Локализованные имена меток".
`LabelProperty`	Определяет свойство "Метка".

SignatureList — класс

Объектная модель карточки "Список подписей".

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class SignatureList : BaseCard
```

Свойства

Имя	Описание
`Processes`	Возвращает коллекцию бизнес-процессов. Карточка не поддерживает работу с БП, поэтому метод возвращает null.
`Signatures`	Возвращает коллекцию подписей.
`SystemInfo`	Возвращает системную информацию. Карточка не поддерживает хранение системной информации, поэтому метод возвращает null.

Поля

Имя	Описание
`SignaturesProperty`	Определяет свойство "Подписи".

Заметки

В классе переопределены обязательные для карточки данной библиотеки свойства `Processes` и `SystemInfo`, в связи с отсутствием необходимости в хранении указанной информации.

SignatureType — перечисление

Определяет типы подписи

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum SignatureType
```

Члены

Имя члена	Описание
`Unknown = 0`	Неизвестный тип подписи.
`Hash = 1`	`Hash`.

Имя члена	Описание
`Cades_BES = 2`	`Cades_BES`.
`Cades_T = 3`	`Cades_T`.
`Cades_C = 4`	`Cades_C`.
`Cades_XLT1 = 5`	`Cades_XLT1`.
`Cades_A = 6`	`Cades_A`.

Staff — класс

Класс Staff представляет объектную модель *Справочника сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class Staff : BaseDictionaryCard
```

Свойства

Имя	Описание
`ADsMapping`	Задаёт или возвращает коллекцию соответствий между именем поля и атрибутом ADs.
`AllDepViewFields`	Задаёт или возвращает коллекцию отображаемых полей подразделений.
`AllEmplViewFields`	Задаёт или возвращает коллекцию отображаемых полей сотрудников.
`AllGrpViewFields`	Задаёт или возвращает коллекцию отображаемых полей групп.
`Groups`	Возвращает коллекцию групп из *Справочника сотрудников*.
`Positions`	Возвращает коллекцию должностей из *Справочника сотрудников*.

Имя	Описание
Roles	Возвращает коллекцию ролей из *Справочника сотрудников*.
Units	Задаёт или возвращает коллекцию подразделений верхнего уровня иерархии.
UserSettings	Возвращает пользовательские настройки.

Методы

Имя	Описание
AddGroup(StaffGroup)	Добавляет существующую группу в *Справочник сотрудников*.
AddPosition(StaffPosition)	Добавляет должность.
AddRole(StaffRole)	Добавляет роль.
AddUnit(StaffUnit)	Добавляет подразделение.
ExistGroup(StaffGroup)	Проверяет наличие заданной группы.
ExistRole(StaffRole)	Проверяет наличие заданной роли.
RemoveGroup(StaffGroup)	Удаляет группу.
RemovePosition(StaffPosition)	Удаляет должность.
RemoveRole(StaffRole)	Удаляет роль.

Поля

Имя	Описание
ADsMappingProperty	Представляет свойство "Соответствие полей ADs".
AllDepViewFieldsProperty	Представляет свойство "Отображаемые поля подразделений".
AllEmplViewFieldsProperty	Представляет свойство "Отображаемые поля сотрудников".

Имя	Описание
AllGrpViewFieldsProperty	Представляет свойство "Отображаемые поля группы".
GroupsProperty	Представляет свойство "Группы".
PositionsProperty	Представляет свойство "Должности".
RolesProperty	Представляет свойство "Роли".
UnitsProperty	Представляет свойство "Подразделения".
UserSettingsProperty	Представляет свойство "Пользовательские настройки".

Staff.AddGroup — метод (StaffGroup)

Добавляет существующую группу в *Справочник сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public void AddGroup(StaffGroup group)
```

Параметры

group

Тип: StaffGroup

Группа пользователей

Заметки

Данный метод используется при нестандартном добавлении группы, объектная модель которой уже сформирована, в Справочник сотрудников. В остальных ситуациях рекомендуется использовать стандартный метод добавления IStaffService.AddNewGroup.

Staff.Units — свойство

Задаёт или возвращает коллекцию подразделений верхнего уровня иерархии.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<StaffUnit> Units { get; set; }
```

Значение свойства

Тип: ObjectCollection<StaffUnit>

Коллекция подразделений типа StaffUnit

Заметки

Данное свойство возвращает организации расположенные непосредственно на первом уровне иерархии подразделений. Дочерние организации и подразделения могут быть получены через свойство StaffUnit.Units конкретной организации.

StaffAddresse — класс

Предоставляет адресные данные организации в *Справочнике сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StaffAddresse : ObjectBase
```

Свойства

Имя	Описание
Address	Задаёт или возвращает адрес организации.
AddressType	Задаёт или возвращает тип адреса организации.
City	Задаёт или возвращает город.
Country	Задаёт или возвращает страну.

Имя	Описание
ZipCode	Задаёт или возвращает почтовый индекс.

Методы

Имя	Описание
ToString	Возвращает строку, содержащую индекс, город, полный адрес и страну, разделенные запятой.

Поля

Имя	Описание
AddressProperty	Представляет свойство "Адрес".
AddressTypeProperty	Представляет свойство "Тип адреса".
CityProperty	Представляет свойство "Город".
CountryProperty	Представляет свойство "Страна".
ZipCodeProperty	Представляет свойство "Индекс".

StaffAddresseAddressType — перечисление

Определяет тип адреса организации в *Справочнике сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum StaffAddresseAddressType
```

Члены

Имя члена	Описание
ContactAddress	Контактный адрес.
PostAddress	Почтовый адрес.
LegalAddress	Юридический адрес.

StaffADsMapping — класс

Определяет соответствие между атрибутом Active Directory и названием поля в *справочнике сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class StaffADsMapping : ObjectBase
```

Свойства

Имя	Описание
`ADsName`	Задаёт или возвращает название атрибута учетной записи сотрудника в Active Directory.
`FieldName`	Задаёт или возвращает название поля в *справочнике сотрудников*, соответствующее ADsName.

Поля

Имя	Описание
`ADsNameProperty`	Определяет свойство "Атрибут ADs".
`FieldNameProperty`	Определяет свойство "Имя поля".

StaffContain — класс

Представляет контейнер роли в *справочнике сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class StaffContain : ObjectBase
```

Свойства

Имя	Описание
RefId	Задаёт или возвращает идентификатор сотрудника, подразделения, группы или роли.
RefType	Задаёт или возвращает тип ссылки.

Поля

Имя	Описание
RefIDProperty	Определяет свойство "ID ссылки".
RefTypeProperty	Определяет свойство "Тип ссылки".

StaffDeputy — класс

Объектная модель заместителя сотрудника в *справочнике сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StaffDeputy : ObjectBase
```

Свойства

Имя	Описание
Control	Задаёт или возвращает признак того, что заместитель имеет право исполнять те задания замещаемого сотрудника, для которых он назначен ответственным исполнителем (но не те, где замещаемый является рядовым исполнителем).
DeputyAccess	Задаёт или возвращает признак того, что заместителю предоставляются права аналогичные правам замещаемого сотрудника.

Имя	Описание
Employee	Задаёт или возвращает объектную модель сотрудника, являющегося заместителем.
IsNotified	Задаёт или возвращает признак того, что сотрудник получил уведомление (отображается при запуске Windows-клиента) о том, что он является заместителем другого сотрудника.
ParentEmployee	Задаёт или возвращает замещаемого сотрудника.
Performing	Задаёт или возвращает признак того, что заместитель имеет право исполнять те задания, предназначенные для замещаемого им (неактивного) сотрудника, в которых тот является рядовым исполнителем.
Permanent	Задаёт или возвращает признак того, что заместитель может находить, просматривать и исполнять задания замещаемого, даже если основной исполнитель активен и сам его исполняет.
Signature	Задаёт или возвращает признак того, что заместитель имеет право подписывать документ ЭП от имени замещаемого им сотрудника.
SignatureComment	Задаёт или возвращает текст, которым сопровождается подпись заместителя в документах, подписываемых им за замещаемого сотрудника.

Поля

Имя	Описание
ControlProperty	Представляет свойство "Ответственное исполнение".
DeputyAccessProperty	Представляет свойство "Выдавать права доступа".
IsNotifiedProperty	Представляет свойство "Сотрудник уведомлен".
OrderProperty	Представляет свойство "Порядок".
PerformingProperty	Представляет свойство "Исполнение".
PermanentProperty	Представляет свойство "Постоянный заместитель".
SignatureCommentProperty	Представляет свойство "Комментарий к подписи".
SignatureProperty	Представляет свойство "Подпись".
SyncTagProperty	Представляет свойство "Поле синхронизации".

StaffEmployee — класс

Представляет сотрудника подразделения из справочника сотрудников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StaffEmployee : ObjectBase
```

Свойства

Имя	Описание
AccountName	Задаёт или возвращает имя учетной записи сотрудника.
AccountSId	Задаёт или возвращает SID учетной записи.

Имя	Описание
ActiveEmployee	Задаёт или возвращает ссылку на актуального сотрудника. В случае перевода сотрудника в новое подразделение, данное свойство будет содержать ссылку на новую запись в справочнике.
ADsNotSynchronize	Задаёт или возвращает признак отмены синхронизации данных сотрудника с Active Directory.
BirthDate	Задаёт или возвращает дату рождения.
BirthDateSpecified	Возвращает признак того, что день рождения был установлен.
CalendarId	Задаёт или возвращает идентификатор календаря рабочего времени.
CardEmployee	Задаёт или возвращает ссылку на карточку сотрудника.
CardEmployeeKind	Задаёт или возвращает вид карточки сотрудника.
CardEmployeeKindSpecified	Задаёт или возвращает признак того, что вид карточки сотрудника был задан.
ChProperties	Задаёт или возвращает коллекцию свойств для сотрудников.
ClockNumber	Задаёт или возвращает табельный номер.
Comments	Задаёт или возвращает дополнительную информацию по сотруднику.
DelegateFolderNotAvailable	Задаёт или возвращает папку-делегат. Данная папка будет добавлена в персональную папку сотрудника.
DepartmentName	Возвращает название подразделения в котором числиться сотрудник.

Имя	Описание
Deputies	Задаёт или возвращает заместителей сотрудника.
DisplayName	Возвращает отображаемое имя сотрудника.
DisplayString	Задаёт или возвращает строка отображения данных о сотруднике.
Email	Задаёт или возвращает адрес электронной почты.
EmplName	Возвращает значение свойства StaffEmployee.FullName.
EndDate	Задаёт или возвращает дату окончания отсутствия сотрудника на рабочем месте.
Fax	Задаёт или возвращает факс.
FirstName	Задаёт или возвращает имя сотрудника.
FullName	Возвращает строку в формате *Фамилия И.О.*.
Gender	Задаёт или возвращает пол сотрудника.
HomePhone	Задаёт или возвращает номер домашнего телефона.
IDCode	Задаёт или возвращает идентификатор сотрудника, применяемый внутри компании.
IDIssuedBy	Задаёт или возвращает название подразделения выдавшего паспорт.
IDNumber	Задаёт или возвращает номер паспорта.
Importance	Задаёт или возвращает значимость сотрудника.

Имя	Описание
InactiveStatus	Задаёт или возвращает состояние сотрудника в период неактивности.
IPPhone	Задаёт или возвращает номер IP-телефона.
IsDefault	Задаёт или возвращает признак того, что данная запись является учетной записью по умолчанию.
LastName	Задаёт или возвращает фамилию сотрудника.
LockedFrom	Задаёт или возвращает дату, начиная с которой действует блокировка пользователя.
LockedTo	Задаёт или возвращает конечную дату блокировки пользователя.
Manager	Задаёт или возвращает ссылку на руководителя сотрудника.
MiddleName	Задаёт или возвращает отчество сотрудника.
MobilePhone	Задаёт или возвращает номер мобильного телефона.
NameCases	Задаёт или возвращает коллекцию падежей фамилии, имени и отчества.
NotAvailable	Задаёт или возвращает признак того, что сотрудник недоступен для выбора в карточке.
NotSearchable	Задаёт или возвращает признак того, что сотрудник недоступен при поиске.
PersonalFolder	Задаёт или возвращает персональную папку сотрудника.
Phone	Задаёт или возвращает номер местного телефона.

Имя	Описание
Pictures	Задаёт или возвращает коллекцию фотографий сотрудника.
Position	Задаёт или возвращает должность сотрудника.
PositionName	Возвращает название должности.
RoomNumber	Задаёт или возвращает номер комнаты.
RoutingType	Задаёт или возвращает тип маршрутизации.
ShortName	Возвращает строку в формате *Фамилия И.*. Если фамилия не задана, то возвращает имя сотрудника.
ShortPositionName	Возвращает короткое название должности.
StartDate	Задаёт или возвращает дату начала отсутствия сотрудника.
Status	Задаёт или возвращает текущее состояние сотрудника.
Unit	Возвращает подразделение, в котором числиться сотрудник.

Методы

Имя	Описание
ToString	Возвращает значение свойства StaffEmployee.DisplayName

Поля

Имя	Описание
AccountNameProperty	Представляет свойство "Пользователь".
AccountSIDProperty	Представляет свойство "SID учетной записи".

Имя	Описание
ActiveEmployeeProperty	Представляет свойство "Актуальный сотрудник".
ADsNotSynchronizeProperty	Представляет свойство "Не синхронизировать с ADs".
BirthDateProperty	Представляет свойство "Дата рождения".
CalendarIDProperty	Представляет свойство "Календарь рабочего времени".
CardEmployeeKindProperty	Представляет свойство "Вид".
CardEmployeeKindSpecifiedProperty	Представляет свойство "Вид карточки сотрудника задан".
CardEmployeeProperty	Представляет свойство "Ссылка на карточку CardEmployee".
ChPropertiesProperty	Представляет свойство "Свойства для сотрудников".
ClockNumberProperty	Представляет свойство "Табельный номер".
CommentsProperty	Представляет свойство "Дополнительная информация".
DelegateFolderProperty	Представляет свойство "Папка-делегат".
DeputiesProperty	Представляет свойство "Заместители".
DisplayStringProperty	Представляет свойство "Строка отображения".
EmailProperty	Представляет свойство "E-mail".
EndDateProperty	Представляет свойство "Дата окончания отсутствия".
FaxProperty	Представляет свойство "Факс".
FirstNameProperty	Представляет свойство "Имя".
GenderProperty	Представляет свойство "Пол".

Имя	Описание
HomePhoneProperty	Представляет свойство "Домашний телефон".
IDCodeProperty	Представляет свойство "ID код".
IDIssuedByProperty	Представляет свойство "Паспорт выдан".
IDNumberProperty	Представляет свойство "Номер паспорта".
ImportanceProperty	Представляет свойство "Значимость".
InactiveStatusProperty	Представляет свойство "Состояние в период неактивности".
IPPhoneProperty	Представляет свойство "IP-телефон".
IsDefaultProperty	Представляет свойство "Учетная запись по умолчанию".
LastNameProperty	Представляет свойство "Фамилия".
LockedFromProperty	Представляет свойство "Заблокирован с".
LockedToProperty	Представляет свойство "Заблокирован по".
ManagerProperty	Представляет свойство "Руководитель".
MiddleNameProperty	Представляет свойство "Отчество".
MobilePhoneProperty	Представляет свойство "Сотовый телефон".
NameCasesProperty	Представляет свойство "Падежи имени".
NotAvailableProperty	Представляет свойство "Не показывать при выборе".
PersonalFolderProperty	Представляет свойство "Личная папка".
PhoneProperty	Представляет свойство "Местный телефон".
PicturesProperty	Представляет свойство "Фотографии".

Имя	Описание
PositionProperty	Представляет свойство "Должность".
RoomNumberProperty	Представляет свойство "Комната".
RoutingTypeProperty	Представляет свойство "Маршрутизация".
ShowAccountDialogProperty	Представляет свойство "Показывать диалог выбора учетной записи".
StartDateProperty	Представляет свойство "Дата начала отсутствия".
StatusProperty	Представляет свойство "Состояние сотрудника".
SyncTagProperty	Представляет свойство "Поле синхронизации".

StaffEmployee.Deputies — свойство

Задаёт или возвращает заместителей сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<StaffDeputy> Deputies { get; set; }
```

Значение свойства

Тип: ObjectCollection<StaffDeputy>

Коллекция заместителей типа StaffDeputy

Примеры

Ниже приведён пример получения заместителей, имеющих право подписи.

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②
```

```
IEnumerable<StaffEmployee> employees  = staffService.GetCurrentEmployee().Deputies.Where
(t=>t.Signature = true).Select(t=>t.Employee); ③
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение заместителей.

StaffEmployee.DisplayName — свойство

Возвращает отображаемое имя сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public string DisplayName { get; }
```

Значение свойства

Тип: System.String

Отображаемое имя сотрудника

Заметки

Возвращает значение StaffEmployee.DisplayString, а если оно не задано, то вернет StaffEmployee.FullName.

StaffEmployee.InactiveStatus — свойство

Задаёт или возвращает состояние сотрудника в период неактивности.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StaffEmployeeInactiveStatus InactiveStatus { get; set; }
```

Значение свойства

Тип: StaffEmployeeInactiveStatus

Статус сотрудника в период его неактивности

Заметки

Неактивность сотрудника определяется в соответствии с периодом его отсутствия, ограниченным значением "Период отсутствия" в *Справочнике сотрудников* или значениями `StaffEmployee.StartDate` и `StaffEmployee.EndDate` объектной модели.

StaffEmployee.Pictures — свойство

Задаёт или возвращает коллекцию фотографий сотрудника.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public ObjectCollection<StaffPicture> Pictures { get; set; }
```

Значение свойства

Тип: `ObjectCollection<StaffPicture>`

Коллекция элементов типа `StaffPicture`

StaffEmployee.Position — свойство

Задаёт или возвращает должность сотрудника.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public StaffPosition Position { get; set; }
```

Значение свойства

Тип: `StaffPosition`

Должность сотрудника

StaffEmployee.Status — свойство

Задаёт или возвращает текущее состояние сотрудника.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public StaffEmployeeStatus Status { get; set; }
```

Значение свойства

Тип: `StaffEmployeeStatus`

Состояние сотрудника

Заметки

Текущее состояние сотрудника может быть установлено, например, в *Справочнике сотрудников*. Актуализация состояния сотрудника, с проверкой наступления периода отсутствия, производится при вызове метода `IStaffService.ActualizeEmployeesState()`. Актуализация состояния выполняется автоматически бизнес-процессом "Процесс актуализации состояния сотрудников".

StaffEmployee.Unit — свойство

Возвращает подразделение, в котором числиться сотрудник.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public StaffUnit Unit { get; }
```

Значение свойства

Тип: `StaffUnit`

Подразделение

StaffEmployeeStatus — перечисление

Определяет возможные состояния сотрудника организации.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum StaffEmployeeStatus
```

Члены

Имя члена	Описание
Active	Активен.
Sick	Болен.
Vacation	В отпуске.
BusinessTrip	В командировке.
Absent	Отсутствует.
Discharged	Уволен.
Transfered	Переведен.
DischargedNoRestoration	Уволен без возможности восстановления.

StaffEmployeeInactiveStatus — перечисление

Определяет возможные состояния сотрудника в период его неактивности.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum StaffEmployeeInactiveStatus
```

Члены

Имя члена	Описание
Sick	Болен.
Vacation	В отпуске.
BusinessTrip	В командировке.
Absent	Отсутствует.

StaffEmployeesFormat — класс

Формат отображения данных сотрудников подразделения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StaffEmployeesFormat : ObjectBase
```

Свойства

Имя	Описание
FieldName	Задаёт или возвращает название атрибута сотрудника.
FirstLetterOnly	Задаёт или возвращает признак отображения только первого символа выбранного поля.
Order	Задаёт или возвращает порядковый номер формата среди других форматов.
Prefix	Задаёт или возвращает префикс к отображаемому значению.
Suffix	Задаёт или возвращает суффикс к отображаемому значению.

Поля

Имя	Описание
FieldNameProperty	Определяет свойство "Поле".

Имя	Описание
FirstLetterOnlyProperty	Определяет свойство "Только первый символ".
OrderProperty	Определяет свойство "Порядок".
PrefixProperty	Определяет свойство "Префикс".
SuffixProperty	Определяет свойство "Суффикс".

StaffGroup — класс

Группа сотрудников *Справочника сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StaffGroup : ObjectBase
```

Свойства

Имя	Описание
AccountName	Задаёт или возвращает название учетной записи, связанной с группой.
AccountSId	Задаёт или возвращает SID учетной записи, связанной с группой.
ADsNotSynchronize	Задаёт или возвращает признак того, что синхронизация с AD производится не должна.
Comments	Задаёт или возвращает комментарий к группе.
Employees	Возвращает список сотрудников группы.
EmployeesIds	Возвращает список идентификаторов сотрудников группы.

Имя	Описание
GroupFolders	Возвращает список папок группы сотрудников.
GroupItems	Возвращает коллекцию элементов из группы сотрудников.
Groups	Задаёт или возвращает дочерние группы.
GrpViewFields	Задаёт или возвращает массив отображаемых полей группы.
Name	Задаёт или возвращает название группы.
ParentGroup	Возвращает родительскую группы.
Security	Возвращает права доступа к объекту.

Методы

Имя	Описание
AddEmployee(StaffEmployee)	Добавляет указанного сотрудника в группу сотрудников. Устарел. Используйте метод IStaffService.AddEmployeeToGroup.
AddGroup(StaffGroup)	Добавляет в группу указанную дочернюю группу. Устарел. Используйте метод IStaffService.AddNewGroup.
DeleteEmployee(StaffEmployee)	Удаляет сотрудника из группы. Устарел. Используйте метод IStaffService.DeleteEmployee.
DeleteGroup(StaffGroup)	Удаляет из группы подчиненную группу сотрудников. Устарел. Используйте метод IStaffService.DeleteGroup.

Имя	Описание
`EmployeeExist(Guid)`	Проверяет вхождение сотрудника в группу.
`EmployeeExist(StaffEmployee)`	Проверяет вхождение сотрудника в группу.
`ExistGroup(StaffGroup)`	Проверяет существование подчиненной группы.

Поля

Имя	Описание
`AccountNameProperty`	Представляет свойство "Учетная запись".
`AccountSIDProperty`	Представляет свойство "SID учетной записи".
`ADsNotSynchronizeProperty`	Представляет свойство "Не синхронизировать с ADs".
`CommentsProperty`	Представляет свойство "Комментарии".
`DefaultGroupLayoutProperty`	Представляет свойство "Default group layout ".
`DefaultGroupLayoutTimestampProperty`	Представляет свойство "Default layout timestamp".
`GroupFoldersProperty`	Представляет свойство "Папки группы".
`GroupItemsProperty`	Представляет свойство "Группа".
`GroupLayoutProperty`	Представляет свойство "Group layout".
`GroupLayoutTimestampProperty`	Представляет свойство "Group layout timestamp".
`GrpViewFieldsProperty`	Представляет свойство "Отображаемые поля группы".
`NameProperty`	Представляет свойство "Название".
`RefreshADsGroupProperty`	Представляет свойство "Обновлять группу ADs".

StaffGroup.Employees- свойство

Возвращает список сотрудников группы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public IEnumerable<StaffEmployee> Employees { get; }
```

Значение свойства

Тип: System.Collections.Generic.IEnumerable<StaffEmployee>

Список сотрудников типа StaffEmployee

StaffGroup.EmployeesIds — свойство

Возвращает список идентификаторов сотрудников группы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public IEnumerable<Guid> EmployeesIds { get; }
```

Значение свойства

Тип: System.Collections.Generic.IEnumerable<Guid>

Список идентификаторов сотрудников

StaffGroup.GroupFolders — свойство

Возвращает список папок группы сотрудников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<StaffGroupFolder> GroupFolders { get; }
```

Значение свойства

Тип: ObjectCollection<StaffGroupFolder>

Список папок типа StaffGroupFolder

StaffGroup.GroupItems — свойство

Возвращает коллекцию элементов из группы сотрудников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<StaffGroupItem> GroupItems { get; }
```

Значение свойства

Тип: ObjectCollection<StaffGroupItem>

Коллекция объектов типа StaffGroupItem

StaffGroup.Groups — свойство

Задаёт или возвращает дочерние группы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<StaffGroup> Groups { get; set; }
```

Значение свойства

Тип: ObjectCollection<StaffGroup>

Группа сотрудников типа StaffGroup

StaffGroup.ParentGroup — свойство

Возвращает родительскую группы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StaffGroup ParentGroup { get; }
```

Значение свойства

Тип: StaffGroup

Если группа не является подчиненной, то null

StaffGroup.EmployeeExist — метод (Guid)

Проверяет вхождение сотрудника в группу.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public bool EmployeeExist(Guid employeeId)
```

Параметры

employeeId

Тип: System.Guid

Идентификатор сотрудника

Возвращаемое значение

Тип: System.Boolean

`true` — если сотрудник в группе, иначе — `false`

StaffGroup.EmployeeExist — метод (StaffEmployee)

Проверяет вхождение сотрудника в группу.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public bool EmployeeExist(StaffEmployee employee)
```

Параметры

employee

Тип: StaffEmployee

Сотрудник

Возвращаемое значение

Тип: System.Boolean

true — если сотрудник в группе, иначе — false

StaffGroup.ExistGroup — метод (StaffGroup)

Проверяет существование подчиненной группы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public bool ExistGroup(StaffGroup group)
```

Параметры

group

Тип: StaffGroup

Группа сотрудников

Возвращаемое значение

Тип: System.Boolean

true — если группа существует, иначе — false

StaffGroupFolder — класс

Представляет папку определённую в параметрах группы пользователей в *Справочнике сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StaffGroupFolder : ObjectBase
```

Свойства

Имя	Описание
FolderId	Задаёт или возвращает идентификатор папки, отображаемой у сотрудников группы.
ShowInTab	Задаёт или возвращает признак того, что папка будет отображаться в дереве папок Windows-клиента.

Поля

Имя	Описание
FolderIdProperty	Представляет свойство "Идентификатор папки".
ShowInTabProperty	Представляет свойство "Отображать в закладках Windows-клиента".

StaffGroupItem — класс

Представляет сотрудника в группе *Справочника сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StaffGroupItem : ObjectBase
```

Свойства

Имя	Описание
`Employee`	Задаёт или возвращает сотрудника группы.
`EmployeeId`	Задаёт или возвращает идентификатор сотрудника группы.
`Group`	Задаёт или возвращает текущую группу.
`Role`	Задаёт или возвращает роль сотрудника в группе или в *рабочей группе*.

Поля

Имя	Описание
`EmployeeIdProperty`	Представляет свойство "Сотрудник".
`RoleProperty`	Представляет свойство "Роль".
`SyncTagProperty`	Представляет свойство "Поле синхронизации".

StaffGroupRole — перечисление

Определяет роли сотрудника в рабочей группе.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum StaffGroupRole
```

Члены

Имя члена	Описание
`Administrator`	Администратор группы.
`Participant`	Участник.
`Reader`	Только чтение.

StaffPicture — класс

Фотография сотрудника в *справочнике сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class StaffPicture : ObjectBase
```

Конструкторы

Имя	Описание
`StaffPicture()`	Инициализирует экземпляр класса StaffPicture.

Свойства

Имя	Описание
`Image`	Задаёт или возвращает изображение для фотографии сотрудника.
`ImageFormat`	Задаёт или возвращает формат изображения.
`Picture`	Задаёт или возвращает бинарные данные изображения.

Поля

Имя	Описание
`ImageFormatProperty`	Определяет свойство "Формат изображения".
`PictureProperty`	Определяет свойство "Фотография".

Заметки

При создании нового экземпляра `StaffPicture` достаточно установить значение `Image`, а значение `Picture` будет вычислено автоматически.

StaffPicture.Image — свойство

Задаёт или возвращает изображение для фотографии сотрудника.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public Image Image { get; set; }
```

Значение свойства

Тип: Image

Изображение

StaffPicture.ImageFormat — свойство

Задаёт или возвращает формат изображения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StaffPictureImageFormat ImageFormat { get; set; }
```

Значение свойства

Тип: StaffPictureImageFormat

Формат изображения

StaffPicture.Picture — свойство

Задаёт или возвращает бинарные данные изображения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public byte[] Picture { get; set; }
```

Значение свойства

Тип: `System.Byte[]`

Бинарные данные

StaffPictureImageFormat — перечисление

Определяет тип сжатия хранимой фотографии сотрудника в *справочнике сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public enum StaffPictureImageFormat
```

Члены

Имя члена	Описание
`Jpeg`	Формат `.jpeg`.
`Tiff`	Формат `.tiff`.
`Bmp`	Формат `.bmp`.
`Gif`	Формат `.gif`.

StaffPosition — класс

Объектная модель должности сотрудника в *справочнике сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class StaffPosition : ObjectBase
```

Свойства

Имя	Описание
Accusative	Задаёт или возвращает название должности в винительном падеже.
Comments	Задаёт или возвращает содержимое комментария к должности.
Dative	Задаёт или возвращает название должности в дательном падеже.
Genitive	Задаёт или возвращает название должности в родительном падеже.
Instrumental	Задаёт или возвращает название должности в творительном падеже.
Name	Задаёт или возвращает название в именительном падеже.
Prepositional	Задаёт или возвращает название в предложным падеже.
ShortName	Задаёт или возвращает краткое название должности.

Поля

Имя	Описание
AccusativeProperty	Определяет свойство "Винительный".
CommentsProperty	Определяет свойство "Комментарии".
DativeProperty	Определяет свойство "Дательный".
GenitiveProperty	Определяет свойство "Родительный".
ImportanceProperty	Определяет свойство "Значимость".
InstrumentalProperty	Определяет свойство "Творительный".
NameProperty	Определяет свойство "Название".
PrepositionalProperty	Определяет свойство "Предложный".
ShortNameProperty	Определяет свойство "Краткое название".

Имя	Описание
SyncTagProperty	Определяет свойство "Поле синхронизации".

StaffRole — класс

Представляет роль в *справочнике сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StaffRole : ObjectBase
```

Свойства

Имя	Описание
AccountName	Задаёт или возвращает название учетной записи, связанной с ролью.
AccountSId	Задаёт или возвращает SID учетной записи, связанной с ролью.
ADsNotSynchronize	Задаёт или возвращает признак того, что синхронизация с AD производится не должна.
Comments	Задаёт или возвращает комментарий к роли.
Contains	Задаёт или возвращает коллекцию объектов имеющих данную роль.
Name	Задаёт или возвращает название роли.
RoleFolders	Задаёт или возвращает коллекцию папок данной роли.

Поля

Имя	Описание
AccountNameProperty	Представляет свойство "Учетная запись".
AccountSIDProperty	Представляет свойство "SID учетной записи".
ADsNotSynchronizeProperty	Представляет свойство "Не синхронизировать с ADs".
CommentsProperty	Представляет свойство "Комментарии".
ContainsProperty	Представляет свойство "Содержимое".
IsPersonalProperty	Представляет свойство "Персональная роль".
NameProperty	Представляет свойство "Название".
RefreshADsGroupProperty	Представляет свойство "Обновлять группу ADs".
RoleFoldersProperty	Представляет свойство "Папки роли".

StaffRoleFolder — класс

Представляет папку роли в *справочнике сотрудников*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StaffRoleFolder : ObjectBase
```

Свойства

Имя	Описание
FolderId	Задаёт или возвращает идентификатор папки.
ShowInTab	Задаёт или возвращает признак того, что папка будет отображаться в дереве папок Windows-клиента.

Поля

Имя	Описание
`FolderIdProperty`	Определяет свойство "Идентификатор папки".
`ShowInTabProperty`	Определяет свойство "Отображать в закладках Windows-клиента".

StaffUnit — класс

Объектная модель подразделения из *Справочника сотрудников*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class StaffUnit : ObjectBase
```

Свойства

Имя	Описание
`Account`	Задаёт или возвращает расчетный счет организации.
`Address`	Возвращает первый по списку контактный адрес.
`Addresses`	Задаёт или возвращает список адресов компании/подразделения. Включает: фактический адрес, юридический и почтовый.
`ADsId`	Задаёт или возвращает идентификатор подразделения в Active Directory.
`ADsNotSynchronize`	Задаёт или возвращает признак синхронизации записи с Active Directory.
`ADsPath`	Задаёт или возвращает путь в каталоге Active Directory до подразделения.

Имя	Описание
BankName	Задаёт или возвращает название банка, в котором открыт Р/С компании.
BIk	Задаёт или возвращает БИК банка.
CalendarId	Задаёт или возвращает идентификатор бизнес-календаря, согласно которому работает компания.
CardDepartment	Задаёт или возвращает карточку подразделения.
Comments	Задаёт или возвращает дополнительную информацию.
ContactPerson	Задаёт или возвращает сотрудника, являющегося контактным лицом организации.
CorrespondentAccount	Задаёт или возвращает корр. счет.
Email	Задаёт или возвращает адрес эл. почты.
EmployeeKind	Задаёт или возвращает вид карточки сотрудника.
EmployeeKindSpecified	Задаёт или возвращает признак того, что задан вид карточки сотрудника.
Employees	Задаёт или возвращает список сотрудников.
EmployeesFormat	Задаёт или возвращает форматы отображения данных о сотрудниках.
Fax	Задаёт или возвращает номер факса.
FullName	Задаёт или возвращает полное название организации.
IncomingFolder	Задаёт или возвращает папку входящих документов.
INn	Задаёт или возвращает ИНН организации.

Имя	Описание
Kind	Задаёт или возвращает вид карточки подразделения.
KindSpecified	Задаёт или возвращает признак того, что задан вид карточки подразделения.
KPp	Задаёт или возвращает КПП организации.
Manager	Задаёт или возвращает сотрудника, который является руководителем организации.
Name	Задаёт или возвращает краткое название организации.
NotAvailable	Задаёт или возвращает признак недоступности организации для выбора из справочника.
OKONh	Задаёт или возвращает ОКВЭД.
OKPO	Задаёт или возвращает ОКПО.
OutgoingFolder	Задаёт или возвращает папку для исходящих документов.
ParentUnit	Задаёт или возвращает родительскую организацию/подразделение.
Phone	Задаёт или возвращает номер телефона.
Properties	Задаёт или возвращает дополнительные свойства.
ResolutionFolder	Задаёт или возвращает папку для внутренних документов.
RootFolder	Задаёт или возвращает корневую папку, предназначенную для хранения личных папок сотрудников.
Security	Возвращает настройки безопасности.

Имя	Описание
TaskFolder	Задаёт или возвращает папку для заданий.
Telex	Задаёт или возвращает телекс.
TemplateFolder	Задаёт или возвращает шаблонную папку.
Type	Задаёт или возвращает тип контейнера для текущей структурной единицы.
Units	Возвращает список вложенных организаций/подразделений.

Методы

Имя	Описание
AddEmployee(StaffEmployee)	Добавляет сотрудника в организацию.
AddUnit(StaffUnit)	Добавляет дочернее подразделение или организацию.
RemoveUnit(StaffUnit)	Удаляет дочернее подразделение или организацию.

Поля

Имя	Описание
AccountProperty	Определяет свойство "Расчетный счет".
AddressesProperty	Определяет свойство "Адреса".
ADsIDProperty	Определяет свойство "ADs ID".
ADsNotSynchronizeProperty	Определяет свойство "Не синхронизировать с ADs".
ADsPathProperty	Определяет свойство "Путь в ADs".
BankNameProperty	Определяет свойство "Название банка".
BIKProperty	Определяет свойство "БИК".

Имя	Описание
CalendarIDProperty	Определяет свойство "Календарь рабочего времени".
CodeProperty	Определяет свойство "Код".
CommentsProperty	Определяет свойство "Дополнительная информация".
ContactPersonProperty	Определяет свойство "Контактное лицо".
CorrespondentAccountProperty	Определяет свойство "Корреспондентский счет".
DefaultEmployeeLayoutProperty	Определяет свойство "Default employee layout".
DefaultEmployeeLayoutTimestampProperty	Определяет свойство "Default employee layout timestamp".
DefaultUnitLayoutProperty	Определяет свойство "Default unit layout".
DefaultUnitLayoutTimestampProperty	Определяет свойство "Default unit layout timestamp".
DepartmentProperty	Определяет свойство "Ссылка на карточку CardDepartment ".
DepViewFieldsProperty	Определяет свойство "Отображаемые поля подчиненных подразделений".
EmailProperty	Определяет свойство "E-mail".
EmployeeKindProperty	Определяет свойство "Вид карточек сотрудников".
EmployeeKindSpecifiedProperty	Определяет свойство "Вид карточек сотрудников задан".
EmployeeLayoutProperty	Определяет свойство "Employee layout".
EmployeeLayoutTimestampProperty	Определяет свойство "Employee layout timestamp".

Имя	Описание
EmployeesFormatProperty	Определяет свойство "Формат отображения сотрудников".
EmployeesProperty	Определяет свойство "Сотрудники".
EmplViewFieldsProperty	Определяет свойство "Отображаемые поля сотрудников подразделения".
FaxProperty	Определяет свойство "Факс".
FullNameProperty	Определяет свойство "Полное название".
IncomingFolderProperty	Определяет свойство "Папка входящих документов".
INNProperty	Определяет свойство "ИНН".
KindProperty	Определяет свойство "Вид".
KindSpecifiedProperty	Определяет свойство "Вид карточек подразделений задан".
KPPProperty	Определяет свойство "КПП".
ManagerProperty	Определяет свойство "Руководитель".
NameProperty	Определяет свойство "Название".
NotAvailableProperty	Определяет свойство "Не показывать при выборе".
OKONHProperty	Определяет свойство "ОКВЭД".
OKPOProperty	Определяет свойство "ОКПО".
OutgoingFolderProperty	Определяет свойство "Папка исходящих документов".
PhoneProperty	Определяет свойство "Телефон".
PropertiesProperty	Определяет свойство "Свойства".
ResolutionFolderProperty	Определяет свойство "Папка распорядительных документов".
RootFolderProperty	Определяет свойство "Корневая папка".

Имя	Описание
SecurityProperty	Определяет свойство "Безопасность".
SyncTagProperty	Определяет свойство "Поле синхронизации".
TabSectionsProperty	Определяет свойство "Разделы свойств".
TaskFolderProperty	Определяет свойство "Папка заданий".
TelexProperty	Определяет свойство "Телекс".
TemplateFolderProperty	Определяет свойство "Шаблонная папка".
TypeProperty	Определяет свойство "Тип подразделения".
UnitLayoutProperty	Определяет свойство "Unit layout".
UnitLayoutTimestampProperty	Определяет свойство "Unit layout timestamp".
UnitsProperty	Определяет свойство "Подразделения".

StaffUnit.Manager — свойство

Задаёт или возвращает сотрудника, который является руководителем организации.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StaffEmployee Manager { get; set; }
```

Значение свойства

Тип: StaffEmployee

Руководитель организации/подразделения

Заметки

Руководитель подразделения или организации может быть задан, например, в

Справочнике сотрудников. Чтобы руководитель сотрудника записался в поле `Manager`, он должен был явно задан в справочнике сотрудников.

Примеры

В примере выполняется проверка того, что текущий сотрудника является руководителем подразделения, в котором числиться.

```
①

IStaffService staffService = objectContext.GetService<IStaffService>(); ②

StaffEmployee employee = staffService.GetCurrentEmployee(); ③

StaffEmployee manager = employee.Unit.Manager; ④
bool isManager = (employee = manager);
```

① Инициализация контекста объектов.

② Получение сервиса для работы со Справочником сотрудников.

③ Получения текущего сотрудника.

④ Получение руководителя подразделения, в котором числиться сотрудник.

StaffUnit.Units — свойство

Возвращает список вложенных организаций/подразделений.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public ObjectCollection<StaffUnit> Units { get; }
```

Значение свойства

Тип: `ObjectCollection<StaffUnit>`

Коллекция подразделений типа `StaffUnit`

StaffUnit.Type — свойство

Задаёт или возвращает тип контейнера для текущей структурной единицы.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StaffUnitType Type { get; set; }
```

Значение свойства

Тип: StaffUnitType

Тип контейнера

StaffUnitType — перечисление

Определяет тип подразделения в справочнике сотрудников.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum StaffUnitType
```

Члены

Имя члена	Описание
Organization	Организация.
Department	Подразделение.

StatesCardKindStateSetting — класс

Настройки автомата состояний для вида карточки, приведенные в *конструкторе состояний*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StatesCardKindStateSetting : ObjectBase
```

Свойства

Имя	Описание
FirstState	Задаёт или возвращает начальное состояние карточки.
Kind	Задаёт или возвращает вид карточки, которому принадлежат данная настройка.
Operations	Задаёт или возвращает коллекцию операций (перехода или редактирования), доступных для вида карточки.
StateMachineBranches	Задаёт или возвращает переходы автомата состояний.
StateMachineLayout	Задаёт или возвращает разметку для области построения автомата состояний.
States	Задаёт или возвращает коллекцию состояний, доступных виду карточки.

Методы

Имя	Описание
FindBuiltInOperation(BuiltInOperation)	Возвращает операцию (*конструктора состояний*), соответствующую переданной встроенной операции.
FindBuiltInState(BuiltInState)	Возвращает состояние (*конструктора состояний*), соответствующую переданному встроенному состоянию.

Поля

Имя	Описание
FirstStateProperty	Представляет свойство "Первое состояние".
KindProperty	Представляет свойство "Вид карточки".

Имя	Описание
OperationsProperty	Представляет свойство "Операции".
StateMachineBranchesProperty	Представляет свойство "Переходы автомата состояний".
StateMachineLayoutProperty	Представляет свойство "Разметки автомата состояний".
StatesProperty	Представляет свойство "Состояния".

Примеры

Далее приведён пример использования метода GetKindStateSetting при получении начального состояния карточки задание вида На исполнение

```
①

IStateService stateService = objectContext.GetService<IStateService>();
KindsCardKind cardKind = objectContext.GetObject<KindsCardKind>(new Guid("AB801854-70AF-4B6C-AB48-1B59B5D11AA9")); ②

StatesCardKindStateSetting stateSetting = stateService.GetKindStateSetting(cardKind); ③

StatesState startState = stateSetting.FirstState; ④
```

① Инициализация контекста объектов.

② Получение сервиса.

③ Получение настроек вида.

④ Получение начального состояния.

StatesCardKindStateSetting.FirstState — свойство

Задаёт или возвращает начальное состояние, определённое в настройках вида карточки в *Конструкторе состояний*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StatesState FirstState { get; set; }
```

Значение свойства

Тип: StatesState

Начальное состояние

StatesCardKindStateSetting.Kind — свойство

Задаёт или возвращает вид карточки, которому принадлежат данная настройка.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public KindsCardKind Kind { get; set; }
```

Значение свойства

Тип: KindsCardKind

Вид карточки

StatesCardKindStateSetting.Operations — свойство

Задаёт или возвращает коллекцию операций (перехода или редактирования), доступных для вида карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StatesOperationCollection Operations { get; set; }
```

Значение свойства

Тип: StatesOperationCollection

Коллекция операций *Конструктора состояний*

StatesCardKindStateSetting.StateMachineBranches — свойство

Задаёт или возвращает переходы автомата состояний.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StatesStateMachineBranchCollection StateMachineBranches { get; set; }
```

Значение свойства

Тип: StatesStateMachineBranchCollection

Коллекция переходов автомата состояния *Конструктора состояний*

StatesCardKindStateSetting.States — свойство

Задаёт или возвращает коллекцию состояний, доступных виду карточки.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StatesStateCollection States { get; set; }
```

Значение свойства

Тип: StatesStateCollection

Коллекция состояний *Конструктора состояний*

StatesCardKindStateSetting.FindBuiltInOperation — метод (BuiltInOperation)

Возвращает операцию (*конструктора состояний*), соответствующую переданной встроенной операции.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StatesOperation FindBuiltInOperation(BuiltInOperation builtInOperation)
```

Параметры

builtInOperation

Тип: `BuiltInOperation`

Встроенная операция

Возвращаемое значение

Тип: `StatesOperation`

Операция (*конструктора состояний*)

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `builtInOperation`.

StatesCardKindStateSetting.FindBuiltInState — метод (BuiltInState)

Возвращает состояние (*конструктора состояний*), соответствующую переданному встроенному состоянию.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public StatesState FindBuiltInState(BuiltInState builtInState)
```

Параметры

builtInState

Тип: `BuiltInState`

Встроенное состояние

Возвращаемое значение

Тип: `StatesState`

Состояние (*конструктора состояний*)

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр builtInState.

StatesDictionary — класс

Объектная модель карточки "Конструктор состояний".

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StatesDictionary : BaseDictionaryCard
```

Свойства

Имя	Описание
CardKindStateSettings	Задаёт или возвращает коллекцию настроек видов карточек, определённые в *Конструкторе состояний*.
StateMachineLayouts	Задаёт или возвращает коллекцию разметок для автомата состояний в *Конструкторе состояний*.

Поля

Имя	Описание
CardKindStateSettingsProperty	Представляет свойство "Настройки состояния".
StateMachineLayoutsProperty	Представляет свойство "Разметки автомата состояний".

StatesDictionary.CardKindStateSettings — свойство

Задаёт или возвращает коллекцию настроек видов карточек, определённые в *Конструкторе состояний*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<StatesCardKindStateSetting> CardKindStateSettings { get; set; }
```

Значение свойства

Тип: ObjectCollection<StatesCardKindStateSetting>

Коллекция настроек StatesCardKindStateSetting

StatesDictionary.StateMachineLayouts — свойство

Задаёт или возвращает коллекцию разметок для автомата состояний в *Конструкторе состояний*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<StatesStateMachineLayout> StateMachineLayouts { get; set; }
```

Значение свойства

Тип: ObjectCollection<StatesStateMachineLayout>

Коллекция настроек StatesStateMachineLayout

StatesOperation — класс

Представляет операцию, зарегистрированную в *Конструкторе состояний*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StatesOperation : ObjectBase
```

Свойства

Имя	Описание
BuiltInOperation	Задаёт или возвращает идентификатор встроенной операции, соответствующей операции из *Конструктора состояний*.
DefaultName	Задаёт или возвращает название по умолчанию.
EditModeOnly	Задаёт или возвращает признак того, что операция доступна только в режиме редактирования.
LocalizedDescription	Возвращает локализованное описание операции.
LocalizedName	Возвращает локализованное название операции.
OperationDescriptions	Задаёт или возвращает коллекцию всех локализованных описаний операции.
OperationNames	Задаёт или возвращает коллекцию всех локализованных названий операции.

Методы

Имя	Описание
ToString	Возвращает локализованное название операции.

Поля

Имя	Описание
BuiltInOperationProperty	Представляет свойство "Встроенная операция".
DefaultNameProperty	Представляет свойство "Название по умолчанию".
EditModeOnlyProperty	Представляет свойство "Доступна только в режиме редактирования".

Имя	Описание
`OperationDescriptionsProperty`	Представляет свойство "Описание операций".
`OperationNamesProperty`	Представляет свойство "Локализованные названия операций".

StatesOperation.BuiltInOperation — свойство

Задаёт или возвращает идентификатор встроенной операции, соответствующей операции из *Конструктора состояний*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public Guid BuiltInOperation { get; set; }
```

Значение свойства

Тип: `System.Guid`

Идентификатор встроенной операции

StatesOperation.EditModeOnly — свойство

Задаёт или возвращает признак того, что операция доступна только в режиме редактирования.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public bool EditModeOnly { get; set; }
```

Значение свойства

Тип: `System.Boolean`

`true` — доступен исключительно в режиме редактирования, иначе — `false`

StatesOperationCollection — класс

Представляет коллекцию объектов типа StatesOperation.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class StatesOperationCollection : ObjectCollection<StatesOperation>
```

Конструкторы

Имя	Описание
StatesOperationCollection()	Инициализирует новый экземпляр класса StatesOperationCollection.
StatesOperationCollection(ObjectCollection InitializationData)	Инициализирует новый экземпляр класса StatesOperationCollection с помощью указанного значения.

Заметки

Данный класс предоставляет методы для работы с коллекцией операций StatesOperation, зарегистрированных в *Конструкторе состояний*..

StatesOperationDescription — класс

Описание операции в *конструкторе состояний*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StatesOperationDescription : ObjectBase
```

Свойства

Имя	Описание
LocaleId	Задаёт или возвращает номер локали.

Имя	Описание
`Name`	Задаёт или возвращает локализованное описание.

Поля

Имя	Описание
`LocaleIDProperty`	Определяет свойство "Локализация".
`NameProperty`	Определяет свойство "Название".

StatesOperationName — класс

Локализованное название операции в *конструкторе состояний*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class StatesOperationName : ObjectBase
```

Свойства

Имя	Описание
`LocaleId`	Задаёт или возвращает номер локали.
`Name`	Задаёт или возвращает локализованное название операции.

Поля

Имя	Описание
`LocaleIDProperty`	Определяет свойство "Локализация".
`NameProperty`	Определяет свойство "Название".

StatesState — класс

Представляет состояние из конструктора состояний.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StatesState : ObjectBase
```

Свойства

Имя	Описание
BuiltInState	Задаёт или возвращает идентификатор встроенного состояния.
DefaultName	Задаёт или возвращает базовое название состояния.
LocalizedName	Возвращает локализованное название состояния.
StateNames	Задаёт или возвращает коллекцию локализованных имён.

Методы

Имя	Описание
ToString	Возвращает локализованное название.

Поля

Имя	Описание
BuiltInStateProperty	Определяет свойство "Встроенное состояние".
DefaultNameProperty	Определяет свойство "Название по умолчанию".
DynamicProperty	Определяет свойство "Динамическое".
StateNamesProperty	Определяет свойство "Локализованные имена состояний".

StatesState.BuiltInState — свойство

Задаёт или возвращает идентификатор встроенного состояния.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public Guid BuiltInState { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор встроенного состояния

Заметки

Встроенное состояние определяется в объектной модели карточки в отличие от состояния, определённого в Конструкторе состояний.

StatesStateCollection — класс

Представляет коллекцию объектов типа StatesState.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class StatesStateCollection : ObjectCollection<StatesState>
```

Конструкторы

Имя	Описание
StatesStateCollection()	Инициализирует новый экземпляр класса StatesStateCollection.
StatesStateCollection(ObjectCollectionInitializationData)	Инициализирует новый экземпляр класса StatesStateCollection с помощью указанного значения.

Заметки

Класс StatesStateCollection предоставляет методы для работы с коллекцией состояний StatesState, определённых в *Конструкторе состояний*.

StatesStateMachineBranch — класс

Представляет переход автомата состояний.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class StatesStateMachineBranch : ObjectBase
```

Свойства

Имя	Описание
BranchType	Задаёт или возвращает тип перехода состояния в автомате состояний.
BuiltInBranch	Задаёт или возвращает идентификатор встроенного перехода.
Default	Задаёт или возвращает признак того, что переход является переходом по умолчанию.
EndState	Задаёт или возвращает конечное состояние в переходе состояний.
Operation	Задаёт или возвращает операцию перехода состояния.
StartState	Задаёт или возвращает начальное состояние в переходе состояний.

Поля

Имя	Описание
BranchTypeProperty	Определяет свойство "Тип перехода".
BuiltInBranchProperty	Определяет свойство "Встроенный переход".
DefaultProperty	Определяет свойство "По умолчанию".
EndStateProperty	Определяет свойство "Конечное состояние".

Имя	Описание
HotKeyProperty	Определяет свойство "Горячая клавиша".
OperationProperty	Определяет свойство "Операция".
ScriptProperty	Определяет свойство "Скрипт".
StartStateProperty	Определяет свойство "Начальное состояние".

StatesStateMachineBranch.BranchType — свойство

Задаёт или возвращает тип перехода состояния в автомате состояний.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StatesStateMachineBranchBranchType BranchType { get; set; }
```

Значение свойства

Тип: StatesStateMachineBranchBranchType

Тип перехода

StatesStateMachineBranch.BuiltInBranch — свойство

Задаёт или возвращает идентификатор встроенного перехода состояний.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public Guid BuiltInBranch { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор встроенного перехода состояний

Заметки

Встроенный переход состояний определяется в объектной модели карточки в отличие от перехода, определённого в Конструкторе состояний.

StatesStateMachineBranch.EndState — свойство

Задаёт или возвращает конечное состояние в переходе состояний.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StatesState EndState { get; set; }
```

Значение свойства

Тип: StatesState

Конечное состояние перехода

StatesStateMachineBranch.Operation — свойство

Задаёт или возвращает операцию перехода состояния.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StatesOperation Operation { get; set; }
```

Значение свойства

Тип: StatesOperation

Операция перехода

StatesStateMachineBranch.StartState — свойство

Задаёт или возвращает начальное состояние в переходе состояний.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StatesState StartState { get; set; }
```

Значение свойства

Тип: StatesState

Начальное состояние

StatesStateMachineBranchBranchType — перечисление

Определяет тип перехода состояния в автомате состояний.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum StatesStateMachineBranchBranchType
```

Члены

Имя члена	Описание
Point	Без перехода.
Line	Простая смена состояния.
Script	Смена состояния по скрипту.

StatesStateMachineBranchCollection — класс

Представляет коллекцию объектов типа StatesStateMachineBranch.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class StatesStateMachineBranchCollection : ObjectCollection
<StatesStateMachineBranch>
```

Конструкторы

Имя	Описание
`StatesStateMachineBranchCollection()`	Инициализирует новый экземпляр класса StatesStateMachineBranchCollection.
`StatesStateMachineBranchCollection(ObjectCollectionInitializationData)`	Инициализирует новый экземпляр класса `StatesStateMachineBranchCollection` с помощью указанного значения.

Заметки

Предоставляет методы для работы с коллекцией переходов автомата состояний `StatesStateMachineBranch`.

StatesStateMachineLayout — класс

Класс `StatesStateMachineLayout` представляет разметку автомата состояний в *Конструкторе состояний*.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class StatesStateMachineLayout : ObjectBase
```

Свойства

Имя	Описание
`LayoutFileId`	Задаёт или возвращает идентификатор файла разметки автомата состояний.
`LayoutXml`	Задаёт или возвращает разметку для автомата состояний. *Не используется*.

Методы

Имя	Описание
SetSession(UserSession)	Переопределяет пользовательскую сессию, используемую объектом типа StatesStateMachineLayout.

Поля

Имя	Описание
StatesMachineLayoutFileIdProperty	Определяет свойство "Идентификатор файла разметки машины состояний".
StatesMachineLayoutProperty	Определяет свойство "Разметка автомата состояний".

Task — класс

Объектная модель карточки *Задание*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class Task : BaseCard
```

Свойства

Имя	Описание
ActualDelegates	Возвращает коллекцию актуальных делегатов, которым в последний раз было явно делегировано задание.
Comments	Возвращает коллекцию комментариев.
CurrentPerformers	Возвращает коллекцию текущих исполнителей.
Delegates	Возвращает список делегирования (список предыдущих исполнителей задания).

Имя	Описание
MainInfo	Возвращает основные свойства карточки "Задание".
Preset	Возвращает индивидуальные настройки задания.

Поля

Имя	Описание
AcceptanceState	Представляет состояние "На приёмке".
AcceptanceToCompletedByAccept	Представляет переход "приёмка — Завершено".
AcceptanceToNotAcceptedByTakeToRework	Представляет переход "приёмка — На доработке".
AcceptanceToRecalledByRecall	Представляет переход "приёмка — Отозвано".
AcceptOperation	Операция приёмки.
ActualDelegatesProperty	Определяет свойство "Актуальные делегаты".
AddAttachmentOperation	Операция добавления дополнения.
AddCommentOperation	Операция добавления комментария.
AddMainDocumentOperation	Операция добавления основного документа.
AddReportOperation	Операция добавления отчёта.
CancelDelegationOperation	Операция отмены делегирования.
ChangeCurrentPerformerOperation	Операция изменения текущего исполнителя.
ChangeExecutionDateOperation	Операция изменения срока исполнения.
ChangeOperation	Операция изменения.
CommentsProperty	Представляет свойство "Комментарии".
CompletedState	Представляет состояние "Завершено".

Имя	Описание
`CompleteOperation`	Операция завершения.
`CopyChildTaskOperation`	Операция копирования подчинённого задания.
`CreateChildTaskGroupOperation`	Операция создания подчиненной группы заданий.
`CreateChildTaskOperation`	Операция создания подчинённого задания.
`CurrentPerformersProperty`	Представляет свойство "Текущие исполнители".
`DeferOperation`	Операция отложения.
`DeferredState`	Представляет состояние "Отложено".
`DeferredToInWorkByTakeToWork`	Представляет переход "Отложено — В работе".
`DeferredToRecalledByRecall`	Представляет переход "Отложено — Отозвано".
`DeferredToRejectedByReject`	Представляет переход "Отложено — Отклонено".
`DelegatedState`	Представляет состояние "Делегировано".
`DelegatedToDelegatedByCancelDelegation`	Представляет переход "Делегировано — Делегирование отменено".
`DelegatedToDelegatedByDelegate`	Представляет переход "Делегировано — Делегировано".
`DelegatedToInWorkByTakeToWork`	Представляет переход "Делегировано — В работе".
`DelegatedToNotStartedByCancelDelegation`	Представляет переход "Делегировано — Делегирование отменено".
`DelegatedToRecalledByRecall`	Представляет переход "Делегировано — Отозвано".
`DelegatedToRejectedByReject`	Представляет переход "Делегировано — Отклонено".

Имя	Описание
`DelegateOperation`	Операция делегирования.
`DelegatesProperty`	Представляет свойство "Список делегирования".
`DontTakeToWorkOperation`	Операция отказа брать в работу.
`EditCompletionPresetOperation`	Операция редактирования настроек завершения.
`EditPlannedTermsOperation`	Операция ввода плановых сроков.
`EditPresetsOperation`	Операция редактирования общих настроек.
`EditRealTermsOperation`	Операция редактирования фактических параметров исполнения.
`EditRemindersOperation`	Операция редактирования напоминаний.
`InitializationState`	Представляет состояние "Инициализация".
`InitializationToRecalledByRecall`	Представляет переход "Инициализация — Отозвано".
`InitializationToStartedByStart`	Представляет переход "Инициализация — Отправить".
`InWorkState`	Представляет состояние "В работе".
`InWorkToCompletedByComplete`	Представляет переход "В работе — Завершено".
`InWorkToCompletedByStopExecution`	Представляет переход "В работе — Прекратить исполнение".
`InWorkToDeferredByDefer`	Представляет переход "В работе — Отложено".
`InWorkToDelegatedByDelegate`	Представляет переход "В работе — Делегировано".
`InWorkToRecalledByRecall`	Представляет переход "В работе — Отозвано".

Имя	Описание
`InWorkToRejectedByReject`	Представляет переход "В работе в Отклонено".
`MainInfoProperty`	Представляет свойство "Задание".
`ModifyAnyCommentOperation`	Операция изменения любого комментария.
`ModifyOwnCommentOperation`	Операция изменения только собственного комментария
`NotAcceptedState`	Представляет состояние "На доработке".
`NotAcceptedToAcceptanceByComplete`	Представляет переход "На доработке — На приёмке".
`NotAcceptedToCompletedByComplete`	Представляет переход "На доработке — Завершено".
`NotAcceptedToInWorkByTakeToWork`	Представляет переход "На доработке — В работе".
`NotAcceptedToRecalledByRecall`	Представляет переход "На доработке — Отозвано".
`PresetsProperty`	Представляет свойство "Настройки".
`RecalledState`	Представляет состояние "Отозвано".
`RecalledToInitializationByChange`	Представляет переход "Отозвано — Инициализация".
`RecallOperation`	Операция отзыва.
`RejectedState`	Представляет состояние "Отклонено".
`RejectedToInitializationByChange`	Представляет переход "Отклонено — Инициализация".
`RejectedToInWorkByTakeToWork`	Представляет переход "Отклонено — В работе".
`RejectedToRecalledByRecall`	Представляет переход "Отклонено — Отозвано".
`RejectOperation`	Операция отклонения.

Имя	Описание
RemoveAttachmentOperation	Операция удаления дополнения.
RemoveChildTaskGroupOperation	Операция удаления подчиненной группы заданий.
RemoveChildTaskOperation	Операция удаления подчинённого задания.
RemoveMainDocumentOperation	Операция удаления основного документа.
RemoveReportOperation	Операция удаления отчёта.
ReturnFromTheDelegationState	Представляет состояние "Возврат с делегирования".
ReturnFromTheDelegationToAcceptanceByComplete	Представляет переход "Возврат с делегирования — На приёмке".
ReturnFromTheDelegationToCompletedByComplete	Представляет переход "Возврат с делегирования — Завершено".
ReturnFromTheDelegationToInWorkByTakeToWork	Представляет переход "Возврат с делегирования — В работе".
ReturnFromTheDelegationToRecalledByRecall	Представляет переход "Возврат с делегирования — Отозвано".
SelectPerformerOperation	Операция выбора исполнителя.
SetAuthorOperation	Операция редактирования автора.
SetContentOperation	Операция редактирования содержания.
SetControllerOperation	Операция редактирования контролёра.
SetPriorityOperation	Операция установки важности.
StartedState	Представляет состояние "Начато".
StopExecutionOperation	Операция прекращения исполнения.
TakeToReworkOperation	Операция отправки на доработку.
TakeToWorkOperation	Операция приёмки в работу.

Примеры

Ниже приведён пример создания задания, у которого запрещено ручное

делегирование. После создания, задание отправляется исполнителю

```
①

ITaskService taskService = objectContext.GetService<ITaskService>();
IStaffService staffService = objectContext.GetService<IStaffService>();
ILogService iLogService = objectContext.GetService<ILogService>();
IStateService stateService = objectContext.GetService<IStateService>(); ②

KindsCardKind kindsCardKind = objectContext.GetObject<KindsCardKind>(new Guid("00000000-
0000-0000-0000-000000000000")); ③

Task task = taskService.CreateTask(kindsCardKind); ④

taskService.InitializeDefaults(task); ⑤

task.MainInfo.Priority = TaskPriority.High;
task.MainInfo.Content = "Разобрать входящие документы";
task.MainInfo.Name = "Важное задание"; ⑥

task.Preset.AllowDelegateManual = false; ⑦

taskService.AddSelectedPerformer(task.MainInfo, staffService.GetCurrentEmployee()); ⑧

objectContext.SaveObject<Task>(task); ⑨

CardData cardData = userSession.CardManager.GetCardData(objectContext.GetObjectRef<Task
>(task).Id);
task.Description = baseCardService.GenerateDigest(task, cardData, "Важное задание"); ⑩

iLogService.AddLogMessage(task, Task.CreateCardOperation, "Карточка создана");
objectContext.AcceptChanges(); ⑪

taskService.StartTask(task);
StatesStateMachineBranch state = stateService.FindBranchByBuiltIn(Task
.InitializationToStartedByStart, task.SystemInfo.State); ⑫
stateService.ChangeState(task, state);
objectContext.AcceptChanges();
```

① Инициализация контекста объектов.

② Получение необходимых сервисов.

③ Получение вид задания.

④ Создание задания.

⑤ Инициализация задания: заполняется поле автор и настройки вида.

⑥ Повышенный приоритет.

⑦ Запрет ручного делегирования.

⑧ Добавление исполнителей (текущий пользователь).

⑨ Сохранение — для возможности получения данных карточки.

⑩ Получение данных карточки.

⑪ Добавление события в журнал.

⑫ Запуск задания, и смена статуса.

Task.ActualDelegates — свойство

Возвращает коллекцию актуальных делегатов, которым в последний раз было явно делегировано задание.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public ObjectCollection<TaskActualDelegate> ActualDelegates { get; }
```

Значение свойства

Тип: `ObjectCollection<TaskActualDelegate>`

Коллекция типа `TaskActualDelegate`

Заметки

Свойство `ActualDelegates` возвращает коллекцию сотрудников, которым было делегировано задание в последний раз.

Task.Comments — свойство

Возвращает коллекцию комментариев.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public ObjectCollection<TaskComment> Comments { get; }
```

Значение свойства

Тип: ObjectCollection<TaskComment>

Коллекция исполнителей типа TaskComment

Task.CurrentPerformers — свойство

Возвращает коллекцию текущих исполнителей.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<TaskCurrentPerformer> CurrentPerformers { get; }
```

Значение свойства

Тип: ObjectCollection<TaskCurrentPerformer>

Коллекция исполнителей типа TaskCurrentPerformer

Task.Delegates — свойство

Возвращает список делегирования.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<TaskDelegate> Delegates { get; }
```

Значение свойства

Тип: ObjectCollection<TaskDelegate>

Список делегирования типа TaskDelegate

Заметки

Список делегирование представляет собой дерево, содержащее все данные по выполненному событию делегирования.

Task.MainInfo — свойство

Возвращает основные свойства карточки "Задание".

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public TaskMainInfo MainInfo { get; }
```

Значение свойства

Тип: TaskMainInfo

Объектная модель секции "Задание" карточки "Задание"

Заметки

Основная информация содержит сведения о дочерних заданиях, о сроках и исполнителях.

Task.Preset — свойство

Возвращает индивидуальные настройки задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public TaskPreset Preset { get; }
```

Значение свойства

Тип: TaskPreset

Настройки задания

TaskActualDelegate — класс

Предоставляет данные актуального делегата карточки *Задание*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskActualDelegate : BaseCardSectionRow
```

Свойства

Имя	Описание
Employee	Задаёт или возвращает сотрудника, являющегося актуальным делегатом.

Поля

Имя	Описание
EmployeeProperty	Определяет свойство "Сотрудник".

TaskComment — класс

Комментарий к заданию в карточке "Задание".

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskComment : BaseCardSectionRow
```

Свойства

Имя	Описание
Author	Задаёт или возвращает автора комментария.
Content	Задаёт или возвращает содержимое.
CreationDate	Задаёт или возвращает дату добавления комментария.
ShortContent	Возвращает сокращенное содержание. При многострочном комментарии, возвращает первую строчку.

Поля

Имя	Описание
AuthorProperty	Определяет свойство "Автор комментария".
Content	Определяет свойство "Содержание".
CreationDateProperty	Определяет свойство "Дата создания".

TaskCompletionOption — класс

Вариант завершения задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class TaskCompletionOption : BaseCardSectionRow
```

Свойства

Имя	Описание
AdditionalFields	Возвращает дополнительные атрибуты варианта завершения задания.
IconFileId	Задаёт или возвращает идентификатор иконки для кнопки завершения.
Name	Задаёт или возвращает текстовую метку для кнопки завершения.
Operation	Задаёт или возвращает операция перехода.
Value	Задаёт или возвращает значение операции завершения.

Поля

Имя	Описание
AdditionalFieldsProperty	Определяет свойство "Дополнительные атрибуты".

Имя	Описание
IconFileIdProperty	Определяет свойство "Иконка".
NameProperty	Определяет свойство "Название".
OperationProperty	Определяет свойство "Операция".
ValueProperty	Определяет свойство "Значение".

TaskCompletionOptionAdditionalField — класс

Дополнительные атрибуты варианта завершения задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class TaskCompletionOptionAdditionalField : BaseCardSectionRow
```

Свойства

Имя	Описание
Field	Задаёт или возвращает название поля.
Required	Задаёт или возвращает обязательность наличия значения у поля.
SectionId	Задаёт или возвращает идентификатор секции.
ShowDialog	Задаёт или возвращает параметры отображения поля в диалоге завершения задания.

Поля

Имя	Описание
FieldProperty	Определяет свойство "Поле секции".
RequiredProperty	Определяет свойство "Обязательное поле".

Имя	Описание
SectionIdProperty	Определяет свойство "Идентификатор секции".
ShowDialogProperty	Определяет свойство "Показывать диалог".

TaskCompletionOptionAdditionalFieldShowDialog — перечисление

Определяет необходимость отображения поля в диалоге завершения.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum TaskCompletionOptionAdditionalFieldShowDialog
```

Члены

Имя члена	Описание
Always	Отображать.
Never	Не отображать.

TaskCompletionParameter — класс

Параметры завершения задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class TaskCompletionParameter
```

Свойства

Имя	Описание
AdditionalOptions	Возвращает дополнительные параметры завершения задания.

Имя	Описание
`Icon`	Возвращает иконку для параметра завершения.
`IconIco`	Возвращает иконку в формате иконки для параметра завершения.
`Name`	Возвращает название параметра.
`Operation`	Возвращает операцию.
`Value`	Возвращает значение.

TaskCurrentPerformer — класс

Класс `TaskCurrentPerformer` представляет текущего исполнителя задания.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class TaskCurrentPerformer : BaseCardSectionRow
```

Свойства

Имя	Описание
`Employee`	Задаёт или возвращает сотрудника, являющегося текущим исполнителем задания.

Поля

Имя	Описание
`EmployeeProperty`	Определяет свойство "Сотрудник".

TaskCurrentPerformer.Employee — свойство

Задаёт или возвращает сотрудника, являющегося текущим исполнителем задания.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`

- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StaffEmployee Employee { get; set; }
```

Значение свойства

Тип: StaffEmployee

Сотрудник

TaskDelegate — класс

Список делегирований карточки *Задание*.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskDelegate : BaseCardSectionRow
```

Свойства

Имя	Описание
Comment	Задаёт или возвращает комментарий к делегированию.
Date	Задаёт или возвращает время выполнения делегирования.
DelegatedEmployee	Задаёт или возвращает сотрудника, который выполнил делегирование.
DelegatedPerformers	Возвращает список исполнителей.
DelegatedTo	Возвращает список сотрудников, которым было делегировано задание.
Delegates	Возвращает вложенную коллекцию списков делегирования.
Performer	Задаёт или возвращает исполнителя.

Имя	Описание
PreviousDelegate	Возвращает вышестоящий список делегирования.
Reason	Задаёт или возвращает причину делегирования.
Return	Задаёт или возвращает требование "Возврат с делегирования".
ReturnDate	Задаёт или возвращает дату возврата с делегирования, если оно было осуществлено.
Returned	Задаёт или возвращает признак того, что задание было возвращено с делегирования.

Поля

Имя	Описание
CommentProperty	Определяет свойство "Комментарии".
DateProperty	Определяет свойство "Время выполнения делегирования".
DelegatedEmployeeProperty	Определяет свойство "Сотрудник, выполнивший делегирование".
DelegatedPerformersProperty	Определяет свойство "Исполнители".
DelegatedToProperty	Определяет свойство "Кому делегировано".
DelegatesProperty	Определяет свойство "Список делегирования".
PerformerProperty	Определяет свойство "Исполнитель".
ReasonProperty	Определяет свойство "Причина делегирования".
ReturnDateProperty	Определяет свойство "Дата возврата с делегирования".
ReturnedProperty	Определяет свойство "Возвращено".

Имя	Описание
ReturnProperty	Определяет свойство "Признак возврата с делегирования".

TaskDelegateReason — перечисление

Определяет возможные причины делегирования задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public enum TaskDelegateReason
```

Члены

Имя члена	Описание
PerformerNotActive	Исполнитель не активен.
Automatic	Автоматически.
Delegation	Делегирование.

TaskDelegate.DelegatedTo — свойство

Возвращает список сотрудников, которым было делегировано задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public ObjectCollection<TaskDelegatedTo> DelegatedTo { get; }
```

Значение свойства

Тип: ObjectCollection<TaskDelegatedTo>

Коллекция типа TaskDelegatedTo

Заметки

В данной переменной храниться полный список делегатов; актуальные делегаты хранятся в `Task.ActualDelegates`.

TaskDelegatedTo — класс

Предоставляет информацию о том, кому было делегировано задание.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class TaskDelegatedTo : BaseCardSectionRow
```

Свойства

Имя	Описание
`Employee`	Задаёт или возвращает сотрудника, которому было делегировано задание.

Поля

Имя	Описание
`EmployeeProperty`	Определяет свойство "Сотрудник".

TaskDelegatedTo.Employee — свойство

Задаёт или возвращает сотрудника, которому было делегировано задание.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public StaffEmployee Employee { get; set; }
```

Значение свойства

Тип: `StaffEmployee`

Сотрудник

TaskGroup — класс

Объектная модель карточки "Группа заданий".

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public sealed class TaskGroup : BaseCard
```

Свойства

Имя	Описание
`MainInfo`	Возвращает основную информацию по карточке.
`SelectedPerformers`	Возвращает коллекцию выбранных исполнителей.

Поля

Имя	Описание
`CompletedState`	Представляет состояние "Завершено".
`CompleteOperation`	Операция перехода "Завершить".
`EditDocumentsOperation`	Операция редактирования "Редактирование документов".
`EditPlannedTermsOperation`	Операция редактирования "Ввод плановых сроков задания".
`MainInfoProperty`	Представляет свойство "Основная информация".
`PerformanceState`	Представляет состояние "На исполнении"
`PerformanceToCompletedByCompleteBranch`	Представляет переход "На исполнении — Завершено".
`PerformanceToRecalledByRecallBranch`	Представляет переход "На исполнении — Отозвана".

Имя	Описание
`PreparationState`	Представляет состояние "Подготовка"
`PreparationToPerformanceBySendToPerformanceBranch`	Представляет переход "Подготовка — На исполнении".
`RecalledState`	Представляет состояние "Отозвана"
`RecallOperation`	Операция перехода "Отозвать".
`SelectedPerformersProperty`	Представляет свойство "Выбранные исполнители".
`SelectPerformerOperation`	Операция редактирования "Выбор исполнителя".
`SendToPerformanceOperation`	Операция перехода "Отправить на исполнение".
`SetAuthorOperation`	Операция редактирования "Редактирование автора".
`SetContentOperation`	Операция редактирования "Редактирование содержания".
`SetControllerOperation`	Операция редактирования "Редактирование контролёра".
`SetPriorityOperation`	Операция редактирования "Установить важность".
`SetRoutingTypeOperation`	Операция редактирования "Установить тип маршрутизации".

TaskGroupPresets — класс

Представляет индивидуальные настройки исполнителя группы заданий.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class TaskGroupPresets : BaseCardSectionRow
```

Свойства

Имя	Описание
AllowDelegateManual	Задаёт или возвращает признак того, что группа заданий может быть делегирована вручную.
AllowDelegateToAnyEmployee	Задаёт или возвращает признак того, что задание может быть делегировано любому сотрудник.
AllowDelegateToEmployeeFromList	Задаёт или возвращает признак того, что задание может быть делегировано сотрудник из заранее определённого списка.
Comments	Задаёт или возвращает персональный комментарий к заданию сотрудника.
Delegates	Возвращает список сотрудников, к которым относится данная индивидуальная настройка.
DelegateToDeputy	Задаёт или возвращает признак того, что разрешено делегирование заместителю при неактивности исполнителя.
Duration	Задаёт или возвращает длительность (в часах) исполнения задания.
EndDate	Задаёт или возвращает плановую дату завершения исполнения задания.
Laboriousness	Задаёт или возвращает трудоемкость (в часах) исполнения задания.
Reminder	Задаёт или возвращает количество часов, предшествующих сроку выполнения задания, когда исполнителю будет выдано соответствующее информационное сообщение.

Имя	Описание
ReminderDate	Задаёт или возвращает дату, при наступлении которой исполнителю будет выдано информационное сообщение о приближении срока выполнения задания.
ReportFileRequired	Задаёт или возвращает признак того, что при завершении задания необходимо добавить файл отчёта.
ReportRequired	Задаёт или возвращает признак того, что при завершении задания необходимо ввести отчёт.
RequestCommentAtTaskRejection	Задаёт или возвращает признак того, что необходимо запрашивать комментарий при отклонении задания.
RoutingType	Задаёт или возвращает тип маршрутизации задания.
SeparateTasks	Задаёт или возвращает признак того, что для каждого исполнителя будет создано отдельное задание.
StartDate	Задаёт или возвращает плановую дату начала исполнения задания.
UseBusinessCalendar	Задаёт или возвращает признак того, что при расчете плановой даты завершения задания необходимо учитывать бизнес-календарь исполнителя задания.
UseOwnSettings	Задаёт или возвращает признак того, что должны использоваться индивидуальные настройки завершения и делегирования задания.

Поля

Имя	Описание
AllowDelegateManualProperty	Определяет свойство "Разрешено делегировать вручную".
AllowDelegateToAnyEmployeeProperty	Определяет свойство "Разрешено делегирование вручную любому сотруднику".
AllowDelegateToEmployeeFromListProperty	Определяет свойство "Разрешено делегирование вручную сотруднику из списка".
CommentsProperty	Определяет свойство "Комментарии".
DelegatesProperty	Определяет свойство "Настройка исполнителей".
DelegateToDeputyProperty	Определяет свойство "Разрешить делегирование заместителю при неактивности исполнителя".
DurationProperty	Определяет свойство "Длительность".
EndDateProperty	Определяет свойство "Дата завершения плановая".
LaboriousnessProperty	Определяет свойство "Трудоемкость".
ReminderDateProperty	Определяет свойство "Дата напоминания".
ReminderProperty	Определяет свойство "Напомнить за N часов до срока завершения задания".
ReportFileRequiredProperty	Определяет свойство "Запрашивать файл отчёта".
ReportRequiredProperty	Определяет свойство "Ввести отчёт".

TaskGroupPresetsDelegate — класс

Представляет исполнителя в индивидуальных настройках исполнителя группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
TaskGroupPresetsDelegate
```

Свойства

Имя	Описание
Employee	Задаёт или возвращает сотрудника.
Group	Задаёт или возвращает группу.
Role	Задаёт или возвращает роль.
SearchWord	Задаёт или возвращает идентификатор поискового слова.
Unit	Задаёт или возвращает подразделение.

Поля

Имя	Описание
EmployeeProperty	Определяет свойство "Сотрудник".
GroupProperty	Определяет свойство "Группа".
RoleProperty	Определяет свойство "Роль".
SearchWordProperty	Определяет свойство "Поисковое слово".
UnitProperty	Определяет свойство "Подразделение".

TaskGroupSelectedPerformer — класс

Выбранный исполнитель группы заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskGroupSelectedPerformer : BaseCardSectionRow
```

Свойства

Имя	Описание
Employee	Задаёт или возвращает сотрудника, который выбран исполнителем.
Group	Задаёт или возвращает группу, выбранную в качестве исполнителя.
Order	Задаёт или возвращает порядковый номер выбранного исполнителя в списке выбранных исполнителей.
Presets	Задаёт или возвращает основные настройки исполнителя.
Role	Задаёт или возвращает роль, выбранную в качестве исполнителя.
SearchWord	Задаёт или возвращает идентификатор поискового слова, выбранного в качестве исполнителя.
Unit	Задаёт или возвращает подразделение, выбранное в качестве исполнителя.

Поля

Имя	Описание
EmployeeProperty	Определяет свойство "Сотрудник".
GroupProperty	Определяет свойство "Группа".
OrderProperty	Определяет свойство "Номер".
PresetsProperty	Определяет свойство "Настройки".
RoleProperty	Определяет свойство "Роль".
SearchWordProperty	Определяет свойство "Поисковое слово".
UnitProperty	Определяет свойство "Подразделение".

TaskList — класс

Объектная модель карточки "Список ссылок на карточки заданий".

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskList : BaseCard
```

Свойства

Имя	Описание
Processes	Возвращает коллекцию бизнес-процессов прикрепленных к карточке.
SystemInfo	Возвращает вид и состояние карточки.
TaskGroups	Возвращает коллекцию групп заданий.
Tasks	Возвращает коллекцию заданий.

Поля

Имя	Описание
TaskGroupsProperty	Коллекция групп заданий.
TasksProperty	Коллекция заданий.

TaskListTask — класс

Класс TaskListTask представляется объектную модель задания, определённого в списке заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskListTask : BaseCardSectionRow
```

Свойства

Имя	Описание
`Controller`	Задаёт или возвращает сотрудника, выполняющего роль контролёра задания.
`OnControl`	Задаёт или возвращает признак того, что задание на контроле.
`Task`	Задаёт или возвращает задание.

Поля

Имя	Описание
`ControllerProperty`	Определяет свойство "Контролёр".
`OnControlProperty`	Определяет свойство "На контроле".
`OnControlProperty`	Определяет свойство "Задание".

TaskListTaskGroup — класс

Класс `TaskListTaskGroup` представляет объектную модель группы заданий, определённую в списке заданий.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class TaskListTaskGroup : BaseCardSectionRow
```

Свойства

Имя	Описание
`TaskGroup`	Задаёт или возвращает группу заданий.

Поля

Имя	Описание
`TaskGroupProperty`	Определяет свойство "Группа заданий".

TaskMainInfo — класс

Объектная модель секции `MainInfo` карточки Задание#.

- **Пространство имён:** `DocsVision.BackOffice.ObjectModel`
- **Сборка:** `DocsVision.BackOffice.ObjectModel.dll`

Синтаксис

```
public class TaskMainInfo : BaseCardSectionRow
```

Свойства

Имя	Описание
`Author`	Задаёт или возвращает автора задания.
`ChildTaskList`	Задаёт или возвращает подчиненное задание по отношению к текущему.
`CompletedUser`	Задаёт или возвращает сотрудника, который завершил задание.
`CompletionResult`	Задаёт или возвращает результат выполнения задания.
`Content`	Задаёт или возвращает содержание задания.
`ControlDate`	Задаёт или возвращает дату контроля задания.
`ControlKindId`	Задаёт или возвращает вид контроля.
`Controller`	Задаёт или возвращает контроллера задания.
`CreateMessages`	Задаёт или возвращает признак разрешения рассылки сообщений по почте.
`Delegate`	Задаёт или возвращает информацию по делегату.
`DurationActual`	Задаёт или возвращает фактическую продолжительность.

Имя	Описание
EndDate	Задаёт или возвращает дату окончания задания по плану.
EndDateActual	Задаёт или возвращает фактическую дату окончания задания.
EndDateBeforePostponement	Задаёт или возвращает дату окончания выполнения задания до переноса сроков исполнения.
ExecutionStopped	Задаёт или возвращает признак того, что исполнение задания остановлено.
ForManager	Задаёт или возвращает признак того, что задание назначено руководителю.
IsOverdue	Задаёт или возвращает признак того, что исполнение задания просрочено.
Laboriousness	Задаёт или возвращает плановую трудоемкость задания.
LaboriousnessActual	Задаёт или возвращает фактическую трудоемкость задания.
Name	Задаёт или возвращает название задания.
OnControl	Задаёт или возвращает признак того, что задание находится на контроле.
OutlookTask	Задаёт или возвращает содержимое задания для Outlook.
ParentTask	Задаёт или возвращает родительское задание.
PercentCompleted	Задаёт или возвращает процент выполнения.
Performers	Возвращает список исполнителей.
PostponementCount	Задаёт или возвращает количество переносов сроков задания.

Имя	Описание
`Priority`	Задаёт или возвращает приоритет задания.
`ReferenceList`	Задаёт или возвращает список ссылок для текущего задания.
`Reminder`	Задаёт или возвращает количество часов до окончания срока исполнения задания, за которые будет отправлено напоминание.
`ReminderDate`	Задаёт или возвращает дату напоминания о выполнении задания.
`Report`	Задаёт или возвращает содержимое отчёта исполнителя.
`RequiresAcceptance`	Задаёт или возвращает признак того, что заданию требуется приёмка.
`ResponsibleTaskId`	Задаёт или возвращает идентификатор задания ответственного исполнителя.
`ResponsibleTaskPerformerId`	Задаёт или возвращает идентификатор ответственного сотрудника.
`SelectedPerformers`	Задаёт список назначенных исполнителей.
`SignatureList`	Задаёт или возвращает список подписей для текущего задания.
`StartDate`	Задаёт или возвращает плановую дату начала выполнения задания.
`StartDateActual`	Задаёт или возвращает фактическую дату начала выполнения задания.
`StartDateBeforePostponement`	Задаёт или возвращает дату начала выполнения задания до переноса сроков исполнения.
`StartTaskDate`	Задаёт или возвращает дату запуска задания на выполнение.

Имя	Описание
WorkGroup	Задаёт или возвращает рабочую группу.

Поля

Имя	Описание
AuthorProperty	Автор задания.
ChildTaskListProperty	Дочернее задание по отношению к текущему.
CompletedUserProperty	Сотрудник завершивший задание.
CompletionResultProperty	Результат выполнения задания.
ContentProperty	Содержание задания.
ControlDateProperty	Дату контроля задания.
ControlKindIdProperty	Вид контроля.
ControllerProperty	Контроллера задания.
CreateMessagesProperty	Разрешения рассылки сообщений по почте.
DelegateProperty	Делегат.
DurationActualProperty	Фактическая продолжительность.
EndDateActualProperty	Фактическая дата окончания задания.
EndDateBeforePostponementProperty	Дату окончания выполнения задания до переноса сроков исполнения.
EndDateProperty	Дата окончания задания по плану.
ExecutionStoppedProperty	Исполнение задания остановлено.
ForManagerProperty	Задание назначено руководителю.
IsOverdueProperty	Исполнение задания просрочено.
LaboriousnessActualProperty	Фактическая трудоемкость задания.
LaboriousnessProperty	Плановая трудоемкость задания.
NameProperty	Название задания.

Имя	Описание
OnControlProperty	Задание находится на контроле.
OutlookTaskProperty	Содержимое задания для Outlook.
ParentTaskProperty	Родительское задание.
PercentCompletedProperty	Процент выполнения.
PerformersProperty	Список исполнителей.
PostponementCountProperty	Количество переносов сроков задания.
PriorityProperty	Приоритет задания.
ReferenceListProperty	Список ссылок.
ReminderDateProperty	Дата напоминания о выполнении задания.
ReminderProperty	Количество часов до окончания срока исполнения задания, за которые будет отправлено напоминание.
ReportProperty	Содержимое отчёта исполнителя.
RequiresAcceptanceProperty	Требуется приёмка.
ResponsibleTaskIdProperty	Идентификатор задания ответственного исполнителя.
ResponsibleTaskPerformerIdProperty	Идентификатор ответственного сотрудника.
SelectedPerformersProperty	Список назначенных исполнителей.
SignatureListProperty	Список подписей для текущего задания.
StartDateActualProperty	Фактическая дата начала выполнения задания.
StartDateBeforePostponementProperty	Дата начала выполнения задания до переноса сроков исполнения.
StartDateProperty	Плановая дата начала выполнения задания.
StartTaskDateProperty	Дата запуска задания на выполнение.

Имя	Описание
WorkGroupProperty	Определяет свойство "Рабочая группа".

TaskMainInfo.WorkGroup — свойство

Задаёт или возвращает рабочую группу.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StaffGroup WorkGroup { get; set; }
```

Значение свойства

Тип: StaffGroup

Группа сотрудников

TaskPerformer — класс

Класс TaskPerformer представляет назначенного исполнителя здания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskPerformer : BaseCardSectionRow
```

Свойства

Имя	Описание
Employee	Задаёт или возвращает сотрудника, являющегося назначенным исполнителем задания.

Поля

Имя	Описание
EmployeeProperty	Определяет свойство "Сотрудник".

TaskPerformer.Employee — свойство

Задаёт или возвращает сотрудника, являющегося назначенным исполнителем задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public StaffEmployee Employee { get; set; }
```

Значение свойства

Тип: StaffEmployee

Сотрудник

TaskPreset — класс

Настройки задания в карточки "Задание".

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class TaskPreset : BaseCardSectionRow
```

Свойства

Имя	Описание
AllowDelegateManual	Задаёт или возвращает признак того, что задание может быть делегировано вручную.
AllowDelegateToAnyEmployee	Задаёт или возвращает признак того, что задание может быть делегировано любому сотруднику.

Имя	Описание
AllowDelegateToEmployeeFromList	Задаёт или возвращает признак того, что задание может быть делегировано сотруднику из списка.
AttachmentLinkTypes	Возвращает коллекцию типов ссылок для дополнений.
ChildCopyFields	Возвращает коллекцию полей, которые будут скопированы из главного задания в подчиненное.
ChildKind	Возвращает настройки видов подчиненного задания.
Completion	Возвращает настройки завершения задания.
Delegates	Возвращает список сотрудников, котором разрешено осуществлять делегирование. Элемент списка может представлять сотрудника, группу, подразделение, роль или поисковое слово.
DelegateToDeputy	Задаёт или возвращает признак того, что задание разрешено делегировать заместителю при неактивном сотруднике.
GroupChildKind	Возвращает настройки видов подчиненной группы заданий.
Initialized	Задаёт или возвращает признак того, что настройки были проинициализированы.
MainLinkTypes	Возвращает коллекцию типов ссылок для основного документа.
ReportLinkTypes	Возвращает коллекцию типов ссылок для отчёта.
RequestCommentAtTaskRejection	Запрашивать комментарий при отклонении задания.

Имя	Описание
Routing	Возвращает тип маршрутизации задания.
UseBusinessCalendar	Задаёт или возвращает признак использования бизнес-календаря.

Поля

Имя	Описание
AllowDelegateManualProperty	Представляет свойство "Разрешено делегировать вручную".
AllowDelegateToAnyEmployeeProperty	Представляет свойство "Разрешено делегирование вручную любому сотруднику".
AllowDelegateToEmployeeFromListProperty	Представляет свойство "Разрешено делегирование вручную сотруднику из списка".
AttachmentLinkTypesProperty	Представляет свойство "Настройка дополнительных типов ссылок".
ChildCopyFieldsProperty	Представляет свойство "Настройки копирования".
ChildKindProperty	Представляет свойство "Настройки вида подчинённого задания".
CompletionProperty	Представляет свойство "Настройки завершения".
DelegatesProperty	Представляет свойство "Настройка исполнителей".
DelegateToDeputyProperty	Представляет свойство "Разрешить делегирование заместителю при неактивности исполнителя".
GroupChildKindProperty	Представляет свойство "Настройки вида подчиненной группы заданий".
InitializedProperty	Представляет свойство "Инициализированы".

Имя	Описание
MainLinkTypesProperty	Представляет свойство "Настройка основных типов ссылок".
ReportLinkTypesProperty	Представляет свойство "Настройка типов ссылок отчётов".
RequestCommentAtTaskRejectionProperty	Представляет свойство "Запрашивать комментарий при отклонении задания".
RoutingProperty	Представляет свойство "Настройки маршрутизации".
UseBusinessCalendarProperty	Представляет свойство "Использовать бизнес-календарь".

TaskPresetChildCopyField — класс

Класс TaskPresetChildCopyField описывает свойства поля, копируемого из родительского в подчиненное задание.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskPresetChildCopyField : BaseCardSectionRow
```

Свойства

Имя	Описание
FieldAlias	Задаёт или возвращает псевдоним поля.
FieldId	Задаёт или возвращает уникальный идентификатор поля.
FieldName	Задаёт или возвращает локализованное название поля.
SectionName	Задаёт или возвращает локализованное название секции.

Имя	Описание
SectionPath	Задаёт или возвращает последовательность строку, содержащую последовательность секций, полученную из идентификаторов Sections.

Поля

Имя	Описание
FieldAliasProperty	Определяет свойство "Псевдоним поля".
FieldIdProperty	Определяет свойство "Идентификатор поля".
FieldNameProperty	Определяет свойство "Название поля".
SectionNameProperty	Определяет свойство "Название секции".
SectionPathProperty	Определяет свойство "Путь к секции".

TaskPresetChildKind — класс

Класс TaskPresetChildKind предоставляет настройки вида подчинённого задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskPresetChildKind : BaseCardSectionRow
```

Свойства

Имя	Описание
ChildKindSettings	Возвращает виды заданий, доступные при создании подчиненного задания.

Имя	Описание
ChildTaskKindType	Задаёт или возвращает настройки режима создания подчиненного задания.

Поля

Имя	Описание
ChildKindSettingsProperty	Представляет свойство "Виды, доступные для создания подчинённого задания".
ChildTaskKindTypeProperty	Представляет свойство "Настройки вида подчинённого задания".

TaskPresetChildKindSetting — класс

Представляет вид задания, доступный для создания подчиненного задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskPresetChildKindSetting : BaseCardSectionRow
```

Свойства

Имя	Описание
Creatable	Задаёт или возвращает признак разрешения использования вида при создании подчиненного задания.
Kind	Задаёт или возвращает вид подчиненного задания.
KindSettings	Задаёт или возвращает настройки дочернего вида подчиненного задания.

Поля

Имя	Описание
CreatableProperty	Определяет свойство "Создаваемый".
KindProperty	Определяет свойство "Вид".

TaskPresetDelegate — класс

Предоставляет параметры выбора делегата для задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskPresetDelegate : BaseCardSectionRow
```

Свойства

Имя	Описание
Employee	Задаёт или возвращает сотрудника, которому возможно делегировать задание.
Group	Задаёт или возвращает группу сотрудников, которой возможно делегировать задание.
Role	Задаёт или возвращает роль, которой возможно делегировать задание.
SearchWord	Задаёт или возвращает идентификатор поискового слова, по которому возможно делегировать задание.
Unit	Задаёт или возвращает подразделение, которому возможно делегировать задание.

Поля

Имя	Описание
EmployeeProperty	Определяет свойство "Сотрудник".

Имя	Описание
GroupProperty	Определяет свойство "Группа".
RoleProperty	Определяет свойство "Роль".
SearchWordProperty	Определяет свойство "Поисковое слово".
UnitProperty	Определяет свойство "Подразделение".

TaskSelectedPerformer — класс

Класс TaskSelectedPerformer представляет выбранного исполнителя задания.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public class TaskSelectedPerformer : BaseCardSectionRow
```

Свойства

Имя	Описание
Employee	Задаёт или возвращает сотрудника, являющегося выбранным исполнителем задания.
Group	Задаёт или возвращает группу сотрудников, содержащую выбранных исполнителей задания.
Role	Задаёт или возвращает роль сотрудников, в которую входят выбранные исполнители задания.
SearchWord	Задаёт или возвращает поисковое слово, по которому определяются выбранные исполнители задания.

Имя	Описание
Unit	Задаёт или возвращает подразделение/организацию, которое содержит выбранных исполнителей задания.

Поля

Имя	Описание
EmployeeProperty	Определяет свойство "Сотрудник".
GroupProperty	Определяет свойство "Группа".
RoleProperty	Определяет свойство "Роль".
SearchWordProperty	Определяет свойство "Поисковое слово".
UnitProperty	Определяет свойство "Подразделение".

TaskTreeInfo — класс

Класс TaskTreeInfo возвращает информацию из узла дерева заданий, полученного из списка заданий.

- **Пространство имён:** DocsVision.BackOffice.ObjectModel
- **Сборка:** DocsVision.BackOffice.ObjectModel.dll

Синтаксис

```
public sealed class TaskTreeInfo
```

Конструкторы

Имя	Описание
TaskTreeInfo()	Инициализирует новый экземпляр класса TaskTreeInfo.
TaskTreeInfo(Guid)	Инициализирует новый экземпляр класса TaskTreeInfo с помощью указанного значения.

Свойства

Имя	Описание
Author	Возвращает отображаемое имя автора задания.
ChildTasks	Возвращает список дочерних узлов.
CreationDate	Возвращает дату создания задания.
CurrentPerformers	Возвращает список текущих исполнителей задания.
CurrentPerformersDescription	Возвращает текстовое описание для списка текущих исполнителей задания.
Delegates	Возвращает список делегатов задания.
EndDate	Возвращает базовую дату окончания задания.
EndDateActual	Возвращает действительную дату окончания задания.
ExecutionType	Возвращает вариант исполнения для группы заданий.
Id	Возвращает идентификатор задания/группы заданий.
IsTaskGroup	Возвращает признак того, что данный узел дерева заданий является группой заданий.
KindId	Возвращает идентификатор вида задания/группы заданий.
Name	Возвращает название элемента.
OnControl	Возвращает признак того, что задание находится на контроле.
Performers	Возвращает список исполнителей задания.
PerformersDescription	Возвращает текстовое описание для списка исполнителей задания.

Имя	Описание
Priority	Возвращает приоритет исполнения задания.
Responsible	Возвращает отображаемое имя ответственного исполнителя.
StartDate	Возвращает базовую дату начала выполнения задания.
StartDateActual	Возвращает действительную дату начала выполнения задания.
State	Возвращает отображаемое название текущего состояния задания.
StateId	Возвращает идентификатор текущего состояния задания.

Методы

Имя	Описание
Load(TaskTreeInfo)	Загружает данные в экземпляр класса TaskTreeInfo.

DocsVision.BackOffice.WinForms — пространство имён

Данное пространство имён предоставляет коллекцию стандартных элементов управления и набор классов для работы с разметками.

Пространства имён

Пространство имён	Описание
DocsVision.BackOffice.WinForms.Controls	Содержит стандартные компоненты приложения *Базовые объекты*, основанные на элементах управления.
DocsVision.BackOffice.WinForms.Design	Пространство имён DocsVision.BackOffice.WinForms.Design содержит методы управления разметкой.

Классы

Класс	Описание
`ActiveChildCardEventArgs`	Предоставляет данные для события поиска открытой дочерней карточки.
`BarLocalizer`	Возвращает локализованное название указанной строки состояния.
`BaseCardControl`	Базовый класс карточки, поддерживающей механизм настройки разметки (при помощи *конструктора разметок*) и автомат состояний. Является базовым классом для карточек библиотеки *Базовые объекты*.
`CardAfterCreationModeAttribute`	Задаёт описание метода скрипта, вызываемого после создания карточки.
`CardCreationByModeConstants`	Предоставляет идентификаторы режимов создания карточек.
`CardCreationInitiatedByAttribute`	Задаёт описание метода скрипта; определяет типы инициирующих документов.
`CardCreationModeAttribute`	Задаёт описание метода скрипта; определяет режим создания карточки.
`CardDigestEventArgs`	Предоставляет данные для события генерации дайджеста.
`CardFrameForm`	Предоставляет доступ к данным окна карточки.
`CardSavingEventArgs`	Содержит аргументы для события сохранения данных карточки.
`EditorLocalizer`	Предоставляет локализованные названия для некоторых элементов управления.
`FlexibleCollection`	Предоставляет коллекцию основанную на `System.Collections.Generic.List` с кэшем.

Класс	Описание
FlexibleReadOnlyCollection	Вариант коллекции FlexibleCollection в режиме чтения.
GridLocalizer	Предоставляет локализованные названия для элементов таблицы.
KeyValueItem	Определяет пару "ключ-значение".
ObjectContextEventArgs	Содержит аргументы события инициализации контекста объектов.
RibbonPagesEventArgs	Содержит аргументы для события получения методом GetRibbonPages массива RibbonPage.
ScriptAssemblyPack	Содержит описание сборки.
ScriptClassBase	Базовый класс для скрипта Конструктора скриптов.
ScriptEditForm	Предоставляет редактор скриптов.
StateBranchingEventArgs	Содержит аргументы для события смены состояния (описанного в Конструкторе состояний) карточки.
TreeListLocalizer	Предоставляет локализованные названия некоторых частей элемента управления типа дерево.

Интерфейсы

Интерфейс	Описание
ICardView	Определяет дополнительные методы взаимодействия с окном карточки.
ICustomizableControl	Добавляет к базовому классу карточки возможность работы с разметкой.

Перечисления

Перечисление	Описание
CollectionSortOrder	Задаёт порядок сортировки коллекции.

DocsVision.BackOffice.WinForms.Controls — пространство имён

Содержит стандартные компоненты приложения *Базовые объекты*.

Классы

Класс	Описание
`BranchCancelEventArgs`	Определяет аргументы события, предшествующего удалению перехода состояния в автомате состояний. Событие относится к конструктору состояний.
`BranchEventArgs`	Определяет аргументы для событий связанных с диаграммой состояний. Событие относится к конструктору состояний.
`CardChooseBox`	Предоставляет элемент управления, основанный на ChooseBox, и предназначенный для выбора карточек.
`CardLinksView`	Элемент просмотра ссылок карточки.
`CategoryCancelEventArgs`	Предоставляет аргумент событию `CategoryListView.BeforeCategoryChecked`.
`CategoryEventArgs`	Предоставляет аргумент событиям `CategoryListView.CategoryChecked` и `CategoryListView.CategoryUnchecked`.
`CategoryListView`	Элемент управления, предназначенный для выбора категорий карточки из дерева категорий.
`ExportDialog`	Предоставляет классу `ExportForm` элементы управления.
`ExportForm`	Класс `ExportForm` представляет форму экспорта и печати содержимого карточки, позволяет отправлять содержимое по электронной почте.

Класс	Описание
NumeratorBox	Класс элемента управления типа Нумератор.

Перечисления

Перечисление	Описание
AllowedFolderTypes	Определяет допустимые типы папок.
CardOpenMode	Определяет режим открытия карточки.
CategoryListViewMode	Определяет режим отображения списка категорий.
ItemSelectionMode	Определяет режим открытия справочника.
TaskTreeViewColumn	Определяет типы колонок для отображения в элементе управления TaskTreeView.
TaskTreeViewOperation	Определяет список операций, которые могут быть выполнены из элемента управления TaskTreeView.

ExportForm — класс

Класс ExportForm представляет форму экспорта и печати содержимого карточки, позволяет отправлять содержимое по электронной почте.

- **Пространство имён:** DocsVision.BackOffice.WinForms.Controls
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Конструкторы

Имя	Описание
Public ExportForm()	Инициализирует новый экземпляр класса ExportForm.

Свойства

Имя	Описание
`Caption`	Задаёт или возвращает тему письма, отправляемого из формы `ExportForm`.
`CardData`	Задаёт или возвращает содержимое карточки, для которой вызывается диалог экспорта и печати.
`CardHost`	Задаёт или возвращает контейнер карточки.
`CardTypeId`	Задаёт или возвращает идентификатор типа карточки. Данный параметр используется при формировании списка доступных преобразований печати и экспорта.
`CardXml`	Задаёт или возвращает содержимое произвольного XML-документа, предназначенного для вывода в диалог печати и экспорта. Используется вместо XML-документа, полученного из `CardData`.
`Session`	Задаёт или возвращает пользовательскую сессию.
`TemplateAlias`	Задаёт или возвращает псевдоним преобразования, которое будет выбрано по умолчанию.

Заметки

Класс `ExportForm` предоставляет два варианта использования:

1. Используется `CardData` — диалог вызывается для существующей карточки. Параметры `CardXml`, `CardTypeId`, `Caption` указывать не требуется.
2. Используется `CardXml` — диалог вызывается для печати переданной Xml-строки. Параметр `CardData` указывать не требуется. Данный параметр доступен начиная с версии платформы 5.4.

При печати собственного XML-документа список доступных пребразований печати и экспорта будет ограничен преобразованиями имеющими тип `Custom` (см.

раздел Раздел 'Transformations').

Не рекомендуется задавать значения `CardData` и `CardXml` одновременно, так как при этом возможно формирование некорректного списка преобразований.

ItemSelectionMode — перечисление

Определяет режим открытия справочника.

- **Пространство имён:** `DocsVision.BackOffice.WinForms.Controls`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
public enum ItemSelectionMode
```

Члены

Имя члена	Описание
`Positioning`	Позиционирование на записи. Значение `0`.
`OpenForEdit`	Открытие записи в режиме редактирования. Значение `1`.
`OpenForRead`	Открытие записи в режиме чтения. Значение `2`.

Заметки

Элемент данного перечисления передается в массиве параметров активации открытия карточки и определяет режим открытия указанной записи.

NumeratorBox — класс

Класс элемента управления типа `Нумератор`.

- **Пространство имён:** `DocsVision.BackOffice.WinForms.Controls`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Конструкторы

Имя	Описание
NumeratorBox()	Инициализирует экземпляр класса NumeratorBox.

Свойства

Имя	Описание
ManualEditAllowed	Задаёт или возвращает признак, разрешающий ручной ввод номера в нумераторе.
ObjectContext	Задаёт или возвращает контекст объектов.
Rule	Задаёт или возвращает идентификатор правила нумерации, используемого элементом управления.

Методы

Имя	Описание
Allocate	Выделяет номер из связанного нумератора.
ManualEdit	Вызывает форму ручного ввода номера в нумераторе
Release	Освобождает выделенный номер.

Примеры

Далее приведён пример получения из кода элемента управления типа *Нумератор* с последующим выделением номера.

```
private void NumeratorAllocate_Click(System.Object sender, System.EventArgs e)
{
 ICustomizableControl customizableControl = CardControl;
 NumeratorBox numeratorBox = customizableControl.FindPropertyItem<NumeratorBox
>("НомерДокумента");
 numeratorBox.Allocate();
}
```

NumeratorBox.Rule — свойство

Задаёт или возвращает идентификатор правила нумерации, используемого элементом управления.

- **Пространство имён:** DocsVision.BackOffice.WinForms.Controls
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public Guid Rule { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор правила нумерации

TaskTreeViewColumn — перечисление

Определяет типы колонок для отображения в элементе управления TaskTreeView.

Это перечисление имеет атрибут FlagsAttribute, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** DocsVision.BackOffice.WinForms.Controls
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
[System.Flags]
public enum TaskTreeViewColumn
```

Члены

Имя члена	Описание
None	Не задано. Будут отображены все колонки. Значение 0.
Name	Название. Значение 1.
Author	Автор задания. Значение 2.
State	Состояние. Значение 4.

Имя члена	Описание
`StartDate`	Дата начала плановая. Значение `8`.
`EndDate`	Дата завершения плановая. Значение `10`.
`ActualStartDate`	Дата начала фактическая. Значение `20`.
`ActualEndDate`	Дата завершения фактическая. Значение `40`.
`Performer`	Назначенный исполнитель. Значение `80`.
`CurrentPerformer`	Текущий исполнитель. Значение `100`.
`DelegatePerformer`	Исполнитель. Значение `200`.
`DelegatedEmployee`	Делегировал. Значение `400`.
`DelegationReason`	Причина. Значение `800`.
`Return`	Признак возврата. Значение `1000`.
`Returned`	Возвращено. Значение `2000`.
`DelegationDate`	Дата делегирования. Значение `4000`.
`StateIcon`	Иконка состояния. Значение `8000`.
`ExecutionType`	Тип маршрутизации. Значение `10000`.
`Responsible`	Ответственный исполнитель. Значение `20000`.

TaskTreeViewOperation — перечисление

Определяет список операций, которые могут быть выполнены из элемента управления `TaskTreeView`.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.BackOffice.WinForms.Controls`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
[System.Flags]
public enum TaskTreeViewOperation
```

Члены

Имя члена	Описание
None	В списке отсутствую доступные операции. Значение 0.
CreateChildTask	Создание задания. Значение 1.
CopyChildTask	Копирование. Значение 2.
RemoveChildTask	Удаление задания. Значение 4.
CreateChildTaskGroup	Создание группы заданий. Значение 8.
RemoveChildTaskGroup	Удаление группы заданий. Значение 20.
CopyResultsToParenTask	Перенос результатов в родительское задание. Значение 40.

DocsVision.BackOffice.WinForms.Design — пространство имён

Пространство имён DocsVision.BackOffice.WinForms.Design содержит методы управления разметкой.

Пространства имён

Пространство имён	Описание
PropertyControls	Пространство имён PropertyControls содержит интерфейсы, реализуемые некоторыми элементами управления *Конструктора разметок*.
LayoutItems	Пространство имён LayoutItems содержит базовые сущности разметки карточки.

LayoutItems — пространство имён

Пространство имён LayoutItems содержит базовые сущности разметки карточки.

Интерфейсы

Интерфейс	Описание
ILayoutPropertyItem	Интерфейс ILayoutPropertyItem добавляет базовую функциональность элемента разметки.
IReferencePropertyItem	Интерфейс IReferencePropertyItem определяет параметры связанного поля.
IReusableLayoutItem	Интерфейс IReusableLayoutItem определяет явную поддержку в компоненте элемента управления механизма повторного использования разметки.
ITableControl	Определяет дополнительные возможности элемента управления типа *Таблица* в *Конструкторе разметок*.

ILayoutPropertyItem — интерфейс

Интерфейс ILayoutPropertyItem добавляет базовую функциональность элемента разметки.

- **Пространство имён:** DocsVision.BackOffice.WinForms.Design.LayoutItems
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public interface ILayoutPropertyItem
```

Свойства

Имя	Описание
ControlValue	Задаёт или возвращает значение элемента управления.
PropertyType	Возвращает тип элемента управления.

Методы

Имя	Описание
`Commit`	Сохраняет изменённое значение элемента управления.

Примеры

Пример скрипта карточки, который сбрасывает значение "Темы" карточки на значение по умолчанию:

```
private void SetDefaultThemeButton_Click(System.Object sender, System.EventArgs e)
{
 ICustomizableControl control = CardControl;

 ILayoutPropertyItem layoutItem = control.FindPropertyItem<ILayoutPropertyItem>("Theme"
);①

 if(layoutItem = null) return;

 MessageResult result = CardControl.ShowMessage(string.Format("Подтвердите изменение темы
{0} на значение по умолчанию", layoutItem.ControlValue),
  "Сброс темы карточки", MessageType.Question, MessageButtons.OkCancel);②

 if(result != MessageResult.Ok) return;

 layoutItem.ControlValue = "Тема по умолчанию";
 layoutItem.Commit(); ③
}
```

① Получение элемента разметки.

② Запрос подтверждения сброса темы.

③ Сохранение нового значения темы.

IReferencePropertyItem — интерфейс

Интерфейс `IReferencePropertyItem` определяет параметры связанного поля.

- **Пространство имён:** `DocsVision.BackOffice.WinForms.Design.LayoutItems`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
public interface IReferencePropertyItem
```

Свойства

Имя	Описание
CardTypeId	Возвращает идентификатор типа карточки.
SectionTypeId	Возвращает идентификатор секции карточки.
AllowUnresolved	Возвращает признак, определяющий возможность сохранения в элемент управления нераспознанного значения.

Методы

Имя	Описание
GetLinkedFieldValue(String)	Возвращает значение указанного связанного поля.

IReusableLayoutItem — интерфейс

Интерфейс IReusableLayoutItem определяет явную поддержку в компоненте элемента управления механизма повторного использования разметки.

- **Пространство имён:** DocsVision.BackOffice.WinForms.Design.LayoutItems
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public interface IReusableLayoutItem
```

Методы

Имя	Описание
OnLayoutLoad	Вызывается при первоначальной загрузке разметки.

Имя	Описание
OnLayoutReuse	Вызывается при повторном использовании разметки. Вызывается раньше, чем передается контекст в свойство ObjectContext элемента управления.

Заметки

Данный интерфейс реализуется в компоненте элемента управления, если требуется дополнительно обработать события загрузки и повторного использования разметки в элементе управления.

Реализуемые в компоненте элемента управления методы будут вызваны при первоначальной загрузке разметки и при повторном использовании соответственно. Методы будут вызваны, только если разметка поддерживает повторное использование.

Примеры

```
public partial class HtmlBrowser : PanelControl, IReusableLayoutItem
{

 // ...

 public void OnLayoutLoad() ①
 {
  cardDataExportCache.Clear();
 }

 public void OnLayoutReuse() ②
 {
  cardDataExportCache.Clear(); ③
  cardData = null;
  ReflectChanges();
 }
}
```

① Обрабатываем в элементе управления событие первой загрузки разметки. Здесь можно не выполнять никаких действий.

② Обрабатываем в элементе управления событие повторного использования (второй и последующих загрузок) разметки.

③ Здесь нужно очищать данные, которые "достались" от предыдущего использования данного компонента и могут помещать корректной работе или корректному отображению данных.

IReusableLayoutItem.OnLayoutLoad — метод

Вызывается при первоначальной загрузке разметки.

- **Пространство имён:** DocsVision.BackOffice.WinForms.Design.LayoutItems
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
void OnLayoutLoad()
```

IReusableLayoutItem.OnLayoutReuse — метод

Вызывается при повторном использовании разметки. Вызывается раньше, чем передается контекст в свойство ObjectContext элемента управления.

- **Пространство имён:** DocsVision.BackOffice.WinForms.Design.LayoutItems
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
void OnLayoutReuse();
```

Заметки

В реализации следует выполнить действия, связанные с подготовкой элемента управления к использованию в повторно использованной разметке.

ITableControl — интерфейс

Определяет дополнительные возможности элемента управления типа *Таблица* в *Конструкторе разметок*.

- **Пространство имён:** DocsVision.BackOffice.WinForms.Design.LayoutItems
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public interface ITableControl
```

Свойства

Имя	Описание
`FocusedCellIndex`	Задаёт или возвращает номер выделенной ячейки таблицы.
`FocusedRowIndex`	Задаёт или возвращает номер выделенной строки таблицы.
`FocusedRowItem`	Задаёт или возвращает выделенную строку таблицы.
`GridColumns`	Возвращает коллекцию колонок таблицы.
`Item`	Возвращает элемент таблицы с заданным номером строки.
`RowCount`	Возвращает количество строк таблицы.

Методы

Имя	Описание
`AddRow(BaseCard)`	Добавляет новую строку в таблицу.
`GetControl`	Возвращает элемент управления.
`RefreshRow(Int32)`	Обновляет строку с заданным идентификатором.
`RemoveRow(BaseCard, BaseCardProperty)`	Удаляет указанную строку.
`RemoveRow(BaseCard, Int32)`	Удаляет строку с указанным номером.

События

Имя	Описание
`CellValueChanged`	Событие вызывается при изменении значения ячейки.

PropertyControls — пространство имён

Пространство имён `PropertyControls` содержит интерфейсы, реализуемые

некоторыми элементами управления *Конструктора разметок*.

Интерфейсы

Интерфейс	Описание
`IPreviewFileControl`	Добавляет возможность предварительного просмотра файла, полученного по идентификатору, либо по местоположению.
`IPropertyControl`	Определяет наличие стандартных параметров элемента управления.

IPreviewFileControl — интерфейс

Добавляет возможность предварительного просмотра файла, полученного по идентификатору, либо по местоположению.

- **Пространство имён:** `DocsVision.BackOffice.WinForms.Design.PropertyControls`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
public interface IPreviewFileControl
```

Методы

Имя	Описание
`Preview(Guid)`	Предварительный просмотр файла с заданным идентификатором.
`Preview(String)`	Предварительный просмотр файла, расположенного на файловой системе.

События

Имя	Описание
`PreviewCompleted`	Вызывается после отображения файла в области предварительного просмотра.

IPropertyControl — интерфейс

Определяет наличие стандартных параметров элемента управления.

- **Пространство имён:** DocsVision.BackOffice.WinForms.Design.PropertyControls
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public interface IPropertyControl
```

Свойства

Имя	Описание
AllowEdit	Задаёт или возвращает признак разрешения редактирования значения элемента управления.
ControlValue	Задаёт или возвращает значение элемента управления.
ObjectContext	Задаёт или возвращает контекст объектов.
ShowBorder	Задаёт или возвращает признак отображения границы элемента управления.
Signed	Задаёт или возвращает признак того, что поле, соответствующее элементу управления, является подписанным.
TabIndex	Задаёт или возвращает последовательность перехода элемента управления.
TabStop	Задаёт или возвращает признак возможности перевода курсора на элемент управления по TAB.
ToolTip	Задаёт или возвращает всплывающую подсказку для элемента управления.

Методы

Имя	Описание
`Commit`	Сохраняет изменение значения элемента управления. Без вызова данного метода изменение значения через объектную модель зафиксировано не будет.
`GetDefaultControlValue`	Возвращает значение по умолчанию для элемента управления. Необходимо выполнить проверку на значение null, т.к. для большинства элементов управление данное значение не задано.
`GetValueCopy`	Возвращает значение элемента управления.

События

Имя	Описание
`ControlValueChanged`	Событие вызывается после изменения значения элемента управления.

Заметки

Изменение значения элемента управления, выполненное из объектной модели, должно быть зафиксированы вызовом метода `Commit`. Если фиксация изменений произведена не будет, то пользователь не получит запроса на сохранение изменений в карточке при её закрытии.

BaseCardControl — класс

Базовый класс карточки, поддерживающей механизм настройки разметки (при помощи *конструктора разметок*) и автомат состояний. Является базовым классом для карточек библиотеки *Базовые объекты*.

- **Пространство имён:** `DocsVision.BackOffice.WinForms`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
[CardFrameWindowType(typeof(CardFrameForm))]
[ComVisible(true)]
[Customizable]
[Guid("16F0201F-589F-465c-9721-F6A571EB9512")]
public class BaseCardControl : CardControl, ICustomizableControl, ICardView
```

Конструкторы

Имя	Описание
BaseCardControl()	Инициализирует новый экземпляр класса BaseCardControl.

Свойства

Имя	Описание
AvailableBranches	Возвращает список доступных переходов состояний карточки.
BaseObject	Возвращает объектную модель открытой карточки.
Caption	Задаёт или возвращает заголовок окна в котором открыта карточка.
ObjectContext	Возвращает текущий контекст объектов.
ReadOnly	Возвращает true, если карточка открыта на чтение.

Методы

Имя	Описание
ChangeState(StatesStateMachineBranch)	Изменяет состояние карточки следуя предоставленному переходу автомата состояний.
Commit	Сохраняет данные карточки.
HasChanges	Возвращает признак наличия не сохранённых изменений в контексте объектов.

Имя	Описание
ReloadLayout(Boolean)	Перезагружает разметку карточки.
Rollback	Отменяет несохранённые изменения данных карточки.
Save	Сохраняет данные карточки в случае отсутствия в них ошибок.
ValidateProperties	Проверяет данные карточки на наличие ошибок. При наличии ошибок будет выдано соответствующее сообщение.
ValidateProperties(Boolean)	Проверяет данные карточки на наличие ошибок с возможностью отключения сообщения о наличии ошибок.
ValidateProperties(Boolean, IEnumerable<string>, out IEnumerable<string>, out IEnumerable<string>)	Проверяет данные карточки на наличие ошибок с возможностью отключения проверки отдельных данных.

События

Имя	Описание
AfterConvertToTemplate	Событие возникает после конвертации карточки в шаблон.
AfterCreateFromTemplate	Событие возникает после создания карточки из шаблона.
BeforeConvertToTemplate	Событие возникает перед конвертацией карточки в шаблон.
BeforeCreateFromTemplate	Событие возникает перед созданием карточки из шаблона.
CardActivatedAsync	Событие возникает при открытии карточки. Может быть использовано для вынесения бизнес-логики в отдельный поток.

Имя	Описание
`CardActivatedAsyncCompleted`	Событие возникает после завершения исполнения бизнес-логики, привязанной к событию CardActivatedAsync. Может быть использовано для обновления интерфейса пользователя.
`GeneratingDigest`	Событие возникает перед генерацией дайджеста.
`GettingRibbonPages`	Событие возникает перед возвращением результата выполнения метода `ICardView.GetRibbonPages`.
`LayoutChanged`	Событие возникает после смены разметки.
`ObjectContextInitializing`	Событие возникает при инициализации контекста объектов, запущенной обращением к не заданному свойству `base.ObjectContext`.
`QueryActiveChildCard`	Событие возникает перед возвращением результата выполнения метода `ICardView.GetActiveChildCard`. Позволяет переопределить активную дочернюю карту.
`Saved`	Событие возникает после сохранения данных карточки.
`Saving`	Событие возникает перед сохранением данных карточки. Позволяет отменить сохранение, установкой `Cancel` в значение `true`.
`StateBranching`	Событие возникает перед выполнением перехода автомата состояний.
`StateChanged`	Событие возникает после смены состояния карточки.

BaseCardControl — конструктор

Инициализирует новый экземпляр класса `BaseCardControl`.

- **Пространство имён:** `DocsVision.BackOffice.WinForms`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
public BaseCardControl()
```

Заметки

Перед инициализацией компонента осуществляется проверка разрешения на запуск (лицензия).

BaseCardControl.AvailableBranches — свойство

Возвращает список доступных переходов состояний карточки.

- **Пространство имён:** `DocsVision.BackOffice.WinForms`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
public IEnumerable<StatesStateMachineBranch> AvailableBranches { get; }
```

Значение свойства

Тип: `System.Collections.Generic.IEnumerable<StatesStateMachineBranch>`

Коллекция переходов автомата состояний `StatesStateMachineBranch`

Заметки

Коллекция формируется на основе элементов выпадающего меню Windows-клиента `stateMachinePopupMenu` (**Действие/Выполнить**).

BaseCardControl.BaseObject — свойство

Возвращает объектную модель открытой карточки

- **Пространство имён:** `DocsVision.BackOffice.WinForms`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
public virtual BaseCard BaseObject { get; }
```

Значение свойства

Тип: BaseCard

Карточка

Заметки

Компонент карточки, унаследованный от BaseCardControl, должен переопределить данное свойство и вернуть объектную модель самой карточки, из которой можно будет получить доступ непосредственно к данным карточки.

BaseCardControl.Caption — свойство

Задаёт или возвращает заголовок окна в котором открыта карточка.

- **Пространство имён:** DocsVision.BackOffice.WinForms
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public string Caption { get; set; }
```

Значение свойства

Тип: System.String

Заголовок окна

Заметки

Данное свойство никоим образом не влияет на содержимое самой карточки. При повторном открытии заголовок будет установлен в соответствии с данными карточки.

BaseCardControl.ChangeState — метод (StatesStateMachineBranch)

Изменяет состояние карточки следуя предоставленному переходу автомата состояний.

- **Пространство имён:** DocsVision.BackOffice.WinForms

- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public void ChangeState(StatesStateMachineBranch branch)
```

Параметры

branch

Тип: StatesStateMachineBranch

Переход автомата состояний

Исключения

Исключение	Условие
System.InvalidOperationException	Ошибка возвращается в случае, если не задан параметр baseCard или kindsCardProcess.

Заметки

При смене статуса не происходит сохранение карточки — нужно выполнить сохранение самостоятельно.

Примеры

Ниже приведем пример скрипта СУБП, выполняющего смену статуса карточки на "Согласован"

```
using System.Collections.Generic;
using System.Linq;
using DocsVision.BackOffice.ObjectModel.Services;

private void Button_Click(System.Object sender, System.EventArgs e)
{
 IList<StatesStateMachineBranch> statesStateMachineBranch = CardControl.ObjectContext
.GetService<IStateService>().GetStateMachineBranches(BaseObject.SystemInfo.CardKind); ①

 StatesStateMachineBranch statesStateMachineBranchLines = statesStateMachineBranch.Where
(t => t.StartState = base.BaseObject.SystemInfo.State
  && t.BranchType = StatesStateMachineBranchBranchType.Line
  && t.EndState.DefaultName.Equals("Is approved")).FirstOrDefault(); ②

 this.CardControl.ChangeState(statesStateMachineBranchLines); ③
```

```
}
```

① Получение всех значений автомата состояний для карточки.

② Выбирается первый переход с начальным статусом соответствующем текущему статусу карточки, и названием "Согласован".

③ Смена статуса.

BaseCardControl.Commit — метод

Сохраняет данные карточки.

- **Пространство имён:** `DocsVision.BackOffice.WinForms`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
public virtual void Commit()
```

Заметки

Метод `Commit` сохраняет все не сохранённые изменения в контексте объектов, а также генерирует дайджеста карточки. Проверка данных не выполняется.

BaseCardControl.HasChanges — метод

Возвращает признак наличия не сохранённых изменений в контексте объектов.

- **Пространство имён:** `DocsVision.BackOffice.WinForms`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
public virtual bool HasChanges()
```

Возвращаемое значение

Тип: `System.Boolean`

`true` — есть не сохранённые изменения, иначе — `false`

Заметки

Данный метод может быть вызван, к примеру, перед сохранением карточки,

если пользователь должен быть предупреждён об отсутствии или наличии изменений в карточке.

BaseCardControl.ReloadLayout — метод (Boolean)

Перезагружает разметку карточки.

- **Пространство имён:** DocsVision.BackOffice.WinForms
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public virtual void ReloadLayout(bool refreshSize = true)
```

Параметры

refreshSize

Тип: System.Boolean

true — при перезагрузке разметки будут вычислены новые размеры контейнера карточки, иначе — false

Заметки

Данный метод может быть вызван, к примеру, после смены состояния карточки программным способом для приведения разметки к виду, соответствующему новому состоянию.

BaseCardControl.Rollback — метод (Boolean)

Отменяет несохранённые изменения данных карточки.

- **Пространство имён:** DocsVision.BackOffice.WinForms
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public virtual void Rollback()
```

Заметки

Если отменяются изменения, поступившие от элементов управления карточки, то значения, отображаемые в данных элементах управления, сохраняться, но не будут учтены при сохранении карточки.

BaseCardControl.Save — метод

Проверяет данные карточки на корректность и сохраняет её в случае отсутствия ошибок.

- **Пространство имён:** DocsVision.BackOffice.WinForms
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public virtual bool Save()
```

Возвращаемое значение

Тип: System.Boolean

true — сохранение прошло успешно, иначе — false

Заметки

Данный метод может быть вызван из скрипта карточки, и по своему действию эквивалентен нажатию кнопки **Сохранить** в карточке.

BaseCardControl.ValidateProperties — метод

Проверяет данные карточки на наличие ошибок. При наличии ошибок будет выдано соответствующее сообщение.

- **Пространство имён:** DocsVision.BackOffice.WinForms
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public bool ValidateProperties()
```

Возвращаемое значение

Тип: System.Boolean

true — ошибки отсутствуют, иначе — false

BaseCardControl.ValidateProperties — метод (Boolean)

Проверяет данные карточки на наличие ошибок с возможностью отключения сообщения о наличии ошибок.

- **Пространство имён:** DocsVision.BackOffice.WinForms
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public bool ValidateProperties(bool showMessage)
```

Параметры

showMessage

Тип: System.Boolean

false — не отображать сообщение о наличии ошибок, иначе — true

Возвращаемое значение

Тип: System.Boolean

true — ошибки отсутствуют, иначе — false

BaseCardControl.ValidateProperties — метод (Boolean, IEnumerable<string>, out IEnumerable<string>, out IEnumerable<string>, out IEnumerable<string>)

Проверяет данные карточки на наличие ошибок с возможностью отключения проверки отдельных данных.

- **Пространство имён:** DocsVision.BackOffice.WinForms
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public virtual bool ValidateProperties(bool showMessage, IEnumerable<string>
excludedProperties, out IEnumerable<string> requiredFields, out IEnumerable<string>
notResolvedFields, out IEnumerable<string> notResolvedTables)
```

Параметры

showMessage

Тип: System.Boolean

false — не отображать сообщение о наличии ошибок, иначе — true

excludedProperties

Тип: System.Collections.Generic.IEnumerable<string>

Список свойств карточки, которые должны быть исключены из проверки. Под названиями свойств карточки подразумеваются названия соответствующих элементов управления.

requiredFields

Тип: `System.Collections.Generic.IEnumerable<string>`

Получает список свойств, требующих обязательного заполнения. Из списка исключаются свойства, перечисленные в `excludedProperties`.

notResolvedFields

Тип: `System.Collections.Generic.IEnumerable<string>`

Получает список свойств, которые из-за ошибки не могут быть связаны с данными карточки. Из списка исключаются свойства, перечисленные в `excludedProperties`. Данный параметр может сигнализировать, к примеру, о некорректно введённом значении в элемент управления типа `Сотрудник`.

notResolvedTables

Тип: `System.Collections.Generic.IEnumerable<string>`

Получает список таблиц, которые из-за ошибки не могут быть связаны с данными карточки. Из списка исключаются таблицы, перечисленные в `excludedProperties`.

Возвращаемое значение

Тип: `System.Boolean`

`true` — ошибки отсутствуют, иначе — `false`

CardFrameForm — класс

Предоставляет доступ к данным окна карточки.

- **Пространство имён:** `DocsVision.BackOffice.WinForms`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Конструкторы

Имя	Описание
`CardFrameForm()`	Инициализирует экземпляр класса CardFrameForm.

ScriptClassBase — класс

Базовый класс для скрипта Конструктора скриптов.

- **Пространство имён:** `DocsVision.BackOffice.WinForms`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
public class ScriptClassBase
```

Конструкторы

Имя	Описание
`ScriptClassBase()`	Инициализирует новый экземпляр класса ScriptClassBase.

Свойства

Имя	Описание
`BaseObject`	Возвращает карточку из которой вызван скрипт.
`CardControl`	Возвращает компонент карточки
`CardData`	Возвращает данные карточки
`CardFrame`	Возвращает окно карточки.
`Session`	Возвращает сессию пользователя.

Методы

Имя	Описание
`RefreshControls`	Обновляет свойства элементов управления.

ICardView — интерфейс

Определяет дополнительные методы взаимодействия с окном карточки.

- **Пространство имён:** `DocsVision.BackOffice.WinForms`
- **Сборка:** `DocsVision.BackOffice.WinForms.dll`

Синтаксис

```
public interface ICardView
```

Свойства

Имя	Описание
SaveChildCardOnClosing	Задаёт или возвращает признак принудительного сохранения подчиненных карточек при их закрытии.

Методы

Имя	Описание
ActivateChildCard(Guid)	При переопределении в производном классе активирует подчиненную карточку с указанным идентификатором.
GetActivatedChildCards	Возвращает массив активированных подчиненных карточек.
GetActiveChildCard	При переопределении в производном классе возвращает объект типа BaseCardControl, представляющий активную дочернюю карточку.
GetChildCardIds	Возвращает массив идентификаторов подчиненных карточек.
GetRibbonPages	При переопределении в производном классе возвращает массив страниц ленты инструментов.
GetSetting	При переопределении в производном классе возвращает настройки вида карточки. Требует приведения к нужному типу.
HideEmptyRibbonGroups	При переопределении в производном классе скрывает пустые группы из ленты инструментов.

Имя	Описание
HideEmptyRibbonPages(IEnumerable)	При переопределении в производном классе осуществляет поиск пустых страниц ленты инструментов, в переданном списке, и срывает их.
OnChildCardActivated(ActiveChildCardEventArgs)	Обрабатывает событие активации подчиненной карточки.
SetSize(Size, Size)	При переопределении в производном классе задаёт размеры окна.

События

Имя	Описание
ChildCardActivated	Событие возникает после активации компонента карточки.

Заметки

Данный интерфейс реализуется классом DocsVision.BackOffice.WinForms.BaseCardControl.

Под активацией карточки подразумевается активация компонента карточки, которая происходит при открытии карточки. При закрытии карточки её компонент остаётся активированным, т.е. в списке, возвращаемом методом GetActivatedChildCards будут присутствовать в т.ч. карточки, которые были закрыты.

ICustomizableControl — интерфейс

Добавляет к базовому классу карточки возможность работы с разметкой.

- **Пространство имён:** DocsVision.BackOffice.WinForms
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
public interface ICustomizableControl
```

Свойства

Имя	Описание
CardData	Возвращает данные карточки.
LayoutControl	Возвращает разметку карточки.

Методы

Имя	Описание
FindLayoutItem(String)	Осуществляет поиск элемента разметки с указанным именем.
FindPropertyItem<T>(String)	Осуществляет поиск свойства указанного типа T с заданным названием.
PrepareDesignMode	Переводит окно карточки в режим конструктора. Отключает элементы управления расположенные на ленте инструментов.
RefreshLayoutItems(IEnumerable<LayoutsProperty>)	Обновляет свойства элементов представления, полученных из переданного списка.

ICustomizableControl.FindPropertyItem<T> — метод (String)

Осуществляет поиск свойства указанного типа T с заданным названием.

- **Пространство имён:** DocsVision.BackOffice.WinForms
- **Сборка:** DocsVision.BackOffice.WinForms.dll

Синтаксис

```
T FindPropertyItem<T>(string propertyName)
```

Параметры типа

T

Тип искомого свойства

Параметры

propertyName

Тип: System.String

Название искомого свойства

Возвращаемое значение

Тип: T

Искомое свойство

DocsVision.BackOffice.Xml — пространство имён

Данное пространство содержит подпространство имён с классами обеспечивающими сериализацию и десериализацию XML-объектов.

Пространства имён

Пространство имён	Описание
Schemas	Содержит объектную модель схем XML-документов, представляющих сущности приложения *Базовые объекты*.

DocsVision.BackOffice.Xml.Schemas — пространство имён

Содержит объектную модель схем XML-документов, представляющих сущности приложения *Базовые объекты*.

Классы

Класс	Описание
SearchParameterInfo	Класс SearchParameterInfo представляет свойства поля, предназначенного для формирования поискового запроса.
SearchParametersInfo	Класс SearchParametersInfo представляет схему сущности, содержащей параметры поиска объекта в базе Docsvision по полям.

SearchParameterInfo — класс

Класс `SearchParameterInfo` представляет свойства поля, предназначенного для формирования поискового запроса.

- **Пространство имён:** `DocsVision.BackOffice.Xml.Schemas`
- **Сборка:** `DocsVision.BackOffice.Xml.Schemas.dll`

Синтаксис

```
[Serializable]
[DebuggerStepThrough]
[DesignerCategory("code")]
[GeneratedCode("xsd", "2.0.50727.3038")]
[XmlType(Namespace = "http://schemas.docsvision.com/BackOffice/2014-08-
01/SearchParameterInfo")]
public class SearchParameterInfo
```

Конструкторы

Имя	Описание
`SearchParameterInfo`	Инициализирует новый экземпляр класса SearchParameterInfo.

Свойства

Имя	Описание
`Name`	Задаёт или возвращает название поля.
`Type`	Задаёт или возвращает тип поля.
`Value`	Задаёт или возвращает значение поля.

SearchParametersInfo — класс

Класс `SearchParametersInfo` представляет схему сущности, содержащей параметры поиска объекта в базе Docsvision по полям.

- **Пространство имён:** `DocsVision.BackOffice.Xml.Schemas`
- **Сборка:** `DocsVision.BackOffice.Xml.Schemas.dll`

Синтаксис

```
[Serializable]
[DebuggerStepThrough]
[DesignerCategory("code")]
[GeneratedCode("xsd", "2.0.50727.3038")]
[XmlRoot(Namespace = "http://schemas.docsvision.com/BackOffice/2014-08-
01/SearchParameterInfo", IsNullable = false)]
[XmlType(Namespace = "http://schemas.docsvision.com/BackOffice/2014-08-
01/SearchParameterInfo")]
public class SearchParametersInfo
```

Конструкторы

Имя	Описание
SearchParametersInfo	Инициализирует новый экземпляр класса SearchParametersInfo.

Свойства

Имя	Описание
SearchParameterInfo	Задаёт или возвращает массив параметров, представленных в виде схемы XML, поиска объекта в базе Docsvision по полям.

SearchParametersInfo.SearchParameterInfo — свойство

Задаёт или возвращает массив параметров, представленных в виде схемы XML, поиска объекта в базе Docsvision по полям.

- **Пространство имён:** DocsVision.BackOffice.Xml.Schemas
- **Сборка:** DocsVision.BackOffice.Xml.Schemas.dll

Синтаксис

```
[XmlElement("SearchParameterInfo")]
public SearchParameterInfo[] SearchParameterInfo { get; set; }
```

Значение свойства

Тип: SearchParameterInfo[]

Свойства и значение искомого поля

DocsVision.DocumentsManagement — пространство имён

Пространство имён `DocsVision.DocumentsManagement` содержит описание сервиса для отправки согласования.

Пространства имён

Пространство имён	Описание
DocsVision.DocumentsManagement.ObjectModel	Пространство имён `DocsVision.DocumentsManagement.ObjectModel` содержит описание сервиса для отправки согласования.

DocsVision.DocumentsManagement.ObjectModel — пространство имён

Пространство имён `DocsVision.DocumentsManagement.ObjectModel` содержит описание сервиса для отправки согласования.

Интерфейсы

Интерфейс	Описание
IReconcileService	Определяет методы сервиса для работы с согласованиями.

IReconcileService — интерфейс

Определяет методы сервиса для работы с согласованиями.

- **Пространство имён:** `DocsVision.DocumentsManagement.ObjectModel`
- **Сборка:** `DocsVision.DocumentsManagement.ObjectModel.dll`

Синтаксис

```
public interface IReconcileService
```

Методы

Имя	Описание
CreateReconciliationCard(Guid, KindsCardCreationSetting)	Создаёт согласование для документа.

Имя	Описание
`FillFiles(Document, BaseCard)`	Копирует вложенные в инициирующий документ файлы в согласование.
`HandleDocumentAfterReconcileCreated(Document, CardData)`	Запускает согласование и обновляет данные документа после создания согласования.

Заметки

Данный сервис имеет две реализации, в сборках:

- `DocsVision.DocumentsManagement.ObjectModel.dll` — реализует базовую функциональность согласования, использовавшеюся в Управлении документами.
- `DocsVision.ApprovalDesigner.ObjectModel.dll` — реализация функциональность согласования, использующуюся в Конструкторе согласований; реализация основана на реализации согласования для УД.

Примеры

См. пример.

IReconcileService.HandleDocumentAfterReconcileCreated — метод (Guid, KindsCardCreationSetting)

Создаёт согласование для документа.

- **Пространство имён:** `DocsVision.DocumentsManagement.ObjectModel`
- **Сборка:** `DocsVision.DocumentsManagement.ObjectModel.dll`

Синтаксис

```
Guid CreateReconciliationCard(Guid initialCardId, KindsCardCreationSetting
cardCreationSetting)
```

Параметры

initialCardId

Тип: `System.Guid`

Идентификатор карточки, инициирующей согласование

cardCreationSetting

Тип: KindsCardCreationSetting

Настройки режима создания согласования

Возвращаемое значение

Тип: System.Guid Идентификатор созданного согласования

IReconcileService.FillFiles — метод (Document, BaseCard)

Копирует вложенные в инициирующий документ файлы в согласование.

- **Пространство имён:** DocsVision.DocumentsManagement.ObjectModel
- **Сборка:** DocsVision.DocumentsManagement.ObjectModel.dll

Синтаксис

```
void FillFiles(Document initialDocument, BaseCard reconciliationCard)
```

Параметры

initialDocument

Тип: Document

Инициирующий документ, из которого будут скопированы файлы

reconciliationCard

Тип: BaseCard

Карточка согласования

Заметки

Данный метод вызывается автоматически, при создании согласования, и не требуется отдельного вызова. Может быть переопределен при реализации собственного механизма согласования.

IReconcileService.HandleDocumentAfterReconcileCreated — метод (Document, CardData)

Запускает согласование и обновляет данные документа после создания согласования.

- **Пространство имён:** DocsVision.DocumentsManagement.ObjectModel

- **Сборка:** `DocsVision.DocumentsManagement.ObjectModel.dll`

Синтаксис

```
void HandleDocumentAfterReconcileCreated(Document document, CardData
reconciliationCardData)
```

Параметры

document

Тип: `Document`

Инициирующий документ

reconciliationCardData

Тип: `CardData`

Данный карточки согласования

Заметки

Данный метод должен быть вызван после создания согласования методом `CreateReconciliationCard`.

DocsVision.Platform — пространство имён

В пространстве имён `DocsVision.Platform` содержаться классы реализующие API уровня ядра Docsvision.

Пространства имён

Пространство имён	Описание
`DocsVision.Platform.CardHost`	В пространстве имён `DocsVision.Platform.CardHost` содержаться классы и интерфейсы, относящиеся к взаимодействию с контейнером карточек.
`DocsVision.Platform.Data`	Содержит набор классов для взаимодействия с базами данных.

Пространство имён	Описание
DocsVision.Platform.Extensibility	Пространство имён DocsVision.Platform.Extensibility содержит набор классов и интерфейсов расширения функциональности Windows-клиента.
DocsVision.Platform.ObjectManager	Пространство имён DocsVision.Platform.ObjectManager предоставляет классы и интерфейсы, декларирующие API бизнес-логики Docsvision.
DocsVision.Platform.ObjectModel	Пространство имён DocsVision.Platform.ObjectModel содержит классы определяющие фабрики сервисов и преобразователей данных, сущности хранящие данные контекста объектов, а также вспомогательные классы и интерфейсы. Эти классы составляют объектную модель высокого уровня.
DocsVision.Platform.Security	В пространстве имён DocsVision.Platform.Security содержаться объекты связанные с настройкой и реализацией модели безопасности и криптографии Docsvision.
DocsVision.Platform.StorageServer	Пространство имён DocsVision.Platform.StorageServer содержит основные интерфейсы сервера приложений Docsvision, набор прокси-классов для интерфейса сервера IStorageServer, набор генераторов и интерфейсов к базе данных, а также реализацию логики сервера приложений.

Пространство имён	Описание
DocsVision.Platform.SystemCards	Пространство имён DocsVision.Platform.SystemCards содержит реализацию объектов бизнес-логики для системных карточек.
DocsVision.Platform.WinForms	Пространство имён с набором базовых классов и элементов управления WinForms.
DocsVision.Platform.Wpf	Пространство имён DocsVision.Platform.Wpf содержит реализации некоторых базовых классов из DocsVision.Platform.WinForms для разработки пользовательских интерфейсов с использованием WPF.

Классы

Класс	Описание
AssemblyResolver	Класс AssemblyResolver реализует функцию загрузки сборок Docsvision из каталога установки клиента Docsvision в runtime программы.

DocsVision.Platform.CardHost — пространство имён

В пространстве имён DocsVision.Platform.CardHost содержаться классы и интерфейсы, относящиеся к взаимодействию с контейнером карточек.

Классы

Класс	Описание
CardEventArgs	Предоставляет данные для различных событий карточек.
CardFrame	Содержит методы доступа к параметрам окна карточки и методы управления этим окном.
CardHost	Реализует механизм доступа к контейнеру карточек.

Класс	Описание
NavFolderType	Представляет тип папки для Windows-клиента.
ParameterCollection	Представляет хранилище коллекции параметров с возможностью выбора элемента по псевдониму.
RefreshEventArgs	Предоставляет данные для события обновления.

Интерфейсы

Интерфейс	Описание
ICard	Добавляет к производному классу метод обновления карточки и её идентификатор.
ICardContainer	Описывает механизм загрузки карточки в контейнер.
ICardFrame	Описывает методы взаимодействия с окном карточки.
ICardHost	Описывает механизм организации взаимодействия с контейнером карточки.
ICardInitialize	Определяет возможность загрузки данных карточки и сессии в контейнер.
ICardUI	Описывает низкоуровневое взаимодействие с окном карточки.
ICardUIAction	Предоставляет возможность вызова определённого действия карточки.
IEditSecurityDialog	Компонент реализующий окно настроек безопасности.
IExtensionManager	Предоставляет механизм доступа к расширениям Windows-клиента.

Интерфейс	Описание
IMessageService	Сервис IMessageService предоставляется методы для отображения сообщений в Windows-клиенте.
INavigator	Предоставляет возможность хранения идентификатора папки.
IPropertyPageFrame	Определяет механизм доступа к странице свойств карточки или папки
IPropertyPageUI	Предоставляет доступ к механизму взаимодействия с окном карточки.
IReusableCardComponent	Определяет возможность повторного использования компонента карточки.

Делегаты

Делегат	Описание
CardEventHandler(object, CardEventArgs)	Представляет метод, который будет обрабатывать события от карточек.
RefreshEventHandler(object, RefreshEventArgs)	Представляет метод, который будет обрабатывать событие обновления данных карточки.

Перечисления

Перечисление	Описание
ActionFlags	Действия, которые должен выполнить Windows-клиент после обработки события.
ActivateFlags	Дополнительные флаги активации карточки.
ActivateMode	Режим активации карточки.
EditSecurityFlags	Режим открытия редактора политики безопасности.

Перечисление	Описание
MessageButtons	Определяет наборы кнопок, отображаемых в окне сообщения, формируемого методом ShowMessage.
MessageResult	Определяет возможные результаты выбора пользователя в окне сообщения. Возвращается методом ShowMessage.
MessageType	Определяет возможные типы сообщений, формируемых методом ShowMessage.
ReusageMode	Определяет возможные режимы повторного использования компонента карточки.
SystemDialogType	Тип модального окна системного назначения.

CardEventArgs — класс

Предоставляет данные для различных событий карточек.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
public class CardEventArgs
```

Конструкторы

Имя	Описание
CardEventArgs(Guid)	Инициализирует объект CardEventArgs с указанным идентификатором карточки.
CardEventArgs(Guid, Guid)	Инициализирует объект CardEventArgs с указанным идентификатором карточки и папки.

Имя	Описание
`CardEventArgs(Guid, Guid, ActivateMode)`	Инициализирует объект `CardEventArgs` с указанным режимом активации карточки.
`CardEventArgs(Guid, Guid, ActivateMode, ActivateFlags)`	Инициализирует объект `CardEventArgs` с указанными флагами активации.

Свойства

Имя	Описание
`CardId`	Возвращает идентификатор карточки.
`FolderId`	Возвращает идентификатор папки.
`Flags`	Возвращает дополнительные флаги активации карточки.
`IsNew`	Возвращает дополнительные флаги активации карточки.
`Mode`	Возвращает режим активации карточки.

CardHost — класс

Реализует механизм доступа к контейнеру карточек.

- **Пространство имён:** `DocsVision.Platform.CardHost`
- **Сборка:** `DocsVision.Platform.CardHost.dll`

Синтаксис

```
[ClassInterface(ClassInterfaceType.None)]
[ComVisible(true)]
[ToolboxItem(false)]
public sealed class CardHost : Component, IServiceProvider, ICardHost, IDisposable
```

Конструкторы

Имя	Описание
`CardHost(UserSession)`	Инициализирует новый экземпляр класса CardHost.

Свойства

Имя	Описание
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`CreateInstance(UserSession)`	Статический метод получения экземпляр контейнера карточки для указанной сессии пользователя.
`ToIntPtr`	Возвращает внутреннее целочисленное представление объекта.

События

Имя	Описание
`CardChanged`	Событие вызывается при изменении карточки.
`CardClosed`	Событие вызывается при закрытии карточки.
`CardOpened`	Событие вызывается при открытии карточки.
`Refreshed`	Событие вызывается при обновлении данных.

Заметки

См. также описание методов интерфейса `ICardHost`.

CardHost.CreateInstance — метод (UserSession)

Статический метод получения экземпляр контейнера карточки для указанной сессии пользователя.

- **Пространство имён:** `DocsVision.Platform.CardHost`
- **Сборка:** `DocsVision.Platform.CardHost.dll`

Синтаксис

```
public static CardHost CreateInstance(UserSession session)
```

Параметры

session

Тип: UserSession

Текущая сессия

Возвращаемое значение

Тип: CardHost

Экземпляр контейнера

NavFolderType — класс

Представляет тип папки для Windows-клиента.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
public sealed class NavFolderType : INavFolderType
```

Свойства

Имя	Описание
AllowedCardTypes	Возвращает коллекцию разрешённых типов карточек.
AllowedFolderTypes	Возвращает коллекцию разрешённых типов папок.
AllowedTemplates	Возвращает коллекцию разрешённых шаблонов экспорта.
AllowedViews	Возвращает коллекцию разрешённых представлений.

Имя	Описание
`DefaultStyle`	Задаёт или возвращает стиль отображения содержимого папки по умолчанию.
`DefaultTemplateId`	Задаёт или возвращает шаблон экспорта по умолчанию.
`DefaultViewId`	Задаёт или возвращает представление папки по умолчанию.
`Flags`	Задаёт или возвращает дополнительные атрибуты папки.
`FolderCardLocation`	Задаёт или возвращает идентификатор карточки папок.
`FolderTypes`	Задаёт или возвращает дочерние типы папок.
`Icon`	Задаёт или возвращает иконку папки.
`Id`	Возвращает идентификатор папки.
`Name`	Задаёт или возвращает название.
`PropCardId`	Задаёт или возвращает идентификатор ассоциированной карточки папка.
`RefreshTimeout`	Задаёт или возвращает индивидуальный период обновления содержимого папки.
`Url`	Задаёт или возвращает URL папки.
`ViewCycleCount`	Задаёт или возвращает ограничение на размер порции данных, отображаемой в представлении одновременно.

ParameterCollection — класс

Представляет хранилище коллекции параметров с возможностью выбора элемента по псевдониму.

- **Пространство имён:** `DocsVision.Platform.CardHost`
- **Сборка:** `DocsVision.Platform.CardHost.dll`

Синтаксис

```
public sealed class ParameterCollection : IEnumerable
```

Конструкторы

Имя	Описание
ParameterCollection	Инициализирует новый экземпляр класса ParameterCollection.

Свойства

Имя	Описание
Count	Возвращает количество элементов в коллекции.
Item	Возвращает элемент коллекции с заданным индексом.
Item	Возвращает элемент коллекции с заданным псевдонимом.

Методы

Имя	Описание
Clear	Удаляет все элементы коллекции.
Contains(String)	Определяет вхождение в коллекцию элемента с заданным псевдонимом.
GetEnumerator	Возвращает перечислитель.
Remove(Int32)	Удаляет элемент коллекции с заданным индексом.
Remove(String)	Удаляет элемент коллекции с заданным псевдонимом.

RefreshEventArgs — класс

Предоставляет данные для события обновления.

- **Пространство имён:** DocsVision.Platform.CardHost

- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
public class RefreshEventArgs : EventArgs
```

Конструкторы

Имя	Описание
RefreshEventArgs(Boolean)	Инициализирует новый экземпляр класса RefreshEventArgs с возможностью полного обновления данных.

Свойства

Имя	Описание
IsFull	Возвращает признак режима полного обновление.

RefreshEventArgs — конструктор (Boolean)

Инициализирует новый экземпляр класса RefreshEventArgs с возможностью полного обновления данных.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
internal RefreshEventArgs(bool full)
```

Параметры

full

Тип:

Задаёт режим полного обновления.

Заметки

RefreshEventArgs.IsFull — свойство

Возвращает признак режима полного обновление.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
public bool IsFull { get; }
```

Значение свойства

Тип: Boolean

true — полное обновление.

ICardFrame — интерфейс

Описывает методы взаимодействия с окном карточки.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
public interface ICardFrame
```

Свойства

Имя	Описание
Caption	Задаёт или возвращает заголовок окна.
CardHost	Возвращает контейнер карточки.
Id	Возвращает идентификатор окна.
Visible	Задаёт или возвращает режим видимости окна.
Window	Возвращает объект, реализующий доступ низкоуровневым параметрам окна.

Методы

Имя	Описание
`Close`	При переопределении в производном классе немедленно закрывает окно.
`CloseAndCreateNew(Guid)`	При переопределении в производном классе закрывает окно и открывает новое в указанном режиме.
`Commit(ActionFlags)`	При переопределении в производном классе завершает работу карточки с сохранением изменений.
`FinishSelection(Object)`	Завершает выбор, если карточка была открыта на выбор методом `SelectFromCard`.
`Refresh`	При переопределении в производном классе обновляет окно.
`ShowMessage(String, String, String, MessageType, MessageButtons)`	Отображает сообщение в модальном окне.

События

Имя	Описание
`CardChanged`	Происходит после изменения данных карточки.
`CardCloseAndCreate`	Происходит при вызове метода `CloseAndCreateNew`.
`CardClosed`	Происходит при закрытии карточки.
`CardOpened`	Происходит при открытии карточки.

Данный механизм возможно использовать только при работе с Windows-клиентом.

ICardFrame.FinishSelection — метод (Object)

Завершает выбор, если карточка была открыта на выбор.

- **Пространство имён:** `DocsVision.Platform.CardHost`

- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
void FinishSelection(object selectedValue)
```

Параметры

selectedValue

Тип: System.Object

Возвращаемое значение выбора

Заметки

Примеры

В примере, некий метод, вызываемый закрытие карточки, вызывает FinishSelection с передачей результата выполнения описанного запроса. Результат передается только в случае нажатия пользователем кнопки OK, в диалоговом окне.

```
protected override void OnCardClosed(EventArgs e)
{
    if (this.DialogResult = DialogResult.OK)
    {
        this.CardFrame.FinishSelection(query.GetXML(true, null));
    }
    base.OnCardClosed(e);
}
```

ICardHost — интерфейс

Описывает механизм организации взаимодействия с контейнером карточки.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
[ServiceContract(Namespace = "http://DocsVision.Platform.CardHost")]
public interface ICardHost
```

Свойства

Имя	Описание
CardCount	Возвращает количество запущенных немодальных карточек.
ExtensionManager	Возвращает интерфейс взаимодействия с расширениями Windows-клиента.
FolderCardId	Возвращает идентификатор карточки папок.
SearchCardId	Возвращает идентификатор карточки сохранённых поисковых запросов.
ViewCardId	Возвращает идентификатор карточки представления.

Методы

Имя	Описание
ActivateFolder(Guid, Boolean)	При переопределении в производном классе производит позиционирование курсора на указанной папке.
CloseCards	При переопределении в производном классе закрывает все карточки.
CreateCard(Guid, Guid, ParameterCollection)	Создаёт карточку.
GetCardUrl(Guid, Guid)	При переопределении в производном классе формирует URL-адрес для указанной карточки.
GetFolderUrl(Guid, Guid)	При переопределении в производном классе формирует URL-адрес для указанной папки.
GetShortcutUrl(Guid, Guid, Guid)	При переопределении в производном классе формирует URL-адрес для указанного ярлыка.
Refresh(Boolean)	При переопределении в производном классе инициирует обновление данных.

Имя	Описание
SelectCard(String)	При переопределении в производном классе открывает окно выбора со списком карточек. Окно выбора будет иметь заданный заголовок.
SelectCard(String, Object)	При переопределении в производном классе открывает окно выбора с отфильтрованным списком карточек. Окно выбора будет иметь заданный заголовок.
SelectCards(String)	При переопределении в производном классе открывает окно выбора нескольких экземпляров из списка карточек. Окно выбора будет иметь заданный заголовок.
SelectCards(String, String)	При переопределении в производном классе открывает окно выбора нескольких экземпляров из отфильтрованного списка карточек. Окно выбора будет иметь заданный заголовок.
SelectFolder(String)	При переопределении в производном классе открывает окно выбора со списком папок. Окно выбора будет иметь заданный заголовок.
SelectFolder(String, Int)	При переопределении в производном классе открывает окно выбора со списком папок заданного типа. Окно выбора будет иметь заданный заголовок.
SelectFolder(String, Int32, Guid)	При переопределении в производном классе открывает окно выбора со списком папок заданного типа. Курсор будет предустановлен на заданной папке. Окно выбора будет иметь заданный заголовок.

Имя	Описание
SelectFromCard(Guid, String)	При переопределении в производном классе отображает карточку в режиме выбора строки.
SelectFromCard(Guid, String, Object)	При переопределении в производном классе отображает карточку в режиме выбора строки. Позволяет использовать дополнительные параметры активации.
ShowCard(Guid, ActivateMode)	При переопределении в производном классе отображает пользовательский интерфейс карточки в режиме контейнера.
ShowCardModal(Guid, ActivateMode)	При переопределении в производном классе отображает модально пользовательский интерфейс карточки в режиме контейнера.
ShowCard(Guid, Guid, ActivateMode, ActivateFlags, Object)	При переопределении в производном классе отображает модально пользовательский интерфейс карточки в режиме контейнера с передачей параметров активации карточки.
ToIntPtr	При переопределении в производном классе возвращает внутреннее целочисленное представление объекта.

События

Имя	Описание
CardChanged	Происходит после изменения данных карточки.
CardClosed	Происходит при закрытии карточки.
CardOpened	Происходит при открытии карточки.
Refreshed	Происходит при обновлении данных карточки.

ICardHost.ActivateFolder — метод (Guid, Boolean)

При переопределении в производном классе производит позиционирование курсора на указанной папке.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
void ActivateFolder(Guid folderId, bool showSubfolders)
```

Параметры

folderId

Тип: System.Guid

Идентификатор папки, на которой будет выполнено позиционирование курсора

showSubfolders

Тип: System.Boolean

Зарезервированный параметр

Заметки

Данный метод позволяет осуществить установку курсора на папке в приложении Windows-клиент. Может быть использован, к примеру, при реализации расширения Windows-клиента.

Примеры

Пример создания расширения Windows-клиента, использующего данный метод, приведён в разделе Позиционирование курсора на папке

ICardHost.CreateCard — метод (Guid, Guid, ParameterCollection, Boolean)

Создаёт карточку, используя переданные параметры.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
Guid CreateCard(Guid creatorEntryId, Guid initialCardInstanceId, ParameterCollection
parameters, bool showUI)
```

Параметры

creatorEntryId

Тип: System.Guid

Идентификатор режима создания карточки из *справочника видов карточек*

initialCardInstanceId

Тип: System.Guid

Идентификатор карточки, создающий карточку

parameters

Тип: ParameterCollection

Параметры активации карточки

showUI

Тип: System.Boolean

При true, после создания, карточка будет открыта на редактирование

Возвращаемое значение

Тип: System.Guid

Идентификатор созданной карточки

Заметки

Для корректной работы, метод необходимо вызвать из скрипта карточки.

Содержимое parameters зависит от типа создаваемой карточки, а точнее от принимаемых ею параметров.

ICardHost.SelectCard — метод (String)

При переопределении в производном классе открывает окно выбора со списком карточек. Окно выбора будет иметь заданный заголовок.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
Guid SelectCard(string caption)
```

Параметры

caption

Тип: System.String

Заголовок окна выбора карточки

Возвращаемое значение

Тип: System.Guid

Идентификатор выбранной карточки

ICardHost.SelectCard — метод (String, Object)

При переопределении в производном классе открывает окно выбора с отфильтрованным списком карточек. Окно выбора будет иметь заданный заголовок.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
Guid SelectCard(string caption, object selectParams)
```

Параметры

caption

Тип: System.String

Заголовок окна выбора карточки

selectParams

Тип: System.Object

Дополнительные Параметры

Возвращаемое значение

Тип: System.Guid

Идентификатор выбранной карточки

Заметки

Позиционирование на папке "Результаты поиска" (Search results) возможно в Windows-клиенте версии 5.4 и выше.

ICardHost.SelectFromCard — метод (Guid, String)

Отображение карточки в режиме выбора строки.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
object SelectFromCard(Guid cardId, string caption)
```

Параметры

cardId

Тип: System.Guid

Идентификатор карточки вызываемой

caption

Тип:

Текст выбираемой строки

Возвращаемое значение

Тип: System.Object

Выбранные строки или массив строк

ICardHost.SelectFromCard — метод (Guid, String, Object)

При переопределении в производном классе отображает карточку в режиме выбора строки. Позволяет использовать дополнительные параметры активации.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
object SelectFromCard(Guid cardId, string caption, object activateParams)
```

Параметры

cardId

Тип: System.Guid

Идентификатор открываемого справочника

caption

Тип: System.String

Заголовок окна выбора

activateParams

Тип: System.Object

Параметры активации

Возвращаемое значение

Тип: System.Object

Результат выбора

Заметки

В зависимости от параметров активации, результат выбора может представлять собой строку или массив строк.

IPropertyPageFrame — интерфейс

Определяет механизм доступа к странице свойств карточки или папки.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
public interface IPropertyPageFrame : IDisposable, IServiceProvider
```

Свойства

Имя	Описание
CardHost	Возвращает контейнер карточки свойств.
Handle	Возвращает указатель на окно.
Session	Возвращает текущую сессию пользователя.

IReusableCardComponent — интерфейс

Определяет возможность повторного использования компонента карточки.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
public interface IReusableCardComponent
```

Свойства

Имя	Описание
CanBeReused	Возвращает признак того, что компонент может быть повторно использован.
Mode	Возвращает режим повторного использования компонента карточки.

Заметки

Повторно используемый компонент кэшируется во внутреннем хранилище контейнера карточек и при следующем обращении (в т.ч. к карточке аналогичного типа) запрашивается из кэша, вместо повторного создания.

Если установлен режим Mode.OwnLogic, то контейнер карточек получает признак повторного использования компонента из переменной CanBeReused, если режим

Mode.CardHostLogic, то необходимость повторного использования компонента определяется самим контейнером карточек, исходя из внутренней логики. В режиме Mode.NoReusage компонент не кэшируется, а создаётся каждый раз повторно.

Необходимость использования данной функциональности определяется разработчиком.

IReusableCardComponent.CanBeReused — свойство

Возвращает признак того, что компонент может быть повторно использован.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
bool CanBeReused { get; }
```

Значение свойства

Тип: System.Boolean

true — компонент может быть повторно использован, иначе — false

IReusableCardComponent.Mode — свойство

Возвращает режим повторного использования компонента карточки.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
ReusageMode Mode { get; }
```

Значение свойства

Тип: ReusageMode

Режим повторного использования компонента карточки

ActionFlags — перечисление

Действия, которые должен выполнить Windows-клиент после обработки события.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.CardHost`
- **Сборка:** `DocsVision.Platform.CardHost.dll`

Синтаксис

```
[Flags]
public enum ActionFlags
```

Члены

Имя члена	Описание
`None`	Ничего не делать. Значение `0x0000`.
`ContinueAction`	Продолжить текущее действие, если этот флаг не установить — то действие будет прервано! Значение `0x0001`.
`CommittedData`	Данные карточки сохранены, можно закрывать. Значение `0x0002`.
`NoShortcuts`	Не создавать ярлык для карточки при её закрытии. Значение `0x0100`.
`WantReturn`	Передавать карточке нажатия клавиши Enter — тогда как в стандартном режиме Windows-клиент сам перехватывает эти нажатия, и использует их как сигнал к закрытию карточки. Значение `0x0200`.
`HideFrame`	Скрыть окно карточки. Значение `0x0400`.
`ChangeWantReturn`	Признак, что карточка хочет изменить поведение Enter. Значение `0x0800`.

Имя члена	Описание
WantAllKeys	Передавать карточке коды всех нажатых клавиш — тогда как в стандартном режиме Windows-клиент перехватывает нажатия системных клавиш, таких как `Tab`, `Esc` и т.д. Значение 0x1000.
ChangeWantAllKeys	Признак, что карточка хочет изменить поведение системных клавиш. Значение 0x2000.
HideStatusBar	Скрыть строку статуса в окне карточки. Значение 0x4000.

Заметки

Для побитового соединения значений используйте побитовый оператор OR, как показано в следующем примере

```
ActionFlags args = ActionFlags.NoShortcuts` | ActionFlags.CommittedData;
```

ActivateFlags — перечисление

Дополнительные флаги активации карточки.

Это перечисление имеет атрибут FlagsAttribute, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** DocsVision.Platform.CardHost
- **Сборка:** DocsVision.Platform.CardHost.dll

Синтаксис

```
[Flags]
public enum ActivateFlags
```

Члены

Имя члена	Описание
None	Флаги отсутствуют. Значение 0x00.

Имя члена	Описание
`ModalWindow`	Карточка открыта модально. Значение `0x01`.
`FolderPanel`	Карточка папки. Значение `0x02`.
`New`	Создан новый экземпляр карточки. Значение `0x04`.
`NewFromTemplate`	Новая карточка из шаблона. Значение `0x08`.
`ByShortcut`	Карточка открыта по ярлыку. Значение `0x10`.
`ByUrl`	Карточка открыта по ссылке. Значение `0x20`.
`ReadOnly`	Карточка открыта только для чтения. Значение `0x40`.
`NoCreateNew`	Запрет создания новых карточек в этой же папке. Значение `0x80`.

Заметки

Для побитового соединения значений используйте побитовый оператор `OR`, как показано в следующем примере

```
ActivateFlags args = ActivateFlags.New` | ActivateFlags.NewFromTemplate;
```

ActivateMode — перечисление

Режим активации карточки.

- **Пространство имён:** `DocsVision.Platform.CardHost`
- **Сборка:** `DocsVision.Platform.CardHost.dll`

Синтаксис

```
public enum ActivateMode
```

Члены

Имя члена	Описание
`Edit`	Режим редактирования.
`ReadOnly`	Только чтение.
`Preview`	Предварительный просмотр в Windows-клиенте.
`Select`	Открытие справочника в режиме выбора.

EditSecurityFlags — перечисление

Режим открытия редактора политики безопасности.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.CardHost`
- **Сборка:** `DocsVision.Platform.CardHost.dll`

Синтаксис

```
[Flags]
public enum EditSecurityFlags
```

Члены

Имя члена	Описание
`ShowOwner`	Отображать закладку владельца. Значение `0x00000001`.
`ShowAudits`	Отображать закладку аудита. Значение `0x00000002`.
`PermissionsReadOnly`	Отключить возможность изменения прав доступа. Значение `0x00000008`.
`ShowAdvanced`	Отображать кнопку Дополнительно. Значение `0x00000010`.

Имя члена	Описание
`ShowResetSecurity`	Предоставить возможность сброса параметров безопасности. Значение `0x00000020`.
`OwnerReadOnly`	Отключить возможность смены владельца. Значение `0x00000040`.
`ShowProperties`	Отображать свойства. Значение `0x00000080`.
`ShowResetOwnerOnChildren`	Разрешить возможность наследование владельца для дочерних элементов. Значение `0x00000100`.
`ShowPropagateFromParent`	Окно прав распространенных от родителя. Значение `0x00000200`.
`ShowNoPropagateInherit`	Окно "Не распространять наследование". Значение `0x00000400`.
`ShowResetPermissionsOnChildren`	Разрешить возможность наследование прав доступа для дочерних элементов. Значение `0x00004000`.
`ShowResetAuditsOnChildren`	Разрешить возможность наследование политик аудита для дочерних элементов. Значение `0x00008000`.
`ShowEffectivePermissions`	Отображать вкладку эффективных прав доступа. Значение `0x00020000`.
`ShowResetPermissions`	Предоставить возможность сброса прав доступа. Значение `0x00040000`.
`ShowResetAudits`	Предоставить возможность сброса политики аудита. Значение `0x00080000`.
`ShowResetOwner`	Предоставить возможность сброса владельца. Значение `0x00100000`.
`ShowSpecialPermissions`	Просмотр специальных прав. Значение `0x00200000`.
`ViewOnly`	Режим просмотра. Значение `0x00400000`.

Имя члена	Описание
`DefaultSet`	Установить начальные параметры безопасности. Значение `0x005EC333`.
`RequireElevationToEditPermissions`	Требуется повышение для изменения прав. Значение `0x01000000`.
`RequireElevationToEditAudits`	Требуется повышение для изменения политики аудита. Значение `0x02000000`.
`RequireElevationToEditOwner`	Требуется повышение для изменения владельца. Значение `0x04000000`.
`MayWritePermissions`	Полный доступ. Значение `0x10000000`.

MessageButtons — перечисление

Набор кнопок для отображения методом `ShowMessage`.

- **Пространство имён:** `DocsVision.Platform.CardHost`
- **Сборка:** `DocsVision.Platform.CardHost.dll`

Синтаксис

```
[Flags]
public enum MessageButtons
```

Члены

Имя члена	Описание
`Ok`	Ок. Значение `0x000000001`.
`DefButton1`	Кнопка по умолчанию `1`. Значение `0x000000001`.
`OkCancel`	Ок и Отмена. Значение `0x000000011`.
`AbortRetryIgnore`	Прервать, Повторить и Игнорировать. Значение `0x000000021`.
`YesNoCancel`	Да, Нет и Отмена. Значение `0x000000031`.
`YesNo`	Да, Нет. Значение `0x000000041`.

Имя члена	Описание
`RetryCancel`	Повторить, Отмена. Значение `0x000000051`.
`CancelTryContinue`	Прервать, Повторить и Продолжить. Значение `0x00000006`.
`DefButton2`	Кнопка по умолчанию `2`. Значение `0x00000100`.
`DefButton3`	Кнопка по умолчанию `3`. Значение `0x00000200`.
`BtnShowFlag`	Флаг показа. Значение `0x10000000`.
`BtnShowFlagReset`	Сброс флага показа. Значение `0x20000000`.
`BtnShowFlagDefault`	Значение флага показа по умолчанию. Значение `0x40000000`.

MessageResult — перечисление

Результат выбор пользователя после вызова `ShowMessage`.

- **Пространство имён:** `DocsVision.Platform.CardHost`
- **Сборка:** `DocsVision.Platform.CardHost.dll`

Синтаксис

```
public enum MessageResult
```

Члены

Имя члена	Описание
`None`	Кнопка не была нажата
`Ok`	Нажата кнопка Ок.
`Cancel`	Нажата кнопка Отмена.
`Abort`	Нажата кнопка Прервать.
`Retry`	Нажата кнопка Повторить.
`Ignore`	Нажата кнопка Игнорировать.

Имя члена	Описание
`Yes`	Нажата кнопка Да.
`No`	Нажата кнопка Нет.
`TryAgain`	Нажата Заново.
`Continue`	Продолжить.

MessageType — перечисление

Тип сообщения в окне сообщения.

- **Пространство имён:** `DocsVision.Platform.CardHost`
- **Сборка:** `DocsVision.Platform.CardHost.dll`

Синтаксис

```
public enum MessageType
```

Члены

Имя члена	Описание
`Error`	Ошибка.
`Question`	Вопрос.
`Warning`	Предупреждение.
`Information`	Информационное сообщение.

ReusageMode — перечисление

Определяет возможные режимы повторного использования компонента карточки.

- **Пространство имён:** `DocsVision.Platform.CardHost`
- **Сборка:** `DocsVision.Platform.CardHost.dll`

Синтаксис

```
public enum ReusageMode
```

Члены

Имя члена	Описание
`CardHostLogic`	Необходимость повторного использования компонента определяется внутренней логикой контейнера карточек.
`OwnLogic`	Необходимость повторного использования компонента определяется в его свойстве `CanBeReused`.
`NoReusage`	Компонент карточки не должен кэшироваться.

SystemDialogType — перечисление

Тип модального окна системного назначения.

- **Пространство имён:** `DocsVision.Platform.CardHost`
- **Сборка:** `DocsVision.Platform.CardHost.dll`

Синтаксис

```
public enum SystemDialogType
```

Члены

Имя члена	Описание
`Unknown`	Не задан.
`CardPrint`	Окно печати карточки.
`AdvancedFind`	Окно расширенного поиска.
`Parameters`	Окно параметров.
`ViewSettings`	Просмотр настроек.
`Locks`	Окно блокировок.
`Sessions`	Окно пользовательских сессий.
`Log`	Окно лога.

DocsVision.Platform.Data — пространство имён

Содержит набор классов для взаимодействия с базами данных.

Пространства имён

Пространство имён	Описание
DocsVision.Platform.Data.Metadata	Пространство имён DocsVision.Platform.Data.Metadata содержит набор классов для работы с метаданными.

Классы

Класс	Описание
DatabaseCommand	Представляет запрос к базе данных.

DocsVision.Platform.Data.Metadata — пространство имён

Пространство имён `DocsVision.Platform.Data.Metadata` содержит набор классов для работы с метаданными.

Пространства имён

Пространство имён	Описание
DocsVision.Platform.Data.Metadata.CardModel	Содержит объектную модель метаданных карточки.

Интерфейсы

Интерфейс	Описание
IMetadataProvider	Предоставляет доступ к метаданным карточек, секций, и полей.

DocsVision.Platform.Data.Metadata.CardModel — пространство имён

Содержит объектную модель метаданных карточки.

Классы

Класс	Описание
`Card`	Предоставляет доступ к метаданным карточки.
`CardSection`	Предоставляет доступ к метаданным секции карточки.
`SectionField`	Предоставляет доступ к метаданным поля карточки.

Card — класс

Предоставляет доступ к метаданным карточки.

- **Пространство имён:** `DocsVision.Platform.Data.Metadata.CardModel`
- **Сборка:** `DocsVision.Platform.StorageServer.dll`

Синтаксис

```
[CLSCompliant(true)]
public sealed class Card : CardNamedObject, ICloneable
```

Конструкторы

Имя	Описание
`Card()`	Инициализирует новый экземпляр класса Card.

Свойства

Имя	Описание
`Actions`	Возвращает коллекцию методов карточки.
`CardFlags`	Задаёт или возвращает флаги карточки.
`DefaultSecurity`	Задаёт или возвращает набор прав по умолчанию, которые будут назначаться на экземпляры карточки при их создании.

Имя	Описание
FetchMode	Задаёт или возвращает механизм чтения карточки.
Icon	Задаёт или возвращает иконку.
IconFile	Задаёт или возвращает путь к файлу иконки.
LibraryId	Задаёт или возвращает идентификатор библиотеки.
Modes	Возвращает коллекцию режимов работы карточки.
ProgId	Задаёт или возвращает программный идентификатор COM-компонента, реализующего функциональность карточки.
Sections	Возвращает коллекцию секций карточки.
Transformations	Возвращает коллекцию возможных преобразований данных карточки.
TypeName	Задаёт или возвращает полное имя класса компонента карточки.
Version	Задаёт или возвращает номер версии карточки.
VirtualFields	Возвращает коллекцию виртуальных полей.

Методы

Имя	Описание
Accept(CardObjectVisitor)	Применяет внесенные изменения.
Clone	Создаёт копию карточки со всеми секциями.
GetSectionByAlias(String)	Возвращает секцию карточки с указанным псевдонимом.

Имя	Описание
`Load(XmlReader)`	Загружает схему карточки из объекта XmlReader.
`Save(XmlWriter)`	Сериализует схему карточки в объект XmlWriter.

CardSection — класс

Предоставляет доступ к метаданным секции карточки.

- **Пространство имён:** `DocsVision.Platform.Data.Metadata.CardModel`
- **Сборка:** `DocsVision.Platform.StorageServer.dll`

Синтаксис

```
[CLSCompliant(true)]
public sealed class CardSection : CardNamedObject, ICloneable
```

Конструкторы

Имя	Описание
`CardSection()`	Инициализирует новый экземпляр класса CardSection.

Свойства

Имя	Описание
`Constraints`	Возвращает коллекцию ограничений на поля секции.
`DisplayFields`	Задаёт или возвращает поля, из которых составляются описания (дайджесты) строк секции.
`Fields`	Возвращает коллекцию полей секции.
`PropDispValue`	Задаёт или возвращает поле, содержащее отображаемое значение свойства, если секция используется для хранения свойств.

Имя	Описание
PropName	Задаёт или возвращает поле, содержащее название свойства, если секция используется для хранения свойств.
PropType	Задаёт или возвращает поле, определяющее тип свойства, если секция используется для хранения свойств.
PropValue	Задаёт или возвращает поле, содержащее значение свойства, если секция используется для хранения свойств.
RowAlias	Задаёт или возвращает строку, указывающую наименование соответствующего класса при использовании утилит авто-генерации кода по описанию карточки.
SearchWords	Возвращает коллекцию контекстных/поисковых слов.
SectionFlags	Задаёт или возвращает флаги секции.
Sections	Возвращает коллекцию подсекции секции.
SectionType	Задаёт или возвращает тип секции.
SortFields	Возвращает коллекцию описаний полей секции, по которым производится сортировка строк при обращении к данным секции.

Методы

Имя	Описание
Accept(CardObjectVisitor)	Сохраняет внесенные изменения.
Clone	Создаёт копию текущей секции.

Имя	Описание
`FindField(String)`	Возвращает поле с указанным псевдонимом.
`FindLinkedField(String)`	Возвращает связанное поле с указанным псевдонимом.
`Load(XmlReader)`	Загружает данные секции из XML.
`Save(XmlWriter)`	Сохраняет данные секции в XML.

SectionField — класс

Предоставляет доступ к метаданным поля карточки.

- **Пространство имён:** `DocsVision.Platform.Data.Metadata.CardModel`
- **Сборка:** `DocsVision.Platform.StorageServer.dll`

Синтаксис

```
[CLSCompliant(true)]
public sealed class SectionField : CardNamedObject, ICloneable
```

Конструкторы

Имя	Описание
`SectionField`	Инициализирует экземпляр класса SectionField.

Свойства

Имя	Описание
`CopyBehavior`	Задаёт или возвращает режим копирования данных поля.
`DefaultValue`	Задаёт или возвращает значение поля, которое будет автоматически задаваться при создании новых экземпляров карточки.

Имя	Описание
Description	Задаёт или возвращает словесное описание поля, которое будет отображаться в виде подсказки к полю в диалогах поиска и представлений.
EnumValues	Возвращает элементы перечисления, если тип поля: перечисление.
FieldFlags	Задаёт или возвращает дополнительные флаги.
FieldType	Задаёт или возвращает тип поля.
LinkedCardTypeId	Задаёт или возвращает уникальный идентификатор типа карточки, с полями или секциями которой связано данное поле.
LinkedFields	Возвращает коллекцию полей связанной карточки.
LinkedSectionId	Задаёт или возвращает идентификатор секции связанной карточки, с полями которого осуществляется связь.
LinkType	Задаёт или возвращает тип ссылки для ссылочного поля.
Max	Задаёт или возвращает ограничение длины поля.

Методы

Имя	Описание
Accept(CardObjectVisitor)	Применяет изменения.
Clone	Создаёт копию поля.
Load(XmlReader)	Загружает данные поля из XML.
Save(XmlWriter)	Сохраняет данные поля в XML.

IMetadataProvider — интерфейс

Предоставляет доступ к метаданным карточек, секций, и полей.

- **Пространство имён:** DocsVision.Platform.Data.Metadata
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
public interface IMetadataProvider
```

Свойства

Имя	Описание
Cards	Возвращает коллекцию метаданных карточек, зарегистрированных на сервере.
Libraries	Возвращает коллекцию метаданных библиотек карточек, зарегистрированных на сервере.

Методы

Имя	Описание
GetCard(Guid)	Возвращает метаданные карточки с заданным идентификатором.
GetField(Guid)	Возвращает метаданные поля с заданным идентификатором.
GetLibrary(Guid)	Возвращает метаданные библиотеки карточек с заданным идентификатором.
GetSection(Guid)	Возвращает метаданные секции с заданным идентификатором.

Заметки

Код регистрация провайдера метаданных выглядит следующим образом:

```
IMetadataProvider metadataProvider = DocsVisionObjectFactory.CreateMetadataProvider
(userSession);
objectContext.AddService<IMetadataManager>(DocsVisionObjectFactory.CreateMetadataManager(
metadataProvider, userSession));
```

```
objectContext.AddService<IMetadataProvider>(metadataProvider);
```

Доступ к поставщику метаданных аналогичен получения сервиса:

```
IMetadataProvider iMetadataProvider = objectContext.GetService<IMetadataProvider>();
```

DatabaseCommand — класс

Представляет запрос к базе данных.

- **Пространство имён:** DocsVision.Platform.Data
- **Сборка:** DocsVision.Platform.StorageServer.Data.dll

Синтаксис

```
public abstract class DatabaseCommand : IDbCommand, IDisposable
```

Свойства

Имя	Описание
CommandText	Возвращает или задаёт инструкцию Transact-SQL, имя таблицы или хранимую процедуру, выполняемую для источника данных.
CommandTimeout	Возвращает или задаёт время ожидания перед завершением попытки выполнить команду и созданием ошибки.
CommandType	Возвращает или задаёт значение, указывающее, как будет интерпретироваться свойство CommandText.
Connection	Задаёт или возвращает соединение с базой данных.
Parameters	Возвращает коллекцию параметров запроса к базе данных.

Имя	Описание
`Transaction`	Задаёт или возвращает транзакцию Transact-SQL, которая должна быть выполнена над базой данных SQL Server.
`UpdatedRowSource`	Задаёт или возвращает способ применения к обновляемой строке результатов команды запроса.

DocsVision.Platform.Extensibility — пространство имён

Пространство имён `DocsVision.Platform.Extensibility` содержит набор классов и интерфейсов расширения функциональности Windows-клиента.

Классы

Класс	Описание
`NavCommand`	Содержит свойства команды расширения Windows-клиента.
`NavCommandLocation`	Определяет местоположение команды в меню Windows-клиента.
`NavFolderInfo`	Определяет идентификатор и параметры отображения папки в Windows-клиенте.
`NavPropertyPage`	Предоставляет данные для страницы свойств карточки, реализуемой для расширения Windows-клиента.

Интерфейсы

Интерфейс	Описание
`INavCardType`	Определяет параметры подтипа карточки.
`INavCardTypeCollection`	Содержит методы для работы с коллекцией `INavCardType`.
`INavCommandExtension`	Описывает командное расширение Windows-клиента.

Интерфейс	Описание
INavFolderTreeExtension	Описывает расширение Windows-клиента, добавляющее в интерфейс пользователя дополнительные отображаемые папки.
INavFolderTypeCollection	Содержит методы для работы с коллекцией INavFolderType.
INavReportExtension	Описывает расширение Windows-клиента, выполняющее функцию экспорта.

Перечисления

Перечисление	Описание
MenuLocation	Определяет типы объектов для привязки меню.
NavCardTypeFlags	Дополнительные флаги подтипа карточки.
NavCommandStatus	Определяет статус команды в расширении меню Windows-клиента.
NavCommandTypes	Типы команд расширения Windows-клиента.
NavExtensionTypes	Типы расширений Windows-клиента.
NavFolderControlFlags	Предопределяет действия Windows-клиента для методов расширения контроля папки.
NavFolderControlType	Предоставляет для расширения Windows-клиента типы папок.
NavFolderLocationFlags	Определяет местоположение папки в кусте дерева папок Windows-клиента.
NavFolderTreeFlags	Определяет местоположение папки заданное в расширении дерева Windows-клиента и панели избранного.
NavPickerAccountTypes	Определяет тип учетной записи.

Перечисление	Описание
NavPickerNameFormat	Определяет формат имени учетной записи.
NavPropertyPageTypes	Определяет тип страницы свойств расширения Windows-клиента.

NavCommand — класс

Содержит свойства команды расширения Windows-клиента.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public class NavCommand
```

Конструкторы

Имя	Описание
NavCommand()	Инициализирует новый экземпляр типа NavCommand.

Свойства

Имя	Описание
CommandType	Возвращает или задаёт тип команды.
Description	Возвращает или задаёт описание команды расширения.
Icon	Возвращает или задаёт иконку команды.
Location	Возвращает или задаёт расположение команды.
Name	Возвращает или задаёт локализованное название.

NavCommand — конструктор

Инициализирует новый экземпляр класса `NavCommand`.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public class NavCommand
```

NavCommand.CommandType — свойство

Возвращает или задаёт тип команды.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public virtual NavCommandTypes CommandType { get; set; }
```

Значение свойства

Тип: `NavCommandTypes`

Тип команды

NavCommand.Description — свойство

Возвращает или задаёт описание команды расширения.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public virtual string Description { get; set; }
```

Значение свойства

Тип: `System.String`

Описание

Заметки

Данный текст будет отображаться во всплывающей подсказке с меню.

NavCommand.Icon — свойство

Возвращает или задаёт иконку команды.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public virtual Icon Icon { get; set; }
```

Значение свойства

Тип: System.Drawing.Icon

Иконка кнопки

NavCommand.Location — свойство

Возвращает или задаёт расположение команды.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public virtual NavCommandLocation Location { get; set; }
```

Значение свойства

Тип: NavCommandLocation

Расположение команды в меню

NavCommandLocation — класс

Определяет местоположение команды в меню Windows-клиента.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public class NavCommandLocation
```

Конструкторы

Имя	Описание
`NavCommandLocation()`	Инициализирует новый экземпляр класса NavCommandLocation.

Свойства

Имя	Описание
`Menu`	Возвращает или задаёт типы объектов, поддерживающих использование данного меню.

NavCommandLocation — конструктор

Инициализирует новый экземпляр класса `NavCommandLocation`.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public NavCommandLocation()
```

NavFolderInfo — класс

Определяет идентификатор и параметры отображения папки в Windows-клиенте.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public class NavFolderInfo
```

Свойства

Имя	Описание
`Flags`	Задаёт или возвращает флаги, определяющие местоположение папки в Windows-клиенте.
`FolderId`	Задаёт или возвращает идентификатор папки, предназначенной для отображения в Windows-клиенте.
`Location`	Задаёт или возвращает порядок расположения папки среди других папок.

NavFolderInfo.Flags — свойство

Задаёт или возвращает флаги, определяющие местоположение папки в Windows-клиенте.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public NavFolderTreeFlags Flags { get; set; }
```

Значение свойства

Тип: `NavFolderTreeFlags`

Маска местоположения папки расширения

NavFolderInfo.FolderId — свойство

Задаёт или возвращает идентификатор папки, предназначенной для отображения в Windows-клиенте.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public Guid FolderId { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор существующей папки

Заметки

Пользователь должен иметь соответствующие права на данную папку.

NavFolderInfo.Location — свойство

Задаёт или возвращает порядок расположения папки среди других папок.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public NavFolderLocationFlags Location { get; set; }
```

Значение свойства

Тип: NavFolderLocationFlags

Флаг местоположения папки среди других папок

Заметки

При наличии нескольких папок с одинаковым Location сортировка производится по алфавиту.

NavPropertyPage — класс

Предоставляет данные для страницы свойств карточки, реализуемой для расширения Windows-клиента.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public sealed class NavPropertyPage
```

Конструкторы

Имя	Описание
NavPropertyPage	Инициализирует новый экземпляр класса NavPropertyPage.

Свойства

Имя	Описание
Clsid	Возвращает или задаёт идентификатор элемента управления, реализующего страницу.
License	Возвращает или задаёт лицензию на компонент.
Name	Возвращает или задаёт заголовок страницы свойств.
PageType	Задаёт или возвращает тип страницы свойств.
Progid	Возвращает или задаёт программный идентификатор компонента, реализующего страницу.
TypeName	Возвращает или задаёт полное имя класса программного компонента.

Примеры

```
protected override IEnumerable<NavPropertyPage> CreatePropertyPages()
{
    return new NavPropertyPage[] {
        new NavPropertyPage() {
            PageType = NavPropertyPageTypes.All,
            Name = "TestPropertyPage",
            Clsid = typeof(TestPropertyPage).GUID
        }
    };
}
```

NavPropertyPage — конструктор

Инициализирует новый экземпляр класса `NavPropertyPage`.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public NavPropertyPage()
```

NavPropertyPage.Clsid — свойство

Возвращает или задаёт идентификатор элемента управления, реализующего страницу.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public Guid Clsid { get; set; }
```

Значение свойства

Тип: `System.Guid`

Идентификатор элемента управления

NavPropertyPage.License — свойство

Возвращает или задаёт лицензию на компонент.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public string License { get; set; }
```

Значение свойства

Тип: `System.String`

Лицензия

NavPropertyPage.Name — свойство

Возвращает или задаёт заголовок страницы свойств.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public string Name { get; set; }
```

Значение свойства

Тип: System.String

Заголовок страницы

NavPropertyPage.PageType — свойство

Задаёт или возвращает тип страницы свойств.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public NavPropertyPageTypes PageType { get; set; }
```

Значение свойства

Тип: NavPropertyPageTypes

Тип страницы

Заметки

Фактически тип страницы определяет допустимость использования данной страницы для определённых типов объектов.

NavPropertyPage.Progid — свойство

Возвращает или задаёт программный идентификатор компонента, реализующего страницу.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public string Progid { get; set; }
```

Значение свойства

Тип: System.String

Идентификатор элемента управления

NavPropertyPage.TypeName — свойство

Возвращает или задаёт полное имя класса программного компонента.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public string TypeName { get; set; }
```

Значение свойства

Тип: System.String

Название класс

INavCardType — интерфейс

Определяет параметры подтипа карточки.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public interface INavCardType
```

Свойства

Имя	Описание
`Alias`	Задаёт или возвращает псевдоним подтипа карточки.
`CardTypes`	Возвращает массив дочерних подтипов карточек.
`Flags`	Возвращает или задаёт вспомогательные флаги подтипа.
`Icon`	Возвращает или задаёт иконку подтипа.
`Id`	Возвращает уникальный идентификатор.
`Name`	Возвращает или задаёт локализованное имя подтипа.

INavCardTypeCollection — интерфейс

Содержит методы для работы с коллекцией `INavCardType`.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public interface INavCardTypeCollection : IEnumerable<INavCardType>, IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов коллекции.
`Item`	Возвращает элемент коллекции с заданным индексом.
`Item`	Возвращает элемент коллекции с заданным идентификатором.

Методы

Имя	Описание
`AddNew(Guid)`	Добавляет в коллекцию объект типа INavCardType с заданным идентификатором и возвращает ссылку на него.
`Clear`	Удаляет все элементы коллекции.
`Contains(Guid)`	Определяет вхождение в коллекцию подтипа с заданным идентификатором.
`Remove(Guid)`	Удаляет из коллекции подтип с заданным идентификатором.
`Remove(Int32)`	Удаляет по индексу элемент коллекции.

INavFolderTreeExtension — интерфейс

Описывает расширение Windows-клиента, добавляющее в интерфейс пользователя дополнительные отображаемые папки.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public interface INavFolderTreeExtension
```

Методы

Имя	Описание
`GetTreeExtensionFolders`	Возвращает массив идентификаторов папок и параметров отображения в интерфейсе Windows-клиента.

Заметки

Данное расширение позволяет поместить папку на нужный уровень дерева папок Windows-клиента или панели избранного, избегая использования папок-делегатов.

Пример доступен в разделе Пример реализации расширения дерева папок Windows-клиента.

INavFolderTreeExtension.GetTreeExtensionFolders — метод

Возвращает массив идентификаторов папок и параметров отображения в интерфейсе Windows-клиента.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
NavFolderInfo[] GetTreeExtensionFolders()
```

Возвращаемое значение

Тип: NavFolderInfo[]

Массив настроек папок для отображения в Windows-клиенте

INavFolderTypeCollection — интерфейс

Содержит методы для работы с коллекцией INavFolderType.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public interface INavFolderTypeCollection : IEnumerable<INavFolderType>, IEnumerable
```

Свойства

Имя	Описание
Count	Возвращает количество элементов коллекции.
Item	Возвращает элемент коллекции с заданным индексом.
Item	Возвращает элемент коллекции с заданным идентификатором.

Методы

Имя	Описание
`AddNew(Guid)`	Добавляет в коллекцию объект типа `INavFolderType` с заданным идентификатором и возвращает ссылку на него.
`Clear`	Удаляет все элементы коллекции.
`Contains(Guid)`	Определяет вхождение в коллекцию подтипа с заданным идентификатором.
`Remove(Guid)`	Удаляет из коллекции тип с заданным идентификатором.
`Remove(Int32)`	Удаляет по индексу элемент коллекции.

INavCommandExtension — интерфейс

Описывает командное расширение Windows-клиента.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
public interface INavCommandExtension
```

Свойства

Имя	Описание
`Count`	Возвращает количество зарегистрированных команд.

Методы

Имя	Описание
`GetCommand(Int32)`	Возвращает команду по индексу.
`InvokeCommand(Int32, NavFolderControlType, Guid, Guid[])`	Запускает команду с заданным индексом.

Имя	Описание
QueryStatus(Int32, NavFolderControlType, Guid, Guid[])	Возвращает результат выполнения команды с заданным индексом.
QueryStatus(Int32, NavFolderControlType, Guid, Guid[])	Возвращает результаты выполнения команд с указанными индексами.

Заметки

Пример доступен в подразделе Пример реализации командного расширения Windows-клиента.

INavReportExtension — интерфейс

Описывает расширение Windows-клиента, выполняющее функцию экспорта.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public interface INavReportExtension
```

Методы

Имя	Описание
CreateReport(Object, InfoRowCollection, Guid)	При переопределении в производном классе запускает механизм экспорта.

INavReportExtension.CreateReport — метод (Object, InfoRowCollection, Guid)

При переопределении в производном классе запускает механизм экспорта.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
void CreateReport(object grid, InfoRowCollection rowCollection, Guid folderId)
```

Параметры

grid

Тип: `System.Object`

Ссылка на элемент управления `JanusGrid`, используемый для отображения данных представления в Windows-клиенте. Не рекомендуется записывать какие-либо данные непосредственно в `JanusGrid` или изменять его настройки — это может привести к сбоям в работе Windows-клиента.

rowCollection

Тип: `InfoRowCollection`

Данные текущего представления

folderId

Тип: `System.Guid`

Идентификатор текущей папки

MenuLocation — перечисление

Определяет типы объектов для привязки меню.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
[Flags]
public enum MenuLocation
```

Члены

Имя члена	Описание
`None`	Не задано. Значение `0x00`.
`Card`	Карточка. Значение `0x01`.
`Folder`	Папка. Значение `0x02`.

Имя члена	Описание
`Grid`	Область представления. Значение `0x04`.
`Application`	Весь Windows-клиент. Значение `0x08`.

NavCommandStatus — перечисление

Определяет статус команды в расширении меню Windows-клиента.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
[Flags]
public enum NavCommandStatus
```

Члены

Имя члена	Описание
`None`	Не задан. Значение `0x0`.
`Supported`	Команда применима к данному типу объектов. Значение `0x1`.
`Enabled`	Команда активна. Значение `0x2`.
`Latched`	Кнопка нажата и пункт меню отмечен. Значение `0x4`.
`HideStandard`	Спрятать стандартные команды. Значение `0x8`. Не используется.

NavCommandTypes — перечисление

Определяет тип команды расширения Windows-клиента.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.Extensibility`

- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
[Flags]
public enum NavCommandTypes
```

Члены

Имя члена	Описание
None	Тип не указан. Значение 0x00.
Menu	Команда меню. Значение 0x01.
ToolBar	Команда панели инструментов. Значение 0x02.
Picture	Команда имеет иконку. Значение 0x04.
Separator	Разделитель. Значение 0x08.
Subcommand	Команда расположена в подменю. Значение 0x10.
ContextMenuFolder	Команда контекстного меню папки. Значение 0x20.
ContextMenuCard	Команда контекстного меню карточки. Значение 0x40.
ContextMenuGrid	Команда контекстного меню области вне карточек и папок. Значение 0x80.
DropDownButton	Команда кнопки с выпадающим списком. Значение 0x100.
DropDownMenu	Команда выпадающего списка. Значение 0x200.
StatusBar	Команда располагается в области уведомления. Значение 0x400.

Имя члена	Описание
`RefreshExtensionCommands`	Дополнительный флаг для команды. После выполнения команды будет выполнена перезагрузка командного расширения, что необходимо для поддержания динамически изменяемых меню. Значение `0x800`.

NavExtensionTypes — перечисление

Типы расширений Windows-клиента.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
[Flags]
public enum NavExtensionTypes
```

Члены

Имя члена	Описание
`None`	Тип не указан. Значение `0x0000`.
`VirtualFolder`	Расширение виртуальных папок. Значение `0x0001`.
`Report`	Расширение экспорта. Значение `0x0002`.
`Command`	Командное расширение. Значение `0x0004`.
`Event`	Расширение событий. Значение `0x0008`.
`Picker`	Расширение-пикер (диалоговое окно выбора). Значение `0x0010`.
`Control`	Расширение контроля папок. Значение `0x0020`.

Имя члена	Описание
`PropertyPages`	Расширение страниц свойств. Значение `0x0040`.
`CardTypes`	Расширение типов карточек. Значение `0x0080`.
`FolderTypes`	Расширение типов папок. Значение `0x0100`.
`StandardCommand`	Стандартные команды. Значение `0x0200`.
`FolderTree`	Расширение дерева папок. Позволяет поместить свою папку в дерево папок Windows-клиента без использования папок-делегатов. Значение `0x2000`.
`All`	Все типы расширений. Значение `0x1FFF`.

NavFolderControlFlags — перечисление

Предопределяет действия Windows-клиента для методов расширения контроля папки.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
[Flags]
public enum NavFolderControlFlags
```

Члены

Имя члена	Описание
`None`	Не задано (поведение по умолчанию). Значение `0x00`.
`HideFolder`	Скрыть папку. Значение `0x01`.

Имя члена	Описание
`HideUnreadCount`	Не показывать количество непрочитанных карточек. Значение `0x02`.
`DoNotAskChildren`	Не показывать дочерние папки. Значение `0x04`.

NavFolderControlType — перечисление

Предоставляет для расширения Windows-клиента типы папок.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
[Flags]
public enum NavFolderControlType
```

Члены

Имя члена	Описание
`Unknown`	Неизвестный тип папки. Значение `0x0000`.
`PublicRoot`	Папка общедоступных документов. Значение `0x0001`.
`Folder`	Обычная папка. Значение `0x0002`.
`CardsRoot`	Системная папка "Карточки". Значение `0x04`.
`CardLibrary`	Папка библиотеки карточек в системной ветке "Карточки". Значение `0x0008`.
`CardType`	Папка типа карточек в системной ветке "Карточки". Значение `0x0010`.

Имя члена	Описание
`SearchResults`	Системная папка "Результаты поиска". Значение `0x0020`.
`References`	Системная папка "Справочники". Значение `0x0040`.
`RecycleBin`	Системная папка "Корзина". Значение `0x0080`.
`FavoritesPersonal`	Папка "Избранное". Значение `0x0200`.

NavPickerAccountTypes — перечисление

Определяет тип учетной записи.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
[Flags]
public enum NavPickerAccountTypes
```

Члены

Имя члена	Описание
`None`	Неизвестный тип. Значение `0x0`.
`User`	Пользователь. Значение `0x1`.
`Group`	Группа. Значение `0x2`.
`Computer`	Компьютер. Значение `0x4`.
`Multiselect`	Множественный выбор. Значение `0x8`.

NavPickerNameFormat — перечисление

Определяет формат имени учетной записи.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public enum NavPickerNameFormat
```

Члены

Имя члена	Описание
Account	Учетная запись.
Class	Тип объекта. Например, пользователь или группа.
CommonName	Короткое имя.
Default	По умолчанию.
Full	Полное имя.
Locking	Имя пользователя, заблокировавшего объект.
Sid	Идентификатор безопасности SID.

NavPropertyPageTypes — перечисление

Определяет тип страницы свойств расширения Windows-клиента.

Это перечисление имеет атрибут FlagsAttribute, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
[Flags]
public enum NavPropertyPageTypes
```

Члены

Имя члена	Описание
`None`	Страница не применима ни для каких объектов. Значение `0x0`.
`Folder`	Страница свойств папки. Значение `0x1`.
`Card`	Страница свойств карточки. Значение `0x2`.
`CardType`	Страница свойств типа карточки. Значение `0x4`.
`All`	Страница для всех типов объектов. Значение `0x7`.

NavFolderTreeFlags — перечисление

Определяет местоположение папки заданное в расширении дерева и панели избранного Windows-клиента.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.Extensibility`
- **Сборка:** `DocsVision.Platform.Extensibility.dll`

Синтаксис

```
[Flags]
public enum NavFolderTreeFlags
```

Члены

Имя члена	Описание
`None`	Местоположение не задано. Значение `0x00`.
`ShowSubfolders`	Дополнительный флаг, разрешающий отображение вложенных папок у папки, определённой в расширении. Значение `0x01`.

Имя члена	Описание
ShowInFavoriteFolders	Помещает папку в панель избранного. Значение 0x02.
ShowInFolders	Помещает папку в качестве вложенной папки в раздел дерева "Папки". Значение 0x04.
ShowInPersonalFolder	Помещает папку в личные папки пользователя. Значение 0x08.
ShowAsRoot	Помещает папку на верхнем уровне дерева папок. Значение 0x10.

NavFolderLocationFlags — перечисление

Определяет местоположение папки в кусте дерева папок Windows-клиента.

- **Пространство имён:** DocsVision.Platform.Extensibility
- **Сборка:** DocsVision.Platform.Extensibility.dll

Синтаксис

```
public enum NavFolderLocationFlags
```

Члены

Имя члена	Описание
AsRegular	Расположение папки определяют те же правила, что и для других папок.
BeyondRegular	Папка располагается ниже остальных папок.
AboveRegular	Папка располагается выше остальных папок.

DocsVision.Platform.ObjectManager — пространство имён

Пространство имён DocsVision.Platform.ObjectManager предоставляет классы и интерфейсы, декларирующие API бизнес-логики Docsvision.

Пространства имён

Пространство имён	Описание
DocsVision.Platform.ObjectManager.Metadata	Предоставляет методы для работы с метаданными карточек.
DocsVision.Platform.ObjectManager.SearchModel	Пространство имён DocsVision.Platform.ObjectManager.SearchModel содержит классы предоставляющие методы доступа к серверному поиску.
DocsVision.Platform.ObjectManager.SystemCards	Содержит классы с реализацией объектов бизнес-логики для **системных карточек**.
DocsVision.Platform.ObjectManager.ViewModel	Содержит классы описывающие объектную модель **представлений** и методы для работы с представлениями.

Классы

Класс	Описание
AccessManager	Представляет объектную модель **менеджера объектов безопасности**. Содержит методы для работы с зашифрованными и подписанными объектами. Осуществляет управление правами (дескрипторами безопасности) на объекты системы.
AutoLocker	Содержит вспомогательные методы для работы с блокировкой объектов.
AutoUpdater	Содержит вспомогательные методы для работы с режимом отложенной блокировки.
CardData	Содержит данные карточки.
CardDataCollection	Предоставляет методы для работы с коллекцией CardData.
CardLabelCollection	Представляет коллекцию меток карточек.

Класс	Описание
CardManager	Представляет объектную модель **менеджера карточек**. Содержит методы чтение, изменение, создание и удаление карточек.
CitationFileItem	Список цитат, сгруппированных по файлам.
Connection	Соединение с сервером Docsvision.
DigestViewColumns	Представляет колонку в представление.
ExportCardErrorEventArgs	Предоставляет данные для события ExportCardInspector.Error
ExportCardInspector	Инспектор, позволяющий гибко управлять логикой экспорта.
ExtensionManager	Представляет объектную модель **менеджера расширений**. Содержит методы для работы с серверными расширениями.
ExtensionMethod	Описывает структуру серверного расширения.
ExtensionMethodParameter	Представляет параметр метода серверного расширения.
ExtensionMethodParameterCollection	Содержит методы для работы с коллекцией ExtensionMethodParameter.
FilesCitationInfo	Список цитат, сгруппированных по карточкам.
FileData	Содержит методы для организации работы с простыми файлами.
FileDataCollection	Содержит методы для работы с коллекцией FileData.
FileManager	Представляет объектную модель **менеджера файлов**. Содержит методы для работы с файлами.

Класс	Описание
`ImportCardInspector`	Инспектор, позволяющий гибко управлять логикой импорта.
`InfoRow`	Представляет элемент коллекции поддерживающей страничное чтение.
`InfoRowCollection`	Содержит методы для работы с коллекцией InfoRow.
`LicenseManager`	Класс `LicenseManager` содержит функциональность по работе с дополнительными опциями лицензии Docsvision, в которых присутствует ограничение по количеству соединений.
`LockedObject`	Представляет объект поддерживающий блокировку.
`LockedObjectCollection`	Содержит методы для работы с коллекцией `LockedObject`.
`LockManager`	Представляет объектную модель **менеджера блокировок**. Содержит методы управления блокировками на объекты.
`LogManager`	Представляет объектную модель **менеджера журнала**. Содержит методы работы с журналом системы.
`LogSearchQuery`	Представляет фильтр данных для журнала.
`QueryParameter`	Предоставляет хранилище для параметров запроса.
`QueryParameterCollection`	Содержит методы для работы с коллекцией `QueryParameter`.
`Report`	Представляет данные для хранимой процедуры.
`ReportCollection`	Содержит методы для работы с коллекцией `Report`.

Класс	Описание
`ReportManager`	Представляет объектную модель *менеджера хранимых процедур*.
`ReportParameter`	Осуществляет хранение параметров хранимых процедур.
`ReportParameterCollection`	Содержит методы для работы с коллекцией `ReportParameter`.
`RowData`	Представляет строку данных карточки.
`RowDataCollection`	Содержит методы для работы с коллекцией `RowData`.
`RowField`	Описывает поле секции карточки.
`RowFieldCollection`	Содержит методы для работы с коллекцией `RowField`.
`SearchAggregationItemInfo`	Список объектов.
`SearchAggregationItemsRequest`	Объект запроса.
`SearchFilesCitationsRequest`	Запрос цитируемых данных из файла.
`SectionData`	Представляет секцию карточки.
`SectionDataCollection`	Содержит методы для работы с коллекцией `SectionData`.
`SectionDataReader`	Серверный курсор для чтения данных секции.
`SessionManager`	Представляет объектную модель **менеджера сессий**. Содержит методы управления сессиями пользователей.
`SubSectionData`	Представляет подсекцию карточки.
`QueryParameter`	Осуществляет хранение параметра запроса.
`QueryParameterCollection`	Содержит методы для работы с коллекцией `QueryParameter`.
`ViewReadRequest`	Запрос представления.
`ViewSource`	Источник данных представления.

Класс	Описание
UserSession	Содержит данные сессии пользователя, а также предоставляет доступ к менеджерам управления объектами системы

Интерфейсы

Интерфейс	Описание
ILockable	Разрешает для объекта управление режимом отложенной блокировки.
IProcessInfo	Определяет возможности по управлению состоянием конкретного бизнес-процесса и получению базовой информации о самом бизнес-процессе.
ISecurable	Разрешает объекту реализовывать модель безопасности.
IUpdatable	Разрешает для объекта режим отложенной записи.
IXmlExportable	Разрешает сохранять данные объекта в XML-формате.

Перечисления

Перечисление	Описание
ArchiveOptions	Режим архивации объекта.
ArchiveState	Определяет признак архивирования объекта.
ClearLogStrategy	Определяет стратегию автоматической очистки журнала.
CryptObjectType	Тип крипто-объекта.
EventType	Определяет типы записей в журнале событий.
ExportFlags	Задаёт формат экспорта данных.
LinkType	Маска типа ссылки.

Перечисление	Описание
LockedObjectType	Типы блокированных объектов.
LockStatus	Состояние блокировки объекта.
LogStrategy	Определяет стратегию ведения журнала.
ObjectState	Определяет состояние объекта.
ObjectStatus	Состояние объекта в режиме отложенной записи.
OfflineState	Состояние хранения файла.
ParameterValueType	Предоставляет тип для параметра метода серверного расширения.
RecordStatus	Состояние задействованности объекта.
SessionLoginFlags	Описывает флаги открытия новой сессии.

DocsVision.Platform.ObjectManager.Metadata — пространство имён

Представляет методы для работы с метаданными карточек.

Классы

Класс	Описание
CardAction	Представляет свойства метода карточки.
CardLibrary	Содержит описание библиотеки карточек.
CardLibraryCollection	Содержит методы для работы с коллекцией CardLibrary.
CardSection	Предоставляет доступ к метаданным секции карточки.
CardType	Содержит методы для работы с типами карточек и описывающие эти объекты свойства.

Класс	Описание
`CardTypeCollection`	Содержит методы для работы с коллекцией CardType.
`CardVirtualField`	Представляет виртуальное поле карточки.
`CardVirtualFieldCollection`	Содержит методы для работы с коллекцией CardVirtualField.
`Field`	Представляет описание поля строки карточки.

Перечисления

Перечисление	Описание
`FetchMode`	Определяет режим загрузки данных с сервера.
`FieldType`	Определяет тип поля.

CardAction — класс

Представляет свойства метода карточки. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.Metadata`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class CardAction
```

Свойства

Имя	Описание
`Alias`	Возвращает псевдоним команды.
`Id`	Возвращает идентификатор вызываемой команды.
`Name`	Возвращает название команды.

Имя	Описание
Session	Возвращает текущую сессию пользователя.

Заметки

Возможные действия определяются в момент создания схемы карточки.

CardAction.Alias — свойство

Возвращает псевдоним команды.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Alias { get; }
```

Значение свойства

Тип: System.String

Псевдоним

CardAction.Id — свойство

Возвращает идентификатор вызываемой команды.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid Id { get; }
```

Значение свойства

Тип: System.Guid

Идентификатор действия

Заметки

Идентификатор определяет команду, которая будет вызвана из клиентского компонента карточки, и была реализована в соответствующей бибилиотеке.

CardAction.Name — свойство

Возвращает название команды.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Name { get; }
```

Значение свойства

Тип: System.String

Название команды

Заметки

Как правило, это имя будет использовано в Windows-клиенте для отображения в меню карточки.

CardAction.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Сессия пользователя

CardLibrary — класс

Содержит описание библиотеки карточек.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.Metadata`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class CardLibrary
```

Свойства

Имя	Описание
`Alias`	Возвращает псевдоним библиотеки.
`CardTypes`	Возвращает коллекцию описаний карточек, входящих в библиотеку.
`Clsid`	Возвращает идентификатор класса программного компонента библиотеки.
`Icon`	Возвращает иконку библиотеки.
`Id`	Возвращает идентификатор библиотеки.
`Installers`	Возвращает сведения о пакетах инсталляции библиотеки.
`Log`	Возвращает набор журналируемых операций.
`Name`	Возвращает название библиотеки.
`ProgId`	Возвращает идентификатор программного компонента библиотеки.
`Session`	Возвращает текущая сессию пользователя.
`Version`	Возвращает версию библиотеки.

CardLibraryCollection — класс

Содержит методы для работы с коллекцией `CardLibrary`. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class CardLibraryCollection : IEnumerable<CardLibrary>, IEnumerable
```

Конструкторы

Имя	Описание
CardLibraryCollection()	Инициализирует новый экземпляр класса CardLibraryCollection.

Свойства

Имя	Описание
Count	Возвращает количество элементов в коллекции.
Item	Возвращает элемент коллекции с заданным индексом.
Item	Возвращает элемент коллекции с идентификатором библиотеки.
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
Contains(Guid)	Проверяет наличие элемента коллекции с заданным идентификатором библиотеки.
GetEnumerator	Возвращает перечислитель, осуществляющий перебор элементов экземпляра класса CardLibraryCollection.

CardSection — класс

Предоставляет доступ к метаданным секции карточки.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.Metadata`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class CardSection
```

Свойства

Имя	Описание
`Alias`	Задаёт или возвращает псевдоним секции.
`CardType`	Возвращает тип карточки.
`ChildSections`	Возвращает подчиненные секции.
`DisplayFields`	Возвращает коллекцию полей, доступную только для чтения.
`Fields`	Возвращает коллекцию полей секции.
`Flags`	Возвращает дополнительные установленные флаги.
`Id`	Возвращает идентификатор секции.
`Name`	Возвращает локализованное название секции.
`Names`	Возвращает коллекцию локализованное названий секции.
`ParentSection`	Возвращает родительскую секцию.
`Session`	Возвращает текущую сессию пользователя.
`Type`	Задаёт или возвращает тип секции.

Методы

Имя	Описание
`Reload`	Загружает заново свойства секции.
`Save`	Сохраняет изменённые значения.

CardType — класс

Содержит методы для работы с типами карточек и описывающие эти объекты свойства.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.Metadata`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class CardType
```

Конструкторы

Имя	Описание
`CardType()`	Инициализирует новый экземпляр класса CardType.

Свойства

Имя	Описание
`Actions`	Возвращает коллекцию методов данного карточки.
`Alias`	Возвращает псевдоним типа карточки.
`AllSections`	Возвращает коллекцию всех секций карточки (без учета иерархии).
`Clsid`	Возвращает идентификатор класса программного компонента карточки.
`Definition`	Полное XML-описание карточки
`FetchMode`	Возвращает режим загрузки данных карточки.

Имя	Описание
Flags	Задаёт или возвращает дополнительные атрибуты карточки.
Icon	Возвращает иконку типа карточки.
Id	Возвращает идентификатор типа.
LibraryId	Возвращает идентификатор библиотеки, которой принадлежит карточка.
LicenseKey	Лицензионный ключ для активации COM-компонента библиотеки.
Modes	Возвращает коллекцию режимов работы карточки.
Name	Возвращает локализованное название типа.
ProgId	Возвращает идентификатор программного компонента библиотеки.
Schema	Возвращает строку содержащую XSD-схему описания карточки.
Sections	Возвращает коллекцию секций верхнего уровня карточки.
Session	Возвращает текущую сессию пользователя.
Transformations	Возвращает коллекция возможных преобразований карточки.
TypeName	Возвращает полное название класса программного компонента карточки.
Views	Возвращает коллекцию описаний представлений карточки.
VirtualFields	Возвращает коллекцию виртуальных полей карточки.

Методы

Имя	Описание
AccessCheck(int)	Осуществляет проверку наличия прав доступа.
GetAccessControl	Возвращает описатель прав на объект.
GetAccessControl(AccessControlSections)	Возвращает описатель прав на объект из выбранных разделов безопасность.
SetAccessControl(CardDataSecurity)	Сохраняет изменённые права доступа к объекту.

CardType.VirtualFields — свойство

Возвращает коллекцию виртуальных полей карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardVirtualFieldCollection VirtualFields { get; }
```

Значение свойства

Тип: CardVirtualFieldCollection

Коллекция виртуальных полей

Заметки

Виртуальные поля служат для извлечения данных из нескольких таблиц с применением к ним определённых операций.

CardTypeCollection — класс

Содержит методы для работы с коллекцией CardType.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class CardTypeCollection : IEnumerable<CardType>, IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов коллекции.
`Session`	Возвращает текущую сессию пользователя.
`Item`	Возвращает элемент коллекции по заданному индексу.
`Item`	Возвращает элемент коллекции с указанным идентификатором типа.

Методы

Имя	Описание
`Contains(Guid)`	Определяет вхождение в коллекцию элемента с заданным идентификатором типа.
`GetEnumerator`	Возвращает перечислитель.

CardVirtualField — класс

Представляет виртуальное поле карточки. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.Metadata`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class CardVirtualField
```

Свойства

Имя	Описание
`Alias`	Возвращает псевдоним виртуального поля.
`Description`	Возвращает описание виртуального поля.

Имя	Описание
Id	Возвращает идентификатор поля.
Name	Локализованное название виртуального поля.
SectionId	Идентификатор секции карточки, источника данных виртуального поля.
Session	Текущая сессия пользователя.

Методы

Имя	Описание
Export	Извлекает определение виртуального поля для последующего добавления в конкретное описание представления.

CardVirtualField.Alias — свойство

Возвращает псевдоним виртуального поля.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Alias { get; }
```

Значение свойства

Тип: System.String

Псевдоним

Заметки

По псевдониму возможно обращение к полю внутри программы.

CardVirtualField.Description — свойство

Возвращает описание виртуального поля

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Description { get; }
```

Значение свойства

Тип: System.String

Примечание к полю

CardVirtualField.Id — свойство

Возвращает идентификатор поля.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid Id { get; }
```

Значение свойства

Тип: System.Guid

Идентификатор поля

CardVirtualField.Name — свойство

Локализованное название виртуального поля.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Name { get; }
```

Значение свойства

Тип: System.String

Название поля

Заметки

Язык возвращаемого значения зависит от настроек языка на сервере.

CardVirtualField.SectionId — свойство

Идентификатор секции карточки, источника данных виртуального поля.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid SectionId { get; }
```

Значение свойства

Тип: System.Guid

Идентификатор секции

Заметки

Идентификатор секции из которой данное виртуальное поле получает данные.

CardVirtualField.Session — свойство

Текущая сессия пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Текущая сессия

CardVirtualField.Export — метод

Извлекает определение виртуального поля для последующего добавления в конкретное описание представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract VirtualField Export()
```

Возвращаемое значение

Тип: VirtualField

Описание виртуального поля

CardVirtualFieldCollection — класс

Содержит методы для работы с коллекцией CardVirtualField.

- **Пространство имён:** DocsVision.Platform.ObjectManager.Metadata
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class CardVirtualFieldCollection : IEnumerable<CardVirtualField>,
IEnumerable
```

Свойства

Имя	Описание
Count	Возвращает количество элементов коллекции.
Session	Возвращает текущую сессию пользователя.
Item	Возвращает элемент коллекции с заданным идентификатором.
Item	Возвращает элемент коллекции с указанным индексом.
Item	Возвращает элемент коллекции с заданным псевдонимом.

Методы

Имя	Описание
`Contains(Guid)`	Проверяет вхождение в коллекцию элемента с заданным идентификатором.
`GetEnumerator`	Возвращает перечислитель.

Field — класс

Представляет поле строки карточки.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.Metadata`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class Field
```

Свойства

Имя	Описание
`Alias`	Задаёт или возвращает псевдоним поля.
`DefaultValue`	Задаёт или возвращает значение по умолчания.
`Description`	Возвращает описание.
`Descriptions`	Возвращает коллекцию локализованных описаний.
`EnumValues`	Возвращает значения перечисления.
`EnumValues`	Задаёт или возвращает дополнительные флаги.
`Id`	Возвращает идентификатор поля.
`LinkedCardType`	Задаёт или возвращает тип карточки связанной с полем.

Имя	Описание
`LinkedFields`	Возвращает коллекцию связанных полей.
`LinkedSection`	Задаёт или возвращает привязанную секцию.
`LinkType`	Задаёт или возвращает тип связи.
`Max`	Задаёт или возвращает максимальный размер поля.
`Name`	Возвращает локализованное название поля.
`Names`	Возвращает коллекцию локализованных имён.
`Section`	Возвращает секцию в которой находится поле.
`Session`	Возвращает текущую сессию пользователя.
`Type`	Возвращает или задаёт тип поля.
`ValueType`	Возвращает тип значения поля.

Методы

Имя	Описание
`Reload`	Перезагружает поле.
`Save`	Сохраняет изменённое значение поля.

FetchMode — перечисление

Определяет режим загрузки данных с сервера.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.Metadata`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
[Flags]
public enum FetchMode
```

Члены

Имя члена	Описание
`None`	Не определён. Значение `0x00`.
`Card`	Загружаются все данные из всех секций карточки (этот режим актуален только для карточки в целом). Значение `0x01`.
`Section`	При первом же обращении к коллекции строк секции её данные будут загружены полностью. Значение `0x02`.
`SubSection`	Загружаются только данные конкретной подсекции. Значение `0x04`.
`Level`	Загружаются данные подсекции только для конкретного уровня в дереве. Значение `0x08`.
`Row`	Режим построчной загрузки. В данный момент не реализован. Значение `0x10`.

Заметки

Режим загрузки определяет размер порции данных, получаемых за одно обращение к серверу.

FieldType — перечисление

Определяет тип поля.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.Metadata`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum FieldType
```

Члены

Имя члена	Описание
`Int`	Целое число.
`Bool`	Логическое значение.
`DateTime`	Дата и время.
`Date`	Дата.
`Time`	Время.
`Enum`	Перечисление.
`Bitmask`	Битовая маска.
`UniqueId`	Уникальный идентификатор (`Guid`).
`UserId`	Идентификатор пользователя.
`String`	Строка ANSI.
`Unistring`	Строка UNICODE.
`FileId`	Идентификатор файла.
`Float`	Дробное.
`RefId`	Ссылка на строку.
`RefCardId`	Ссылка на карточку.
`Text`	Текст ANSI.
`Unitext`	Текст UNICODE.
`Binary`	Двоичные данные.
`Image`	Изображение.
`SDId`	Описатель прав.
`Decimal`	Десятичное число.
`Variant`	Значение переменного типа.

ObjectState — перечисление

Определяет состояние объекта.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`

- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum ObjectState
```

Члены

Имя члена	Описание
`NotExisting`	Карточки не существует.
`Existing`	Карточка существует и может быть получена.
`Deleted`	Карточка существует, но помечена как удалённая, работа с ней невозможна.

ObjectStatus — перечисление

Состояние объекта в режиме отложенной записи.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
[Flags]
public enum ObjectStatus
```

Члены

Имя члена	Описание
`Normal`	Исходное состояние. Значение `0x0`.
`New`	Новый объект. Значение `0x1`.
`Deleted`	Объект удалён. Значение `0x2`.
`Changed`	Данные объекта были изменены. Значение `0x4`.

Имя члена	Описание
`Moved`	Изменено положение объекта в иерархии. Значение `0x8`.

OfflineState — перечисление

Состояние хранения файла.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum OfflineState
```

Члены

Имя члена	Описание
`Online`	Данные файла находятся в базе, с ним можно работать как обычно.
`Offline`	Данные файла выгружены во внешнее хранилище.
`OfflineAutoRestore`	Данные файла выгружена во внешнее хранилище, однако будут возвращены в базу при первом же обращении, что для пользователя гарантирует прозрачность работы с его данными.

ParameterValueType — перечисление

Предоставляет тип для параметра метода серверного расширения.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum ParameterValueType
```

Члены

Имя члена	Описание
`String`	Строка.
`Integer`	Целое число.
`Double`	Число с плавающей запятой.
`DateTime`	Дата и время.
`Bool`	Логический.
`Guid`	Уникальный идентификатор.

DocsVision.Platform.ObjectManager.SearchModel — пространство имён

Пространство имён `DocsVision.Platform.ObjectManager.SearchModel` содержит классы предоставляющие методы доступа к серверному поиску.

Классы

Класс	Описание
`AttributiveSearch`	Содержит методы для выполнения атрибутивного поиска.
`CardTypeQuery`	Представляет данные для поиска по типу карточки.
`CardTypeQueryCollection`	Содержит методы для работы с коллекцией `CardTypeQuery`.
`Condition`	Представляет набор условий для серверного поиска.
`ConditionCollection`	Предоставляет методы для работы с коллекцией `Condition`.
`ConditionGroup`	Представляет иерархическую группу условий поиска.
`ConditionGroupCollection`	Предоставляет методы для работы с коллекцией `ConditionGroup`.
`ControlLayout`	Определяет макет для параметров поискового запроса внутри формы.

Класс	Описание
ControlLayoutCollection	Предоставляет методы для работы с коллекцией ControlLayout.
FullTextSearch	Реализует модель полнотекстового серверного поиска.
JoinSection	Предоставляет данные присоединенной секции в поисковом запросе.
JoinSectionCollection	Предоставляет методы для работы с коллекцией JoinSection.
Scope	Описывает ограничения поиска.
ScopeCard	Описывает ограничение поиска карточкой.
ScopeCardCollection	Предоставляет методы для работы с коллекцией ScopeCard.
ScopeCardType	Описывает ограничение поиска типом карточки.
ScopeCardTypeCollection	Предоставляет методы для работы с коллекцией ScopeCardType.
ScopeFileType	Описывает ограничение поиска типом файла.
ScopeFolder	Описывает ограничение поиска папкой.
ScopeFolderCollection	Предоставляет методы для работы с коллекцией ScopeFolder.
SearchLayout	Настройки расположения параметров поискового запроса.
SearchQuery	Базовый объект для построения поискового запроса.
SectionQuery	Содержит методы для организации поиска по секциям карточки.
SectionQueryCollection	Предоставляет методы для работы с коллекцией SectionQuery.

Перечисления

Перечисление	Описание
ConditionGroupOperation	Логическая операция для группировки условий.
ConditionOperation	Тип сравнительной операции для поискового запроса
FieldSubtype	Определяет тип значения в поле.
FullTextSearchMode	Определяет режим полнотекстового поиска.
SearchAggregationItemsResultType	Результат поиска агрегатов.
SectionQueryOperation	Определяет логическую операцию для полнотекстового поиска.

AttributiveSearch — класс

Предоставляет методы для выполнения атрибутивного поиска.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class AttributiveSearch
```

Свойства

Имя	Описание
SearchQuery	Возвращает экземпляр поискового запроса.
CardTypeQueries	Возвращает параметры поиска по типу карточки.

Заметки

Атрибутивный поиск позволяет осуществлять выборку по различным данным карточек.

Примеры

```
const string ID_CARDTYPE = "{C1FED883-08DE-420F-8FB4-C16CEFFC1630}";
const string ID_SECTION = "{8C77892A-21CC-4972-AD71-A9919BCA8187}";
SearchQuery searchQuery = session.CreateSearchQuery();

CardTypeQuery typeQuery = searchQuery.AttributiveSearch.CardTypeQueries.AddNew(new Guid
(ID_CARDTYPE)); ①

SectionQuery sectionQuery = typeQuery.SectionQueries.AddNew(new Guid(ID_SECTION)); ②

sectionQuery.ConditionGroup.Conditions.AddNew("Name", FieldType.Unistring,
ConditionOperation.StrEquals, "Doc1"); ③

string query = searchQuery.GetXml(true, null); ④

CardDataCollection coll = session.CardManager.FindCards(query); ⑤
```

① Поиск по типу карточки.

② Поиск по секции.

③ Поиск по значению поля.

④ Получение текста запроса.

⑤ Выполнение запроса.

AttributiveSearch.CardTypeQueries — свойство

Возвращает параметры поиска по типу карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardTypeQueryCollection CardTypeQueries { get; }
```

Значение свойства

Тип: CardTypeQueryCollection

Коллекция поисковых запросов по заданным типам карточек.

AttributiveSearch.SearchQuery — свойство

Возвращает экземпляр поискового запроса.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SearchQuery SearchQuery { get; }
```

Значение свойства

Тип: SearchQuery

Экземпляр поискового запроса.

CardTypeQuery — класс

Представляет данные для поиска по типу карточки. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class CardTypeQuery
```

Свойства

Имя	Описание
AllCards	Возвращает или задаёт признак включения в поисковый запрос всех карточек.
AttributiveSearch	Возвращает объект содержащий параметры атрибутивного поиска.
CardTypeId	Возвращает или задаёт идентификатор типа карточки.
SearchQuery	Возвращает экземпляр поискового запроса.
SectionQueries	Возвращает коллекцию поисковых запросов по секциям карточки.

CardTypeQuery.AllCards — свойство

Возвращает или задаёт признак включения в поисковый запрос всех карточек.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool AllCards { get; set; }
```

Значение свойства

Тип: System.Boolean

true — поиск по всем карточкам, false — поиск согласно заданным атрибутам

Заметки

При значении true, поиск будет осуществляться по всем карточкам, несмотря на установленные другими параметрами ограничения.

CardTypeQuery.AttributiveSearch — свойство

Возвращает экземпляр атрибутивного поиска.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract AttributiveSearch AttributiveSearch { get; }
```

Значение свойства

Тип: AttributiveSearch

Экземпляр атрибутивного поиска

CardTypeQuery.CardTypeId — свойство

Возвращает или задаёт идентификатор типа карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid CardTypeId { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор типа карточки

CardTypeQuery.SearchQuery — свойство

Возвращает экземпляр поискового запроса.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SearchQuery SearchQuery { get; }
```

Значение свойства

Тип: SearchQuery

Поисковый запрос

CardTypeQuery.SectionQueries — свойство

Возвращает коллекцию поисковых запросов по секциям карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SectionQueryCollection SectionQueries { get; }
```

Значение свойства

Тип: SectionQueryCollection

Коллекция запросов

Заметки

CardTypeQueryCollection — класс

Содержит методы для работы с коллекцией `CardTypeQuery`. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class CardTypeQueryCollection : IEnumerable<CardTypeQuery>, IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов коллекции.
`Item`	Возвращает элемент коллекции с указанным индексом.
`Item`	Возвращает элемент коллекции с указанным идентификатором типа карточки.
`SearchQuery`	Возвращает экземпляр поискового запроса.

Методы

Имя	Описание
`AddNew(Guid)`	Добавляет тип карточки для запроса и возвращает сформированный поисковый запрос.
`Clear`	Удаляет все элементы коллекции.
`Contains(Guid)`	Определяет, входит ли элемент с заданным идентификатором в коллекцию.

Имя	Описание
Remove(Guid)	Производит удаление элемента коллекции с заданным идентификатором типа карточки.
Remove(Int32)	Выполняет удаление элемента коллекции с указанным индексом.

CardTypeQueryCollection.Count — свойство

Возвращает количество элементов коллекции..

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract int Count { get; }
```

Значение свойства

Тип: System.Int32

Число элементов коллекции

CardTypeQueryCollection.Item — свойство

Возвращает элемент коллекции с указанным индексом.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardTypeQuery this[int index] { get; }
```

Параметры

index

Тип: System.Int32

Индекс элемента

Значение свойства

Тип: CardTypeQuery

Элемент коллекции

CardTypeQueryCollection.Item — свойство

Возвращает элемент коллекции с указанным идентификатором типа карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardTypeQuery this[Guid cardTypeId] { get; }
```

Параметры

cardTypeId

Тип: System.Guid

Идентификатор типа карточки

Значение свойства

Тип: CardTypeQuery

Элемент коллекции

CardTypeQueryCollection.SearchQuery — свойство

Возвращает экземпляр поискового запроса.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SearchQuery SearchQuery { get; }
```

Значение свойства

Тип: SearchQuery

Экземпляр запроса

CardTypeQueryCollection.AddNew — метод (Guid)

Добавляет тип карточки для запроса и возвращает сформированный поисковый запрос.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardTypeQuery AddNew(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: System.Guid

Идентификатор типа карточки

Возвращаемое значение

Тип: CardTypeQuery

Созданный запрос

CardTypeQueryCollection.Clear — метод

Удаляет все элементы коллекции.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Clear()
```

CardTypeQueryCollection.Contains — метод (Guid)

Определяет, входит ли элемент с заданным идентификатором в коллекцию.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool Contains(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: System.Guid

Идентификатор типа карточки

Возвращаемое значение

Тип: System.Boolean

true — элемент входит в коллекцию, иначе — false

CardTypeQueryCollection.Remove — метод (Guid)

Производит удаление элемента коллекции с заданным идентификатором типа карточки

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Remove(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: System.Guid

Идентификатор типа карточки

CardTypeQueryCollection.Remove — метод (Int32)

Выполняет удаление элемента коллекции с указанным индексом.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Remove(int index)
```

Параметры

index

Тип: System.Int32

Индекс удаляемого элемента

Condition — класс

Представляет набор условий для серверного поиска. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class Condition
```

Свойства

Имя	Описание
AggregateConditionGroup	Возвращает подгруппу условий, которые будут накладываться на агрегируемые значения.
AggregateDistinctValues	Возвращает или задаёт признак, должна ли агрегационная функция учитывать только различные значения, или все.
AggregateFunction	Имя агрегирующей функции для значения секции.
Alias	Возвращает псевдоним условия.
ConditionGroup	Возвращает группу, которой принадлежит условие.

Имя	Описание
FieldAlias	Возвращает или задаёт псевдоним поля по которому строится условие.
FieldSubtype	Возвращает или задаёт подтип свойства.
FieldSubtypeId	Возвращает или задаёт идентификатор типа универсального справочника
FieldType	Тип поля по которому строиться условие.
Index	Возвращает индекс условия внутри группы условий.
IsParameter	Требует подставлять конкретное значение данного условия в окончательную строку поиска.
IsSearchWord	Признак наличия в условии специального поискового слова разрешаемого в значение карточкой.
IsSystemField	Возвращает или задаёт признак принадлежности поля к системным полям.
Operation	Сравнительная операция для данного условия.
ParameterAlias	Возвращает или задаёт псевдоним параметра, соответствующего данному условию.
ParameterId	Возвращает или задаёт уникальный идентификатор параметра, соответствующего данному условию.
ParameterName	Имя параметра, соответствующего данному условию.
SectionAlias	Псевдоним секции, к полям которой применяется данное условие.
SectionQuery	Возвращает основную секцию запроса.

Имя	Описание
Value	Возвращает или задаёт значение с которым производится сравнение.

Condition.AggregateConditionGroup — свойство

Возвращает подгруппу условий, которые будут накладываться на агрегируемые значения.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ConditionGroup AggregateConditionGroup { get; }
```

Значение свойства

Тип: ConditionGroup

Подгруппа условий

Заметки

Например, с помощью таких условий можно построить запрос вида "Число исполненных задач по документу = X".

Condition.AggregateDistinctValues — свойство

Возвращает или задаёт признак, должна ли агрегационная функция учитывать только различные значения, или все.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool AggregateDistinctValues { get; set; }
```

Значение свойства

Тип: System.Boolean

`true` — при агрегации будут учтены только уникальные значения, иначе — `false`

Condition.AggregateFunction — свойство

Имя агрегирующей функции для значения секции.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract string AggregateFunction { get; set; }
```

Значение свойства

Тип: `System.String`

Имя агрегирующей функции

Заметки

В качестве значений используются SQL-имена функций (например COUNT, MIN, MAX и т.д.). Агрегация производится по основной секции запроса (SectionQuery). Например, при помощи агрегирующих функций можно построить условия вида "Количество связанных файлов = X", или "Количество согласующих лиц — максимальное".

Condition.Alias — свойство

Возвращает псевдоним условия.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract string Alias { get; }
```

Значение свойства

Тип: `System.String`

Псевдоним

Заметки

Может использоваться при необходимости построить два разных условия по одному и тому же полю.

Condition.ConditionGroup — свойство

Возвращает группу, которой принадлежит условие.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ConditionGroup ConditionGroup { get; }
```

Значение свойства

Тип: ConditionGroup

Группа условий

Condition.FieldAlias — свойство

Возвращает или задаёт псевдоним поля по которому строится условие.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string FieldAlias { get; set; }
```

Значение свойства

Тип: System.String

Псевдоним поля

Condition.FieldSubtype — свойство

Возвращает или задаёт подтип свойства.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract FieldSubtype FieldSubtype { get; set; }
```

Значение свойства

Тип: FieldSubtype

Подтип

Заметки

Используется для корректной типизации значений при поиске среди стандартных свойств.

Condition.FieldSubtypeId — свойство

Возвращает или задаёт идентификатор типа универсального справочника.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid FieldSubtypeId { get; set; }
```

Значение свойства

Тип: System.Int32

Идентификатор подтипа

Заметки

Используется при поиске по свойствам.

Condition.FieldType — свойство

Тип поля по которому строиться условие.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract FieldType FieldType { get; set; }
```

Значение свойства

Тип: FieldType

Тип поля

Condition.Index — свойство

Возвращает индекс условия внутри группы условий.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract int Index { get; }
```

Значение свойства

Тип: System.Int32

Индекс

Condition.IsParameter — свойство

Требует подставлять конкретное значение данного условия в окончательную строку поиска.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool IsParameter { get; set; }
```

Значение свойства

Тип: System.Boolean

Если true, тогда при получении окончательной строки поиска на его место может быть подставлено конкретное значение, иначе — false

Condition.IsSearchWord — свойство

Признак наличия в условии специального поискового слова разрешаемого в значение карточкой.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract bool IsSearchWord { get; set; }
```

Значение свойства

Тип: `System.Boolean`

`true` — в условии фигурирует специальное поисковое слово, иначе — `false`

Заметки

Признак того, что в условии фигурирует специальное поисковое слово, которое разрешается в актуальное значение самой карточкой (поисковые слова задаются в схеме карточки, например, `Today` или `I`).

Condition.IsSystemField — свойство

Возвращает или задаёт признак принадлежности поля к системным полям.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract bool IsSystemField { get; set; }
```

Значение свойства

Тип: `System.Boolean`

`true` — является системным полем, иначе — `false`

Заметки

К числу таких полей относятся поля системных таблиц, например, `IsTemplate`, `Deleted`, `RowID` и т.п.

Condition.Operation — свойство

Сравнительная операция для данного условия.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ConditionOperation Operation { get; set; }
```

Значение свойства

Тип: ConditionOperation

Тип операции

Condition.ParameterAlias — свойство

Возвращает или задаёт псевдоним параметра, соответствующего данному условию.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string ParameterAlias { get; set; }
```

Значение свойства

Тип: System.String

Псевдоним

Condition.ParameterId — свойство

Возвращает или задаёт уникальный идентификатор параметра, соответствующего данному условию.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid ParameterId { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор параметра

Заметки

Этот идентификатор автоматически генерируется при установке соответствующего признака (IsParameter = true).

Condition.ParameterName — свойство

Имя параметра, соответствующего данному условию.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string ParameterName { get; set; }
```

Значение свойства

Тип: System.String

Имя параметра

Заметки

Это имя будет отображаться пользователю в диалоге параметров при выполнении запроса.

Condition.SectionAlias — свойство

Псевдоним секции, к полям которой применяется данное условие.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string SectionAlias { get; set; }
```

Значение свойства

Тип: System.String

Псевдоним секции

Заметки

Если значение не указано, то это основная секция, запроса SectionQuery, в противном случае — присоединенная секция.

Condition.SectionQuery- свойство

Возвращает основную секцию запроса.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
Public abstract SectionQuery SectionQuery { get; }
```

Значение свойства

Тип: SectionQuery

Секция запроса

Condition.Value — свойство

Возвращает или задаёт значение с которым производится сравнение.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract object Value { get; set; }
```

Значение свойства

Тип: `System.Object`

Объект сравнения

ConditionGroup — класс

Представляет иерархическую группу условий поиска.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class ConditionGroup
```

Свойства

Имя	Описание
`Alias`	Возвращает псевдоним группы.
`ConditionGroups`	Возвращает коллекцию вложенных групп условий поиска.
`Conditions`	Возвращает коллекцию условий поиска.
`Index`	Возвращает индекс текущей группы.
`Operation`	Задаёт или возвращает логическое условие применимое к группе поиска.
`Parent`	Возвращает вышестоящий элемент в иерархии поиска.
`SectionQuery`	Возвращает методы поиска по секциям.

JoinSection — класс

Предоставляет данные присоединенной секции в поисковом запросе.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class JoinSection
```

Свойства

Имя	Описание
Alias	Задаёт или возвращает псевдоним данного присоединения (присоединения секции).
Id	Задаёт или возвращает идентификатор секции, к которой выполнено присоединение.
JoinWith	Задаёт или возвращает псевдоним присоединения, если данное присоединение от другого присоединения.
ReturnFields	*Не используется.*
SectionField	Задаёт или возвращает название поля, к которому выполнено присоединение.
SectionQuery	Задаёт или возвращает текущий поисковый запрос.
TableName	Задаёт или возвращает название SQL-таблицы, если выполнено присоединение к таблице базы данных. Указывать ID не требуется.
WithField	Задаёт или возвращает псевдоним поля, от которого присоединяется секция.

Примеры

В качестве примера, найдем документы, автором которых является сотрудник с табельным номером 456.

```
SearchQuery searchQuery = userSession.CreateSearchQuery(); ①

CardTypeQuery cardTypeQuery = searchQuery.AttributiveSearch.CardTypeQueries.AddNew
```

```
(CardDocument.ID); ②

SectionQuery sectionQuery = cardTypeQuery.SectionQueries.AddNew(CardDocument.MainInfo.
ID); ③

JoinSection joinSection = sectionQuery.JoinSections.AddNew("Author1"); ④

joinSection.Id = RefStaff.Employees.ID; ⑤

joinSection.SectionField = "RowID"; ⑥

joinSection.WithField = CardDocument.MainInfo.Author; ⑦

Condition condition = sectionQuery.ConditionGroup.Conditions.AddNew(RefStaff.Employees
.ClockNumber, FieldType.Unistring, ConditionOperation.Equals, "456"); ⑧

condition.SectionAlias = joinSection.Alias; ⑨

CardDataCollection result = userSession.CardManager.FindCards(searchQuery.GetXml()); ⑩
```

① Создаем поисковый запрос.

② Создаем поиск по типу карточки.

③ Добавляем поиск по секции карточки Основная информация.

④ Создаем присоединяемую секцию с псевдонимом `Author1`.

⑤ Указываем идентификатор секции, к которой присоединяемся — сотрудники Справочника сотрудников.

⑥ Указываем идентификатор поля, к которому присоединяемся — идентификатор сотрудника.

⑦ Указываем псевдоним поля основной секции, по которому выполняется присоединение — автор документа.

⑧ Добавляем условие поиска по табельному номеру.

⑨ Указываем псевдоним секции, по которой выполняется поиск — псевдоним присоединенной секции.

⑩ Выполняем поиск и получаем результат — карточки, автором которых является сотрудник с табельным номером 456.

SearchQuery — класс

Базовый объект для построения поискового запроса. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class SearchQuery
```

Свойства

Имя	Описание
AllCards	Возвращает или задаёт признак включения в поисковый запрос всех карточек.
AttributiveSearch	Возвращает методы атрибутивного поиска.
CombineResults	Возвращает или задаёт флаг объединения результатов поиска.
DirectXml	Возвращает или задаёт значение, указывающее, может ли пользователь непосредственно изменить XML.
FullTextSearch	Возвращает методы полнотекстового поиска.
IncludeArchived	Возвращает или задаёт требование включения в область поиска архивных карточек.
IncludeDeleted	Возвращает или задаёт требование включения в область поиска удалённых карточек.
IncludeLinkedCards	Возвращает или задаёт требование включения в область поиска карточки связанные с найденными ссылкой.
IncludeTopicCards	Позволяет включить в результаты поиска карточки, принадлежащие к одной теме обработки с найденными.

Имя	Описание
Limit	Возвращает или задаёт максимальное количество карточек включённых в результат поиска.
LinkDirections	Возвращает или задаёт направление ссылок при истинности значения IncludeLinkedCards.
LinkTypes	Возвращает или задаёт тип ссылок при истинности значения IncludeLinkedCards.
Parameters	Коллекция параметров атрибутивного поиска.
Scope	Возвращает область поиска.
Session	Возвращает текущую открытую сессию пользователя.

Методы

Имя	Описание
Clear	Удаление результатов поиска.
GetParametrizedConditions	Возвращает все параметризованные условия, используемые в поисковом запросе.
GetParametrizedConditions(String)	Возвращает условие с указанным псевдонимом.
GetResolvedParameters	Возвращает параметры поиска.
GetXml	Возвращает сформированную строку запроса в формате XML.
GetXml(QueryParameterCollection)	Возвращает сформированную строку запроса в формате XML с указанием дополнительных параметров поиска.
GetXml(QueryParameterCollection, Boolean)	Возвращает сформированную строку запроса в формате XML с указанием дополнительных параметров поиска и признаком отсутствия пустых настроек.

Имя	Описание
`ParseXml(String)`	Загружает все параметры поиска из XML-строки.

SectionQuery — класс

Содержит методы для организации поиска по секциям карточки.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class SectionQuery
```

Свойства

Имя	Описание
`CardTypeQuery`	Возвращает запрос к карточке.
`ConditionGroup`	Возвращает поисковую группу.
`JoinSections`	Возвращает присоединенные секции.
`Limit`	Задаёт или возвращает максимальное количество карточек в результате поиска.
`Operation`	Задаёт или возвращает логическую операцию при группировке условий.
`Parameters`	Возвращает коллекцию параметров для поискового запроса.
`ReturnFields`	Возвращает или задаёт псевдонимы полей, возвращаемых из поискового запроса.
`SearchQuery`	Возвращает объект содержащий поисковый запрос.
`SectionId`	Возвращает или задаёт идентификатор секции к которой применяется запрос.

Имя	Описание
`Session`	Возвращает текущую сессию пользователя.
`TableName`	Возвращает или задаёт название системной таблицы.

Методы

Имя	Описание
`GetParametrizedConditions`	Возвращает все параметризованные условия, используемые в поисковом запросе.
`GetParametrizedConditions(String)`	Возвращает условие с указанным псевдонимом.
`GetResolvedParameters`	Возвращает параметры поиска.
`GetXml`	Возвращает сформированную строку запроса в формате XML.
`GetXml(QueryParameterCollection)`	Возвращает сформированную строку запроса в формате XML с указанием дополнительных параметров поиска.
`GetXml(QueryParameterCollection, Boolean)`	Возвращает сформированную строку запроса в формате XML с указанием дополнительных параметров поиска и признаком отсутствия пустых настроек.
`ParseXml(String)`	Загружает все параметры поиска из XML-строки.
`SelectCondition(String, String, ConditionOperation)`	Возвращает условие применимое к указанной секции и полю.

SectionQueryCollection — класс

Предоставляет методы для работы с коллекцией `SectionQuery`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class SectionQueryCollection : IEnumerable<SectionQuery>, IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов коллекции.
`Item`	Возвращает элемент коллекции с заданным идентификатором.
`Item`	Возвращает элемент коллекции с заданным индексом.
`SearchQuery`	Возвращает объект типа `SearchQuery`.

Методы

Имя	Описание
`AddNew(Guid)`	Добавляет к поисковому запросу секцию с заданным идентификатором.
`Clear`	Удаляет все элементы коллекции.
`Contains(Guid)`	Проверяет вхождение в коллекцию заданной секции.
`GetEnumerator`	Возвращает перечислитель.
`Remove(Guid)`	Удаляет из коллекции заданную секцию.
`Remove(Int32)`	Удаляет из элемента с указанным индексом.

ConditionGroupOperation — перечисление

Логическая операция для группировки условий.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum ConditionGroupOperation
```

Члены

Имя члена	Описание
And	Логическое И.
Or	Логическое ИЛИ.

ConditionOperation — перечисление

Тип сравнительной операции для поискового запроса.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public enum ConditionOperation
```

Члены

Имя члена	Описание
None	Не задано.
Equals	Равно.
NotEquals	Не равно.
LessEqual	Меньше или равно.
LessThan	Меньше.
GreaterEqual	Больше или равно.
GreaterThan	Больше.
StartsWith	Строка начинает с.
EndsWith	Заканчивается на.
Contains	Содержит.
NotContains	Не содержит.

Имя члена	Описание
`IsNull`	Равно null.
`IsNotNull`	Не null.
`OneOf`	Значение в списке аргументов.
`NotOneOf`	Значение не в списке аргументов.
`StrEquals`	Строки равны. Рекомендуется к использованию Equals.
`StrNotEquals`	Строки не равны. Рекомендуется к использованию `NotEquals`.
`Between`	Значение внутри диапазона аргументов.
`NotBetween`	Значение за пределами диапазона аргументов.
`IsChildOf`	Является дочерним элементом в иерархии.
`IsParentOf`	Является родительским элементом в иерархии.

FieldSubtype — перечисление

Определяет тип значения в поле.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum FieldSubtype
```

Члены

Имя члена	Описание
`None`	Не установлен.
`String`	Строка.
`Int`	Целое число.

Имя члена	Описание
`Double`	Дробное число.
`Date`	Дата.
`Bool`	Булевое значение.
`Employee`	Сотрудник.
`Department`	Подразделение.
`Group`	Группа доступа.
`Role`	Исполняемая роль.
`Universal`	Значение универсального справочника.
`Time`	Время.
`DateTime`	Дата и время.
`PartnerEmployee`	Сотрудник контрагента.
`PartnerDepartment`	Подразделение контрагента.
`CardRef`	Ссылка на карточку.
`CardType`	Вид (из справочника типов).
`CardState`	Состояние (из справочника типов).

Заметки

Используется для корректной типизации значений при поиске среди стандартных свойств.

FullTextSearch.Mode — свойство

Получает или задаёт режим поиска.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract FullTextSearchMode Mode { get; set; }
```

Значение свойства

Тип: `FullTextSearchMode`

Режим полнотекстового поиска.

Заметки

`FullTextSearchMode` *может принимать следующие значения:*

- Карточки и файлы: `CardsAndFiles = 0`
- Карточки: `Cards = 1`
- Файлы: `Files = 2`
- Поиск по имени файла: `FilesByName = 3`

Поиск цитат возможен только если режим поиска включает файлы, т.е. свойство `Mode` выбрано как `CardsAndFiles` или `Files`.

FullTextSearch.QueryString — свойство

Получает или задаёт значение поисковой строки.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract string QueryString { get; set; }
```

Значение свойства

Тип: `System.String`

Поисковая строка.

FullTextSearch.SearchQuery — свойство

Получает поисковый запрос.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract SearchQuery SearchQuery { get; }
```

Значение свойства

Тип: SearchQuery

Поисковый запрос.

FullTextSearch.WithCitations — свойство

Определяет, будут ли использованы цитаты из файлов в результатах поиска.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SearchModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual bool WithCitations
```

Значение свойства

Тип: System.Boolean

true — результаты поиска дополняются цитатами из файлов, подходящим под поисковый запрос. false — не дополняются.

Заметки

Поиск цитат возможен только если режим поиска включает файлы, т.е. свойство Mode выбрано как CardsAndFiles или Files.

Пример

```
SearchQuery searchQuery = session.CreateSearchQuery("<Search Version=\"4300\"
Limit=\"100\" CombineResults=\"OR\"><Scope/><FulltextSearch
Mode=\"CardsAndFiles\"><QueryString>проверить</QueryString></FulltextSearch></Search>");
searchQuery.FullTextSearch.WithCitations = true;
```

Чтобы получить цитаты, необходимо применить представление на результаты соответствующего поиска любым доступным в платформе способом.

Например, наполнить папку результатами поиска и применить представление на неё. На представление ограничений нет.
Цитаты будут храниться в системном поле представления _FilesCitations. Если

цитат нет или поиск их не поддерживал, значение будет `null`, либо пустым.

Цитаты в столбце хранятся в виде `.xml` следующего формата:

```
<Citations HasMore="False"/>
<File ID="GUID" HasMore = "True">
    <Citation> ... </Citation>
...
</File>
...
</Citations>
```

Символ `<` экранируется как `<`, а символ `>` как `>`.

FullTextSearch — класс

Реализует модель полнотекстового серверного поиска.

Синтаксис

```
public abstract class FullTextSearch
```

Свойства

Имя	Описание
Mode	Получает или задаёт режим поиска.
QueryString	Получает или задаёт значение поисковой строки.
SearchQuery	Получает поисковый запрос.
WithCitations	Определяет, будут ли использованы цитаты из файлов в результатах поиска.

SearchAggregationItemsResultType — перечисление

Результат поиска агрегатов.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SearchModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum SearchAggregationItemsResultType
```

Члены

Имя	Описание
`Cards`	Только сохранить результаты фильтрации в папку.
`Aggregations`	Вернуть только список агрегатов.
`All`	Вернуть список агрегатов и сохранить результаты фильтрации в папку.

DocsVision.Platform.ObjectManager.SystemCards — пространство имён

Содержит классы с реализацией объектов бизнес-логики для **системных карточек**.

Классы

Класс	Описание
`FileCommentCollection`	Содержит методы для работы с коллекцией FileComment.
`FileVersion`	Дополняет класс файла возможностью хранения версий файла.
`FileVersionCollection`	Содержит методы для работы с коллекцией FileVersion.
`Folder`	Специализированный тип отражающий семантику объекта 'Папка'.
`FolderCard`	Специализированный тип отражающий семантику объекта "Карточка папок".
`FolderCardType`	Определяет тип "Карточки папок".
`FolderCardTypeCollection`	Содержит методы для работы с коллекцией `FolderCardType`.
`FolderCollection`	Содержит методы для работы с коллекцией Folder.

Класс	Описание
FolderIcon	Предоставляет иконку папки.
FolderIconCollection	Содержит методы для работы с коллекцией FolderIconCollection.
FolderSubtype	Определяет тип подпапки допустимый для папки.
FolderSubtypeCollection	Содержит методы для работы с коллекцией FolderSubtype.
FolderTemplate	Определяет шаблон экспорта допустимый для папки.
FolderTemplateCollection	Содержит методы для работы с коллекцией FolderTemplate.
FolderView	Определяет представление допустимое для папки.
FolderViewCollection	Содержит методы для работы с коллекцией FolderView.
NavigatorCard	Представляет системную карточку Windows-клиент.
NumeratorCard	Содержит свойства и методы работы с экземпляром нумератора.
NumeratorRange	Содержит методы для работы с диапазоном номеров нумератора.
NumeratorZone	Содержит методы для работы с зонами нумератора.
SavedView	Предоставляет доступ к сохранённому описанию представления.
SavedViewCollection	Содержит методы для работы с коллекцией SavedView.
SavedViewGroup	Представляет группу представлений.
SavedViewGroupCollection	Содержит методы для работы с коллекцией SavedViewGroup.

Класс	Описание
SavedVirtualField	Представляет сохранённое описание виртуального поля.
SavedVirtualFieldCollection	Содержит методы для работы с коллекцией SavedVirtualField.
SearchCard	Объектная модель карточки сохранённых запросов.
SearchHistoryCollection	Содержит методы для работы с коллекцией SearchHistory.
Shortcut	Специализированный тип отражающий семантику объекта "Ярлык".
ShortcutCollection	Содержит методы для работы с коллекцией Shortcut.
SystemCard	Базовый класс для системных карточек.
UserProfileCard	Карточка содержащая профиль пользователя.
VersionedFileCard	Реализует логику карточки **файл с версиями**.
ViewCard	Представляет системную карточку сохранённых представлений.
ViewColumnSettings	Представляет настройки колонки представления.
ViewColumnSettingsCollection	Содержит методы для работы с коллекцией ViewColumnSettings.
ViewGrouping	Представляет настройки группировки представления.
ViewSettings	Предоставляет доступ к параметрам представления.
ViewSettingsCollection	Содержит методы для работы с коллекцией ViewSettings.
ViewSorting	Представляет настройки сортировки представления.

Класс	Описание
`ViewTemplate`	Представляет настройки шаблона экспорта представления.
`ViewTemplateCollection`	Содержит методы для работы с коллекцией ViewTemplate.

Перечисления

Перечисление	Описание
`AggregationType`	Определяет операцию агрегации для табличных данных на стороне клиента.
`AlignType`	Определяет тип выравнивания элемента в представлении.
`FolderFlags`	Дополнительные атрибуты папки.
`FolderRestrictions`	Определяет типы ограничений у папки
`FoldersInfoFlags`	Дополнительные управляющие флаги отображения элементов в папке.
`FolderStyles`	Определяет стиль отображения папки.
`FolderTypes`	Определяет тип папок для поисковых запросов.
`GridLineModes`	Определяет режим отображения линий таблицы.
`GridLineStyle`	Определяет стиль линий таблицы.
`NumberStatus`	Определяет статус номера в **нумераторе**.
`ViewFlags`	Определяет дополнительные флаги регулирующие вид представления.

FileCommentCollection — класс

Содержит методы для работы с коллекцией `FileComment`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class FileCommentCollection : IEnumerable<FileComment>, IEnumerable
```

Свойства

Имя	Описание
Count	Возвращает количество элементов коллекции.
Session	Возвращает сессию пользователя.
Item	Возвращает элемент коллекции с указанным идентификатором комментария.
Item	Возвращает элемент коллекции с заданным индексом.

Методы

Имя	Описание
AddNew(String)	Добавляет комментарий в коллекцию.
AddNew(String, Guid)	Добавляет в коллекцию комментарий от имени указанного сотрудника.
Clear	Удаляет все элементы коллекции.
Contains(Guid)	Проверяет вхождение в коллекцию элемента с заданным идентификатором.
GetEnumerator	Возвращает перечислитель.
Remove(Guid)	Удаляет комментарий с заданным идентификатором из коллекции.
Remove(Int32)	Удаляет комментарий с заданным индексом.

FileVersion — класс

Дополняет класс файла возможностью хранения версий файла. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class FileVersion : FileData
```

Свойства

Имя	Описание
AssociatedFiles	Коллекция ассоциированных с версией файлов
AuthorId	Идентификатор автора версии файла.
Comments	Коллекция комментариев к версии файла.
ExtCardId	Возвращает или задаёт идентификатор карточки расширения.
IncrementalNumber	Порядковый номер версии.
Level	Уровень версии файла.
VersionId	Уникальный идентификатор версии.
VersionNumber	Номер версии на данном уровне.
Versions	Коллекция подверсий файла.
VersionString	Возвращает строковое представление текущей версии.

Методы

Имя	Описание
GetHierarchy	Получение иерархии версий и подверсий файла.
MakeCurrent	Установка данной версии в качестве текущей.

Folder — класс

Специализированный тип отражающий семантику объекта 'Папка'.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class Folder : IUpdatable
```

Свойства

Имя	Описание
AllowedCardTypes	Возвращает список типов карточек, которые можно создавать в этой папке.
AllowedFolderTypes	Возвращает список типы подпапок, которые можно создавать в этой папке.
AllowedStyles	Задаёт или возвращает стили, разрешённые к использованию в данной папке.
AllowedTemplates	Возвращает список разрешённых к использованию в папке шаблонов экспорта.
AllowedViews	Возвращает список представлений, которые можно использовать в этой папке.
CreateDate	Возвращает дату создания папки.
CreatedBy	Возвращает название учетной записи создателя папки.
CurrentStyle	Задаёт или возвращает текущий стиль отображения папки.
CurrentViewId	Задаёт или возвращает идентификатор текущего представления папки.
DefaultStyle	Задаёт или возвращает стиль отображения содержимого папки по умолчанию.
DefaultTemplateId	Задаёт или возвращает идентификатор шаблон экспорта.

Имя	Описание
DefaultViewId	Задаёт или возвращает идентификатор представления папки по умолчанию.
Deleted	Возвращает признак того, что папка была удалена.
ExtTypeId	Задаёт или возвращает идентификатор расширенного типа папки.
Flags	Задаёт или возвращает дополнительные атрибуты папки.
Folders	Возвращает коллекцию дочерних папок.
HasSubfolders	Признак наличия дочерних папок.
IconId	Возвращает или задаёт идентификатор иконки.
Id	Возвращает идентификатор папки.
InUpdate	Возвращает признак включения для объекта режима отложенных изменения.
Name	Возвращает или задаёт название папки.
ParentFolder	Возвращает ссылку на родительскую папку.
PropCardId	Задаёт или возвращает идентификатор карточки папки.
RefId	Задаёт или возвращает идентификатор оригинальной папки, для папки-делегата.
RefreshTimeout	Возвращает или задаёт индивидуальный период обновления содержимого папки.
Restrictions	Возвращает или задаёт признак наличия ограничений папки.

Имя	Описание
SavedParameters	Возвращает коллекцию сохранённых значений параметров поискового запроса.
Session	Возвращает текущую сессию пользователя.
Shortcuts	Возвращает коллекцию ярлыков папки.
Type	Задаёт или возвращает тип папки.
Url	Задаёт или возвращает URL папки.
ViewCycleCount	Задаёт или возвращает максимальное количество выводимых записей.
ViewCyclingEnabled	Задаёт или возвращает признак режима ограничения количества выводимых записей.

Методы

Имя	Описание
BeginUpdate	Включает режим отложенных изменений.
CancelUpdate	Отменяет несохранённые изменения и выключает режим отложенных изменений.
Copy(Guid)	Копирование папки в указанную папку.
Copy(Guid, String)	Копирование папки в указанную папку с изменением конечного имени.
CopyData(Guid, String, Boolean)	Копирование данных папки в указанную папку с изменением имени конечного каталога.
EndUpdate	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.

Имя	Описание
FillBySearch(Guid)	Заполнение папки результатом поискового запроса с заданным идентификатором.
FillBySearch(String)	Заполнение папки результатом переданного поискового запроса.
FillBySearch(Guid, QueryParameterCollection)	Заполнение папки результатом поискового запроса с заданным идентификатором и указанными параметрами.
GetHierarchy	Возвращает идентификаторы родительских папок, согласно иерархии.
GetUnreadCount	Возвращает количество непрочитанных карточек в папке.
GetUnreadCount(Boolean)	Возвращает количество непрочитанных карточек в папке, включая архивированные элементы.
MarkAll(Boolean)	Изменяет статус прочитанности для всех карточек.
MarkCardForDeletion(Guid)	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
Move(Guid)	Перемещает папку в указанную папку.
Purge(Boolean, DateTime)	Удаляет ярлыки принадлежащие папки и помеченные к удалению, в соответствии с заданными условиями.
Refresh	Обновляет информацию о папке.
UpdateNow	Позволяет отправить накопленные изменения на сервер.

Folder.CurrentStyle — свойство

Задаёт или возвращает текущий стиль отображения папки.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract FolderStyles CurrentStyle { get; set; }
```

Значение свойства

Тип: FolderStyles

Стиль отображения папки

Folder.Flags — свойство

Задаёт или возвращает дополнительные атрибуты папки.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract FolderFlags Flags { get; set; }
```

Значение свойства

Тип: FolderFlags

Атрибуты папки

Folder.Restrictions — свойство

Возвращает или задаёт признак наличия ограничений папки.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract FolderRestrictions Restrictions { get; set; }
```

Значение свойства

Тип: FolderRestrictions

Установленные ограничения

Заметки

Наличие флага может использоваться для превентивной проверки ограничений, без дорогостоящего чтения всех списков ограничений.

Folder.PropCardId — свойство

Задаёт или возвращает идентификатор карточки папки.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid PropCardId { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор карточки

Заметки

Карточка папки — карточка, ассоциированная с данной папкой, которая может отображаться в правой части Windows-клиента при переходе в папку.

FolderCard — класс

Специализированный тип отражающий семантику объекта 'Карточка папок'.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class FolderCard : SystemCard
```

Свойства

Имя	Описание
`Folders`	Возвращает коллекцию папок верхнего уровня.
`Icons`	Возвращает коллекцию иконок папок.
`SearchCardId`	Задаёт или возвращает идентификатор карточки сохранённых поисковых запросов.
`ViewCardId`	Задаёт или возвращает идентификатор текущей карточки сохранённых представлений.

Методы

Имя	Описание
`CreateFolder(Guid, String)`	Создаёт в указанной папке с подпапку с указанным именем.
`CreateIcon(Icon, String)`	Загружает иконку с описанием в коллекцию иконок.
`CreateShortcut(Guid, Guid, Boolean)`	Создаёт для заданной карточки ярлык в указанной папке.
`DeleteCard(Guid, Boolean)`	Помечает к удалению, либо удаляет карточку с заданным идентификатором.
`DeleteFolder(Guid, Boolean)`	Помечает к удалению, либо удаляет папку с указанным идентификатором.
`DeleteIcon(Guid)`	Удаляет заданную иконку.
`DeleteShortcut(Guid, Boolean)`	Помечает к удалению, либо удаляет ярлык с указанным идентификатором.
`FindHardLink(Guid)`	Получение сильного ярлыка на карточку.
`FolderExists(Guid)`	Проверяет существование папки с указанным идентификатором.
`GetFolder(Guid)`	Получает объект типа `Folder` по идентификатору папки.

Имя	Описание
GetFolderState(Guid)	Проверяет состояния папки с указанным идентификатором.
GetIcon(Guid)	Возвращает объект типа FolderIcon по идентификатору иконки.
GetShortcut(Guid)	Получает объект для работы с ярлыком по его идентификатору.
GetShortcuts(Guid)	Получает все ярлыки для заданной карточки.
GetShortcutState(Guid)	Определяет состояние ярлыка с указанным идентификатором.
RestoreFolder(Guid)	Восстанавливает папку, помеченную к удалению.
RestoreShortcut(Guid)	Восстанавливает ярлык помеченный к удалению.
ShortcutExists(Guid)	Проверяет существование ярлыка с заданным идентификатором.

FolderCard.DeleteCard — метод (Guid, Boolean)

Помечает к удалению, либо удаляет карточку с заданным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void DeleteCard(Guid cardId, bool permanently)
```

Параметры

cardId

Тип: System.Guid

Идентификатор карточки

permanently

Тип: System.Boolean

`true` — производит стандартное удаление; при `false` — помечает карточку к удалению

Заметки

Карточка удаляет с учетом существующих ярлыков.

Примеры

В примере, карточка с идентификатором `00000000-0000-0000-0000-000000000000` помечает к удалению. `userSession` — текущая сессия пользователя

```
FolderCard folderCard = (FolderCard)userSession.CardManager.GetDictionary(new Guid
("DA86FABF-4DD7-4A86-B6FF-C58C24D12DE2")); ①

folderCard.DeleteCard(new Guid("00000000-0000-0000-0000-000000000000"), false); ②
```

① Получаем карточку папок.

② Помечаем карточку к удалению.

NavigatorCard — класс

Представляет системную карточку **Windows-клиент**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class NavigatorCard : SystemCard
```

Свойства

Имя	Описание
`ChangeDate`	Дата последнего изменения данных в карточке.
`Settings`	Возвращает объект содержащий дополнительные настройки Windows-клиента.
`ViewSettings`	Возвращает коллекцию настроек представлений.

Имя	Описание
ViewTemplates	Возвращает коллекцию шаблонов представлений.
FolderRootName	Возвращает или задаёт имя корневой папки.
UserSettings	Возвращает пользовательские настройки Windows-клиента.

Методы

Имя	Описание
GetViewSettings(Guid)	Возвращает настройки заданного представления.

NavigatorCard.ViewSettings — свойство

Возвращает коллекцию настроек представлений.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ViewSettingsCollection ViewSettings { get; }
```

Значение свойства

Тип: ViewSettingsCollection

Коллекция настроек представлений

NumeratorCard — класс

Содержит свойства и методы работы с экземпляром нумератора.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class NumeratorCard : SystemCard
```

Свойства

Имя	Описание
Available	Признак доступности нумератора для выдачи номеров.
EndNumber	Возвращает конечный номер нумератора.
Name	Получает или задаёт название нумератора.
Ranges	Возвращает коллекцию диапазонов нумератора.
StartNumber	Возвращает начальный номер нумератора.
Zones	Возвращает коллекцию зон нумератора.

Методы

Имя	Описание
AllocateNumbers(Guid, String, Int32, Int32)	Резервирует за пользователем указанный диапазон номеров.
ChangeEndNumber(Int32, Boolean)	Изменяет конечный номер нумератора.
ChangeStartNumber(Int32, Boolean)	Изменяет начальный номер нумератора.
GetNumberInfo(Guid)	Получение информации по заданному номеру нумератора.
GetNumbersStatus(String, Int32, Int32)	Возвращает статусы номеров для заданной зоны из заданного диапазона.
ReleaseNumbers(String, Int32, Int32)	Освобождение группы номеров нумератора.

NumeratorRange — класс

Содержит методы для работы с диапазоном номеров нумератора.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class NumeratorRange : IUpdatable
```

Свойства

Имя	Описание
EndNumber	Возвращает конечный номер диапазона.
Id	Возвращает идентификатор диапазона.
InUpdate	Возвращает признак включения для объекта режима отложенных изменения.
OwnerId	Возвращает идентификатор владельца диапазона номеров.
Session	Возвращает текущую сессию пользователя.
StartNumber	Возвращает начальный номер диапазона.

Методы

Имя	Описание
BeginUpdate	Включает режим отложенных изменений
CancelUpdate	Отменяет несохранённые изменения и выключает режим отложенных изменений.

Имя	Описание
EndUpdate	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
MarkCardForDeletion(Guid)	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
SetEndNumber(Int32)	Устанавливает конченый номер диапазона.
SetOwnerID(String)	Задаёт владельца текущего диапазона номеров.
SetStartNumber(Int32)	Задаёт начальный номер диапазона номеров.
UpdateNow	Позволяет отправить накопленные изменения на сервер.

NumeratorZone — класс

Содержит методы для работы с зонами нумератора.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class NumeratorZone : IUpdatable
```

Свойства

Имя	Описание
Id	Возвращает идентификатор зоны.
InUpdate	Возвращает признак включения для объекта режима отложенных изменения.
Name	Возвращает или задаёт название зоны.

Имя	Описание
`Numerator`	Возвращает нумератор владеющий зоной.
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`AllocateNumber(Guid, Int32)`	Резервирует в текущей зоне заданный номер за заданным пользователем.
`BeginUpdate`	Включает режим отложенных изменений
`CancelUpdate`	Отменяет несохранённые изменения и выключает режим отложенных изменений.
`EndUpdate`	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
`GetNumber(Guid)`	Возвращает номер по его идентификатору.
`GetNumber(Guid, Boolean, Guid)`	Получает номер для заданного пользователя с заданным идентификатором.
`GetNumberId(Int32)`	Возвращает идентификатор по номеру.
`GetNumberStatus(Int32)`	Получение статуса указанного номера.
`MarkCardForDeletion(Guid)`	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
`ReleaseNumber(Guid)`	Освобождение номера с указанным идентификатором.
`ReleaseNumber(Int32)`	Освобождение указанного номера.

Имя	Описание
UpdateNow	Позволяет отправить накопленные изменения на сервер.

Заметки

Зона нумератора представляет собой именованную совокупность номеров. Внутри одного нумератора может существовать несколько параллельных зон, номера в которых совпадают.

SavedView — класс

Предоставляет доступ к сохранённому описанию представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class SavedView : IUpdatable
```

Свойства

Имя	Описание
Comments	Задаёт или возвращает дополнительную информацию о представлении.
Hidden	Задаёт или возвращает признак того, что представление является скрытым.
Id	Возвращает идентификатор представления.
InUpdate	Возвращает признак включения для объекта режима отложенных изменения.
Name	Возвращает или задаёт название представления.
Session	Возвращает текущую сессию пользователя.

Имя	Описание
Sortings	Возвращает правило сортировки в колонках представления.
Text	Задаёт или возвращает строку XML-описания представления.

Методы

Имя	Описание
BeginUpdate	Включает режим отложенных изменений.
CancelUpdate	Отменяет несохранённые изменения и выключает режим отложенных изменений.
Copy(Guid)	Создаёт копию представления в указанной группе представлений.
EndUpdate	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
Export	Позволяет получить объект для упрощенной работы с представлением.
Import(View)	Заполняет XMl-описание данными из переданного представления.
MarkCardForDeletion(Guid)	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
Move(Guid)	Изменяет группу текущего описания представления.
Publish	Инициирует безусловную генерацию хранимых процедур для получения данных представления.
Revoke	Удаляет хранимые процедуры представления из БД.

Имя	Описание
UpdateNow	Позволяет отправить накопленные изменения на сервер.

SavedView.Comments — свойство

Задаёт или возвращает дополнительную информацию о представлении.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Comments { get; set; }
```

Значение свойства

Тип: System.String

Комментарий к представлению

SavedView.Hidden — свойство

Задаёт или возвращает признак того, что представление является скрытым.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool Hidden { get; set; }
```

Значение свойства

Тип: System.Boolean

true — представление скрыто, иначе — false

Заметки

Скрытое представление не отображает пользователю в Windows-клиенте. Аналогично может использоваться в сторонних решениях.

SavedView.Id — свойство

Возвращает идентификатор представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid Id { get; }
```

Значение свойства

Тип: System.Int32

Идентификатор

SavedView.InUpdate — свойство

Возвращает признак включения для объекта режима отложенных изменения.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool InUpdate { get; }
```

Значение свойства

Тип: System.Boolean

`true` — режим отложенных изменений включён, иначе — `false`

SavedView.Name — свойство

Возвращает или задаёт название представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Name { get; set; }
```

Значение свойства

Тип: System.String

Название представления

SavedView.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Сессия пользователя

SavedView.Sortings — свойство

Возвращает правило сортировки в колонках представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract object Sortings { get; }
```

Значение свойства

Тип: System.Object

Вложенные группы сортировки

Заметки

Высока вероятность получения исключения типа System.NotSupportedException.

SavedView.Text — свойство

Задаёт или возвращает строку XML-описания представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Text { get; set; }
```

Значение свойства

Тип: System.String

Описание представления

SavedView.BeginUpdate — метод

Включает режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void BeginUpdate()
```

SavedView.CancelUpdate — метод

Отменяет несохранённые изменения и выключает режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void CancelUpdate()
```

SavedView.Copy — метод (Guid)

Создаёт копию представления в указанной группе представлений.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SavedView Copy(Guid parentGroupId)
```

Параметры

parentGroupId

Тип: System.Guid

Идентификатор группы представлений, в которой будет создана копия

Возвращаемое значение

Тип: SavedView

Созданная копия представления

SavedView.EndUpdate — метод

Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void EndUpdate()
```

SavedView.Export — метод

Позволяет получить объект для упрощенной работы с представлением.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract View Export()
```

Возвращаемое значение

Тип: View

Описание представления

Примеры

```
const string VIEW_CARD_ID = "17F8F0B3-7E93-45E9-B250-EED4E93F3FA3";
ViewCard vcard = (ViewCard)session.CardManager.GetDictionary(new Guid(VIEW_CARD_ID));

SavedView sview = vcard.Views[0];
DocsVision.Platform.ObjectManager.ViewModel.View view = sview.Export();
```

SavedView.Import — метод (View)

Заполняет XMI-описание данными из переданного представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Import(View view)
```

Параметры

view

Тип: View

Описание представления

SavedView.MarkCardForDeletion — метод (Guid)

Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void MarkCardForDeletion(Guid cardId)
```

Параметры

cardId

Тип: System.Guid

Идентификатор карточки

SavedView.Move — метод (Guid)

Изменяет группу текущего описания представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Move(Guid parentGroupId)
```

Параметры

parentGroupId

Тип: System.Guid

Идентификатор группы назначения

SavedView.Publish — метод

Инициирует безусловную генерацию хранимых процедур для получения данных представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Publish()
```

Заметки

Повторную генерацию требуется производить при каждом изменении описания представления.

SavedView.Revoke — метод

Удаляет хранимые процедуры представления из БД.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Revoke()
```

SavedView.UpdateNow — метод

Позволяет отправить накопленные изменения на сервер.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void UpdateNow()
```

SavedViewGroup — класс

Представляет группу представлений.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class SavedViewGroup : IUpdatable
```

Свойства

Имя	Описание
Groups	Возвращает коллекцию групп представлений верхнего уровня.
Id	Возвращает идентификатор текущей группы.

Имя	Описание
InUpdate	Возвращает признак включения для объекта режима отложенных изменения.
Name	Задаёт или возвращает название группы.
ParentGroup	Возвращает родительскую группу представлений.
Session	Возвращает текущую сессию пользователя.
Views	Возвращает коллекцию представлений группы.

Методы

Имя	Описание
BeginUpdate	Включает режим отложенных изменений.
CancelUpdate	Отменяет несохранённые изменения и выключает режим отложенных изменений.
Copy(Guid)	Копирует группу в другую группу представлений.
EndUpdate	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
GetHierarchy	Получение иерархии идентификаторов групп начиная к корневого элемента.
MarkCardForDeletion(Guid)	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
Move(Guid)	Переносит текущую группы в новую родительскую группу представлений.

Имя	Описание
Refresh	Обновляет информацию о группе.
UpdateNow	Позволяет отправить накопленные изменения на сервер.

SavedViewGroupCollection — класс

Содержит методы для работы с коллекцией SavedViewGroup.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class SavedViewGroupCollection : IEnumerable<SavedViewGroup>, IEnumerable
```

Свойства

Имя	Описание
Count	Возвращает количество элементов в коллекции.
Item	Возвращает элемент коллекции с заданным идентификатором.
Item	Возвращает элемент коллекции с указанным индексом.
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
AddNew(String)	Создаёт элемент типа SavedViewGroup с заданным именем.
Clear	Удаляет все элементы коллекции.
Contains(Guid)	Проверяет вхождение в коллекцию элемента с заданным идентификатором.

Имя	Описание
GetByName(String)	Возвращает сохранённую группу представлений по названию.
GetEnumerator	Возвращает перечислитель.
Remove(Guid)	Удаляет элемент коллекции с указанным идентификатором.
Remove(Int32)	Удаляет элемент коллекции с указанным индексом.

SavedViewCollection — класс

Содержит методы для работы с коллекцией SavedView.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class SavedViewCollection : IEnumerable<SavedView>, IEnumerable
```

Свойства

Имя	Описание
Count	Возвращает количество элементов в коллекции.
Item	Возвращает элемент коллекции с заданным идентификатором.
Item	Возвращает элемент коллекции с указанным индексом.
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
AddNew	Создаёт объект типа SavedView.

Имя	Описание
Clear	Удаляет все элементы коллекции.
Contains(Guid)	Проверяет вхождение в коллекцию элемента с заданным идентификатором.
GetEnumerator	Возвращает перечислитель.
Remove(Guid)	Удаляет элемент с заданным идентификатором.
Remove(Int32)	Удаляет элемент с указанным индексом.

SavedVirtualField — класс

Представляет сохранённое описание виртуального поля.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class SavedVirtualField : IUpdatable
```

Свойства

Имя	Описание
Id	Возвращает идентификатор виртуального поля.
InUpdate	Возвращает признак включения для секции режима отложенных изменения.
Name	Задаёт или возвращает название поля.
SectionId	Возвращает идентификатор родительской секции.
Session	Возвращает текущую сессию пользователя.

Имя	Описание
`Text`	Задаёт или возвращает строку XML-описания поля.

Методы

Имя	Описание
`BeginUpdate`	Включает режим отложенных изменений.
`CancelUpdate`	Отменяет несохранённые изменения и выключает режим отложенных изменений.
`EndUpdate`	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
`Export`	Создаёт на основе сохранённого поля типа `VirtualField`.
`Import(VirtualField)`	Загружает описание в сохранённое поле из объекта типа `VirtualField`.
`MarkCardForDeletion(Guid)`	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
`UpdateNow`	Позволяет отправить накопленные изменения на сервер.

SavedVirtualFieldCollection — класс

Представляет коллекцию виртуальных полей.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class SavedVirtualFieldCollection : IEnumerable<SavedVirtualField>,
IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов в коллекции.
`Item`	Возвращает элемент с заданным идентификатором.
`Item`	Возвращает элемент с указанным индексом.
`Item`	Возвращает элемент с заданным названием.
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`AddNew`	Создаёт объект типа `SavedVirtualField`.
`Clear`	Удаляет все элементы коллекции.
`Contains(Guid)`	Определяет вхождение в коллекцию элемента с заданным идентификатором.
`GetEnumerator`	Возвращает перечислитель.
`Remove(Guid)`	Удаляет элемент с заданным идентификатором.
`Remove(Int32)`	Удаляет элемент с указанным индексом.

Shortcut — класс

Специализированный тип отражающий семантику объекта "Ярлык".

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class Shortcut : IUpdatable
```

Свойства

Имя	Описание
CardId	Задаёт или возвращает идентификатор карточки, на которую ссылается ярлык.
CreationDate	Возвращает дату создания ярлыка.
Deleted	Возвращает признак того, что ярлык помечен к удалению.
Description	Задаёт или возвращает примечание к ярлыку.
FolderId	Возвращает идентификатор папки в которой размещен ярлык.
Id	Возвращает идентификатор ярлыка.
InUpdate	Возвращает признак включения для объекта режима отложенных изменения.
IsHardLink	Возвращает признак того, что ярлык является "сильным".
ModeId	Задаёт или возвращает идентификатор режима открытия карточки на которую ссылается ярлык.
Recalled	Возвращает признак отозванности ярлыка.
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
BeginUpdate	Включает режим отложенных изменений.

Имя	Описание
CancelUpdate	Отменяет несохранённые изменения и выключает режим отложенных изменений.
Copy(Guid)	Копирует ярлык в указанную папку.
EndUpdate	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
MarkCardForDeletion(Guid)	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
Move(Guid)	Перемещает ярлык в указанную папку.
Refresh	Обновляет данные ярлыка.
ShortcutEquals(Shortcut)	Сравнивает текущий и заданный ярлыки.
TurnToHardLink	Изменяет ярлык со "слабого" на "сильный".
UpdateNow	Позволяет отправить накопленные изменения на сервер.

Shortcut.ModeId — свойство

Задаёт или возвращает идентификатор режима открытия карточки на которую ссылается ярлык.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid ModeId { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор режима открытия

Заметки

Данное свойство позволяет реализовать дополнительную логику открытия карточки, для различных ярлыков указывающих на одну карточку.

VersionedFileCard — класс

Реализует логику карточки **файл с версиями**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class VersionedFileCard : SystemCard
```

Свойства

Имя	Описание
`Barcode`	Возвращает или задаёт код файла в строчном формате.
`CheckedOut`	Возвращает признак наличия постоянно блокировки.
`CheckinDate`	Дата изменения файла текущей версии.
`CheckoutDate`	Дата установки блокировки.
`CheckoutPath`	Возвращает путь в файловой системе, куда выгружен файл для единоличного владения.
`CheckoutUser`	Возвращает идентификатор пользователя, забравшего файл в единоличное владение.
`Comments`	Возвращает коллекцию комментариев к файлу.
`CurrentVersion`	Возвращает текущую версия файла.

Имя	Описание
`ExtCardId`	Задаёт или возвращает идентификатор карточки расширения.
`Name`	Задаёт или возвращает название карточки файла.
`ParentCardId`	Задаёт или возвращает идентификатор родительской карточки.
`Versions`	Возвращает коллекцию версий файла.

Методы

Имя	Описание
`CheckIn(String, Int32, Boolean, Boolean)`	Снимет постоянную блокировку и создаёт новую версию файла.
`CheckOut(String, Guid, Boolean)`	Выгружает файл на локальную файловую систему и устанавливает постоянную блокировку.
`DeleteVersion(Guid)`	Удаляет версию файла с указанным идентификатором.
`GetNextVersion(Int32)`	Возвращает номер следующей версии файла на заданном уровне.
`GetVersion(Guid)`	Возвращает версию файла с заданным идентификатором.
`Initialize(String, Guid, Boolean, Boolean)`	Создание новой карточки на базе содержимого файла, и создание первой версии.
`UndoCheckout`	Отменяет режим единоличного владения файлом.

ViewCard — класс

Представляет системную карточку сохранённых представлений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class ViewCard : SystemCard
```

Свойства

Имя	Описание
Groups	Возвращает коллекцию групп представлений.
Views	Возвращает коллекцию сохранённых описаний представлений.

Методы

Имя	Описание
CreateGroup(Guid, String)	Создаёт подгруппу в заданной группе представлений.
DeleteGroup(Guid)	Удаляет заданную группу представлений.
DeleteView(Guid)	Удаляет заданное представление.
GetGroup(Guid)	Получает группу представлений с заданным идентификатором.
GetView(Guid)	Получает представление с заданным идентификатором.
GetVirtualFields(Guid)	Получает коллекцию сохранённых описаний виртуальных полей для секции заданного типа.
GroupExists(Guid)	Проверяет существование группы представлений с заданным идентификатором.
ViewExists(Guid)	Проверяет существование представления с заданным идентификатором.

Заметки

Карточка сохранённых представлений является справочником, поэтому её

экземпляр всегда можно получить по идентификатору 17F8F0B3-7E93-45E9-B250-EED4E93F3FA3.

Примеры

Получение карточки сохранённых представлений:

```
const string VIEW_CARD_ID = "17F8F0B3-7E93-45E9-B250-EED4E93F3FA3";
ViewCard vcard = (ViewCard)session.CardManager.GetDictionary(new Guid(VIEW_CARD_ID));
```

ViewCard.Groups — свойство

Возвращает коллекцию групп представлений.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SavedViewGroupCollection Groups { get; }
```

Значение свойства

Тип: SavedViewGroupCollection

Коллекция групп представлений

ViewCard.Views — свойство

Возвращает коллекцию сохранённых описаний представлений.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SavedViewCollection Views { get; }
```

Значение свойства

Тип: SavedViewCollection

Коллекция сохранённых описаний представлений

ViewCard.CreateGroup — метод (Guid, String)

Создаёт подгруппу в заданной группе представлений

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SavedViewGroup CreateGroup(Guid parentGroupId, string name)
```

Параметры

parentGroupId

Тип: System.Guid

Идентификатор родительской группы

name

Тип: System.String

Название создаваемой группы

Возвращаемое значение

Тип: SavedViewGroup

Созданная группа представлений

ViewCard.DeleteGroup — метод (Guid)

Удаляет заданную группу представлений.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void DeleteGroup(Guid groupId)
```

Параметры

groupId

Тип: System.Guid

Идентификатор группы представлений

ViewCard.DeleteView — метод (Guid)

Удаляет заданное представление.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void DeleteView(Guid viewId)
```

Параметры

viewId

Тип: System.Guid

Идентификатор представления

ViewCard.GetGroup — метод (Guid)

Получает группу представлений с заданным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SavedViewGroup GetGroup(Guid groupId)
```

Параметры

groupId

Тип: System.Guid

Идентификатор группы

Возвращаемое значение

Тип: SavedViewGroup

Группа представлений

ViewCard.GetView — метод (Guid)

Получает представление с заданным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SavedView GetView(Guid viewId)
```

Параметры

viewId

Тип: System.Guid

Идентификатор представления

Возвращаемое значение

Тип: SavedView

Представление

ViewCard.GetVirtualFields — метод (Guid)

Получает коллекцию сохранённых описаний виртуальных полей для секции заданного типа.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SavedVirtualFieldCollection GetVirtualFields(Guid sectionId)
```

Параметры

sectionId

Тип: System.Guid

Идентификатор типа секции

Возвращаемое значение

Тип: SavedVirtualFieldCollection

Коллекция описаний виртуальных полей

ViewCard.GroupExists — метод (Guid)

Проверяет существование группы представлений с заданным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool GroupExists(Guid groupId)
```

Параметры

groupId

Тип: System.Guid

Идентификатор группы

Возвращаемое значение

Тип: System.Boolean

true — группа существует, иначе — false

ViewCard.ViewExists — метод (Guid)

Проверяет существование представления с заданным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager.SystemCards
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool ViewExists(Guid viewId)
```

Параметры

viewId

Тип: System.Guid

Идентификатор представления

Возвращаемое значение

Тип: `System.Boolean`

`true` — представление существует, иначе — `false`

ViewSettings — класс

Предоставляет доступ к параметрам представления.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class ViewSettings : IUpdatable
```

Свойства

Имя	Описание
`Aggregation`	Возвращает или задаёт клиентскую операцию агрегации табличных данных.
`AggregationColumn`	Псевдоним колонки, по которой осуществляется агрегация.
`AggregationText`	Текст строки содержащей агрегацию.
`Columns`	Коллекция настроек колонок представления.
`FilterId`	Идентификатор дополнительного фильтра для данных представления.
`Flags`	Дополнительные атрибуты, регулирующие внешний вид представления.
`FolderLevel`	Глубина вложенности папок, из которых извлекаются данные в представление.

Имя	Описание
GridLineMode	Возвращает или задаёт режим отображения линий таблицы.
GridLineStyle	Возвращает или задаёт стиль линий таблицы.
InUpdate	Возвращает признак включения для объекта режима отложенных изменения.
LinkLevel	Возвращает или задаёт глубину раскрытия ссылок между карточками.
PreviewColumn	Имя колонки, отображаемой в области предварительного просмотра.
RowHeight	Возвращает или задаёт высоту строк представления.
RowHeight	Возвращает или задаёт высоту строк представления.
Session	Возвращает текущую сессию пользователя.
Timestamp	Возвращает или задаёт отметку времени установленную для представления.
UserLayout	Возвращает или задаёт пользовательский макет представления.
UserLayoutState	Возвращает или задаёт статус пользовательского макета представления.
UserLayoutTimestamp	Возвращает или задаёт отметку времени установленную для пользовательского макета представления.
ViewId	Возвращает идентификатор представления, к которому относятся настройки.

Методы

Имя	Описание
BeginUpdate	Включает режим отложенных изменений.
CancelUpdate	Отменяет несохранённые изменения и выключает режим отложенных изменений.
Clear	Удаляет настройки представления.
Copy(Guid)	Создаёт копию настроек указанного представления.
EndUpdate	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
GetFont(ViewObjectType)	Получает шрифт указанного представления.
GetGrouping(int)	Получение настроек группировки представления.
GetSorting(int)	Получение настроек сортировки представления.
MarkCardForDeletion(Guid)	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
SetFont(ViewObjectType, Font)	Устанавливает заданный шрифт для представления.
UpdateNow	Позволяет отправить накопленные изменения на сервер.

Примеры

```
const string NAVIGATOR_CARD_TYPE = "A7F9784B-96A4-4B3E-B820-2E714A2A1463";
NavigatorCard nav = (NavigatorCard)session.CardManager.GetDictionary(new Guid
(NAVIGATOR_CARD_TYPE)); ①

ViewSettings settings = nav.ViewSettings[0]; ②
```

```
foreach(ViewColumnSettings col in settings.Columns) ③
{
    MessageBox.Show(col.Caption);
}
```

① Получение карточки Windows-клиента.

② Получение настроек первого представления.

③ Перебор всех колонок и вывод их названий.

ViewSettingsCollection — класс

Содержит методы для работы с коллекцией ViewSettings.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class ViewSettingsCollection : IEnumerable<ViewSettings>, IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов массива.
`Item`	Возвращает элемент коллекции с заданным идентификатором.
`Item`	Возвращает элемент коллекции с заданным индексом.
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`AddNew(Guid)`	Создаёт объект типа ViewSettings с заданным идентификатором.
`Clear`	Удаляет все элементы коллекции.

Имя	Описание
`Contains(Guid)`	Определяет вхождение в коллекцию элемента с заданным идентификатором.
`GetEnumerator`	Возвращает перечислитель.
`Remove(Guid)`	Удаляет элемент с указанным идентификатором.
`Remove(Int32)`	Удаляет элемент с заданным индексом.

AggregationType — перечисление

Определяет операцию агрегации для табличных данных на стороне клиента.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum AggregationType
```

Члены

Имя члена	Описание
`None`	Агрегация отсутствует.
`Count`	Количество строк.
`Sum`	Арифметическая сумма значений.
`Average`	Среднее значение.
`Min`	Минимальное значение.
`Max`	Максимальное значение.
`Dev`	Среднеквадратичное отклонение.
`ValueCount`	Количество элементов.

FolderFlags — перечисление

Дополнительные атрибуты папки.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
[Flags]
public enum FolderFlags
```

Члены

Имя члена	Описание
`None`	Отсутствует. Значение `0x00`.
`VirtualWithSubfolders`	Показывать дочерние подпапки виртуальной папки. Значение `0x01`.
`NoAutoRefresh`	Отключить авто-обновление содержимого папки. Значение `0x02`.
`NoUnreadCards`	Отключить подсветку количества непрочитанных карточек в папке. Значение `0x04`.
`CustomRefresh`	Автоматическое обновление содержимого папки в режиме "Специальное". Значение `0x08`.
`NoChangeFolderRefresh`	Не обновлять содержимое при переходе в папку. Значение `0x10`.

FolderRestrictions — перечисление

Определяет типы ограничений у папки.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
[Flags]
public enum FolderRestrictions
```

Члены

Имя члена	Описание
`None`	Не задано. Значение `0x00`.
`Views`	Установлены ограничения на использования представлений. Значение `0x01`.
`Types`	Установлены ограничения на создание типов карточек. Значение `0x02`.
`Templates`	Установлены ограничения на используемые шаблоны экспорта данных. Значение `0x04`.

Заметки

Наличие флага может использоваться для превентивной проверки ограничений, без дорогостоящего чтения всех целых списков ограничений.

FolderStyles — перечисление

Определяет стиль отображения папки.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
[Flags]
public enum FolderStyles
```

Члены

Имя члена	Описание
`None`	Стиль не определён. Значение `0x00`.
`View`	Представление. Значение `0x01`.
`Card`	Список карточек. Значение `0x02`.
`Url`	Список URL. Значение `0x04`.
`Digest`	Дайджест. Значение `0x08`.
`All`	Все стили. Значение `0x0F`.

FolderTypes — перечисление

Определяет тип папок для поисковых запросов.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.SystemCards`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
[Flags]
public enum FolderTypes
```

Члены

Имя члена	Описание
`None`	Тип не определён. Значение `0x00`.
`Regular`	Обычная папка. Значение `0x01`.
`Root`	Корневая папка. Значение `0x02`.
`Virtual`	Виртуальная папка. Значение `0x04`.
`Delegate`	Папка-делегат. Значение `0x08`.
`SystemHidden`	Системная папка. Значение `0x10`.
`All`	Все типы одновременно. Значение `0x1F`.

DocsVision.Platform.ObjectManager.ViewModel — пространство имён

Содержит классы описывающие объектную модель *представлений* и методы для работы с представлениями.

Классы

Класс	Описание
`Aggregation`	Предоставляет доступ к настройкам агрегирования.
`CaseItem`	Реализует логический переключатель для вычисляемого элемента.
`CaseItemCollection`	Содержит методы для работы с коллекцией `CaseItem`.
`ComputationGroup`	Представляет вычисляемую группу для представления.
`ComputationGroupCollection`	Содержит методы для работы с коллекцией `ComputationGroup`.
`ComputationPart`	Представляет источник данных для вычисляемого поля.
`ComputationPartCollection`	Содержит методы для работы с коллекцией `ComputationPart`.
`ComputedField`	Класс `ComputedField` предоставляет свойства вычисляемого поля.
`ComputedFieldCollection`	Содержит методы для работы с коллекцией `ComputedField`.
`DataCondition`	Представляет фильтр данных.
`DataConditionCollection`	Содержит методы для работы с коллекцией `DataCondition`.
`DataConditionGroup`	Представляет группу фильтров данных.
`DataConditionGroupCollection`	Содержит методы для работы с коллекцией `DataConditionGroup`.
`DataConditionItem`	Представляет элемент фильтра данных.
`DataConditionItemCollection`	Содержит методы для работы с коллекцией `DataConditionItem`.

Класс	Описание
DataItem	Содержит свойства элемента данных для представления.
JoinDefinition	Правило присоединения к представлению секции или таблицы.
JoinDefinitionCollection	Содержит методы для работы с коллекцией JoinDefinition.
SectionField	Представляет поле секции.
SectionFieldCollection	Содержит методы для работы с коллекцией SectionField.
TreeDefinition	Позволяет реализовать дерево правил присоединения к представлению.
View	Базовый класс представления.
ViewColumn	Представляет колонку представления.
ViewColumnCollection	Содержит методы для работы с коллекцией ViewColumn.
ViewDataSorting	Содержит правило сортировки данных.
ViewDataSortingCollection	Содержит методы для работы с коллекцией ViewDataSorting.
ViewElement	Содержит источник данных для представления.
ViewElementCollection	Содержит методы для работы с коллекцией ViewElement.
VirtualField	Представляет свойства виртуального поля и методы для работы с полем.

Перечисления

Перечисление	Описание
AggregateFunction	Определяет функцию агрегации.
ComputationOperation	Определяет тип операции вычисления.
ComputationPartType	Определяет тип вычисляемого элемента.

Перечисление	Описание
`ContentType`	Определяет тип содержимого представления.
`DataType`	Определяет тип данных.
`JoinDefinitionType`	Тип правила присоединения поля к представлению.
`VirtualFieldType`	Определяет тип виртуального поля.

ComputedField — класс

Класс `ComputedField` предоставляет свойства вычисляемого поля.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.ViewModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class ComputedField
```

Свойства

Имя	Описание
`Alias`	Возвращает название вычисляемого поля.
`ComputationGroup`	Возвращает группу вычислений.
`ResultSwitch`	Возвращает коллекцию логических переключателей.
`View`	Возвращает представление к которому привязано вычисляемое поле.

ComputedFieldCollection — класс

Содержит методы для работы с коллекцией `ComputedField`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.ViewModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class ComputedFieldCollection : IEnumerable<ComputedField>, IEnumerable
```

Свойства

Имя	Описание
Count	Возвращает количество элементов коллекции.
Item	Возвращает элемент коллекции с указанным псевдонимом.
Item	Возвращает элемент коллекции с заданным индексом.
View	Возвращает представление к которому привязана коллекция вычисляемых полей.

Методы

Имя	Описание
AddNew(String)	Добавляет элемент коллекции с заданным псевдонимом и возвращает на него ссылку.
Clear	Удаляет все элементы коллекции.
Contains(String)	Проверяет вхождение в коллекцию элемента с заданным псевдонимом.
GetEnumerator	Возвращает перечислитель.
Remove(Int32)	Удаляет элемент коллекции с заданным индексом.
Remove(String)	Удаляет элемент коллекции с указанным псевдонимом.

JoinDefinitionCollection — класс

Содержит методы для работы с коллекцией JoinDefinition.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class JoinDefinitionCollection : IEnumerable<JoinDefinition>, IEnumerable
```

Свойства

Имя	Описание
Count	Возвращает количество элементов коллекции.
Item	Возвращает элемент коллекции с указанным псевдонимом.
Item	Возвращает элемент коллекции с заданным индексом.
View	Возвращает представление к которому привязана коллекция правил.

Методы

Имя	Описание
AddNew(String)	Добавляет элемент коллекции с заданным псевдонимом и возвращает на него ссылку.
AddNew(String, JoinDefinitionType)	Добавляет элемент коллекции с заданным псевдонимом и типом присоединения, и возвращает на него ссылку.
Clear	Удаляет все элементы коллекции.
Contains(String)	Проверяет вхождение в коллекцию элемента с заданным псевдонимом.
GetEnumerator	Возвращает перечислитель.
Remove(Int32)	Удаляет элемент коллекции с заданным индексом.
Remove(String)	Удаляет элемент коллекции с указанным псевдонимом.

SectionFieldCollection — класс

Содержит методы для работы с коллекцией `SectionField`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.ViewModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class SectionFieldCollection : IEnumerable<SectionField>, IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов коллекции.
`Item`	Возвращает элемент коллекции с указанным псевдонимом.
`Item`	Возвращает элемент коллекции с заданным индексом.
`View`	Возвращает представление к которому привязана коллекция полей секции.

Методы

Имя	Описание
`AddNew(String)`	Добавляет элемент коллекции с заданным псевдонимом и возвращает на него ссылку.
`Clear`	Удаляет все элементы коллекции.
`Contains(String)`	Проверяет вхождение в коллекцию элемента с заданным псевдонимом.
`GetEnumerator`	Возвращает перечислитель.
`Remove(Int32)`	Удаляет элемент коллекции с заданным индексом.
`Remove(String)`	Удаляет элемент коллекции с указанным псевдонимом.

View — класс

Базовый класс содержащий методы управления описанием представления.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.ViewModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class View
```

Свойства

Имя	Описание
`Columns`	Возвращает коллекцию колонок представления.
`ContentType`	Задаёт или возвращает тип содержимого представления.
`Elements`	Возвращает коллекцию элементов представления.
`Parameters`	Возвращает коллекцию параметров представления.
`Session`	Возвращает текущую сессия пользователя.
`Sorting`	Возвращает коллекцию правил сортировки данных в колонках представления.

Методы

Имя	Описание
`GetResolvedParameters`	Возвращает коллекцию параметров поискового запроса.
`GetXml`	Возвращает описание представление в XML-формате.
`ParseXml(String)`	Загружает параметры представления из XML-строки.

View.Columns — свойство

Возвращает коллекцию колонок представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ViewColumnCollection Columns { get; }
```

Значение свойства

Тип: ViewColumnCollection

Коллекция колонок представления

View.ContentType — свойство

Задаёт или возвращает тип содержимого представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ContentType ContentType { get; set; }
```

Значение свойства

Тип: ContentType

Тип содержимого

View.Elements — свойство

Возвращает коллекцию элементов представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ViewElementCollection Elements { get; }
```

Значение свойства

Тип: `ViewElementCollection`

Коллекция представлений

Заметки

Элементы определяют источники данных для колонок представления.

View.Parameters — свойство

Возвращает коллекцию параметров представления.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.ViewModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract QueryParameterCollection Parameters { get; }
```

Значение свойства

Тип: `QueryParameterCollection`

Коллекция параметров

View.Session — свойство

Текущая сессия пользователя.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.ViewModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: `UserSession`

Текущая сессия

View.Sorting — свойство

Возвращает коллекцию правил сортировки данных в колонках представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ViewDataSortingCollection Sorting { get; }
```

Значение свойства

Тип: ViewDataSortingCollection

Коллекция правил сортировки

View.GetResolvedParameters — метод

Возвращает коллекцию параметров поискового запроса.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract QueryParameterCollection GetResolvedParameters()
```

Возвращаемое значение

Тип: QueryParameterCollection

Коллекция параметров поискового запроса

View.GetXml — метод

Возвращает описание представление в XML-формате.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string GetXml()
```

Возвращаемое значение

Тип: System.String

Описание представления

View.ParseXml — метод (String)

Загружает параметры представления из XML-строки.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void ParseXml(string viewXml)
```

Параметры

viewXml

Тип: System.String

XML-строка

Заметки

Параметры загружаются в текущее представление.

ViewColumnCollection — класс

Содержит методы для работы с коллекцией ViewColumn.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class ViewColumnCollection : IEnumerable<ViewColumn>, IEnumerable
```

Свойства

Имя	Описание
Count	Возвращает количество элементов коллекции.

Имя	Описание
`Item`	Возвращает элемент коллекции с указанным псевдонимом.
`Item`	Возвращает элемент коллекции с заданным индексом.
`View`	Возвращает представление к которому привязана коллекция колонок.

Методы

Имя	Описание
`AddNew(String)`	Добавляет элемент коллекции с заданным псевдонимом и возвращает на него ссылку.
`Clear`	Удаляет все элементы коллекции.
`Contains(String)`	Проверяет вхождение в коллекцию элемента с заданным псевдонимом.
`GetEnumerator`	Возвращает перечислитель.
`Remove(Int32)`	Удаляет элемент коллекции с заданным индексом.
`Remove(String)`	Удаляет элемент коллекции с указанным псевдонимом.

ViewDataSortingCollection — класс

Содержит методы для работы с коллекцией `ViewDataSorting`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.ViewModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class ViewDataSortingCollection : IEnumerable<ViewDataSorting>,
IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов коллекции.
`Item`	Возвращает элемент коллекции с указанным названием.
`Item`	Возвращает элемент коллекции с заданным индексом.
`View`	Возвращает представление к которому привязана коллекция правил сортировки.

Методы

Имя	Описание
`AddNew(String)`	Добавляет элемент коллекции с заданным названием и возвращает на него ссылку.
`Clear`	Удаляет все элементы коллекции.
`Contains(String)`	Проверяет вхождение в коллекцию элемента с заданным названием.
`GetEnumerator`	Возвращает перечислитель.
`GetXML`	Возвращает коллекцию правил в формате XML.
`ParseXML(String)`	Загружает правила из XML-строки.
`Remove(Int32)`	Удаляет элемент коллекции с заданным индексом.
`Remove(String)`	Удаляет элемент коллекции с указанным псевдонимом.

ViewElement — класс

Содержит источник данных для представления. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class ViewElement
```

Свойства

Имя	Описание
ComputedFields	Возвращает набор правил для создания вычисляемых полей.
ConditionGroup	Возвращает условия для фильтрации результирующего набора.
JoinDefinitions	Возвращает коллекцию правил присоединения к представлению других секций или таблиц.
SectionFields	Возвращает набор полей секций.
SectionId	Идентификатор секции карточки из которой извлекаются данные.
SkipInstanceLimit	Возвращает или задаёт признак отключения ограничения на количество экземпляров карточки.
View	Возвращает представление к которому привязан источник.

Методы

Имя	Описание
ExtractVirtualField(String)	Извлекает из представления виртуальное поле с заданным псевдонимом.
MergeVirtualField(VirtualField, String, String)	Добавляет в представление существующее виртуальное поле.
RenameJoinDefinition(String, String)	Переименовывает присоединенную секцию или таблицу.

ViewElement.ComputedFields — свойство

Возвращает набор правил для создания вычисляемых полей.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ComputedFieldCollection ComputedFields { get; }
```

Значение свойства

Тип: ComputedFieldCollection

Коллекция вычисляемых полей

ViewElement.JoinDefinitions — свойство

Возвращает коллекцию правил присоединения к представлению других секций или таблиц.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract JoinDefinitionCollection JoinDefinitions { get; }
```

Значение свойства

Тип: JoinDefinitionCollection

Коллекция правил присоединения

ViewElement.SkipInstanceLimit — свойство

Возвращает или задаёт признак отключения ограничения на количество экземпляров карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool SkipInstanceLimit { get; set; }
```

Значение свойства

Тип: System.Boolean

`true` — ограничение отключено, иначе — `false`

Заметки

При срабатывании правил выбора данных из таблиц, в представление может попасть несколько экземпляров одной карточки. Данный параметр ограничивает количество экземпляров выводимых в представление до одного.

ViewElement.SectionId — свойство

Идентификатор секции карточки из которой извлекаются данные.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid SectionId { get; }
```

Значение свойства

Тип: System.Guid

Идентификатор секции

ViewElement.SectionFields — свойство

Возвращает набор полей секций.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SectionFieldCollection SectionFields { get; }
```

Значение свойства

Тип: SectionFieldCollection

Коллекция полей секций

ViewElement.ExtractVirtualField — метод (String)

Извлекает из представления виртуальное поле с заданным псевдонимом.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract VirtualField ExtractVirtualField(string columnAlias)
```

Параметры

columnAlias

Тип: System.String

Псевдоним поля

Возвращаемое значение

Тип: VirtualField

Виртуальное поле

ViewElement.MergeVirtualField — метод (VirtualField, String, String)

Добавляет в представление существующее виртуальное поле.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void MergeVirtualField(VirtualField virtualField, string
targetSectionAlias, string columnAlias)
```

Параметры

virtualField

Тип: `VirtualField`

Виртуальное поле

targetSectionAlias

Тип: `System.String`

Псевдоним секции, к которой добавляется поле

columnAlias

Тип: `System.String`

Псевдоним колонки

ViewElementCollection — класс

Содержит методы для работы с коллекцией `ViewElement`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.ViewModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class ViewElementCollection : IEnumerable<ViewElement>, IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов коллекции.
`Item`	Возвращает элемент коллекции с указанным идентификатором.
`Item`	Возвращает элемент коллекции с заданным индексом.
`View`	Возвращает представление к которому привязана коллекция источников данных.

Методы

Имя	Описание
AddNew(Guid)	Добавляет элемент коллекции с заданным идентификатором и возвращает на него ссылку.
Clear	Удаляет все элементы коллекции.
Contains(Guid)	Проверяет вхождение в коллекцию элемента с заданным идентификатором.
GetEnumerator	Возвращает перечислитель.
Remove(Guid)	Удаляет элемент коллекции с указанным идентификатором.
Remove(Int32)	Удаляет элемент коллекции с заданным индексом.

VirtualField — класс

Представляет свойства виртуального поля и методы для работы с полем.

- **Пространство имён:** DocsVision.Platform.ObjectManager.ViewModel
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class VirtualField
```

Свойства

Имя	Описание
ComputedField	Возвращает вычисляемое поле.
DataType	Задаёт или возвращает тип данных в источнике.
FieldType	Задаёт или возвращает тип данных виртуального поля.
JoinDefinitions	Возвращает коллекцию правил присоединения.

Имя	Описание
`SectionField`	Возвращает поле секции.
`SectionId`	Возвращает идентификатор секции.
`View`	Возвращает представление в котором находится виртуальное поле.

Методы

Имя	Описание
`GetXml`	Выгружает описание виртуального поля в XML-строку.
`ParseXml(String)`	Загружает описание виртуального поля из XML-строки.

Заметки

Виртуальные поля представляют собой правила для формирования данных, которые физически не хранятся в самой карточке, но могут быть интересны при построении представлений (отчётов) с участием карточек такого типа

ContentType — перечисление

Определяет тип содержимого представления.

- **Пространство имён:** `DocsVision.Platform.ObjectManager.ViewModel`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum ContentType
```

Члены

Имя члена	Описание
`Regular`	Обычный
`Digest`	Дайджест

AccessManager — класс

Представляет объектную модель **менеджера объектов безопасности**. Содержит методы для работы с зашифрованными и подписанными объектами. Осуществляет управление правами (дескрипторами безопасности) на объекты системы.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class AccessManager
```

Свойства

Имя	Описание
`SecuritySchema`	Возвращает строку содержащую схему безопасности в XML-формате.
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`AccessCheck(SecureObjectType, Guid, Guid, Int32)`	Проверка наличия заданных прав на объект у пользователя.
`CreateCustomSecureObject(Guid)`	Создаёт объект безопасности.
`DeleteCryptObject(Guid)`	Удаляет крипто-объект с заданным идентификатором.
`DeleteCryptObjects(Guid, CryptObjectType)`	Удаляет все крипто-объекты заданного типа для указанного файла.
`GetAccessControl(SecureObjectType, Guid, Guid)`	Возвращает описатель прав для объекта.
`GetCryptObject(Guid)`	Возвращает подпись, либо ключ шифрования для крипто-объекта с заданным идентификатором.

Имя	Описание
`GetCryptObjectInfo(Guid)`	Возвращает коллекцию нетепизированных данных для заданного крипто-объекта.
`GetCryptObjects(Guid, CryptObjectType)`	Возвращает коллекцию нетепизированных данных крипто-объектов, сопоставляемых указанному файлу.
`GetKey(Guid)`	Получает ключ шифрования для заданного объекта.
`SetAccessControl(SecureObjectType, Guid, Guid, DVObjectSecurity)`	Устанавливает права на объект.
`StoreCryptObject(Guid, Guid, CryptObjectType, Byte[], String, String, Guid, String)`	Сохраняет новую подпись или ключ шифрования.
`StoreKey(Guid, Byte[])`	Сохраняет ключ шифрования объекта.
`StoreKey(Guid, Byte[], String)`	Сохраняет ключ шифрования объекта для указанного пользователя.
`StoreSignature(Guid, Byte[], String)`	Сохраняет подпись объекта.
`StoreSignature(Guid, Byte[], String, Guid, String)`	Сохраняет подпись объекта с дополнительным описанием.

AccessManager.SecuritySchema — свойство

Возвращает строку содержащую схему безопасности в XML-формате.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract string SecuritySchema { get; }
```

Значение свойства

Тип: `System.String`

Схема безопасности

AccessManager.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Текущая сессия

AccessManager.AccessCheck — метод (SecureObjectType, Guid, Guid, Int32)

Проверка наличия заданных прав на объект у пользователя. Метод проверяет права для пользователя, от имени которого создана пользовательская сессия.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool AccessCheck(SecureObjectType objectType, Guid objectId, Guid
parentId, int desiredAccess)
```

Параметры

objectType

Тип: SecureObjectType

Тип объекта безопасности

objectId

Тип: System.Guid

Идентификатор проверяемого объекта

parentId

Тип: `System.Guid`

Идентификатор родительского объекта для случаев, когда это уместно. Если неуместно использовать `parentID`, можно передавать `Guid.Empty`. `parentId` необходим при проверке прав на строке (`SecureObjectType.Row`), в таком случае в `parentId` передается идентификатор секции, в `objectId` — идентификатор строки. Такой подход используется ещё в ряде методов модуля Платформа.

desiredAccess

Тип: `System.Int32`

Запрашиваемая маска проверяемого уровня доступа. Т.к. Docsvision поддерживает различные схемы прав безопасности и нет универсального перечисления с масками доступа, для разных классов объектов эти флаги\перечисления будут разными. В различных случаях могут использоваться `read`, `write`, `delete`, `own` в generic\specific вариантах, в других — `Read`, `Use`.

Возвращаемое значение

Тип: `System.Boolean`

`true` — пользователь владеет заданными правами, иначе — `false`

AccessManager.CreateCustomSecureObject — метод (Guid)

Создаёт объект безопасности.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract ISecurable CreateCustomSecureObject(Guid securityClassId)
```

Параметры

securityClassId

Тип: `System.Guid`

Идентификатор класса безопасности

Возвращаемое значение

Тип: `ISecurable`

Объект безопасности

ArchiveOptions — перечисление

Режим архивации объекта.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
[Flags]
public enum ArchiveOptions
```

Члены

Имя члена	Описание
`None`	Не задан. Значение `0x0`
`Delay`	Использовать режим отложенной архивации. Значение `0x1`
`IncludeLinkedCards`	Архивируются также другие карточки, связанные с данной по ссылке. Значение `0x2`
`IncludeLinkedFiles`	Архивируются также связанные с карточкой файлы. Значение `0x4`

Заметки

Отложенная архивация происходит по расписанию.

ArchiveState — перечисление

Определяет признак архивирования объекта.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public enum ArchiveState
```

Члены

Имя члена	Описание
NotArchived	Не архивирован
Archived	Архивирован
PreparedToArchive	Подготовлен к помещению в архив
PreparedToDearchive	Подготовлен к извлечению из архива

CardData — класс

Содержит данные карточки. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class CardData : IUpdatable, ILockable, IXmlExportable
```

Свойства

Имя	Описание
ArchiveState	Возвращает признак нахождения файла в архиве.
Barcode	Выполняет получение или установку штрих-кода карточки.
ChangeDate	Возвращает дату последнего изменения.
CreateDate	Возвращает дату создания карточки.
Description	Возвращает или задаёт описание экземпляра карточки (дайджест).

Имя	Описание
FetchMode	Возвращает или задаёт режим загрузки данных карточки.
IconId	Возвращает или задаёт идентификатор иконки карточки.
Id	Возвращает идентификатор экземпляра карточки.
InUpdate	Признак режима отложенных изменения.
IsTemplate	Возвращает или задаёт признак того, что карточка является шаблоном.
Labels	Возвращает коллекцию меток карточки.
LockOwner	Возвращает название учетной записи субъекта заблокировавшего карточку.
LockStatus	Возвращает состояние блокировки карточки.
RecordStatus	Возвращает или задаёт состояние карточки.
Sections	Возвращает коллекцию секций карточки.
Session	Возвращает текущую открытую сессию пользователя.
Topic	Возвращает или задаёт тему карточки.
TopicId	Идентификатор темы карточки.
TopicIndex	Индекс карточки внутри темы обработки
TopicOrder	Возвращает порядковый номер в теме обработки.
TopicParentId	Идентификатор родительской карточки в теме карточек.
Type	Описанию метаданных типа карточки.

Имя	Описание
WasRead	Задаёт или возвращает признак прочитанности ярлыка.

Методы

Имя	Описание
AccessCheck(System.Int32)	Проверяет наличие заданного доступа у текущего пользователя.
AddToTopicChain(Guid)	Добавляет текущую карточку к теме обработки другой карточки.
Archive(ArchiveOptions)	Помещает текущую карточку в архив.
BeginUpdate	Включить режим отложенных изменений.
CancelUpdate	Отменить несохранённые изменения и выключить режим отложенных изменений.
Copy	Серверное копирование данных карточек.
Dearchive(ArchiveOptions)	Восстанавливает карточку из архива.
EndUpdate	Записать изменения и выключить режим отложенных изменений.
ForceUnlock	Принудительно снимает блокировку карточки.
GetAccessControl	Получает описатель прав на объект.
GetAccessControl(AccessControlSections)	Возвращает описатель прав на объект для заданного раздела дескриптора безопасности.
MarkCardForDeletion(Guid)	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
PlaceLock	Установка временной блокировки.

Имя	Описание
PlaceLock(Boolean)	Установка постоянной или временной блокировки.
PurgeCache	Удаление карточек из кэша менеджера объектов.
Refresh	Принудительное обновление данных карточки с сервера.
Refresh(Boolean)	Принудительное обновление данных карточки с сервера, независимо от наличия или отсутствия изменений.
RemoveLock	Снимает блокировку сделанную в своей сессии.
SaveXml(Stream)	Экспорт данных карточки в XML-формате.
SaveXml(Stream, ExportFlags)	Экспорт данных карточки в XML-формате, с установленными флагами формата экспорта.
SaveXml(Stream, ExportFlags, ExportCardInspector)	Экспорт данных карточки в XML-формате, с заданной логикой экспорта.
SetAccessControl(CardDataSecurity)	Задаёт описатель прав на карточку.
UpdateNow	Отправляет накопленные изменения на сервер.

CardData.ArchiveState — свойство

Возвращает признак нахождения файла в архиве.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ArchiveState ArchiveState { get; }
```

Значение свойства

Тип: ArchiveState

Признак архивации

CardData.Barcode — свойство

Выполняет получение или установку штрих-кода карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Barcode { get; set; }
```

Значение свойства

Тип: String

Штрих-код

CardData.ChangeDate — свойство

Возвращает дату последнего изменения.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract DateTime ChangeDate { get; }
```

Значение свойства

Тип: DateTime

Дата изменения

CardData.CreateDate — свойство

Возвращает дату создания карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract DateTime CreateDate { get; }
```

Значение свойства

Тип: DateTime

Дата создания

CardData.Description — свойство

Возвращает или задаёт описание экземпляра карточки (дайджест).

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Description { get; set; }
```

Значение свойства

Тип: String

Описание

CardData.FetchMode — свойство

Возвращает или задаёт режим загрузки данных карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract FetchMode FetchMode { get; set; }
```

Значение свойства

Тип: FetchMode

Режим загрузки

Заметки

Режим загрузки используется для снижения объема передаваемых данных (загрузка с сервера больших справочников), либо наоборот для увеличения скорости доступа (небольшую карточку можно загрузить целиком, вместо организации нескольких обращений на чтение к серверу).

CardData.IconId — свойство

Возвращает или задаёт идентификатор иконки карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid IconId { get; set; }
```

Значение свойства

Тип: Guid

Идентификатор значка

CardData.Id — свойство

Возвращает идентификатор экземпляра карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid Id { get; }
```

Значение свойства

Тип: Guid

Идентификатор

CardData.InUpdate — свойство

Признак режима отложенных изменения.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool InUpdate { get; }
```

Значение свойства

Тип: Boolean

false — режим выключен; true — режим включён

CardData.IsTemplate — свойство

Возвращает или задаёт признак того, что карточка является шаблоном.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool IsTemplate { get; set; }
```

Значение свойства

Тип: Boolean

true — карточка является шаблоном; false — карточка не является шаблоном

Заметки

CardData.Labels — свойство

Возвращает коллекцию меток карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardLabelCollection Labels { get; }
```

Значение свойства

Тип: CardLabelCollection

Метки

CardData.LockOwner — свойство

Возвращает название учетной записи субъекта заблокировавшего карточку.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string LockOwner { get; }
```

Значение свойства

Тип: String

Учетная запись

Заметки

Имеет значение в случае, если на карточке установлена блокировка.

CardData.LockStatus — свойство

Возвращает состояние блокировки карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract LockStatus LockStatus { get; }
```

Значение свойства

Тип: LockStatus

Состояние блокировки

CardData.RecordStatus — свойство

Возвращает или задаёт состояние карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract RecordStatus RecordStatus { get; set; }
```

Значение свойства

Тип: RecordStatus

Состояние объекта.

CardData.Sections — свойство

Возвращает коллекцию секций карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SectionDataCollection Sections { get; }
```

Значение свойства

Тип: SectionDataCollection

Коллекция секций

Заметки

Дальнейшее обращение к элементам секции осуществляется в стандартном порядке обращения с коллекцией.

CardData.Session — свойство

Возвращает текущую открытую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Сессия пользователя

CardData.Topic — свойство

Возвращает или задаёт тему карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Topic { get; set; }
```

Значение свойства

Тип: String

Тема карточки

CardData.TopicId — свойство

Идентификатор темы карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid TopicId { get; }
```

Значение свойства

Тип: Guid

Идентификатор

CardData.TopicIndex — свойство

Индекс карточки внутри темы обработки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract int TopicIndex { get; }
```

Значение свойства

Тип: System.Int32

Позиция

CardData.TopicOrder — свойство

Возвращает порядковый номер в теме обработки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract int TopicOrder { get; }
```

Значение свойства

Тип: System.Int32

Порядковый номер

CardData.TopicParentId — свойство

Идентификатор родительской карточки в теме карточек.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid TopicParentId { get; }
```

Значение свойства

Тип: `Guid`

Идентификатор

CardData.Type — свойство

Описанию метаданных типа карточки.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract CardType Type { get; }
```

Значение свойства

Тип: `CardType`

Метаданные

CardData.WasRead — свойство

Задаёт или возвращает признак прочитанности ярлыка.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract bool WasRead { get; set; }
```

Значение свойства

Тип: `Boolean`

`true` — ярлык прочитан; `false` — ярлык не прочитан

Заметки

По значению определяется — открывалась карточка на просмотр или нет. Значение параметра уникально для пользователя.

CardData.AccessCheck — метод (System.Int32)

Проверяет наличие заданного доступа у текущего пользователя.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public bool AccessCheck(int desiredAccess)
```

Параметры

desiredAccess

Тип: `System.Int32`

Требуемый доступ

Возвращаемое значение

Тип: `Boolean`

`true` — пользователь обладает требуемым доступом, иначе — `false`

CardData.AddToTopicChain — метод (Guid)

Добавляет текущую карточку к теме обработки другой карточки.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract void AddToTopicChain(Guid cardId)
```

Параметры

cardId

Тип: `Guid`

Идентификатор целевой карточки

CardData.Archive — метод (ArchiveOptions)

Помещает текущую карточку в архив.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Archive(ArchiveOptions archiveOptions)
```

Параметры

archiveOptions

Тип: ArchiveOptions

Режим архивации карточки

Примеры

В следующем примере демонстрируется возможность перемещения карточки в архив.

```
CardData card = session.CardManager.GetCardData(new System.Guid(
"идентификатор_карточки")); ①

if (card.ChangeDate.Year < DateTime.Now.Year -1) ②
{
    card.Archive(ArchiveOptions.IncludeLinkedCards); ③
}
```

① Получение с сервера данных карточки с известным идентификатором.

Где session — объект типа UserSession, предоставляющий текущую открытую сессию.

② Проверка даты последнего изменения.

③ Перемещние карточки в архив, с архивацией связанных карточек.

CardData.BeginUpdate — метод

Включить режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void BeginUpdate()
```

Заметки

Подробнее о данном режиме читайте в разделе Режим отложенных изменений`

CardData.CancelUpdate — метод

Отменить несохранённые изменения и выключить режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void CancelUpdate()
```

CardData.Copy — метод

Серверное копирование данных карточек.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardData Copy()
```

Возвращаемое значение

Тип: CardData

Экземпляр карточки

Заметки

Копирует данные текущей карточки. В результат получается новый независимый экземпляр карточки с другим идентификатором.

Примеры

Ниже приведён пример использования метода Copy при создании копии карточки с изменением описания

```
①

CardData cardData = userSession.CardManager.GetCardData(new Guid("00000000-0000-0000-
0000-000000000000"), false); ②

CardData newCardData = userSession.CardManager.GetCardData(cardData.Id, false).Copy(); ③
newCardData.Description = newCardData.Description + " Copy";
```

① Инициализация контекста объектов.

② Получение карточки.

③ Создание копии.

CardData.Dearchive — метод (ArchiveOptions)

Восстанавливает карточку из архива.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Dearchive(ArchiveOptions archiveOptions)
```

Параметры

archiveOptions

Тип: ArchiveOptions

Режим архивации карточки

CardData.EndUpdate — метод

Записать изменения и выключить режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void EndUpdate()
```

CardData.ForceUnlock — метод

Принудительно снимает блокировку карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void ForceUnlock()
```

Заметки

Может быть снята, в том числе, блокировка, установленная в другой пользовательской сессии.

Для снятия "чужой" блокировки необходимы права администратора Docsvision.

CardData.GetAccessControl — метод

Получает описатель прав на объект.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public CardDataSecurity GetAccessControl()
```

Возвращаемое значение

Тип: CardDataSecurity

Описатель прав

CardData.GetAccessControl — метод (AccessControlSections)

Возвращает описатель прав на объект для заданного раздела дескриптора безопасности.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public CardDataSecurity GetAccessControl(AccessControlSections includeSections)
```

Параметры

includeSections

Тип: AccessControlSections

Раздел дескриптора безопасности

Возвращаемое значение

Тип: CardDataSecurity

Описатель прав

CardData.MarkCardForDeletion — метод (Guid)

Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void MarkCardForDeletion(Guid cardId)
```

Параметры

cardId

Тип: Guid

Идентификатор удаляемой карточки

Заметки

При отмене пакета изменений (в режиме отложенных изменений), карточка будет автоматически удалена. Этот метод позволяет избавить разработчика от необходимости самому следить за жизненным циклом объектов, которые он

создаёт в режиме отложенных изменений.

CardData.PlaceLock — метод

Установка временной блокировки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void PlaceLock()
```

Заметки

Позволяет заблокировать ресурс с целью недопущения его использования в других сессиях.

CardData.PlaceLock — метод (Boolean)

Установка постоянной или временной блокировки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void PlaceLock(bool permanently)
```

Параметры

permanently

Тип: Boolean

true — постоянная блокировка; false — временная блокировка;

Заметки

Временная блокировка ограничена существованием сессии, из которой эта блокировка установлена. Постоянная блокировка должна быть снята тем, кто её установил, либо администратором системы.

CardData.PurgeCache — метод

Удаление карточек из кэша менеджера объектов.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void PurgeCache()
```

Заметки

При следующем обращение к серверу данные будут принудительно пересчитаны.

CardData.Refresh — метод

Принудительное обновление данных карточки с сервера.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void Refresh()
```

Заметки

Данные будут загружены только при наличии в них изменений с момента предыдущего обновления.

CardData.Refresh — метод (Boolean)

Принудительное обновление данных карточки с сервера, независимо от наличия или отсутствия изменений

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Refresh(bool ignoreTimestamp)
```

Параметры

ignoreTimestamp

Тип: `Boolean`

`true` — игнорировать отсутствие изменений; `false` — учитывать отсутствие изменений

Заметки

При установке ignoreTimestamp в значение true, данные объектов будут получены с сервера в любом случае.

CardData.RemoveLock — метод

Снимает блокировку сделанную в своей сессии.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract void RemoveLock()
```

Примеры

```
//Получение с сервера данных карточки с известным идентификатором
CardData card = session.CardManager.GetCardData(new System.Guid(
"идентификатор_карточки"));

//Проверка блокировки на карточке
if (card.LockStatus != LockStatus.Free)
{
    //Объект заблокирован -- проверка владельца
    if (card.LockStatus = LockStatus.OwnerLocked)
    {
        card.RemoveLock();
    }
    else
    {
        MessageBox.Show("Объект заблокирован пользователем " + card.LockOwner);
    }
}
```

CardData.SaveXml — метод (Stream)

Экспорт данных карточки в XML-формате.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void SaveXml(Stream stream)
```

Параметры

stream

Тип: Stream

Поток для данных

CardData.SaveXml — метод (Stream, ExportFlags)

Экспорт данных карточки в XML-формате, с установленными флагами формата экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void SaveXml(Stream stream, ExportFlags exportFlags)
```

Параметры

stream

Тип: Stream

Поток для данных

exportFlags

Тип: ExportFlags

Флаги экспорта

CardData.SaveXml — метод (Stream, ExportFlags, ExportCardInspector)

Экспорт данных карточки в XML-формате, с заданной логикой экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void SaveXml(Stream stream, ExportFlags exportFlags, ExportCardInspector
inspector)
```

Параметры

stream

Тип: Stream

Поток для данных

exportFlags

Тип: ExportFlags

Флаги экспорта

inspector

Тип: ExportCardInspector

Логика экспорта

CardData.SetAccessControl — метод (CardDataSecurity)

Задаёт описатель прав на карточку.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void SetAccessControl(CardDataSecurity cardSecurity)
```

Параметры

cardSecurity

Тип: CardDataSecurity

Описатель прав

Примеры

```
CardData card = session.CardManager.GetCardData(new System.Guid(
"идентификатор_карточки")); ①

CardDataSecurity rights = card.GetAccessControl(); ②

CardDataAccessRule rule = new CardDataAccessRule("TESTDOMAIN\\IvanovII", CardDataRights
.Read, AccessControlType.Allow); ③

rights.SetAccessRule(rule); ④

card.SetAccessControl(rights); ⑤
```

① Получение с сервера данных карточки с известным идентификатором.

② Получение описателя прав карточки.

③ Создание нового разрешения — совокупного права чтения.

④ Добавление нового разрешения к описателю прав.

⑤ Сохранение изменённого описателя прав.

CardData.UpdateNow — метод

Отправляет накопленные изменения на сервер.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract void UpdateNow()
```

Заметки

Отправляет накопленные изменения на сервер, на клиенте они очищаются из кэша, режим продолжает действовать. Все изменения после этого момента также будут накапливаться.

CardDataCollection — класс

Предоставляет методы для работы с коллекцией `CardData`. Этот класс является

абстрактным.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class CardDataCollection : IEnumerable<CardData>, IEnumerable,
IXmlExportable
```

Свойства

Имя	Описание
Count	Возвращает количество элементов в коллекции.
Item	Возвращает элемент коллекции с заданным индексом.
Item	Возвращает элемент коллекции с заданным идентификатором карточки.
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
Contains	Проверяет наличие в коллекции элемента с заданным идентификатором карточки.
SaveXml(Stream)	Метод сохраняет представление данных объекта в формате XML, в поток.
SaveXml(Stream, ExportFlags)	Метод сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта.

Имя	Описание
SaveXml(Stream, ExportFlags, ExportCardInspector)	Метод сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта и заданной логикой экспорта.

CardDataCollection.Count — свойство

Возвращает количество элементов в коллекции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract int Count { get; }
```

Значение свойства

Тип: System.Int32

Количество элементов

CardDataCollection.Item — свойство

Возвращает элемент коллекции с заданным индексом.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardData this[int index] { get; }
```

Параметры

index

Тип: System.Int32

Индекс элемента коллекции

Значение свойства

Тип: CardData

Карточка

CardDataCollection.Item — свойство

Возвращает элемент коллекции с заданным идентификатором карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardData this[Guid cardId] { get; }
```

Параметры

cardId

Тип: System.Guid

Идентификатор карточки

Значение свойства

Тип: CardData

Карточка

CardDataCollection.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Сессия пользователя

CardDataCollection.Contains — метод (Guid)

Проверяет наличие в коллекции элемента с заданным идентификатором карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool Contains(Guid cardId)
```

Параметры

cardId

Тип: System.Guid

Идентификатор

Возвращаемое значение

Тип: System.Boolean

true — коллекция содержит искомый элемент, иначе — false

CardDataCollection.SaveXml — метод (Stream)

Метод сохраняет представление данных объекта в формате XML, в поток.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void SaveXml(Stream stream)
```

Параметры

stream

Тип: System.IO.Stream

Поток для сохранения

CardDataCollection.SaveXml — метод (Stream, ExportFlags)

Метод сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void SaveXml(Stream stream, ExportFlags exportFlags)
```

Параметры

stream

Тип: System.IO.Stream

Поток для сохранения данных

exportFlags

Тип: ExportFlags

Параметры экспорта

CardDataCollection.SaveXml — метод (Stream, ExportFlags, ExportCardInspector)

Метод сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта и заданной логикой экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void SaveXml(Stream stream, ExportFlags exportFlags, ExportCardInspector
inspector)
```

Параметры

stream

Тип: System.IO.Stream

Поток для сохранения данных

exportFlags

Тип: `ExportFlags`

Параметры экспорта

inspector

Тип: `ExportCardInspector`

Логика экспорта

CardLabelCollection — класс

Представляет коллекцию меток карточек.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class CardLabelCollection : IEnumerable<LabelObject>, IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов коллекции.
`Item`	Возвращает метку с заданным идентификатором.
`Count`	Возвращает элемент коллекции с указанным индексом.
`Session`	Возвращает текущую сессию клиента.

Методы

Имя	Описание
`Add(LabelObject)`	Добавляет в коллекцию указанный элемент.
`Clear`	Удаляет все элементы коллекции.

Имя	Описание
`Contains(Guid)`	Определяет вхождение в коллекцию метки с заданным идентификатором.
`GetEnumerator`	Возвращает перечислитель.
`Remove(Guid)`	Удаляет метку с заданным идентификатором.

CardManager — класс

Представляет объектную модель **менеджера карточек**. Содержит методы чтение, изменение, создание и удаление карточек. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class CardManager
```

Свойства

Имя	Описание
`CardLibraries`	Возвращает коллекцию библиотек карточек, зарегистрированных в текущей базе данных.
`CardTypes`	Возвращает коллекцию типов карточек, зарегистрированных в текущей базе данных.
`Session`	Возвращает сессию текущего пользователя.

Методы

Имя	Описание
`CreateCard(Guid)`	Создаёт карточку и возвращает ссылку на интерфейс компонента карточки.

Имя	Описание
CreateCard(Guid, Guid)	Создаёт карточку с заданным идентификатором и возвращает ссылку на интерфейс компонента карточки.
CreateCardData(Guid)	Создание нового экземпляра карточки указанного типа.
CreateCardData(Guid, Guid)	Создание нового экземпляра карточки указанного типа с заданным идентификатором.
DeleteCard(Guid)	Удаляет карточку.
DeleteCard(Guid, Boolean)	Удаляет карточку с возможностью восстановления.
GetCard(Guid)	Возвращает ссылку на программный интерфейс карточки.
GetCard(Guid, Boolean)	Возвращает ссылку на программный интерфейс карточки с обновлением кэша.
GetCardData(Guid)	Возвращает по идентификатору карточки объект для работы с её данными.
GetCardData(Guid, Boolean)	Возвращает по идентификатору карточки объект для работы с её данными, с обновлением кэша.
GetCardData(Guid, Guid)	Возвращает данные карточки с секцией заданного типа и заданным идентификатором строки.
GetDictionary(Guid)	Возвращает ссылку на программный интерфейс справочника.
GetDictionary(Guid, Boolean)	Возвращает ссылку на программный интерфейс справочника с обновлением кэша.
GetDictionaryData(Guid)	Возвращает данные справочника указанного типа.

Имя	Описание
GetDictionaryData(Guid, Boolean)	Возвращает данные справочника указанного типа с обновлением кэша.
GetDictionaryDataAsReadonly(Guid)	Возвращает данные справочника в режиме "только чтение".
GetDictionaryAsReadonly(Guid)	Возвращает объект справочника в режиме "только чтение".
GetCards(Guid)	Возвращает коллекцию всех карточек данного типа.
GetCardState(Guid)	Возвращает статус карточки. Позволяет определить, что карточка удалена.
PurgeCards(DateTime)	Окончательно удаляет карточки помеченные к удалению с учетом даты удаления.
PurgeCards(DateTime, Guid)	Окончательно удаляет карточки помеченные к удалению с учетом даты удаления и типа карточки.
RestoreCard(Guid)	Восстанавливает карточку, помеченную к удалению.
BatchUpdate(Guid[])	Выполняет пакетное обновление данных для указанных карточек.
PurgeCache()	Очистка кэша менеджера объектов.
ImportCards(Stream)	Импортирует карточку из потока XML данных.
ImportCards(Stream, ImportCardInspector)	Импортирует карточку из потока XML данных, с заданным алгоритмом импорта.
UpdateMetadata(string)	Обновляет динамические метаданные указанными XML-данными.
ClearLink(Guid, Guid, String)	Позволяет очистить ссылку, хранящуюся в указанном поле указанной строки.

Имя	Описание
GetLinksFromCard(Guid, LinkType, int)	Возвращает коллекцию карточек, на которые ссылается данная карточка, с заданным типом ссылки и глубиной вложенности.
GetLinksToCard(Guid, LinkType, int)	Возвращает коллекцию карточек, ссылающихся на данную, с заданным типом ссылки и глубиной вложенности.
GetViewData(ViewReadRequest)	Возвращает данные представления карточки для заданного запроса представления.
GetViewData(ViewSource)	Возвращает данные представления карточки для заданного источника данных представлений.
GetViewData(ViewSource, Guid)	Возвращает данные сохранённого представления с указанным идентификатором.
GetViewData(ViewSource, String)	Возвращает данные представления с заданным описанием.
GetViewData(ViewSource, Guid, QueryParameterCollection)	Возвращает данные сохранённого представления с указанным идентификатором, с указанием параметров поиска.
GetViewData(ViewSource, string, QueryParameterCollection)	Возвращает данные представления с учетом приведенного описания, с указанием параметров поиска.
GetViewRequestFromCardLib	Создание экземпляра запроса представления для указанной библиотеки карточек.
GetViewRequestFromCards	Создание экземпляра запроса представления для карточек.
GetViewRequestFromCardType(Guid)	Создание экземпляра запроса представления для заданного типа карточек.

Имя	Описание
GetViewRequest	Создание экземпляра запроса представления.
GetViewRequestFromFolder(Guid)	Создание экземпляра запроса представления для заданной папки.
GetViewRequestFromInstance(Guid)	Создание экземпляра запроса представления для заданного экземпляра.
GetViewRequestFromRecycleBin	Создание экземпляра запроса представления для системной папки "Корзина".
GetViewRequestFromReferences	Создание экземпляра запроса представления для системной папки "Справочники".
GetViewRequestFromSearch(Guid)	Создание экземпляра запроса представления для существующего поискового запроса.
GetViewRequestFromSearch(String)	Создание экземпляра запроса представления для переданного поискового запроса.
GetViewRequestFromSearchFolder(Guid)	Создание экземпляра запроса представления для виртуальной папки.
GetViewRequestFromSearchResults(Guid)	Создание экземпляра запроса представления для папки результатов поиска.
FindCards(String)	Ищет карточки, удовлетворяющие поисковому запросу.
FindCardsInfo(Guid)	Ищет информацию по карточкам согласно заданному идентификатору поискового запроса.
FindCardsInfo(String)	Ищет информацию по карточкам согласно заданному поисковому запросу.

Имя	Описание
FindCardsInfo(Guid, QueryParameterCollection)	Ищет информацию по карточкам согласно заданному идентификатору поискового запроса с учетом параметров поиска.
FindCardsInfo(String, QueryParameterCollection)	Ищет информацию по карточкам согласно заданному поисковому запросу с учетом параметров поиска.
FindAggregationCardsInfo (SearchAggregationItemsRequest)	Получает список агрегатов и/или результаты фильтрации.
GetSearchAggregationItemsRequest (SearchAggregationItemsResultType, Guid, Guid, string, int?)	Возвращает результат агрегации.
SearchFilesCitations(SearchFilesCitationsRequest)	Создаёт класс запроса, включающий параметры поиска и формирования цитат.
GetSearchFilesCitationsRequest(SearchFilesCitationParentObjectsType, string)	Создаёт класс запроса, включающий параметры поиска и формирования цитат.

Примеры

В примере, отправляем в архив карточки в которых последнее изменение было внесено более года назад. Поиск осуществляется в открытой сессии пользователя userSession.

```
CardData cardData = userSession.CardManager.GetCardData(new System.Guid("00000000-0000-0000-0000-000000000000")); ①

if (cardData.ChangeDate.Year < DateTime.Now.Year -1) ②
{
    cardData.Archive(ArchiveOptions.IncludeLinkedCards);
}
```

① Получение с сервера данных карточки с идентификатором 00000000-0000-0000-0000-000000000000.

② Проверка даты последнего изменения.

CardManager.CardLibraries — свойство

Возвращает коллекцию библиотек карточек, зарегистрированных в текущей базе данных.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardLibraryCollection CardLibraries { get; }
```

Значение свойства

Тип: CardLibraryCollection

Коллекция библиотек карточек

CardManager.CardTypes — свойство

Возвращает коллекцию типов карточек, зарегистрированных в текущей базе данных.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardTypeCollection CardTypes { get; }
```

Значение свойства

Тип: CardTypeCollection

Коллекция типов карточек

CardManager.Session — свойство

Возвращает сессию текущего пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Сессия пользователя

CardManager.BatchUpdate — метод (Guid[])

Выполняет пакетное обновление данных для указанных карточек.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void BatchUpdate(Guid[] cardIds)
```

Параметры

cardIds

Тип: Guid[]

Массив идентификаторов карточек

Заметки

Данный метод допустим для карточек находящихся в режиме отложенных изменений. Метод осуществляет отправку изменений, накопившихся в карточках, на сервер одним пакетом.

CardManager.ClearLink — метод (Guid, Guid, String)

Позволяет очистить ссылку, хранящуюся в указанном поле указанной строки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void ClearLink(Guid sectionId, Guid rowId, string fieldAlias)
```

Параметры

sectionId

Тип: `Guid`

Идентификатор типа секции со ссылкой

rowId

Тип: `Guid`

Идентификатор строки со ссылкой

fieldAlias

Тип: `String`

Название поля со ссылкой

CardManager.CreateCard — метод (Guid)

Создаёт карточку и возвращает ссылку на интерфейс компонента карточки.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public object CreateCard(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: `Guid`

Идентификатор типа карточки

Возвращаемое значение

Тип: `Object`

Интерфейс карточки

Заметки

В дальнейшем возвращенное значение можно привести к конкретному типу карточки.

Примеры

```
const string FILE_CARD_TYPE = "6E39AD2B-E930-4D20-AAFA-C2ECF812C2B3"; ①
VersionedFileCard fileCard = (VersionedFileCard)session.CardManager.CreateCard(new Guid
(FILE_CARD_TYPE)); ②
fileCard.Initialize(@"..\Test.doc", Guid.Empty, false, false); ③
FileVersion fileVersion = fileCard.CurrentVersion;
MessageBox.Show(fileVersion.VersionString); ④
```

① Константа, идентификатор типа карточки-файла.

② Создание карточки файла.

③ Первичная инициализация карточки файла.

④ Получение первой версии файла.

CardManager.CreateCard — метод (Guid, Guid)

Создаёт карточку с заданным идентификатором и возвращает ссылку на интерфейс компонента карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract object CreateCard(Guid cardTypeId, Guid cardId)
```

Параметры

cardTypeId

Тип: Guid

Идентификатор типа карточки

cardId

Тип: Guid

Идентификатор карточки

Возвращаемое значение

Тип: Object

Интерфейс карточки

CardManager.CreateCardData — метод (Guid)

Создание нового экземпляра карточки указанного типа.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public CardData CreateCardData(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: Guid

Идентификатор типа карточки

Возвращаемое значение

Тип: CardData

Карточка

CardManager.CreateCardData — метод (Guid, Guid)

Создание нового экземпляра карточки указанного типа с заданным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardData CreateCardData(Guid cardTypeId, Guid cardId)
```

Параметры

cardTypeId

Тип: Guid

Идентификатор типа карточки

cardId

Тип: Guid

Идентификатор карточки

Возвращаемое значение

Тип: CardData

Карточка

CardManager.DeleteCard — метод (Guid)

Удаляет карточку.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void DeleteCard(Guid cardId)
```

Параметры

cardId

Тип: Guid

Идентификатор карточки

CardManager.DeleteCard — метод (Guid, Boolean)

Удаляет карточку с возможностью восстановления.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void DeleteCard(Guid cardId, bool permanently)
```

Параметры

cardId

Тип: Guid

Идентификатор карточки

permanently

Тип: `Boolean`

Если `false`, то удаляет с возможностью восстановления, иначе без такой возможности

CardManager.FindAggregationCardsInfo — метод (SearchAggregationItemsRequest)

Получает список агрегатов и/или результаты фильтрации.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public virtual IEnumerable<SearchAggregationItemInfo> FindAggregationCardsInfo
(SearchAggregationItemsRequest request)
```

Параметры

request

Тип: `SearchAggregationItemsRequest`

Объект запроса.

Возвращаемое значение

Тип: `SearchAggregationItemInfo`

Возвращает список объектов.

CardManager.FindCards — метод (String)

Ищет карточки, удовлетворяющие поисковому запросу.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract CardDataCollection FindCards(string queryXml)
```

Параметры

queryXml

Тип: `String`

Поисковый запрос

Возвращаемое значение

Тип: `CardDataCollection`

Коллекция карточек

CardManager.FindCardsInfo — метод (Guid)

Ищет информацию по карточкам согласно заданному идентификатору поискового запроса.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public InfoRowCollection FindCardsInfo(Guid queryId)
```

Параметры

queryId

Тип: `Guid`

Идентификатор запроса

Возвращаемое значение

Тип: `InfoRowCollection`

Коллекция данных

Заметки

Поиск осуществляется согласно заданному запросу, сохранённому в системе.

CardManager.FindCardsInfo — метод (String)

Ищет информацию по карточкам согласно заданному поисковому запросу.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public InfoRowCollection FindCardsInfo(string queryXml)
```

Параметры

queryXml

Тип: String

Текст запроса

Возвращаемое значение

Тип: InfoRowCollection

Коллекция данных

CardManager.FindCardsInfo — метод (Guid, QueryParameterCollection)

Ищет информацию по карточкам согласно заданному идентификатору поискового запроса с учетом параметров поиска.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public InfoRowCollection FindCardsInfo(Guid queryId, QueryParameterCollection parameters)
```

Параметры

queryId

Тип: Guid

Идентификатор запроса

parameters

Тип: QueryParameterCollection

Параметр поиска

Возвращаемое значение

Тип: InfoRowCollection

Коллекция данных

Заметки

Поиск осуществляется согласно заданному запросу, сохранённому в системе.

CardManager.FindCardsInfo — метод (String, QueryParameterCollection)

Ищет информацию по карточкам согласно заданному поисковому запросу с учетом параметров поиска.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public InfoRowCollection FindCardsInfo(string queryXml, QueryParameterCollection
parameters)
```

Параметры

queryXml

Тип: String

Текст запроса

parameters

Тип: QueryParameterCollection

Параметр поиска

Возвращаемое значение

Тип: InfoRowCollection

Коллекция данных

CardManager.GetCard — метод (Guid)

Возвращает ссылку на программный интерфейс карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public object GetCard(Guid cardId)
```

Параметры

cardId

Тип: Guid

Идентификатор карточки

Возвращаемое значение

Тип: Object

Интерфейс карточки

Заметки

Данный метод работает только с карточками входящими в библиотеку "Системные карточки" ("Карточка папок", "Карточка настроек пользователя" и т.д.). Для простых карточек метод вернет ошибку System.IO.FileNotFoundException.

CardManager.GetCard — метод (Guid, Boolean)

Возвращает ссылку на программный интерфейс карточки с обновлением кэша.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract object GetCard(Guid cardId, bool refresh)
```

Параметры

cardId

Тип: Guid

Идентификатор карточки

refresh

Тип: Boolean

`true` — обновить кэш карточек; `false` — не обновлять

Возвращаемое значение

Тип: `Object`

Интерфейс карточки

CardManager.GetCardData — метод (Guid)

Возвращает по идентификатору карточки объект для работы с её данными.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public CardData GetCardData(Guid cardId)
```

Параметры

cardId

Тип: `Guid`

Идентификатор карточки

Возвращаемое значение

Тип: `CardData`

Карточка

CardManager.GetCardData — метод (Guid, Boolean)

Возвращает по идентификатору карточки объект для работы с её данными, с обновлением кэша.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract CardData GetCardData(Guid cardId, bool refresh)
```

Параметры

cardId

Тип: `Guid`

Идентификатор карточки

refresh

Тип: `Boolean`

`true` — обновить кэш карточек; `false` — не обновлять

Возвращаемое значение

Тип: `CardData`

Карточка

Заметки

Параметр `refresh` не влияет на обновление данных карточки при наличии незавершённых обновления. В этом случае рекомендуется выполнить принудительное обновление — вызвать метод `CardData.Refresh`.

CardManager.GetCardData — метод (Guid, Guid)

Возвращает данные карточки с секцией заданного типа и заданным идентификатором строки.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract CardData GetCardData(Guid rowId, Guid sectionId)
```

Параметры

rowId

Тип: `Guid`

Идентификатор строки

sectionId

Тип: `Guid`

Идентификатор секции

Возвращаемое значение

Тип: `CardData`

Карточка

Заметки

Этот метод удобно использовать, если идентификатор самой карточки неизвестен, но известна одна из её строк.

CardManager.GetCards — метод (Guid)

Возвращает коллекцию всех карточек данного типа.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract CardDataCollection GetCards(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: `Guid`

Идентификатор типа карточек

Возвращаемое значение

Тип: `CardDataCollection`

Коллекция карточек

CardManager.GetCardState — метод (Guid)

Возвращает статус карточки. Позволяет определить, что карточка удалена.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract ObjectState GetCardState(Guid cardId)
```

Параметры

cardId

Тип: Guid

Идентификатор карточки

Возвращаемое значение

Тип: ObjectState

Статус карточки

CardManager.GetDictionary — метод (Guid)

Возвращает ссылку на программный интерфейс справочника.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public object GetDictionary(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: Guid

Идентификатор типа справочника

Возвращаемое значение

Тип: Object

Интерфейс справочника

CardManager.GetDictionary — метод (Guid, Boolean)

Возвращает ссылку на программный интерфейс справочника с обновлением кэша.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract object GetDictionary(Guid cardTypeId, bool refresh)
```

Параметры

cardTypeId

Тип: Guid

Идентификатор типа справочника

refresh

Тип: Boolean

true — обновить кэш карточек; false — не обновлять

Возвращаемое значение

Тип: Object

Интерфейс справочника

CardManager.GetDictionaryData — метод (Guid)

Возвращает данные справочника указанного типа.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public CardData GetDictionaryData(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: Guid

Идентификатор типа справочника

Возвращаемое значение

Тип: `CardData`

Справочник

CardManager.GetDictionaryData — метод (Guid, Boolean)

Возвращает данные справочника указанного типа с обновлением кэша.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract CardData GetDictionaryData(Guid cardTypeId, bool refresh)
```

Параметры

cardTypeId

Тип: `Guid`

Идентификатор типа справочника

refresh

Тип: `Boolean`

`true` — обновить кэш карточек; `false` — не обновлять

Возвращаемое значение

Тип: `CardData`

Справочник

GetDictionaryDataAsReadonly — метод (Guid)

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public virtual CardData GetDictionaryDataAsReadonly(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: System.Guid

Идентификатор типа карточки.

GetDictionaryAsReadonly — метод (Guid)

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual object GetDictionaryAsReadonly(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: System.Guid

Идентификатор типа карточки.

CardManager.GetLinksToCard — метод (Guid, LinkType, int)

Возвращает коллекцию карточек, ссылающихся на данную, с заданным типом ссылки и глубиной вложенности.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardDataCollection GetLinksFromCard(Guid cardId, LinkType linkType, int
recurseDepth)
```

Параметры

cardId

Тип: Guid

Идентификатор карточки

linkType

Тип: LinkType

Тип ссылки

recurseDepth

Тип: System.Int32

Глубина рекурсии

Возвращаемое значение

Тип: CardDataCollection

Коллекция карточек

Заметки

Параметр recurseDepth определяет на сколько шагов ссылающиеся карточки могут отстоять от данной.

CardManager.GetLinksFromCard — метод (Guid, LinkType, int)

Возвращает коллекцию карточек, на которые ссылается данная карточка, с заданным типом ссылки и глубиной вложенности.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardDataCollection GetLinksToCard(Guid cardId, LinkType linkType, int
recurseDepth)
```

Параметры

cardId

Тип: Guid

Идентификатор карточки

linkType

Тип: LinkType

Тип ссылки

recurseDepth

Тип: System.Int32

Глубина рекурсии

Возвращаемое значение

Тип: CardDataCollection

Коллекция карточек

Заметки

Параметр recurseDepth определяет на сколько шагов карточки могут отстоять от данной.

CardManager.GetSearchAggregationItemsRequest — метод (SearchAggregationItemsResultType, Guid, Guid, string, int?)

Возвращает результат агрегации.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual SearchAggregationItemsRequest GetSearchAggregationItemsRequest
(SearchAggregationItemsResultType resultType, Guid folderId, Guid queryId, string
queryXml, int? localeId)
```

Параметры

resultType

Тип: SearchAggregationItemsResultType

Тип получаемого результата.

folderId

Тип: System.Guid

Идентификатор папки-хранилища для сохранения результатов фильтрации.

queryId

Тип: System.Guid

Идентификатор поиска. Альтернатива `QueryXml`. Если он не `Guid.Empty`, будет использоваться именно сохранённый запрос.

queryXml

Тип: `System.String`

Xml поиска. Должно быть заполнено как минимум поле QueryString полнотекстового поиска. Также могут быть использованы любые другие части — как и сам атрибутивный поиск, так и ограничения по результатам (ограничение на папку например).

`QueryXml` можно не заполнять, но в таком случае не будет заполнена папка результатами поиска, но список агрегатов вернётся.

localeId

Тип: `System.Int32`

Номер локализации в которой необходимо вернуть имя агрегации. Если значение не задано или не найдено в настройках агрегата, будет использоваться английская локаль. Если английская локаль не найдена, будет использоваться первая найденная. Если никакие настройки не заданы, в качестве значения будет использоваться алиас агрегата.

Возвращаемое значение

Тип: `SearchAggregationItemsRequest`

Возвращает

CardManager.GetSearchFilesCitationsRequest — метод (SearchFilesCitationParentObjectsType, string)

Метод создаёт класс запроса, включающий параметры поиска и формирования цитат.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public virtual SearchFilesCitationsRequest GetSearchFilesCitationsRequest
(SearchFilesCitationParentObjectsType objectsType, string fullTextQuery)
```

Параметры

objectsType

Тип: `SearchFilesCitationParentObjectsType`

Тип родительского объекта, по которому будет осуществляться фильтрация. Это либо `Cards` — карточки, либо `Files` — файлы.

fullTextQuery

Тип: `System.String`

Строка, по которой будет осуществляться полнотекстовый поиск. Именно фраза для поиска, а не весь XML поиска.

Возвращаемое значение

`SearchFilesCitationsRequest`

Тип: `SearchFilesCitationsRequest`

Запрос цитируемых данных из файла.

CardManager.GetViewData — метод (ViewReadRequest)

Возвращает данные представления карточки для заданного запроса представления.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract InfoRowCollection GetViewData(ViewReadRequest request)
```

Параметры

request

Тип: `ViewReadRequest`

Запрос представления

Возвращаемое значение

Тип: `InfoRowCollection`

Коллекция данных

CardManager.GetViewData — метод (ViewSource)

Возвращает данные представления карточки для заданного источника данных представлений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public InfoRowCollection GetViewData(ViewSource viewSource)
```

Параметры

viewSource

Тип: ViewSource

Источника данных представления

Возвращаемое значение

Тип: InfoRowCollection

Коллекция данных

CardManager.GetViewData — метод (ViewSource, Guid)

Возвращает данные сохранённого представления с указанным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public InfoRowCollection GetViewData(ViewSource viewSource, Guid viewId)
```

Параметры

viewSource

Тип: ViewSource

Источника данных представления

viewId

Тип: Guid

Идентификатор представления

Возвращаемое значение

Тип: InfoRowCollection

Коллекция данных

CardManager.GetViewData — метод (ViewSource, String)

Возвращает данные представления с заданным описанием.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public InfoRowCollection GetViewData(ViewSource viewSource, string viewXml)
```

Параметры

viewSource

Тип: ViewSource

Источника данных представления

viewXml

Тип: String

Описание представления

Возвращаемое значение

Тип: InfoRowCollection

Коллекция данных

CardManager.GetViewData — метод (ViewSource, Guid, QueryParameterCollection)

Возвращает данные сохранённого представления с указанным идентификатором, с указанием параметров поиска.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public InfoRowCollection GetViewData(ViewSource viewSource, Guid viewId,
QueryParameterCollection parameters)
```

Параметры

viewSource

Тип: ViewSource

Источника данных представления

viewId

Тип: Guid

Идентификатор представления

parameters

Тип: QueryParameterCollection

Параметры запроса

Возвращаемое значение

Тип: InfoRowCollection

Коллекция данных

CardManager.GetViewData — метод (ViewSource, string, QueryParameterCollection)

Возвращает данные представления с учетом приведенного описания, с указанием параметров поиска.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public InfoRowCollection GetViewData(ViewSource viewSource, string viewXml,
QueryParameterCollection parameters)
```

Параметры

viewSource

Тип: `ViewSource`

Источника данных представления

viewXml

Тип: `String`

Описание представления

parameters

Тип: `QueryParameterCollection`

Параметры запроса

Возвращаемое значение

Тип: `InfoRowCollection`

Коллекция данных

CardManager.GetViewRequest — метод

Создание экземпляра запроса представления.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract ViewReadRequest GetViewRequest()
```

Возвращаемое значение

Тип: `ViewReadRequest`

Запрос представления

CardManager.GetViewRequestFromCardLib — метод (Guid)

Создание экземпляра запроса представления для указанной библиотеки карточек.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ViewReadRequest GetViewRequestFromCardLib(Guid cardLibId)
```

Параметры

cardLibId

Тип: Guid

Идентификатор библиотеки карточек

Возвращаемое значение

Тип: ViewReadRequest

Запрос представления

CardManager.GetViewRequestFromCards — метод

Создание экземпляра запроса представления для карточек.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ViewReadRequest GetViewRequestFromCards()
```

Возвращаемое значение

Тип: ViewReadRequest

Запрос представления

CardManager.GetViewRequestFromCardType — метод (Guid)

Создание экземпляра запроса представления для заданного типа карточек.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ViewReadRequest GetViewRequestFromCardType(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: Guid

Идентификатор типа карточек

Возвращаемое значение

Тип: ViewReadRequest

Запрос представления

CardManager.GetViewRequestFromFolder — метод (Guid)

Создание экземпляра запроса представления для заданной папки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ViewReadRequest GetViewRequestFromFolder(Guid folderId)
```

Параметры

folderId

Тип: Guid

Идентификатор папки

Возвращаемое значение

Тип: ViewReadRequest

Запрос представления

CardManager.GetViewRequestFromInstance — метод (Guid)

Создание экземпляра запроса представления для заданного экземпляра.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ViewReadRequest GetViewRequestFromInstance(Guid instanceId)
```

Параметры

instanceId

Тип: Guid

Идентификатор объекта

Возвращаемое значение

Тип: ViewReadRequest

Запрос представления

CardManager.GetViewRequestFromRecycleBin — метод

Создание экземпляра запроса представления для системной папки "Корзина".

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ViewReadRequest GetViewRequestFromRecycleBin()
```

Возвращаемое значение

Тип: ViewReadRequest

Запрос представления

CardManager.GetViewRequestFromReferences — метод

Создание экземпляра запроса представления для системной папки "Справочники".

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ViewReadRequest GetViewRequestFromReferences()
```

Возвращаемое значение

Тип: ViewReadRequest

Запрос представления

CardManager.GetViewRequestFromSearch — метод (Guid)

Создание экземпляра запроса представления для существующего поискового запроса.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ViewReadRequest GetViewRequestFromSearch(Guid searchQueryId)
```

Параметры

searchQueryId

Тип: Guid

Идентификатор запроса

Возвращаемое значение

Тип: ViewReadRequest

Запрос представления

CardManager.GetViewRequestFromSearch — метод (String)

Создание экземпляра запроса представления для переданного поискового запроса.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ViewReadRequest GetViewRequestFromSearch(string searchQuery)
```

Параметры

searchQuery

Тип: String

Поисковый запрос

Возвращаемое значение

Тип: ViewReadRequest

Запрос представления

CardManager.GetViewRequestFromSearchFolder — метод (Guid)

Создание экземпляра запроса представления для виртуальной папки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ViewReadRequest GetViewRequestFromSearchFolder(Guid folderId)
```

Параметры

folderId

Тип: Guid

Идентификатор папки

Возвращаемое значение

Тип: ViewReadRequest

Запрос представления

CardManager.GetViewRequestFromSearchResults — метод (Guid)

Создание экземпляра запроса представления для папки результатов поиска.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ViewReadRequest GetViewRequestFromSearchFolder(Guid folderId)
```

Параметры

folderId

Тип: Guid

Идентификатор виртуальной папки

Возвращаемое значение

Тип: ViewReadRequest

Запрос представления

CardManager.ImportCards — метод (Stream)

Импортирует карточку из потока XML данных.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public CardDataCollection ImportCards(Stream stream)
```

Параметры

stream

Тип: Stream

Поток данных

Возвращаемое значение

Тип: CardDataCollection

Коллекция карточек

Заметки

Входящие данные должны иметь XML-формат. В потоке может быть несколько

карточек.

Примеры

```
FileStream file = new FileStream(@"..\TestCard.xml", FileMode.Open);
ImportCardInspectorReplacer inspector = new ImportCardInspectorReplacer(session);
CardDataCollection cards = session.CardManager.ImportCards(file);
```

CardManager.ImportCards — метод (Stream, ImportCardInspector)

Импортирует карточку из потока XML данных, с заданным алгоритмом импорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardDataCollection ImportCards(Stream stream, ImportCardInspector
cardInspector)
```

Параметры

stream

Тип: Stream

Поток данных

cardInspector

Тип: ImportCardInspector

Поток данных

Возвращаемое значение

Тип: CardDataCollection

Коллекция карточек

Заметки

В потоке может быть несколько карточек.

UpdateMetadata — метод (string)

Обновляет динамические метаданные указанными XML-данными.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void UpdateMetadata(string metadataXml);
```

Параметры

string

Тип: metadataXml

Метаданные

CardManager.PurgeCache — метод

Очистка кэша менеджера объектов.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void PurgeCache()
```

CardManager.PurgeCards — метод (DateTime)

Окончательно удаляет карточки помеченные к удалению с учетом даты удаления.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void PurgeCards(DateTime startDate)
```

Параметры

startDate

Тип: DateTime

Начальная дата удаления

Заметки

Карточки помечаются к удалению методом CardManager.DeleteCard.

CardManager.PurgeCards — метод (DateTime, Guid)

Окончательно удаляет карточки помеченные к удалению с учетом даты удаления и типа карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void PurgeCards(DateTime startDate, Guid cardTypeId)
```

Параметры

startDate

Тип: DateTime

Начальная дата удаления

cardTypeId

Тип: Guid

Идентификатор типа карточки

Заметки

Для выбора карточек, подлежащих удалению, можно использовать дату удаления и тип карточки

CardManager.RestoreCard — метод (Guid)

Восстанавливает карточку помеченную к удалению.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void RestoreCard(Guid cardId)
```

Параметры

cardId

Тип: Guid

Идентификатор карточки

CardManager.SearchFilesCitations — метод (SearchFilesCitationsRequest)

Метод осуществляет непосредственно получение данных.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual IEnumerable<FilesCitationInfo> SearchFilesCitations
(SearchFilesCitationsRequest request)
```

Параметры

request

Тип: SearchFilesCitationsRequest

Запрос цитируемых данных из файла.

Возвращаемое значение

Тип: `FilesCitationInfo`

Результат поиска — список цитат, сгруппированных сначала по карточкам — FilesCitationInfo, потом по файлам — CitationFileItem.

Connection — класс

Соединение с сервером Docsvision.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class Connection
```

Свойства

Имя	Описание
ComplexOperationsTimeout	Задаёт или возвращает таймаут для сложных операций.
ConnectionString	Задаёт или возвращает строку соединения с сервером.
ConnectionTimeout	Задаёт или возвращает таймаут для операций.
CurrentThreadId	Возвращает
DataSource	Возвращает
ErrorDetails	Возвращает
ErrorDetailsEx	Возвращает
ErrorInfo	Возвращает
NetworkControlEnabled	Задаёт или возвращает
SimpleOperationsTimeout	Задаёт или возвращает

Методы

Имя	Описание
CancelRequest(Int32)	
Close	
IsInRequest(Int32)	
Open	

CitationFileItem — класс

Список цитат, сгруппированных по файлам.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class CitationFileItem
```

Свойства

Имя	Описание
FileId	Идентификатор файла.
Citations	Список цитат.
HasMore	Флаг, определяющий, есть ли ещё цитаты в файле сверх заданного ограничения.

Примечание

Цитаты также группируются по карточкам — FilesCitationInfo.

ClearLogStrategy — перечисление

Определяет стратегию автоматической очистки журнала.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public enum ClearLogStrategy
```

Члены

Имя члена	Описание
Never	Никогда не очищать журнал.
ByCount	Очищать при превышении количества записей.
ByDate	Очищать при превышении времени хранения.

Заметки

Подробнее стратегия очистки описывается параметрами соответствующего

метода `SetLogStrategy`.

CryptObjectType — перечисление

Тип крипто-объекта.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum CryptObjectType
```

Члены

Имя члена	Описание
`Signature`	Подпись (ЭП).
`Key`	Публичный ключ шифрования.

FilesCitationInfo — класс

Список цитат, сгруппированных по карточкам.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class FilesCitationInfo
```

Свойства

Имя	Описание
`CardId`	Идентификатор карточки.
`Files`	Список файлов с цитатами.
`HasMore`	Флаг, определяющий, есть ли ещё файлы с цитатами сверх заданного ограничения.

Примечание

Цитаты также группируются по файлам — CitationFileItem.

SearchFilesCitationsRequest — класс

Запрос цитируемых данных из файла.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class SearchFilesCitationsRequest
```

Свойства

Имя	Описание
ParentObjects	Список идентификаторов карточек или файлов (в зависимости от типа заданного в objectsType), для которых будет осуществлён поиск цитат. Если список пустой, будут возвращены все доступные с учётом выставленных ограничений результаты.
FilesPerCardLimit	Ограничение количества файлов для которых будут отображены цитаты в рамках одной карточки. По умолчанию: 2.
CitationsCount	Ограничение количества цитат в рамках одного файла. По умолчанию: 1.
FragmentSize	Количество символов, которое будет содержать выводимая цитата. По умолчанию: 200.

Имя	Описание
`PreTags`	Список HTML-тегов, которые предшествуют цитате. По умолчанию: `<mark>`. `PreTags` является массивом, значение необходимо сохранять в виде стерилизованной в json-строки.
`PostTags`	Список HTML-тегов, которые идут после цитаты. По умолчанию: `</mark>`. `PostTags` является массивом, значение необходимо сохранять в виде стерилизованной в json-строки.

Примечание

Указанные выше параметры являются необязательными. Если они не заданы, будут использованы значения из БД. Если в БД значения также не заданы, будут использованы значения по умолчанию.

В рамках БД значения параметров задаются через процедуру **dvsys_setting_set 'SettingName', 'SettingValue'**, подробнее можно узнать в соответствующем разделе.

Имена настроек:

- `FilesPerCardLimit` — `"FullTextIndexing_CitationsFilesPerCardLimit"`
- `CitationsCount` — `"FullTextIndexing_CitationsCount"`
- `FragmentSize` — `"FullTextIndexing_CitationsFragmentSize"`
- `PreTags` — `"FullTextIndexing_CitationsPreTags"`
- `PostTags` — `"FullTextIndexing_CitationsPostTags"`

EventType — перечисление

Определяет типы записей в журнале событий.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum EventType
```

Члены

Имя члена	Описание
`Error`	Ошибка.
`Information`	Информация.
`Warning`	Предупреждение.
`Audit`	Аудит.

ExportFlags — перечисление

Задаёт формат экспорта данных.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
[Flags]
public enum ExportFlags
```

Члены

Имя члена	Описание
`None`	Условия не заданы. Значение `0x003`.

Имя члена	Описание
`Namespace`	Указывать пространство имён. Значение `0x013`.
`LinkedCards`	Включать связанные карточки. Значение `0x023`.
`LinkedFiles`	Включать данные файлов. Значение `0x043`.
`LinkedRows`	Включать связанные строки. Значение `0x083`.
`LinkedFilesNoData`	Включать атрибуты файлов (без данных). Значение `0x103`.
`ChildRows`	Включать дочерние строки иерархии. Значение `0x203`.
`ChildSections`	Включать строки дочерних секций. Значение `0x403`.
`Security`	Включать дескрипторы безопасности объектов. Значение `0x803`.

ExportCardErrorEventArgs — класс

Предоставляет данные для события `ExportCardInspector.Error`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public class ExportCardErrorEventArgs : EventArgs
```

Конструкторы

Имя	Описание
`ExportCardErrorEventArgs()`	Инициализирует новый экземпляр класса ExportCardErrorEventArgs.

Имя	Описание
`ExportCardErrorEventArgs(Int32, String)`	Инициализирует новый экземпляр класса ExportCardErrorEventArgs с указанным кодом и текстом ошибки.

Свойства

Имя	Описание
`Cancel`	Возвращает или задаёт требование отмены экспорта.
`ErrorCode`	Возвращает или задаёт код ошибки экспорта.
`ErrorMessage`	Возвращает или задаёт текст ошибки экспорта.

ExportCardErrorEventArgs — конструктор

Инициализирует новый экземпляр класса `ExportCardErrorEventArgs`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public ExportCardErrorEventArgs()
```

ExportCardErrorEventArgs — конструктор (Int32, String)

Инициализирует новый экземпляр класса `ExportCardErrorEventArgs` с указанным кодом и текстом ошибки.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public ExportCardErrorEventArgs(int errorCode, string errorMessage)
```

Параметры

errorCode

Тип: System.Int32

Код ошибки

errorMessage

Тип: System.String

Текст сообщения

ExportCardErrorEventArgs.Cancel — свойство

Возвращает или задаёт требование отмены экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public bool Cancel { get; set; }
```

Значение свойства

Тип: System.Boolean

true — экспорт следует отменить, иначе — false

ExportCardErrorEventArgs.ErrorCode — свойство

Возвращает или задаёт код ошибки экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public int ErrorCode { get; set; }
```

Значение свойства

Тип: System.Int32

Код ошибки

ExportCardErrorEventArgs.ErrorMessage — свойство

Возвращает или задаёт текст ошибки экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public string ErrorMessage { get; set; }
```

Значение свойства

Тип: System.String

Текст ошибки

ExportCardInspector — класс

Инспектор, позволяющий гибко управлять логикой экспорта. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class ExportCardInspector
```

Свойства

Имя	Описание
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
AfterExport	Инициируется после экспорта всех данных.
AfterExportCard(CardData)	Инициируется после экспорта данных очередной карточки.

Имя	Описание
AfterExportField(RowData, Field)	Инициируется после экспорта данных поля.
AfterExportFile(FileData)	Инициируется после экспорта данных файла.
AfterExportRow(RowData)	Инициируется после экспорта данных строки.
AfterExportSection(SectionData, Guid)	Инициируется после экспорта данных секции.
BeforeExport(Boolean)	Инициируется перед началом экспорта.
BeforeExportCard(CardData, Boolean)	Инициируется перед началом экспорта данных карточки.
BeforeExportField(RowData, Field, Boolean)	Инициируется перед началом экспорта данных поля.
BeforeExportFile(FileData, Boolean)	Инициируется перед началом экспорта данных файла.
BeforeExportRow(RowData, Boolean)	Инициируется перед началом экспорта данных строки.
BeforeExportSection(SectionData, Guid, Boolean)	Инициируется перед началом экспорта данных секции.
OnError(Int32, String, Boolean)	Позволяет обработать ошибку экспорта.

События

Имя	Описание
Error	Событие получает сигнал, когда обнаружена ошибка в процессе экспорта.

ExportCardInspector.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Текущая сессия

ExportCardInspector.AfterExport — метод

Инициируется после экспорта всех данных.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void AfterExport()
```

ExportCardInspector.AfterExportCard — метод (CardData)

Инициируется после экспорта данных очередной карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void AfterExportCard(CardData cardData)
```

Параметры

cardData

Тип: CardData

Текущая карточка

ExportCardInspector.AfterExportField — метод (RowData, Field)

Инициируется после экспорта данных поля.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void AfterExportField(RowData rowData, Field rowField)
```

Параметры

rowData

Тип: RowData

Строка секции

rowField

Тип: Field

Поле строки секции

ExportCardInspector.AfterExportFile — метод (FileData)

Инициируется после экспорта данных файла.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void AfterExportFile(FileData fileData)
```

Параметры

fileData

Тип: FileData

Простой файл

ExportCardInspector.AfterExportRow — метод (RowData)

Инициируется после экспорта данных строки.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void AfterExportRow(RowData rowData)
```

Параметры

rowData

Тип: RowData

Строка секции

ExportCardInspector.AfterExportSection — метод (SectionData, Guid)

Инициируется после экспорта данных секции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void AfterExportSection(SectionData sectionData, Guid parentRowId)
```

Параметры

sectionData

Тип: SectionData

Секция карточки

parentRowId

Тип: System.Guid

Идентификатор родительской строки

ExportCardInspector.BeforeExport — метод (Boolean)

Инициируется перед началом экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void BeforeExport(ref bool cancel)
```

Параметры

cancel

Тип: System.Boolean

true — отменяет экспорт

ExportCardInspector.BeforeExportCard — метод (CardData, Boolean)

Инициируется перед началом экспорта данных карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void BeforeExportCard(CardData cardData, ref bool cancel)
```

Параметры

cardData

Тип: CardData

Карточка

cancel

Тип: System.Boolean

true — отменяет экспорт

Заметки

Можно организовать дополнительную проверку данных карточки и принять решение о недопущении экспорта данных, для чего необходимо установить значение параметра cancel в значение true.

ExportCardInspector.BeforeExportField — метод (RowData, Field, Boolean)

Инициируется перед началом экспорта данных поля.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void BeforeExportField(RowData rowData, Field rowField, ref bool cancel)
```

Параметры

rowData

Тип: RowData

Строка секции

rowField

Тип: Field

Поле строки секции

cancel

Тип: System.Boolean

true — отменяет экспорт

ExportCardInspector.BeforeExportFile — метод (FileData, Boolean)

Инициируется перед началом экспорта данных файла.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void BeforeExportFile(FileData fileData, ref bool cancel)
```

Параметры

fileData

Тип: FileData

Простой файл

cancel

Тип: System.Boolean

true — отменяет экспорт

ExportCardInspector.BeforeExportRow — метод (RowData, Boolean)

Инициируется перед началом экспорта данных строки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void BeforeExportRow(RowData rowData, ref bool cancel)
```

Параметры

rowData

Тип: RowData

Строка секции

cancel

Тип: System.Boolean

true — отменяет экспорт

ExportCardInspector.BeforeExportSection — метод (SectionData, Guid, Boolean)

Инициируется перед началом экспорта данных секции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void BeforeExportSection(SectionData sectionData, Guid parentRowId, ref
bool cancel)
```

Параметры

sectionData

Тип: `SectionData`

Секция карточки

parentRowId

Тип: `System.Guid`

Идентификатор родительской строки

cancel

Тип: `System.Boolean`

`true` — отменяет экспорт

ExportCardInspector.Error — событие

Событие получает сигнал, когда обнаружена ошибка в процессе экспорта.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public event EventHandler<ExportCardErrorEventArgs> Error
```

Заметки

ExtensionManager — класс

Представляет объектную модель **менеджера расширений**. Содержит методы для работы с серверными расширениями. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class ExtensionManager
```

Свойства

Имя	Описание
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
GetExtensionMethod(String, String)	Получает объект для вызова метода серверного расширения.

ExtensionManager.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Текущая сессия

ExtensionManager.GetExtensionMethod — метод (String, String)

Получает объект для вызова метода серверного расширения.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ExtensionMethod GetExtensionMethod(string extensionName, string
methodName)
```

Параметры

extensionName

Тип: System.String

Имя расширения. Данное имя должно быть записано в реестре сервера

methodName

Тип: System.String

Имя вызываемого метода

Возвращаемое значение

Тип: ExtensionMethod

Объект для доступа к методу расширения

Заметки

В случае отсутствия указанного в параметрах расширения или метода, GetExtensionMethod вернет ошибку.

Примеры

В приведенном примере, вызывается метод GetFileContent расширения SampleExtension, с передачей в качестве параметра расположения файла на диске.

```
ExtensionMethod method = session.ExtensionManager.GetExtensionMethod("SampleExtension",
"GetFileContent"); ①

method.Parameters.AddNew("FilePath", ParameterValueType.String).Value = "..\\Test.txt";
②

string content = method.Execute().ToString(); ③
```

① Получаем метод расширения.

② Добавляем параметр метода.

③ Вызов метода.

ExtensionMethod — класс

Описывает структуру серверного расширения.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class ExtensionMethod
```

Свойства

Имя	Описание
ExtensionName	Возвращает название серверного расширения.
MethodName	Возвращает название метода.
Parameters	Возвращает коллекцию параметров вызываемого метода.
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
Execute	Инициирует выполнение метода серверного расширения и возвращает скалярный результат.
ExecuteReader	Инициирует выполнение метода серверного расширения и возвращает набор данных.

ExtensionMethod.ExtensionName — свойство

Возвращает название серверного расширения.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string ExtensionName { get; }
```

Значение свойства

Тип: System.String

Название расширения

ExtensionMethod.MethodName — свойство

Возвращает название метода.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string MethodName { get; }
```

Значение свойства

Тип: System.String

Название метода

ExtensionMethod.Parameters — свойство

Возвращает коллекцию параметров метода.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ExtensionMethodParameterCollection Parameters { get; }
```

Значение свойства

Тип: ExtensionMethodParameterCollection

Параметры метода

ExtensionMethod.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Текущая сессия пользователя

ExtensionMethod.Execute — метод

Инициирует выполнение метода серверного расширения и возвращает скалярный результат.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract object Execute()
```

Возвращаемое значение

Тип: System.Object

Результат запроса к базе данных

ExtensionMethod.ExecuteReader — метод

Инициирует выполнение метода серверного расширения и возвращает набор данных.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract InfoRowCollection ExecuteReader()
```

Возвращаемое значение

Тип: InfoRowCollection

Результат выполнения запроса к базе данных

ExtensionMethodParameter — класс

Представляет параметр метода серверного расширения.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class ExtensionMethodParameter
```

Свойства

Имя	Описание
Name	Возвращает название параметра.
Session	Возвращает текущую сессию пользователя.
Value	Задаёт или возвращает значение параметра.
ValueType	Возвращает тип параметра.

ExtensionMethodParameter.Name — свойство

Возвращает название параметра.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Name { get; }
```

Значение свойства

Тип: System.String

Название параметра

ExtensionMethodParameter.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Сессия пользователя

ExtensionMethodParameter.Value — свойство

Задаёт или возвращает значение параметра.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract object Value { get; set; }
```

Значение свойства

Тип: System.Object

Значение параметра

ExtensionMethodParameter.ValueType — свойство

Возвращает тип параметра.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ParameterValueType ValueType { get; }
```

Значение свойства

Тип: ParameterValueType

Тип параметра

ExtensionMethodParameterCollection — класс

Содержит методы для работы с коллекцией ExtensionMethodParameter.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class ExtensionMethodParameterCollection : IEnumerable
<ExtensionMethodParameter>, IEnumerable
```

Свойства

Имя	Описание
Count	Возвращает количество элементов в коллекции.
Item	Возвращает элемент коллекции с заданным индексом.
Item	Возвращает элемент коллекции с заданным псевдонимом.
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
AddNew(String, ParameterValueType)	Добавляет пустой элемент коллекции.
AddNew(String, ParameterValueType, Object)	Добавляет элемент коллекции.

Имя	Описание
Clear	Удаляет все элементы коллекции.
Contains(String)	Определяет вхождение элемента в коллекцию элемента с заданным псевдонимом.
GetEnumerator	Возвращает перечислитель.

FileData — класс

Содержит методы для организации работы с простыми файлами.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class FileData : IUpdatable, ILockable
```

Свойства

Имя	Описание
ArchiveState	Статус архивного файла.
ChangeDate	Дата последнего изменения файла.
CreateDate	Дата создания файла.
Encrypted	Признак шифрования файла.
ExtAttributes	Дополнительные атрибуты файла.
Id	Идентификатор файла.
InUpdate	Возвращает признак включения для файла режима отложенных изменения.
LockOwner	Возвращает учетную запись владельца блокировки.
LockStatus	Возвращает статус блокировки.
Name	Название файла.
OfflineState	Место хранения файла.

Имя	Описание
`OwnerCardId`	Идентификатор карточки к которой привязан данный файл.
`Session`	Возвращает открытую пользовательскую сессию.
`Signed`	Признак наличия у файла электронной подписи.
`Size`	Возвращает размер файла.
`StdAttributes`	Стандартные атрибуты файла.
`Url`	URL-адрес для доступа к файлу по протоколу http.

Методы

Имя	Описание
`AccessCheck(Int32)`	Проверка прав доступа.
`Archive(Boolean)`	Перенос файла в архив
`BeginUpdate`	Включает режим отложенных изменений.
`BringOnline`	Восстановление файла из `offline` хранилища.
`CancelUpdate`	Отменяет несохранённые изменения и выключает режим отложенных изменений.

Имя	Описание
Copy	Создание копии файла на сервере. Следует учитывать, что при копировании файла будет скопирована только его информационная часть, бинарная часть не копируется. Информационные части оригинала и копии будут указывать на одни бинарные данные до момента их изменения. При изменении бинарных данных (полученных с использованием стандартного API) оригинала или копии, произойдет разделение ссылок: оригинал и копия станут указывать на собственный экземпляр бинарных данных.
Dearchive(Boolean)	Восстановление файла из архива.
Download(String)	Извлечение файла на файловую систему по указанному пути.
EndUpdate	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
ForceUnlock	Принудительно снимает блокировку.
GetAccessControl	Получает описатель прав для файла.
GetAccessControl(AccessControlSections)	Возвращает описатель прав на объект для заданного раздела дескриптора безопасности.
MarkCardForDeletion(Guid)	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
OpenReadStream	Создание потока для чтения файла.
OpenWriteStream	Создание потока для записи файла.
PlaceLock	Устанавливает временную блокировку.

Имя	Описание
`PlaceLock(Boolean)`	Позволяет установить постоянную блокировку.
`Refresh`	Обновляет информацию о файле.
`RemoveLock`	Снимает блокировку, сделанную в своей сессии.
`SetAccessControl(FileDataSecurity)`	Задаёт описатель прав на файл.
`TakeOffline(Boolean)`	Перенос файла в `offline` хранилище.
`UpdateNow`	Позволяет отправить накопленные изменения на сервер.
`Upload(String)`	Загрузка файла из файловой системы на сервер.

FileManager — класс

Представляет объектную модель **менеджера файлов**. Содержит методы для работы с файлами.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class FileManager
```

Свойства

Имя	Описание
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`CopyFile(Guid)`	Возвращает копию объекта `FileData` с заданным идентификатором.

Имя	Описание
`CreateFile(String)`	Создаёт объект типа `FileData` с заданным именем.
`CreateFileEx(String, Guid)`	Создаёт объект типа `FileData` с заданным именем и идентификатором.
`DeleteFile(Guid)`	Удаляет файл с заданным идентификатором.
`FileExists(Guid)`	Проверяет наличие файла с заданным идентификатором.
`FindFiles(FileSearchQuery)`	Осуществляет поиск файлов с использованием полнотекстового поиска.
`GetFile(Guid)`	Возвращает объект типа `FileData` с заданным идентификатором.

ILockable — интерфейс

Разрешает для объекта управление режимом отложенной блокировки.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public interface ILockable
```

Свойства

Имя	Описание
`LockOwner`	При переопределении в производном классе возвращает учетную запись владельца блокировки.
`LockStatus`	При переопределении в производном классе возвращает статус блокировки.

Методы

Имя	Описание
ForceUnlock	При переопределении в производном классе принудительно снимает блокировку.
PlaceLock	При переопределении в производном классе устанавливает временную блокировку.
PlaceLock(Boolean)	При переопределении в производном классе позволяет установить постоянную блокировку.
RemoveLock	При переопределении в производном классе снимает собственную установленную блокировку.

ILockable.LockOwner — свойство

При переопределении в производном классе возвращает учетную запись владельца блокировки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
string LockOwner { get; }
```

Значение свойства

Тип: String

Имя владельца блокировки

ILockable.LockStatus — свойство

При переопределении в производном классе возвращает статус блокировки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
LockStatus LockStatus { get; }
```

Значение свойства

Тип: LockStatus

Статус блокировки объекта

ILockable.ForceUnlock — метод

При переопределении в производном классе принудительно снимает блокировку.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void ForceUnlock()
```

Заметки

Метод позволяет снять блокировку из любой сессии. Требуются административные привилегии.

ILockable.PlaceLock — метод

При переопределении в производном классе устанавливает временную блокировку.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void PlaceLock()
```

ILockable.PlaceLock — метод (Boolean)

При переопределении в производном классе позволяет установить постоянную блокировку.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void PlaceLock(bool permanently)
```

Параметры

permanently

Тип: Boolean

true — постоянная блокировка; false — временная;

Заметки

Временная блокировка ограничена существованием сессии, из которой эта блокировка установлена. Постоянная блокировка должна быть снята тем, кто её установил, либо администратором системы.

ILockable.RemoveLock — метод

При переопределении в производном классе снимает собственную установленную блокировку.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void RemoveLock()
```

Заметки

Данный метод не может быть использован для снятия блокировок, установленных другим пользователем.

ImportCardInspector — класс

Инспектор, позволяющий гибко управлять логикой импорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class ImportCardInspector
```

Свойства

Имя	Описание
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`AfterCreateCard(Object, CardData, Boolean)`	Вызывается после создания новой карточки.
`AfterCreateFile(Object, FileData, Boolean)`	Вызывается после создания нового файла.
`AfterCreateRow(Object, FileData, RowData)`	Вызывается после создания строки.
`AfterCreateSection(Object, Object, SectionData)`	Вызывается после создания новой секции.
`AfterImport(Object, CardDataCollection)`	Вызывается после импорта всех карточек.
`AfterImportCard(Object, Object, CardData)`	Вызывается после импорта карточки.
`AfterImportField(Object, Object, Field)`	Вызывается после импорта поля.
`AfterImportFile(Object, FileData)`	Вызывается после импорта файла.
`AfterImportRow(Object, Object, RowData)`	Вызывается после импорта строки.
`AfterImportSection(Object, Object, SectionData)`	Вызывается после импорта секции.
`BeforeCreateCard(Object, Boolean)`	Вызывается перед созданием новой карточки.
`BeforeCreateFile(Object, Boolean)`	Вызывается перед созданием нового файла.
`BeforeCreateRow(Object, Object, Boolean)`	Вызывается перед созданием строки.

Имя	Описание
BeforeCreateSection(Object, Object, SectionData, Guid, Boolean)	Вызывается перед созданием новой секции.
BeforeImport(Object, Boolean)	Вызывается перед импортом карточек.
BeforeImportCard(Object, Object, CardData, Boolean)	Вызывается перед импортом карточки.
BeforeImportField(Object, Object, Field, Boolean)	Вызывается перед импортом поля.
BeforeImportFile(Object, FileData, Boolean)	Вызывается перед импортом файла.
BeforeImportRow(Object, Object, Boolean)	Вызывается перед импортом строки.
BeforeImportSection(Object, Object, SectionData, Guid, Boolean)	Вызывается перед импортом секции.
GetAttributeValue(Object, String)	Возвращает из объекта значение атрибута с заданным псевдонимом.
GetXml(Object)	Получает из заданного объекта XML-строку.
GetXmlDocument(Object, XmlReaderSettings)	Получает объект типа XmlDocument из заданного объекта.
GetXmlSchema(Object, XmlReaderSettings, ValidationEventHandler)	Получает объект типа XmlSchema из заданного объекта.
OnError(Int32, String, Boolean)	Вызывает событие Error.
SetAttributeValue(Object, String, String)	Задаёт атрибуту объекта заданное значение.

События

Имя	Описание
Error	Событие получает сигнал, когда обнаружена ошибка в процессе импорта.

InfoRowCollection — класс

Содержит методы для работы с коллекцией `InfoRow`. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class InfoRowCollection : IEnumerable<InfoRow>, IEnumerable, IDisposable
```

Свойства

Имя	Описание
`Count`	Возвращает число элементов в коллекции.
`CurrentPage`	Возвращает текущую страницу.
`CursorId`	Возвращает идентификатор серверного курсора.
`Fields`	Возвращает названия полей строк коллекции.
`Item`	Возвращает элемент коллекции с заданным индексом.

Методы

Имя	Описание
`Dispose`	Очищает ресурсы.
`Dispose(Boolean)`	Очищает неуправляемые ресурсы.
`GetEnumerator`	Возвращает перечислитель.
`SaveXml(Stream)`	Сохраняет коллекцию в поток в XML-формате.
`UpdateRow(Guid, ViewReadRequest)`	Обновление данных одной строки.

IProcessInfo — интерфейс

Определяет возможности по управлению состоянием конкретного бизнес-процесса и получению базовой информации о самом бизнес-процессе.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.ObjectManager.Interop.dll`

Синтаксис

```
[Guid("1CCBAB8B-0A93-421A-BDB1-688F52FC226A"), TypeLibType(TypeLibTypeFlags.FDual` |
TypeLibTypeFlags.FNonExtensible` | TypeLibTypeFlags.FDispatchable)]
[ComImport]
public interface IProcessInfo
```

Свойства

Имя	Описание
`Id`	Возвращает идентификатор процесса.
`Name`	Возвращает название процесса.
`Description`	Возвращает описание процесса.
`Author`	Возвращает автора процесса.
`Created`	Возвращает дату и время создания.
`State`	Возвращает состояние процесса.

Методы

Имя	Описание
`Start`	Запускает бизнес-процесс на исполнение.
`Stop`	Останавливает исполнение процесса.
`Pause`	Приостанавливает исполнение.
`Resume`	Возобновляет исполнение приостановленного процесса.

ISecurable — интерфейс

Разрешает объекту реализовывать модель безопасности.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public interface ISecurable : ISecureObject
```

Свойства

Имя	Описание
Modified	Возвращает признак того, что объект был изменён.

Методы

Имя	Описание
SetFullName(String)	Задаёт полное название объекта.
SetName(String)	Задаёт название объекта.

/dv6/programmer/dv6/Platform-ObjectManager-IUpdatable/IProcessInfo_IN/[IProcessInfo — интерфейс]

/dv6/programmer/dv6/Platform-ObjectManager-IUpdatable/ISecurable_IN/[ISecurable — интерфейс]

IUpdatable — интерфейс

Разрешает для объекта режим отложенной записи.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public interface IUpdatable
```

Свойства

Имя	Описание
InUpdate	При переопределении в производном классе возвращает признак включения для объекта режима отложенных изменения.

Методы

Имя	Описание
BeginUpdate	При переопределении в производном классе включает режим отложенных изменений.
CancelUpdate	При переопределении в производном классе отменяет несохранённые изменения и выключает режим отложенных изменений.
EndUpdate	При переопределении в производном классе отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
MarkCardForDeletion(Guid)	При переопределении в производном классе требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
UpdateNow	При переопределении в производном классе позволяет отправить накопленные изменения на сервер.

Заметки

Подробнее о данном режиме читайте в разделе Режим отложенных изменений.

IUpdatable.InUpdate — свойство

При переопределении в производном классе возвращает признак включения для объекта режима отложенных изменения.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
bool InUpdate { get; }
```

Значение свойства

Тип: Boolean

false — режим выключен; true — режим включен

IUpdatable.BeginUpdate — метод

При переопределении в производном классе включает режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void BeginUpdate()
```

IUpdatable.CancelUpdate — метод

При переопределении в производном классе отменяет несохранённые изменения и выключает режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void CancelUpdate()
```

IUpdatable.EndUpdate — метод

При переопределении в производном классе отправляет накопленные изменения на сервер и выключает режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void EndUpdate()
```

IUpdatable.MarkCardForDeletion — метод (Guid)

При переопределении в производном классе требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void MarkCardForDeletion(Guid cardId)
```

Параметры

cardId

Тип: Guid

Идентификатор удаляемой карточки

Заметки

При отмене пакета изменений (в режиме отложенных изменений), карточка будет автоматически удалена. Этот метод позволяет избавить разработчика от необходимости самому следить за жизненным циклом объектов, которые он создаёт в режиме отложенных изменений.

IUpdatable.UpdateNow — метод

При переопределении в производном классе позволяет отправить накопленные изменения на сервер.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void UpdateNow()
```

Заметки

Отправляет накопленные изменения на сервер, на клиенте они очищаются из кэша, режим продолжает действовать. Все изменения после этого момента также будут накапливаться.

IXmlExportable — интерфейс

Разрешает сохранять данные объекта в XML-формате.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public interface IXmlExportable
```

Методы

Имя	Описание
SaveXml(Stream)	При переопределении в производном классе, метод сохраняет представление данных объекта в формате XML, в поток.
SaveXml(Stream, ExportFlags)	При переопределении в производном классе, метод сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта.
SaveXml(Stream, ExportFlags, ExportCardInspector)	При переопределении в производном классе, метод сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта и заданной логикой экспорта.

IXmlExportable.SaveXml — метод (Stream)

При переопределении в производном классе, метод сохраняет представление данных объекта в формате XML, в поток.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void SaveXml(Stream stream)
```

Параметры

stream

Тип: Stream

Поток для данных

IXmlExportable.SaveXml — метод (Stream, ExportFlags)

При переопределении в производном классе, метод сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void SaveXml(Stream stream, ExportFlags exportFlags)
```

Параметры

stream

Тип: Stream

Поток для данных

exportFlags

Тип: ExportFlags

Флаги экспорта

IXmlExportable.SaveXml — метод (Stream, ExportFlags, ExportCardInspector)

При переопределении в производном классе, метод сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта и заданной логикой экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void SaveXml(Stream stream, ExportFlags exportFlags, ExportCardInspector inspector)
```

Параметры

stream

Тип: Stream

Поток для данных

exportFlags

Тип: ExportFlags

Флаги экспорта

inspector

Тип: ExportCardInspector

Логика экспорта

LicenseManager — класс

Класс LicenseManager содержит функциональность по работе с дополнительными опциями лицензии Docsvision, в которых присутствует ограничение по количеству соединений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class LicenseManager
```

Свойства

Имя	Описание
`Session`	Возвращает сессию пользователя.

Методы

Имя	Описание
`DecrementFeatureUsage(Guid)`	Уменьшает число активных соединений в счетчике дополнительной опции лицензии.
`IncrementFeatureUsage(Guid)`	Увеличивает число активных соединений в счетчике дополнительной опции лицензии.

Заметки

На основе методов `DecrementFeatureUsage` и `IncrementFeatureUsage` реализуется механизм контроля числа пользователей, использующих определённый модуль Docsvision, если для данного модуля предусмотрено лицензионное ограничение на количество одновременно работающих пользователей.

Примеры

К примеру, при осуществлении доступа к дополнительному модулю Docsvision, необходимо выполнить проверку того, не достигнут ли предел подключений пользователей:

```
private void Take(UserSession userSession, Guid featureId) ①
{
 try
 {
  userSession.LicenseManager.IncrementFeatureUsage(featureId); ②
  return;
 } catch (StorageServerException ex)
 {
  if (ex.ErrorCode = (int)ErrorCode.FeatureLimit) ③
  {
④
   return;
  } else
  {
⑤
```

```
    return;
   }
  }
 }
```

① `userSession` — сессия пользователя

`featureId` — идентификатор дополнительной опции в лицензии Docsvision.

② Увеличение числа соединений с модулем.

③ Если достигнут предел подключений, то:

④ Вывод сообщения о превышении количества пользователей.

⑤ Вывод сообщения о прочей ошибке.

После выхода пользователя из модуля, соединение должно быть освобождено:

```
private void Release(UserSession userSession, Guid featureId)
{
 try
 {
  userSession.LicenseManager.DecrementFeatureUsage(featureId); ①
  return;
 } catch (StorageServerException ex)
 {
②
  return;
 }
}
```

① Уменьшение числа соединений с модулем.

② Вывод сообщения о прочей ошибке.

LinkType — перечисление

Маска типа ссылки.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
[Flags]
public enum LinkType
```

Члены

Имя члена	Описание
`None`	Не указан. Значение `0x0`.
`Weak`	Слабая ссылка. Значение `0x1`.
`Hard`	Сильная ссылка. Значение `0x2`.
`Auto`	Автоматическое определение. Значение `0x4`.
`All`	Все типы ссылок. Значение `0x7`.

LockedObject — класс

Представляет объект поддерживающий блокировку.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class LockedObject : ILockable
```

Свойства

Имя	Описание
`CardId`	Возвращает идентификатор карточки, если заблокированный объект — карточка.
`CardTypeId`	Возвращает идентификатор типа карточки, если заблокированный объект — карточка.
`Id`	Возвращает идентификатор объекта.

Имя	Описание
`LockOwner`	Возвращает название учетной записи владельца блокировки.
`LockStatus`	Возвращает статус блокировки.
`Name`	Возвращает название объекта.
`ObjectType`	Возвращает тип заблокированного объекта.
`SectionId`	Возвращает тип секции, если заблокированный объект — строка.
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`ForceUnlock`	Принудительно снимает блокировку из другой сессии.
`RemoveLock`	Снятие блокировки.

LockedObjectCollection — класс

Содержит методы для работы с коллекцией `LockedObject`. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class LockedObjectCollection : IEnumerable<LockedObject>, IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает размер коллекции.

Имя	Описание
`Item`	Возвращает элемент коллекции по идентификатору элемента.
`Item`	Возвращает элемент коллекции по его индексу.
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`Contains`	Определяет вхождение в коллекцию элемента с указанным идентификатором.
`GetEnumerator`	Возвращает перечислитель.

LockedObjectType — перечисление

Типы блокированных объектов.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
[Flags]
public enum LockedObjectType
```

Члены

Имя члена	Описание
`Unknown`	Неизвестный тип. Значение `0x0`.
`Card`	Карточка. Значение `0x1`.
`File`	Файл. Значение `0x2`.

Имя члена	Описание
`Row`	Строка карточки. Значение `0x3`.
`All`	Все типы. Значение `0x3`.

LockManager — класс

Представляет объектную модель **менеджера блокировок**. Содержит методы управления блокировками на объекты. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class LockManager
```

Свойства

Имя	Описание
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`GetLockableObject(Guid)`	Возвращает ссылку на заблокированный объект по его идентификатору.
`GetLockedObjects(Boolean, LockedObjectType)`	Получение коллекции всех заблокированных объектов

LockStatus — перечисление

Состояние блокировки объекта.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum LockStatus
```

Члены

Имя члена	Описание
Free	Блокировка отсутствует.
Locked	Установлена временная блокировка другим пользователем.
CheckedOut	Установлена постоянная блокировка другим пользователем.
OwnerLocked	Установлена временная блокировка текущим пользователем.
OwnerCheckedOut	Установлена постоянная блокировка текущим пользователем.

Примеры

```
CardData card = session.CardManager.GetCardData(new System.Guid(
"идентификатор_карточки")); ①
if (card.LockStatus != LockStatus.Free) ②
{
    if (card.LockStatus = LockStatus.OwnerLocked) ③
    {
        card.RemoveLock();
    }
    else
    {
        MessageBox.Show("Объект заблокирован пользователем " + card.LockOwner);
    }
}
```

① Получение с сервера данных карточки с известным идентификатором.

② Проверка блокировки на карточке.

③ Объект заблокирован — проверка владельца.

LogManager — класс

Представляет объектную модель **менеджера журнала**. Содержит методы

работы с журналом системы.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class LogManager
```

Свойства

Имя	Описание
ClearLogCutCount	Возвращает количество записей, которое должно оставаться в журнале после очистки.
ClearLogCutDays	Возвращает максимальное время хранения в журнале.
ClearLogMaxCount	Возвращает максимальное количество записей в журнале.
ClearLogStrategy	Стратегия очистки журнала.
LogStrategy	Стратегия записи в журнал.
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
DeleteMessages(LogSearchQuery)	Удаление записей из журнала согласно заданному фильтру.
ExportMessages(String)	Экспорт сообщений из журнала в строку XML-формата.
FindMessages(LogSearchQuery)	Получение данных из журнала согласно заданному фильтру.
FindMessages(String)	Получение данных из журнала согласно заданному поисковому запросу.

Имя	Описание
GetMessage(Int32)	Возвращает сообщение из журнала согласно заданному идентификатору.
GetMessageDetails(Int32)	Возвращает информационную часть сообщения из журнала согласно заданному идентификатору.
ImportMessages(String)	Импортирует заданные сообщения для последующей обработки.
LogMessage(EventType, Int32, String, Guid, Guid, Guid, String)	Осуществляет запись сообщения в журнал.
LogMessageEx(EventType, Guid, String, Guid, Guid, Guid, String)	Выполняет запись сообщения в журнал с указанием идентификатора соответствующей операции из описания библиотеки карточек.
SetLogStrategy(LogStrategy, ClearLogStrategy, Int32, Int32, Int32)	Установка параметров стратегии ведения журнала событий.

Примеры

Ниже пример кода, где организуется поиск записей в журнале по идентификатору карточки.

```
LogSearchQuery search = new LogSearchQuery();
search.ObjectId = new Guid("идентификатор_карточки");

InfoRowCollection messages = session.LogManager.FindMessages(search);

foreach(InfoRow row in messages)
{
    System.Diagnostics.Debug.Print("Дата: " + row["Date"].ToString() + "; сообщение:" +
row["Data"].ToString());
}
```

LogManager.ClearLogCutCount — свойство

Возвращает количество записей, которое должно оставаться в журнале после очистки.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract int ClearLogCutCount { get; }
```

Значение свойства

Тип: System.Int32

Количество записей

LogManager.ClearLogCutDays — свойство

Возвращает максимальное время хранения в журнале.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract int ClearLogCutDays { get; }
```

Значение свойства

Тип: System.Int32

Количество хранимых дней

Заметки

Имеет значение при стратегии очистки журнала ClearLogStrategy.ByDate.

LogManager.ClearLogMaxCount — свойство

Возвращает максимальное количество записей в журнале.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract int ClearLogMaxCount { get; }
```

Значение свойства

Тип: System.Int32

Количество хранимых сообщений

Заметки

Имеет значение при стратегии очистки журнала ClearLogStrategy.ByCount.

LogManager.ClearLogStrategy — свойство

Стратегия очистки журнала.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ClearLogStrategy ClearLogStrategy { get; }
```

Значение свойства

Тип: ClearLogStrategy

Стратегия очистки журнала

LogManager.LogStrategy — свойство

Стратегия записи в журнал.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract LogStrategy LogStrategy { get; }
```

Значение свойства

Тип: LogStrategy

Стратегия записи в журнал.

LogManager.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Текущая сессия

LogManager.DeleteMessages — метод (LogSearchQuery)

Удаление записей из журнала согласно заданному фильтру.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual void DeleteMessages(LogSearchQuery searchQuery)
```

Параметры

searchQuery

Тип: LogSearchQuery

Фильтр

Примеры

Ниже пример удаления сообщений за текущий день.

```
LogSearchQuery search = new LogSearchQuery();
search.CreatedAfter = DateTime.Now.Subtract(new TimeSpan(1, 0, 0, 0));

session.LogManager.DeleteMessages(search);
```

LogManager.ExportMessages — метод (String)

Экспорт сообщений из журнала в строку XML-формата.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string ExportMessages(string queryXml)
```

Параметры

queryXml

Тип: System.String

Фильтр, согласно которому будут удаляться сообщения

Возвращаемое значение

Тип: System.String

Сообщения в виде строки XML

LogManager.FindMessages — метод (LogSearchQuery)

Получение данных из журнала согласно заданному фильтру.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public virtual InfoRowCollection FindMessages(LogSearchQuery searchQuery)
```

Параметры

searchQuery

Тип: LogSearchQuery

Фильтр сообщений

Возвращаемое значение

Тип: InfoRowCollection

Коллекция нетипизированных данных

Заметки

Так как журнал часто содержит большое количество записей, получение данных из него может занять длительное время. Поэтому рекомендуется максимально специфицировать параметры фильтра, чтобы минимизировать число получаемых сообщений.

Состав полей в возвращаемой коллекции:

- **ID** — идентификатор сообщения в журнале.
- **ResourceID** — идентификатор объекта.
- **TypeID** — тип объекта.
- **ParentID** — идентификатор родительского объекта.
- **Date** — дата операции.
- **Type** — тип операции.
- **Operation** — код операции.
- **Data** — дополнительная информация.

Примеры

```
LogSearchQuery search = new LogSearchQuery();
search.ObjectId = new Guid("идентификатор_карточки");

InfoRowCollection messages = session.LogManager.FindMessages(search);

foreach(InfoRow row in messages)
{
    System.Diagnostics.Debug.Print("Дата: " + row["Date"].ToString() + "; сообщение:" +
row["Data"].ToString());
}
```

LogManager.FindMessages — метод (String)

Получение данных из журнала согласно заданному поисковому запросу.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract InfoRowCollection FindMessages(string queryXml)
```

Параметры

queryXml

Тип: System.String

Поисковый запрос

Возвращаемое значение

Тип: InfoRowCollection

Коллекция нетипизированных данных

Заметки

Так как журнал часто содержит большое количество записей, получение данных из него может занять длительное время. Поэтому рекомендуется максимально специфицировать параметры фильтра, чтобы минимизировать число получаемых сообщений.

Состав полей в возвращаемой коллекции:

- **ID** — идентификатор сообщения в журнале.
- **ResourceID** — идентификатор объекта.
- **TypeID** — тип объекта.
- **ParentID** — идентификатор родительского объекта.
- **Date** — дата операции.
- **Type** — тип операции.
- **Operation** — код операции.
- **Data** — дополнительная информация.

LogManager.GetMessage — метод (Int32)

Возвращает сообщение из журнала согласно заданному идентификатору.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract InfoRowCollection GetMessage(int messageId)
```

Параметры

messageId

Тип: System.Int32

Идентификатор сообщения

Возвращаемое значение

Тип: InfoRowCollection

Коллекция нетипизированных данных

Заметки

Состав полей в возвращаемой коллекции:

- **ID** — идентификатор сообщения в журнале.
- **ResourceID** — идентификатор объекта.
- **TypeID** — тип объекта.
- **ParentID** — идентификатор родительского объекта.
- **Date** — дата операции.
- **Type** — тип операции.
- **Operation** — код операции.
- **Data** — дополнительная информация.

LogManager.GetMessageDetails — метод (Int32)

Возвращает информационную часть сообщения из журнала согласно заданному идентификатору.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string GetMessageDetails(int messageId)
```

Параметры

messageId

Тип: System.Int32

Идентификатор сообщения

Возвращаемое значение

Тип: System.String

Информативная часть сообщения

Заметки

В отличие от GetMessage данный метод возвращает только информативную часть сообщения (Data).

LogManager.ImportMessages — метод (String)

Импортирует заданные сообщения для последующей обработки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract InfoRowCollection ImportMessages(string xml)
```

Параметры

xml

Тип: System.String

Строка содержащая импортируемые события

Возвращаемое значение

Тип: InfoRowCollection

Коллекция нетипизированных данных

Заметки

Импортируемые данные не сохраняются в журнал событий, а служат для последующей обработки и загрузки методами LogMessage или LogMessageEx.

Метод возвращает коллекцию нетипизированных данных `InfoRowCollection`, поля которой соответствуют полям системного журнала (`dvsys_log`).

LogManager.LogMessage — метод (EventType, Int32, String, Guid, Guid, Guid, String)

Осуществляет запись сообщения в журнал.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract void LogMessage(EventType eventType, int eventOperation, string
resourceName, Guid resourceId, Guid typeId, Guid parentId, string eventData)
```

Параметры

eventType

Тип: `EventType`

Тип сообщения

eventOperation

Тип: `System.Int32`

Код операции. Может быть либо одним из стандартных кодов операций, либо собственной операцией (код `0`)

resourceName

Тип: `System.String`

Имя объекта, с которым произведена операция (формируется в зависимости от контекста — например, при операциях с карточкой это может быть её описание)

resourceId

Тип: `System.Guid`

Идентификатор объекта, с которым производится операция (ID карточки, строки, файла и т.д.)

typeId

Тип: `System.Guid`

Тип объекта (например, для карточки — `CardTypeID`)

parentId

Тип: `System.Guid`

Идентификатор родительского объекта (например, для операции со строкой — родительским объектом будет секция)

eventData

Тип: `System.String`

Любые дополнительные данные, связанные с событием

Заметки

Начиная с версии Docsvision 4.5, появился новый метод для записи в журнал приложений `LogMessageEx`.

Примеры

Например, для записи собственного сообщения:

```
session.LogManager.LogMessage(EventType.Information, 0, "Моя карточка", data.ID,
Guid.Empty, Guid.Empty, "Добавлен исполнитель");
```

Вот как будет выглядеть такое сообщение в стандартном диалоге просмотра журнала:

```
Тип сообщения: Информация.
Операция: Не определено.
Код: 0.
Имя ресурса: Моя карточка.
Идентификатор ресурса: {1EA06712-EBB1-4E33-B2E2-6A4AD05991E5}.
Идентификатор типа ресурса: <Неизвестный идентификатор>.
Идентификатор родительского объекта: <Неизвестный идентификатор>.
Новый идентификатор: <Неизвестный идентификатор>.
Данные сообщения: Добавлен исполнитель.
```

LogManager.LogMessageEx — метод (EventType, Guid, String, Guid, Guid, Guid, String)

Выполняет запись сообщения в журнал с указанием идентификатора соответствующей операции из описания библиотеки карточек.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void LogMessageEx(EventType eventType, Guid eventOperation, string resourceName, Guid resourceId, Guid typeId, Guid parentId, string eventData)
```

Параметры

eventType

Тип: EventType

Тип сообщения

eventOperation

Тип: System.Guid

Идентификатор соответствующей операции из описания библиотеки карточек

resourceName

Тип: System.String

Имя объекта, с которым произведена операция (формируется в зависимости от контекста — например, при операциях с карточкой это может быть её описание)

resourceId

Тип: System.Guid

Идентификатор объекта, с которым производится операция (ID карточки, строки, файла и т.д.)

typeId

Тип: System.Guid

Тип объекта (например, для карточки — CardTypeID)

parentId

Тип: System.Guid

Идентификатор родительского объекта (например, для операции со строкой — родительским объектом будет секция)

eventData

Тип: `System.String`

Любые дополнительные данные, связанные с событием

LogManager.SetLogStrategy — метод (LogStrategy, ClearLogStrategy, Int32, Int32, Int32)

Установка параметров стратегии ведения журнала событий.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract void SetLogStrategy(LogStrategy logStrategy, ClearLogStrategy
clearStrategy, int clearLogMaxCount, int clearLogCutCount, int clearLogCutDays);
```

Параметры

logStrategy

Тип: `LogStrategy`

Стратегия записи сообщений в журнал.

clearStrategy

Тип: `ClearLogStrategy`

Стратегия очистки журнала.

clearLogMaxCount

Тип: `System.Int32`

Максимальное количество записей в журнале

clearLogCutCount

Тип: `System.Int32`

Количество сохраняемых после чистки записей

clearLogCutDays

Тип: System.Int32

Максимальное количество дней хранения записей в журнале

LogSearchQuery — класс

Представляет фильтр данных для журнала.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public class LogSearchQuery
```

Конструкторы

Имя	Описание
LogSearchQuery	Инициализирует новый экземпляр класса LogSearchQuery.
LogSearchQuery(String)	Инициализирует новый экземпляр класса LogSearchQuery с предопределённым значением автора операции.
LogSearchQuery(String, Guid)	Инициализирует новый экземпляр класса LogSearchQuery с предопределённым значением автора операции и идентификатора объекта.
LogSearchQuery(String, Guid, EventType)	Инициализирует новый экземпляр класса LogSearchQuery с предопределённым типом сообщения.

Свойства

Имя	Описание
AccountName	Задаёт или возвращает учетную запись пользователя, выполнявшего операцию.

Имя	Описание
`CreatedAfter`	Задаёт или возвращает нижнюю границу даты совершения операции.
`CreatedBefore`	Задаёт или возвращает верхнюю границу даты совершения операции.
`EventType`	Задаёт или возвращает тип сообщения.
`LogType`	Задаёт или возвращает тип системный уровень сообщения.
`OperationAfter`	Задаёт или возвращает нижнюю идентификатора кода операции.
`OperationBefore`	Задаёт или возвращает верхнюю идентификатора кода операции.
`Operations`	Возвращает коллекцию кодов операций.
`Operations`	Возвращает коллекцию идентификаторов операций.
`ParentId`	Задаёт или возвращает идентификатор родительского объекта.
`ResourceId`	Задаёт или возвращает идентификатор объекта.
`TypeId`	Задаёт или возвращает идентификатор типа объекта.

Методы

Имя	Описание
`GetSectionQueryXml(UserSession)`	Создаёт из установленных ограничений строку поиска в формате XML.

LogStrategy — перечисление

Определяет стратегию ведения журнала.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum LogStrategy
```

Члены

Имя члена	Описание
`NoLogging`	Не вести журнал.
`Filtered`	Журналировать только определённые сообщения.
`Everything`	Журналировать всё.

QueryParameter — класс

Осуществляет хранение параметра запроса. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class QueryParameter
```

Свойства

Имя	Описание
`Alias`	Возвращает псевдоним параметра.
`ContextWord`	Задаёт или возвращает контекстное слово.
`Flags`	Задаёт или возвращает дополнительные флаги.
`Name`	Задаёт или возвращает название параметра.
`Session`	Возвращает текущую сессию пользователя.
`Type`	Задаёт или возвращает тип.

Имя	Описание
`Value`	Задаёт или возвращает значение параметра.

Методы

Имя	Описание
`Resolve`	Получение актуального значения параметра для текущего контекста.

QueryParameterCollection — класс

Содержит методы для работы с коллекцией `QueryParameter`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class QueryParameterCollection : IEnumerable<QueryParameter>, IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает размер коллекции.
`Item`	Возвращает элемент коллекции с указанным индексом.
`Item`	Возвращает элемент коллекции с заданным идентификатором.

Методы

Имя	Описание
`AddNew(String)`	Добавляет в коллекцию элемент с предустановленным псевдонимом и возвращает ссылку.
`Clear`	Удаляет все элементы коллекции.

Имя	Описание
`Contains(String)`	Определяет вхождение в коллекцию элемента с заданным псевдонимом.
`GetEnumerator`	Возвращает перечислитель.
`Remove(Int32)`	Удаляет элемент коллекции с указанным индексом.
`Remove(String)`	Удаляет элемент коллекции с заданным псевдонимом.

RecordStatus — перечисление

Состояние задействованности объекта.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum RecordStatus
```

Члены

Имя члена	Описание
`Operative`	Карточка может быть удалена.
`Record`	Карточка не может быть удалена, установлен запрет на модификацию.
`Withdrawn`	Карточка может быть удалена, установлен запрет на модификацию.

Заметки

Определяет возможности производить изменение объекта.

Report — класс

Представляет данные для хранимой процедуры. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class Report
```

Свойства

Имя	Описание
Alias	Возвращает псевдоним процедуры.
Id	Возвращает идентификатор процедуры.
Name	Возвращает локализованное название процедуры.
Parameters	Содержит список параметров процедуры.
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
AccessCheck(Int32)	Проверяет наличие заданных прав.
GetAccessControl	Получает описатель прав на хранимую процедуру.
GetAccessControl(AccessControlSections)	Возвращает описатель прав на хранимую процедуру для заданного раздела дескриптора безопасности.
GetData	Запускает процедуру на исполнение.
SetAccessControl(ReportSecurity)	Задаёт описатель прав для объекта.

Report.Alias — свойство

Возвращает псевдоним процедуры.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Alias { get; }
```

Значение свойства

Тип: System.String

Псевдоним

Заметки

Псевдоним используется при обращении к процедуре из кода.

Report.Id — свойство

Возвращает идентификатор процедуры.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid Id { get; }
```

Значение свойства

Тип: System.Guid

Идентификатор

Report.Name — свойство

Возвращает локализованное название процедуры.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract string Name { get; }
```

Значение свойства

Тип: System.String

Название процедуры

Заметки

Используется текущие региональные настройки сервера.

Report.Parameters — свойство

Содержит список параметров процедуры.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ReportParameterCollection Parameters { get; }
```

Значение свойства

Тип: ReportParameterCollection

Коллекция параметров

Заметки

Параметры задаются на этапе описания хранимой процедуры в специализированной карточке.

Report.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Сессия пользователя

Report.AccessCheck — метод (Int32)

Проверяет наличие заданных прав.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public bool AccessCheck(int desiredAccess);
```

Параметры

desiredAccess

Тип: System.Int32

Права доступа

Возвращаемое значение

Тип: System.Boolean

true — права доступны, иначе — false

Report.GetAccessControl — метод

Получает описатель прав на хранимую процедуру.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ReportSecurity GetAccessControl();
```

Возвращаемое значение

Тип: ReportSecurity

Описатель прав

Report.GetAccessControl — метод (AccessControlSections)

Возвращает описатель прав на хранимую процедуру для заданного раздела дескриптора безопасности

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public ReportSecurity GetAccessControl(AccessControlSections includeSections)
```

Параметры

includeSections

Тип: System.Security.AccessControl.AccessControlSections

Раздел дескриптора безопасности

Возвращаемое значение

Тип: ReportSecurity

Описатель прав

Report.GetData — метод

Запускает процедуру на исполнение.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract InfoRowCollection GetData()
```

Возвращаемое значение

Тип: InfoRowCollection

Результат запроса

Примеры

```
const string REPORT_ID = "75AD4670-0DF3-43CD-A6FE-C34BC6E36E31";

Report report = session.ReportManager.Reports[new Guid(REPORT_ID)]; ①

report.Parameters["Param1"].Value = "Test";
report.Parameters["Param2"].Value = 0; ②
```

```
InfoRowCollection results = report.GetData(); ③
```

① Получение объекта процедуры по идентификатору.

② Задание значений параметров.

③ Вызов процедуры и получение данных.

Report.SetAccessControl — метод (ReportSecurity)

Задаёт описатель прав для объекта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void SetAccessControl(ReportSecurity reportSecurity)
```

Параметры

reportSecurity

Тип: ReportSecurity

Описатель прав

ReportCollection — класс

Содержит методы для работы с коллекцией Report.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class ReportCollection : IEnumerable<Report>, IEnumerable
```

Свойства

Имя	Описание
Count	Возвращает размер коллекции.

Имя	Описание
`Item`	Возвращает элемент по идентификатору.
`Item`	Возвращает элемент по индексу.
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`Contains(Guid)`	Определяет вхождение в коллекцию элемента с заданным идентификатором.
`GetEnumerator`	Возвращает перечислитель.

ReportManager — класс

Представляет объектную модель *менеджера хранимых процедур*.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class ReportManager
```

Свойства

Имя	Описание
`Reports`	Возвращает коллекцию доступных хранимых процедур.
`Session`	Возвращает текущую сессию пользователя.

ReportManager.Reports — свойство

Возвращает коллекцию доступных хранимых процедур.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ReportCollection Reports { get; }
```

Значение свойства

Тип: ReportCollection

Коллекция хранимых процедур

ReportManager.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Текущая сессия пользователя

ReportParameter — класс

Осуществляет хранение параметров хранимых процедур. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class ReportParameter
```

Свойства

Имя	Описание
`Name`	Возвращает название параметра.
`Session`	Возвращает текущую сессию пользователя.
`Value`	Задаёт или возвращает значение параметра.
`ValueType`	Возвращает тип параметра.

ReportParameterCollection — класс

Содержит методы для работы с коллекцией `ReportParameter`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class ReportParameterCollection : IEnumerable<ReportParameter>,
IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов коллекции.
`Item`	Возвращает элемент коллекции с заданным индексом.
`Item`	Возвращает элемент коллекции с указанным названием.
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`Contains(String)`	Определяет вхождение в коллекцию элемента с заданным названием.
`GetEnumerator`	Возвращает перечислитель.

RowData — класс

Представляет строку данных карточки. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class RowData : IUpdatable, ILockable, IXmlExportable
```

Свойства

Имя	Описание
`AllChildRows`	Возвращает все дочерние строки ниже по дереву иерархии.
`Card`	Возвращает ссылку на карточку, которой принадлежит строка.
`ChildRows`	Возвращает дочерние строки в дереве.
`ChildSections`	Возвращает подчинённые подсекции.
`DisplayString`	Возвращает значение объекта в виде строки.
`HasChildRows`	Определяет наличия дочерних строк.
`Id`	Возвращает идентификатор строки.
`InUpdate`	Признак включения для объекта режима отложенных изменения.
`Item`	Задаёт или возвращает элемент с заданным псевдонимом.
`LockOwner`	Возвращает учетную запись владельца блокировки.

Имя	Описание
LockStatus	Возвращает статус блокировки.
ParentRow	Родительская строка в иерархии строк секции.
Section	Возвращает ссылку на секцию, которой принадлежит строка.
Session	Возвращает текущую сессию пользователя.
Status	Текущее состояние объекта в режиме отложенных изменений.
SubSection	Возвращает ссылку на подсекцию, которой принадлежит строка.

Методы

Имя	Описание
AccessCheck(System.Int32)	Проверяет наличие заданного доступа.
BeginUpdate	Включает режим отложенных изменений.
CancelUpdate	Отменяет несохранённые изменения и выключает режим отложенных изменений.
Copy(Guid, Guid)	Создаёт копию строки в указанном месте дерева иерархии.
EndUpdate	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
ForceUnlock	Принудительно снимает блокировку.
GetAccessControl	Возвращает описатель прав.
GetAccessControl(AccessControlSections)	Возвращает описатель прав для указанного раздела дескриптора безопасности.

Имя	Описание
`GetBoolean(String)`	Возвращает значение поля с указанным псевдонимом в виде объекта типа `Boolean`.
`GetByte(String)`	Возвращает значение поля с указанным псевдонимом в виде объекта типа `Byte`.
`GetBytes(String)`	Возвращает значение поля с указанным псевдонимом в виде массива `Byte`.
`GetChar(String)`	Возвращает значение поля с указанным псевдонимом в виде объекта типа `Char`.
`GetDateTime(String)`	Возвращает значение поля с указанным псевдонимом в виде объекта типа `DateTime`.
`GetDecimal(String)`	Возвращает значение поля с указанным псевдонимом в виде объекта типа `Decimal`.
`GetDouble(String)`	Возвращает значение поля с указанным псевдонимом в виде объекта типа `Double`.
`GetGuid(String)`	Возвращает значение поля с указанным псевдонимом в виде объекта типа `Guid`.
`GetHierarchy`	Получение иерархии строки (идентификаторы всех родительских строк вплоть до самого верхнего уровня).
`GetInt16(String)`	Возвращает значение поля с указанным псевдонимом в виде объекта типа `Int16`.
`GetInt32(String)`	Возвращает значение поля с указанным псевдонимом в виде объекта типа `Int32`.

Имя	Описание
GetInt64(String)	Возвращает значение поля с указанным псевдонимом в виде объекта типа Int64.
GetObject(String)	Возвращает нетипизированное значение поля.
GetSByte(String)	Возвращает значение поля с указанным псевдонимом в виде объекта типа SByte.
GetSingle(String)	Возвращает значение поля с указанным псевдонимом в виде объекта типа Single.
GetString(String)	Возвращает значение поля в виде строки.
GetUInt16(String)	Возвращает значение поля с указанным псевдонимом в виде объекта типа UInt16.
GetUInt32(String)	Возвращает значение поля с указанным псевдонимом в виде объекта типа UInt32.
GetUInt64(String)	Возвращает значение поля с указанным псевдонимом в виде объекта типа UInt64.
GetValueType(String)	Возвращает тип поля с заданным псевдонимом.
MarkCardForDeletion(Guid)	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
Move(Guid, Guid)	Перемещает строку в иерархии строк секции.
PlaceLock	Устанавливает временную блокировку.
PlaceLock(Boolean)	Позволяет установить постоянную блокировку.

Имя	Описание
`Refresh`	Позволяет обновить значение строки.
`Refresh(Boolean)`	Обновляет значение связанных со строкой полей.
`RemoveLock`	Снимает свою блокировку.
`SaveXml(Stream)`	Сохраняет представление данных объекта в формате XML, в поток.
`SaveXml(Stream, ExportFlags)`	Сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта.
`SaveXml(Stream, ExportFlags, ExportCardInspector)`	Сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта и заданной логикой экспорта.
`SetAccessControl(CardDataSecurity)`	Устанавливает права на строку.
`SetBoolean(String, Boolean)`	Присваивает полю с заданным псевдонимом указанное значение типа `Boolean`.
`SetByte(String, Byte)`	Присваивает полю с заданным псевдонимом указанное значение типа `Byte`.
`SetBytes(String, Byte[])`	Присваивает полю с заданным псевдонимом указанное значение массива `Byte`.
`SetChar(String, Char)`	Присваивает полю с заданным псевдонимом указанное значение типа `Char`.
`SetDateTime(String, DateTime)`	Присваивает полю с заданным псевдонимом указанное значение типа `DateTime`.
`SetDecimal(String, Decimal)`	Присваивает полю с заданным псевдонимом указанное значение типа `Decimal`.

Имя	Описание
SetDouble(String, Double)	Присваивает полю с заданным псевдонимом указанное значение типа Double.
SetGuid(String, Guid)	Присваивает полю с заданным псевдонимом указанное значение типа Guid.
SetInt16(String, Short)	Присваивает полю с заданным псевдонимом указанное значение типа Short.
SetInt32(String, Int32)	Присваивает полю с заданным псевдонимом указанное значение типа Int32.

Примеры

```
CardData card = session.CardManager.GetCardData(new System.Guid(
"идентификатор_карточки")); ①

SectionData section = card.Sections[card.Type.Sections["MainInfo"].Id]; ②

RowData row = section.FirstRow; ③

if (row.GetValueType("Number") = FieldType.Int) ④
{
    row.SetInt32("Number", 10);
}
```

① Получение с сервера данных карточки с известным идентификатором.

② Получение данных секции с именем MainInfo.

③ Получение первой строки секции (если строки нет — она будет создана).

④ Запись значения в поле Number.

RowDataCollection — класс

Содержит методы для работы с коллекцией RowData.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class RowDataCollection : IEnumerable<RowData>, IEnumerable,
IXmlExportable
```

Свойства

Имя	Описание
Card	Возвращает ссылку на родительскую карточку.
Count	Возвращает размер коллекции.
Item	Возвращает элемент коллекции с заданным идентификатором.
Item	Возвращает элемент коллекции с заданным индексом
Section	Возвращает ссылку на родительскую секцию.
Session	Возвращает текущую сессию

Методы

Имя	Описание
AddNew	Добавляет в коллекцию строку и возвращает ссылку.
AddNew(Guid)	Добавляет в коллекцию строку с предустановленным идентификатором и возвращает ссылку.
Clear	Удаляет все элементы коллекции.
Contains(Guid)	Определяет вхождение в коллекцию объекта с заданным идентификатором.
Filter(String)	Возвращает отфильтрованные элементы коллекции.
Find(String, Object)	Осуществляет поиск строки с заданным значением указанного поля.
GetEnumerator	Возвращает перечислитель.

Имя	Описание
`Refresh`	Обновляет данные.
`Refresh(Boolean, Boolean)`	Обновляет данные только связанных полей.
`Remove(Guid)`	Удаляет элемент коллекции с заданным идентификатором.
`Remove(Int32)`	Удаляет элемент коллекции с указанным индексом.
`SaveXml(Stream)`	Сохраняет данные строки в поток.
`SaveXml(Stream, ExportFlags)`	Сохраняет данные строки в поток, с указанными параметрами.
`SaveXml(Stream, ExportFlags, ExportCardInspector)`	Сохраняет данные строки в поток, с заданным инспектором.
`Sort(String, Boolean)`	Возвращает сортированный экземпляр коллекции.

RowField — класс

Описывает поле строки карточки.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class RowField
```

Свойства

Имя	Описание
`Alias`	Возвращает псевдоним поля.
`DataType`	Возвращает тип данных поля.
`IsReadOnly`	Поле доступно только на чтение.
`Type`	Возвращает ссылку на описание поля.

Заметки

В отличие от описаний полей в схеме карточки, доступных через объекты метаданных (`DocsVision.Platform.ObjectManager.Metadata.Field`) — `RowField` содержит описание реального поля секции, в т.ч. если поле является связанным или системным (например, поле дайджеста — `Description`).

RowFieldCollection — класс

Содержит методы для работы с коллекцией `RowField`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class RowFieldCollection : IEnumerable<RowField>, IEnumerable
```

Свойства

Имя	Описание
`Count`	Возвращает количество элементов коллекции.
`Item`	Возвращает элемент коллекции с заданным индексом.
`Item`	Возвращает элемент коллекции с указанным псевдонимом.

Методы

Имя	Описание
`Contains(String)`	Определяет вхождение в коллекции поля с указанным псевдонимом.
`GetEnumerator`	Возвращает перечислитель.

Заметки

В отличие от описаний полей в схеме карточки, доступных через объекты метаданных (`CardType.Sections.Fields`) — коллекция `RowFieldCollection` содержит описания реальных полей секции, включая связанные поля других карточек и системные поля (например, поле дайджеста — `Description`).

SearchAggregationItemInfo — класс

Список объектов.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class SearchAggregationItemInfo
```

Свойства

Имя	Описание
Alias	Алиас агрегата.
Name	Локализованное имя агрегата. Если значение не задано или не найдено в настройках агрегата, будет использоваться английская локаль. Если английская локаль не найдена, будет использоваться первая найденная. Если никакие настройки не заданы, в качестве значения будет использоваться алиас агрегата.
Values	Список значений агрегата вида "строка — количество".
IsMain	Является ли агрегат "главным".

SearchAggregationItemsRequest — класс

Объект запроса.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class SearchAggregationItemsRequest
```

Свойства

Имя	Описание
`ResultType`	Тип получаемого результата: `All` — вернуть список агрегатов и сохранить результаты фильтрации в папку; `Aggregation` — вернуть только список агрегатов; `Cards` — только сохранить результаты фильтрации в папку.
`FolderId`	Идентификатор папки-хранилища для сохранения результатов фильтрации.
`QueryXml	Xml поиска. Должно быть заполнено как минимум поле `QueryString` полнотекстового поиска. Также могут быть использованы любые другие части — как и сам атрибутивный поиск, так и ограничения по результатам (ограничение на папку например). `QueryXml` можно не заполнять, но в таком случае не будет заполнена папка результатами поиска, но список агрегатов вернётся.
`QueryId`	Идентификатор поиска. Альтернатива `QueryXml`. Если он не `Guid.Empty`, будет использоваться именно сохранённый запрос.
`Parameters`	Список параметров для запроса. Может использоваться как с `QueryId`, так и с `QueryXml`. В случае `QueryXml` параметры можно включить непосредственно в запрос.

Имя	Описание
`LocaleId`	Номер локализации в которой необходимо вернуть имя агрегации. Если значение не задано или не найдено в настройках агрегата, то будет использоваться английская локаль. Если английская локаль не найдена, то будет использоваться первая найденная. Если настройки не заданы вообще, то в качестве значения будет использоваться алиас агрегата.
`Filters`	Значения для дополнительной фильтрации результатов. Ключ — алиас агрегата, значение — текст для фильтрации. Текст для фильтрации необходимо передавать как сериализованный в JSON список (даже если элемент всего один). См. пример. Фильтрация работает по полному совпадению переданного текста со значениями полей, указанных в соответствующем агрегате. При наличии нескольких полей результаты соединяются по условию "ИЛИ".

SectionData — класс

Представляет секцию карточки. Этот класс является **абстрактным**.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class SectionData : IUpdatable, IXmlExportable
```

Свойства

Имя	Описание
Card	Карточка, которой принадлежит секция.
FetchMode	Возвращает или задаёт режим загрузки данных.
Fields	Возвращает поля строки секции.
FirstRow	Возвращает первую строку секции.
Id	Идентификатор типа секции.
InUpdate	Возвращает признак включения для секции режима отложенных изменения.
Rows	Возвращает коллекцию всех строк верхнего уровня.
Session	Возвращает текущую открытую сессию пользователя.
Type	Предоставляет доступ к метаданным секции карточки.

Методы

Имя	Описание
BeginUpdate	Включает режим отложенных изменений.
CancelUpdate	Отменяет несохранённые изменения и выключает режим отложенных изменений.
CheckRow(Guid, Boolean)	Проверяет наличие в секции строки с заданным идентификатором.
CreateRow	Создаёт новую строку в секции.
CreateRow(Guid)	Создаёт в секции новую строку с заданным идентификатором.

Имя	Описание
`CreateRow(Guid, Guid, Guid)`	Создаёт в секции новую строку с заданным идентификатором, а также с конкретными значениями родительской строки секции и родительской строки дерева.
`DeleteRow(Guid)`	Удаляет из секции строку с заданным идентификатором.
`EndUpdate`	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
`FindRow(String)`	Осуществляет поиск строки в секции с использованием XPath-запроса.
`FindRows(String)`	Осуществляет поиск строк в секции карточки (справочника).
`GetAllRows`	Возвращает все строки из текущей секции.
`GetAllRows(Guid)`	Получение всех строк в секции для конкретной строки родительской секции.
`GetAllRows(Guid, Boolean)`	Получение всех строк в секции для конкретной строки родительской секции, без учета уровня иерархии.
`GetRow(Guid)`	Получение данных строки с заданным идентификатором.
`MarkCardForDeletion(Guid)`	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
`OpenSectionReader(FetchMode, Guid, Guid)`	Открывает серверный курсор для чтения данных секции.
`PurgeCache`	Очистка кэша.
`Refresh`	Принудительное обновление данных с сервера.

Имя	Описание
`Refresh(Boolean)`	Принудительное обновление данных с сервера, независимо от наличия или отсутствия изменений.
`RowExists(Guid)`	Проверка существования в секции строки с заданным идентификатором.
`SaveXml(Stream)`	Сохраняет представление данных объекта в формате XML, в поток.
`SaveXml(Stream, ExportFlags)`	Сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта.
`SaveXml(Stream, ExportFlags, ExportCardInspector)`	Сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта и заданной логикой экспорта.
`UpdateNow`	Позволяет отправить накопленные изменения на сервер.

Примеры

```
CardData card = session.CardManager.GetCardData(new Guid("идентификатор_карточки")); ①

SectionData section = card.Sections[card.Type.Sections["MainInfo"].Id]; ②

RowData row = section.Rows[0]; ③
```

① Получение с сервера данных карточки с известным идентификатором.

② Получение данных секции с именем `MainInfo`.

③ Получение первой строки.

SectionData.Card — свойство

Карточка, которой принадлежит секция.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract CardData Card { get; }
```

Значение свойства

Тип: `CardData`

Карточка

SectionData.FetchMode — свойство

Возвращает или задаёт режим загрузки данных.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract FetchMode FetchMode {get; set;}
```

Значение свойства

Тип: `FetchMode`

Режим загрузки

Заметки

Режим построчной загрузки (`FetchMode.Row`) в данный момент не реализован. Однако получить данные для отдельных строк все же можно — для этого у объекта `SectionData` есть набор методов, работающих с одиночной строкой. Их использование позволяет не загружать на клиента ненужную информацию, что, конечно же, предпочтительнее с точки зрения производительности.

SectionData.Fields — свойство

Возвращает поля строки секции.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract RowFieldCollection Fields { get; }
```

Значение свойства

Тип: RowFieldCollection

Коллекция полей

SectionData.FirstRow — свойство

Возвращает первую строку секции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract RowData FirstRow { get; }
```

Значение свойства

Тип: RowData

Строка секции

Заметки

Если такой строки нет — она будет создана. Самое частое использование этого свойства — получение строки из секции типа структура (например, общее описание карточки).

SectionData.Id — свойство

Идентификатор типа секции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Guid Id { get; }
```

Значение свойства

Тип: Guid

Идентификатор типа

SectionData.InUpdate — свойство

Возвращает признак включения для секции режима отложенных изменения.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool InUpdate { get; }
```

Значение свойства

Тип: Boolean

false — режим выключен; true — режим включён

SectionData.Rows — свойство

Возвращает коллекцию всех строк верхнего уровня.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract RowDataCollection Rows { get; }
```

Значение свойства

Тип: RowDataCollection

Коллекция строк

SectionData.Session — свойство

Возвращает текущую открытую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Сессия

SectionData.Type — свойство

Предоставляет доступ к метаданным секции карточки.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract CardSection Type { get; }
```

Значение свойства

Тип: CardSection

Метаданные

SectionData.BeginUpdate — метод

Включает режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
void BeginUpdate()
```

SectionData.CancelUpdate — метод

Отменяет несохранённые изменения и выключает режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void CancelUpdate()
```

SectionData.CheckRow — метод (Guid, Boolean)

Проверяет наличие в секции строки с заданным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool CheckRow(Guid rowId, bool checkCache)
```

Параметры

rowId

Тип: Guid

Идентификатор строки в секции

checkCache

Тип: Boolean

true — поиск должен вестись в кэше, иначе — false

Возвращаемое значение

Тип: Boolean

true — строка существует, иначе — false

SectionData.CreateRow — метод

Создаёт новую строку в секции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public RowData CreateRow()
```

Возвращаемое значение

Тип: RowData

Новая строка

SectionData.CreateRow — метод (Guid)

Создаёт в секции новую строку с заданным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public RowData CreateRow(Guid rowId)
```

Параметры

rowId

Тип: Guid

Идентификатор присваиваемый строке

Возвращаемое значение

Тип: RowData

Новая строка

SectionData.CreateRow — метод (Guid, Guid, Guid)

Создаёт в секции новую строку с заданным идентификатором, а также с конкретными значениями родительской строки секции и родительской строки дерева.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract RowData CreateRow(Guid parentTreeRowId, Guid parentRowId, Guid rowId)
```

Параметры

parentTreeRowId

Тип: Guid

Идентификатор родительской строки дерева

parentRowId

Тип: Guid

Идентификатор родительской строки

rowId

Тип: Guid

Идентификатор присваиваемый строке

Возвращаемое значение

Тип: RowData

Новая строка

SectionData.DeleteRow- метод (Guid)

Удаляет из секции строку с заданным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void DeleteRow(Guid rowId)
```

Параметры

rowId

Тип: Guid

Идентификатор удаляемой строки

SectionData.EndUpdate — метод

Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void EndUpdate()
```

SectionData.FindRow — метод (String)

Осуществляет поиск строки в секции с использованием XPath-запроса.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract RowData FindRow(string filter)
```

Параметры

filter

Тип: String

XPath-запрос

Возвращаемое значение

Тип: RowData

Искомая строка секции

SectionData.FindRows — метод (String)

Осуществляет поиск строк в секции карточки (справочника).

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract RowDataCollection FindRows(string queryXml)
```

Параметры

queryXml

Тип: String

Поисковый запрос

Возвращаемое значение

Тип: RowDataCollection

Коллекция найденных строк

Примеры

Приведенный далее пример демонстрирует использование серверного поиска при получении сотрудников определённого подразделения

```
①

SectionQuery sectionQuery = userSession.CreateSectionQuery(); ②

Guid departmentID = new Guid(" 00000000-0000-0000-0000-000000000000"); ③

sectionQuery.ConditionGroup.Conditions.AddNew("ParentRowID", FieldType.UniqueId,
ConditionOperation.Equals, departmentID); ④

CardData staffData = userSession.CardManager.GetDictionaryData(RefStaff.ID);
SectionData employeesSection = staffData.Sections[RefStaff.Employees.ID]; ⑤

RowDataCollection employees = employeesSection.FindRows(sectionQuery.GetXml()); ⑥
```

① Инициализация контекста объектов.

② Поиск по секции.

③ Подразделение, из которого осуществляется выборка сотрудников.

④ Добавление условия поиска сотрудников в подразделении departmentId.

⑤ Получение Справочника сотрудников и секции "Сотрудники".

⑥ Выполнение поиска.

SectionData.GetAllRows — метод

Возвращает все строки из текущей секции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public RowDataCollection GetAllRows()
```

Возвращаемое значение

Тип: RowDataCollection

Коллекция найденных строк

SectionData.GetAllRows — метод (Guid)

Получение всех строк в секции для конкретной строки родительской секции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public RowDataCollection GetAllRows(Guid? parentRowId)
```

Параметры

parentRowId

Тип: Guid

Идентификатор строки в родительской секции. Если null, то вернет все строки

Возвращаемое значение

Тип: RowDataCollection

Коллекция искомых строк

SectionData.GetAllRows — метод (Guid, Boolean)

Получение всех строк в секции для конкретной строки родительской секции, без

учета уровня иерархии.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract RowDataCollection GetAllRows(Guid? parentRowId, bool traverseHierarchy)
```

Параметры

parentRowId

Тип: Guid

Идентификатор строки в родительской секции. Если null, то вернет все строки

traverseHierarchy

Тип: Boolean

true — игнорировать значение свойства строки ParentTreeRowID, иначе — false

Возвращаемое значение

Тип: RowDataCollection

Коллекция искомых строк

SectionData.GetRow — метод (Guid)

Получение данных строки с заданным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract RowData GetRow(Guid rowId)
```

Параметры

rowId

Тип: Guid

Идентификатор строки

Возвращаемое значение

Тип: RowData

Строка

SectionData.MarkCardForDeletion — метод (Guid)

Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void MarkCardForDeletion(Guid cardId)
```

Параметры

cardId

Тип: Guid

Идентификатор удаляемой карточки

SectionData.OpenSectionReader — метод (FetchMode, Guid, Guid)

Открывает серверный курсор для чтения данных секции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SectionDataReader OpenSectionReader(FetchMode fetchMode, Guid
parentRowId, Guid parentTreeRowId)
```

Параметры

fetchMode

Тип: FetchMode

Режим загрузки данных с сервера

parentRowId

Тип: `Guid`

Идентификатор родительской строки

parentTreeRowId

Тип: `Guid`

Идентификатор родительской строки дерева

Возвращаемое значение

Тип: `SectionDataReader`

Объект для чтения данных

Заметки

Данный метод рекомендуется использовать при чтении данных секции с очень большим количеством строк. Основное отличие данного метода заключается в построчном чтении данных, без сохранения их в кэше.

SectionData.PurgeCache — метод

Очистка кэша.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract void PurgeCache()
```

Заметки

Приведёт к принудительному перечитыванию данных с сервера при следующем обращении.

SectionData.Refresh — метод

Принудительное обновление данных карточки с сервера.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public void Refresh()
```

Заметки

Данные будут загружены только при наличии в них изменений с момента предыдущего обновления.

SectionData.Refresh — метод (Boolean)

Принудительное обновление данных с сервера, независимо от наличия или отсутствия изменений

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Refresh(bool ignoreTimestamp)
```

Параметры

ignoreTimestamp

Тип: Boolean

true — игнорировать отсутствие изменений; false — учитывать отсутствие изменений

Заметки

При установке ignoreTimestamp в значение true, данные объектов будут получены с сервера в любом случае.

SectionData.RowExists — метод (Guid)

Проверка существования в секции строки с заданным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool RowExists(Guid rowId)
```

Параметры

rowId

Тип: Guid

Идентификатор строки

Возвращаемое значение

Тип: Boolean

true — строка существует, иначе — false

SectionData.SaveXml — метод (Stream)

Сохраняет представление данных объекта в формате XML, в поток.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void SaveXml(Stream stream)
```

Параметры

stream

Тип: Stream

Поток для данных

SectionData.SaveXml — метод (Stream, ExportFlags)

Сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void SaveXml(Stream stream, ExportFlags exportFlags)
```

Параметры

stream

Тип: Stream

Поток для данных

exportFlags

Тип: ExportFlags

Флаги экспорта

SectionData.SaveXml — метод (Stream, ExportFlags, ExportCardInspector)

Сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта и заданной логикой экспорта.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void SaveXml(Stream stream, ExportFlags exportFlags, ExportCardInspector
inspector)
```

Параметры

stream

Тип: Stream

Поток для данных

exportFlags

Тип: ExportFlags

Флаги экспорта

inspector

Тип: ExportCardInspector

Логика экспорта

SectionData.UpdateNow — метод

Позволяет отправить накопленные изменения на сервер.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract void UpdateNow()
```

Заметки

Отправляет накопленные изменения на сервер, на клиенте они очищаются из кэша, режим продолжает действовать. Все изменения после этого момента также будут накапливаться.

SectionDataCollection — класс

Содержит методы для работы с коллекцией `SectionData`.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class SectionDataCollection : IEnumerable<SectionData>, IEnumerable,
IXmlExportable
```

Свойства

Имя	Описание
`Card`	Возвращает ссылку на родительскую карточку.
`Count`	Возвращает размер коллекции.
`Item`	Возвращает элемент коллекции с заданным идентификатором.
`Item`	Возвращает элемент коллекции с указанным индексом.

Имя	Описание
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`Contains(Guid)`	Определяет вхождение в коллекцию элемента с заданным идентификатором.
`GetEnumerator`	Возвращает перечислитель.
`SaveXml(Stream)`	Сохраняет данные секции в поток.
`SaveXml(Stream, ExportFlags)`	Сохраняет данные секции в поток, с учетом формат экспорта.
`SaveXml(Stream, ExportFlags, ExportCardInspector)`	Сохраняет данные секции в поток, с применением инспектора экспорта.

SectionDataReader — класс

Серверный курсор для чтения данных секции.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract class SectionDataReader : IDisposable
```

Свойства

Имя	Описание
`Section`	Возвращает данные секции.
`Session`	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
`Close`	Закрывает серверный курсор.
`Dispose(Boolean)`	Очищает неуправляемые ресурсы.
`GetValue(String)`	Возвращает значение поля.
`GetValueType(String)`	Возвращает значение тип поля.
`Read`	Читает следующую строку данных.

SessionLoginFlags — перечисление

Описывает флаги открытия новой сессии.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public enum SessionLoginFlags
```

Члены

Имя члена	Описание
`None`	Не установлен.
`IncludeCardLibraries`	Включить информацию по библиотекам карточек.
`IncludeCardTypes`	Включить информацию по типам карточек.
`IncludeDictionaries`	Включить справочники.
`IncludeInstallers`	Включить установщики.
`IncludeSecuritySchema`	Включить схему безопасности.
`IncludeUserProfile`	Включить профиль пользователя.

SessionManager — класс

Представляет объектную модель **менеджера сессий**. Содержит методы управления сессиями пользователей. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class SessionManager
```

Методы

Имя	Описание
CloseSession(Guid)	Принудительное закрытие сессии с указанным идентификатором.
Connect(string)	Устанавливает соединение с сервером Docsvision.
Connect(string, string)	Устанавливает соединение с сервером Docsvision.
Connect(string, string, string, string)	Устанавливает соединение с сервером Docsvision.
ConnectProxy(object)	Установка соединения с сервером используя указанное соединение.
CreateInstance	Создание экземпляра менеджера сессий.
CreateInstance(string)	Создание экземпляра менеджера сессий с указанием строки соединения.
CreateSession	Создаёт пользовательскую сессию.
CreateSession(SessionLoginFlags)	Создаёт пользовательскую сессию..
GetLocaleId	Получение идентификатора языка для локализации сообщений.
GetOpenSessions	Получение списка всех открытых сессий пользователя.
SetLocaleId(int)	Установка идентификатора языка для текущей сессии.

Заметки

Для вызова некоторых методов менеджера сессий требуются административные привилегии.

Примеры

В примере организуется соединение с сервером localhost с указанием названия подключаемой базы данных — "Docsvision".

```
string connect =
"ConnectAddress=http://localhost/DocsVision/StorageServer/StorageServerService.asmx;BaseName=Docsvision";
SessionManager manager = SessionManager.CreateInstance();
manager.Connect(connect);
```

SessionManager.CloseSession — метод (Guid)

Принудительное закрытие сессии с указанным идентификатором.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void CloseSession(Guid sessionId)
```

Параметры

sessionId

Тип: Guid

Идентификатор закрываемой сессии.

Заметки

Для использования этого метода требуется наличие у владельца текущей сессии административных привилегий.

SessionManager.Connect — метод (string)

Устанавливает соединение с сервером Docsvision.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Connect(string connectionString)
```

Параметры

connectionString

Тип: String

Строка соединения с сервером.

Заметки

В строке соединения указываются значения параметров соединения, разделенные точкой с запятой. Полный список параметров соединения приведён в приложении Параметры соединения с сервером Docsvision.

SessionManager.Connect — метод (string, string)

Устанавливает соединение с сервером Docsvision.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void Connect(string connectAddress, string baseName)
```

Параметры

connectAddress

Тип: String

Адрес сервера

baseName

Тип: String

Название подключаемой база данных.

SessionManager.Connect — метод (string, string, string, string)

Устанавливает соединение с сервером Docsvision.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void Connect(string connectAddress, string baseName, string userName, string password)
```

Параметры

connectAddress

Тип: String

Адрес сервера

baseName

Тип: String

Название подключаемой база данных

userName

Тип: String

Имя пользователя

password

Тип: String

Пароль пользователя.

SessionManager.CreateInstance — метод

Создание экземпляра менеджера сессий.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public static SessionManager CreateInstance()
```

Возвращаемое значение

Тип: SessionManager

Экземпляр сессии.

SessionManager.CreateInstance — метод (String)

Создание экземпляра менеджера сессий с указанием строки соединения.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public static SessionManager CreateInstance(string connectionString)
```

Параметры

connectionString

Тип: String

Строка соединения.

Возвращаемое значение

Тип: SessionManager

Экземпляр сессии.

Заметки

При вызове метода будет также запущен метод SessionManager.Connect с передачей ему параметра connectionString.

SessionManager.CreateSession — метод

Создаёт пользовательскую сессию.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession CreateSession()
```

Возвращаемое значение

Тип: UserSession

Сессия пользователя.

SessionManager.CreateSession — метод (SessionLoginFlags)

Создаёт пользовательскую сессию.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession CreateSession(SessionLoginFlags flags)
```

Параметры

flags

Тип: SessionLoginFlags

Флаги сессии.

Возвращаемое значение

Тип: UserSession

Сессия пользователя.

SessionManager.GetLocaleId — метод

Получение идентификатора языка для локализации сообщений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract int GetLocaleId()
```

Возвращаемое значение

Тип: System.Int32

Идентификатор языка.

SessionManager.GetOpenSessions — метод

Получение списка всех открытых сессий пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract InfoRowCollection GetOpenSessions()
```

Возвращаемое значение

Тип: InfoRowCollection

Коллекция открытый сессий.

SubSectionData — класс

Представляет подсекцию карточки. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class SubSectionData : IUpdatable, IXmlExportable
```

Свойства

Имя	Описание
Card	Карточка, которой принадлежит подсекция.
FirstRow	Возвращает первую строку подсекции.

Имя	Описание
InUpdate	Возвращает признак включения для подсекции режима отложенных изменения.
ParentRow	Родительская строка из родительской секции.
Rows	Возвращает коллекцию всех строк подсекции.
Section	Секция верхнего уровня, которой принадлежит подсекция.
Session	Возвращает текущую открытую сессию пользователя.

Методы

Имя	Описание
BeginUpdate	Включает режим отложенных изменений.
CancelUpdate	Отменяет несохранённые изменения и выключает режим отложенных изменений.
EndUpdate	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
FindRow(String)	Осуществляет поиск строки в секции с использованием XPath-запроса.
GetAllRows	Получение всех строк секции подсекции.
MarkCardForDeletion(Guid)	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
Refresh	Принудительное обновление данных с сервера.

Имя	Описание
`Refresh(Boolean)`	Принудительное обновление данных с сервера, независимо от наличия или отсутствия изменений.
`SaveXml(Stream)`	Сохраняет представление данных объекта в формате XML, в поток.
`SaveXml(Stream, ExportFlags)`	Сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта.
`SaveXml(Stream, ExportFlags, ExportCardInspector)`	Сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта и заданной логикой экспорта.
`UpdateNow`	Позволяет отправить накопленные изменения на сервер.

SubSectionData.Card — свойство

Карточка, которой принадлежит подсекция.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract CardData Card { get; }
```

Значение свойства

Тип: `CardData`

Карточка

SubSectionData.FirstRow — свойство

Возвращает первую строку подсекции.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract RowData FirstRow { get; }
```

Значение свойства

Тип: RowData

Строка подсекции

SubSectionData.InUpdate — свойство

Возвращает признак включения для подсекции режима отложенных изменения.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract bool InUpdate { get; }
```

Значение свойства

Тип: Boolean

false — режим выключен; true — режим включен

SubSectionData.ParentRow — свойство

Родительская строка из родительской секции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract RowData ParentRow { get; }
```

Значение свойства

Тип: RowData

Строка секции

SubSectionData.Rows — свойство

Возвращает коллекцию всех строк подсекции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract RowDataCollection Rows { get; }
```

Значение свойства

Тип: RowDataCollection

Коллекция строк

SubSectionData.Section — свойство

Секция верхнего уровня, которой принадлежит подсекция.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SectionData Section { get; }
```

Значение свойства

Тип: SectionData

Родительская секция

SubSectionData.Session — свойство

Возвращает текущую открытую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract UserSession Session { get; }
```

Значение свойства

Тип: `UserSession`

Сессия пользователя

SubSectionData.BeginUpdate — метод

Включает режим отложенных изменений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
void BeginUpdate()
```

SubSectionData.CancelUpdate — метод

Отменяет несохранённые изменения и выключает режим отложенных изменений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract void CancelUpdate()
```

SubSectionData.EndUpdate — метод

Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract void EndUpdate()
```

SubSectionData.FindRow — метод (String)

Осуществляет поиск строки в подсекции с использованием XPath-запроса.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract RowData FindRow(string filter)
```

Параметры

filter

Тип: String

XPath-запрос

Возвращаемое значение

Тип: RowData

Искомая строка подсекции

SubSectionData.GetAllRows — метод

Получение всех строк секции подсекции.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public RowDataCollection GetAllRows()
```

Возвращаемое значение

Тип: RowDataCollection

Коллекция найденных строк

SubSectionData.MarkCardForDeletion — метод (Guid)

Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void MarkCardForDeletion(Guid cardId)
```

Параметры

cardId

Тип: Guid

Идентификатор удаляемой карточки

SubSectionData.Refresh — метод

Принудительное обновление данных карточки с сервера.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public void Refresh()
```

Заметки

Данные будут загружены только при наличии в них изменений с момента предыдущего обновления.

SubSectionData.Refresh — метод (Boolean)

Принудительное обновление данных с сервера, независимо от наличия или отсутствия изменений

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void Refresh(bool ignoreTimestamp)
```

Параметры

ignoreTimestamp

Тип: `Boolean`

`true` — игнорировать отсутствие изменений; `false` — учитывать отсутствие изменений

Заметки

При установке `ignoreTimestamp` в значение `true`, данные объектов будут получены с сервера в любом случае.

SubSectionData.SaveXml — метод (Stream)

Сохраняет представление данных объекта в формате XML, в поток.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public void SaveXml(Stream stream)
```

Параметры

stream

Тип: `Stream`

Поток для данных

SubSectionData.SaveXml — метод (Stream, ExportFlags)

Сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public void SaveXml(Stream stream, ExportFlags exportFlags)
```

Параметры

stream

Тип: `Stream`

Поток для данных

exportFlags

Тип: `ExportFlags`

Флаги экспорта

SubSectionData.SaveXml — метод (Stream, ExportFlags, ExportCardInspector)

Сохраняет представление данных объекта в формате XML, в соответствии с установленными флагами экспорта и заданной логикой экспорта.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract void SaveXml(Stream stream, ExportFlags exportFlags, ExportCardInspector
inspector)
```

Параметры

stream

Тип: `Stream`

Поток для данных

exportFlags

Тип: `ExportFlags`

Флаги экспорта

inspector

Тип: `ExportCardInspector`

Логика экспорта

SubSectionData.UpdateNow — метод

Позволяет отправить накопленные изменения на сервер.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract void UpdateNow()
```

Заметки

Отправляет накопленные изменения на сервер, на клиенте они очищаются из кэша, режим продолжает действовать. Все изменения после этого момента также будут накапливаться.

UserSession — класс

Содержит данные сессии пользователя, а также предоставляет доступ к менеджерам управления объектами системы. Этот класс является **абстрактным**.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class UserSession : IDisposable
```

Свойства

Имя	Описание
AccessManager	Предоставляет *менеджера* по работе с объектами безопасности.
CardManager	Предоставляет *менеджера* для работы с карточками.
Connection	Возвращает текущее соединение с сервером Docsvision.
ExtensionManager	Предоставляет *менеджера* для работы с расширениями.

Имя	Описание
FileManager	Предоставляет *менеджера* для работы с файлами.
IconManager	Предоставляет *менеджера* иконок.
Id	Возвращает идентификатор текущей сессии.
LicenseManager	Предоставляет *менеджера* лицензий.
LockManager	Предоставляет *менеджера* блокировок.
LogManager	Предоставляет *менеджера* для работы с журналом работы приложения.
ProfileManager	Предоставляет *менеджера* пользовательских профилей.
Properties	Возвращает коллекцию параметров пользовательской сессии.
ReportManager	Предоставляет *менеджер* для работы с хранимыми процедурами (отчётами).

Методы

Имя	Описание
Awake	Пробуждает приостановленную сессию.
Close	Закрывает текущую сессию.
CreateSearchQuery	Создаёт поисковый запрос.
CreateSearchQuery(string)	Позволяет сформировать заранее описанный поисковый запрос.
CreateSectionQuery	Формирует поисковый запроса по секциям.
CreateSectionQuery(string)	Позволяет сформировать заранее описанный поисковый запрос по секциям.
CreateView	Создаёт представление.

Имя	Описание
`CreateView(string)`	Создаёт представление из переданной строки XML-формата.
`Suspend`	Приостанавливает сессию.

Заметки

В рамках сессии все данные кэшируются, к которым производилось обращение, для ускорения последующего доступа к ним. Размер кэша и продолжительность пребывания в нём данных регулируются `ObjectManager` автоматически, на основании внутренних алгоритмов. Разработчик может только явно потребовать удаления из кэша конкретных объектов (при помощи соответствующих методов этих объектов).

Примеры

В примере создаётся экземпляр менеджера сессий (`manager`), и в открытом соединении создаётся сессия пользователя (`session`).

```
SessionManager manager = SessionManager.CreateInstance();
manager.Connect("http://localhost/DocsVision5/StorageServer/StorageServerService.asmx",
string.Empty);
UserSession session = manager.CreateSession();
MessageBox.Show("IP-адрес клиента: " + session.Properties["ComputerAddress"].Value
.ToString());
```

UserSession.AccessManager — свойство

Предоставляет *менеджера* по работе с объектами безопасности.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract AccessManager AccessManager { get; }
```

Значение свойства

Тип: `AccessManager`

Менеджер объектов безопасности

UserSession.ExtensionManager — свойство

Предоставляет *менеджера* для работы с расширениями.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ExtensionManager ExtensionManager { get; }
```

Значение свойства

Тип: ExtensionManager

Менеджер расширений

UserSession.FileManager — свойство

Предоставляет *менеджера* для работы с файлами.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract FileManager FileManager { get; }
```

Значение свойства

Тип: FileManager

Менеджер файлов

UserSession.LicenseManager — свойство

Предоставляет *менеджера* лицензий.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract LicenseManager LicenseManager { get; }
```

Значение свойства

Тип: `LicenseManager`

Менеджер лицензий

Заметки

Данный менеджер предоставляет методы, позволяющие вести учет подключений к дополнительному модулю Docsvision, если лицензия (в виде дополнительной опции лицензии Docsvision) данного модуля предполагает такие ограничения.

UserSession.LockManager — свойство

Предоставляет *менеджера* блокировок.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract LockManager LockManager { get; }
```

Значение свойства

Тип: `LockManager`

Менеджер блокировок

UserSession.CardManager — свойство

Предоставляет *менеджера* для работы с карточками.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public abstract CardManager CardManager { get; }
```

Значение свойства

Тип: `CardManager`

Менеджер карточек

UserSession.Connection — свойство

Возвращает текущее соединение с сервером Docsvision.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract Connection Connection { get; }
```

Значение свойства

Тип: Connection

Соединение с сервером

UserSession.Properties — свойство

Возвращает коллекцию параметров пользовательской сессии.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract SessionPropertyCollection Properties { get; }
```

Значение свойства

Тип: SessionPropertyCollection

Параметры сессии

Заметки

Список и описание всех параметров приведён в разделе Список свойств пользовательской сессии.

UserSession.CreateSearchQuery — метод

Создаёт поисковый запрос.

- **Пространство имён:** DocsVision.Platform.ObjectManager

- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public SearchQuery CreateSearchQuery()
```

Возвращаемое значение

Тип: SearchQuery

Поисковый запрос

UserSession.ReportManager — свойство

Предоставляет *менеджер* для работы с хранимыми процедурами (отчётами).

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract ReportManager ReportManager { get; }
```

Значение свойства

Тип: ReportManager

Менеджер отчётов

ViewReadRequest — класс

Запрос представления.

- **Пространство имён:** DocsVision.Platform.ObjectManager
- **Сборка:** DocsVision.Platform.ObjectManager.dll

Синтаксис

```
public abstract class ViewReadRequest
```

Свойства

Имя	Описание
CurrentPage	Задаёт или возвращает индекс выбранной страницы.
CursorType	Задаёт или возвращает тип курсора.
FolderLevel	Задаёт или возвращает уровень вложенности включаемых подпапок.
LinkLevel	Задаёт или возвращает уровень вложенности связанных подпапок.
PageSize	Задаёт или возвращает количество строк в странице.
Parameters	Возвращает параметры запроса.
RowLimit	Задаёт или возвращает ограничение количества строк представления.
SearchXml	Задаёт или возвращает XML-строку поискового запроса для получения карточек.
SelectionCursorId	Задаёт или возвращает идентификатор курсора старого представления, для которого указываются выбранные строки.
SelectionItemIds	Задаёт или возвращает порядковые номера выбранных в старом представлении строк.
Session	Возвращает текущую сессию пользователя.
ShowArchived	Задаёт или возвращает признак включения архивных карточек.
ShowDeleted	Задаёт или возвращает признак включения удалённых карточек.
ShowHidden	Задаёт или возвращает признак включения скрытых карточек.
ShowLabels	Задаёт или возвращает признак включения меток.

Имя	Описание
`Sortings`	Возвращает коллекцию параметров сортировки.
`SourceId`	Задаёт или возвращает идентификатор источника данных.
`ViewId`	Задаёт или возвращает идентификатор представления.
`ViewSourceType`	Задаёт или возвращает тип источника данных представления.

ViewSource — класс

Источник данных представления. Этот класс не может быть унаследован.

- **Пространство имён:** `DocsVision.Platform.ObjectManager`
- **Сборка:** `DocsVision.Platform.ObjectManager.dll`

Синтаксис

```
public sealed class ViewSource
```

Свойства

Имя	Описание
`ShowArchived`	Возвращает или задаёт признак необходимости отображения в представлении архивных карточек.
`ShowDeleted`	Возвращает или задаёт признак необходимости отображения в представлении удалённых карточек.
`ShowHidden`	Возвращает или задаёт признак необходимости отображения в представлении скрытых и системных карточек.

Методы

Имя	Описание
FromCardLib(Guid)	В качестве источника данных используется указанная библиотека карточек.
FromCards	Источником данных являются все карточки.
FromCardType(Guid)	Источником данных являются все карточки заданного типа.
FromFolder(Guid)	Источником данных является заданная папка.
FromRecycleBin	Источником является корзина.
FromReferences	Источником являются справочники.
FromSearch(Guid)	Источником является сохранённый поисковый запрос.
FromSearch(String)	Источником является сформированный поисковый запрос (XML).
FromSearchFolder(Guid)	Источником является заданная виртуальная папка.
FromSearchResults(Guid)	Источником является заданная системная папка "Результаты поиска".

DocsVision.Platform.ObjectModel — пространство имён

Пространство имён DocsVision.Platform.ObjectModel содержит классы определяющие фабрики сервисов и преобразователей данных, сущности хранящие данные контекста объектов, а также вспомогательные классы и интерфейсы. Эти классы составляют объектную модель уровня бизнес-логики.

Пространства имён

Пространство имён	Описание
DocsVision.Platform.ObjectModel.Mapping	Пространство имён DocsVision.Platform.ObjectModel.Mapping содержит преобразователи данных.

Пространство имён	Описание
DocsVision.Platform.ObjectModel.Search	Пространство имён DocsVision.Platform.ObjectModel.Search предоставляет типы, используемые для организации поиска.

Классы

Класс	Описание
ContextService	Базовый класс сервисов использующих контекст объектов.
ObjectBase	Базовый класс объектов модели уровня бизнес-логики.
ObjectBaseSecurity	Представляет объект безопасности. Предоставляет методы установки и получения прав доступа.
ObjectCollection<T>	Основной класс коллекции объектов ObjectBase, поддерживающий функцию оповещения об изменении.
ObjectContext	Класс ObjectContext предоставляет методы доступа к сущностям объектной модели уровня бизнес-логики.
ObjectHelper	Содержит метод получения родительского объекта.
ObjectLockInfo	Содержит информацию о блокировке объекта.
ObjectModelException	Представляет общее исключение объектной модели.
ObjectProperty	Предоставляет методы для работы со свойствами объектной модели.
ObjectPropertyValue	Предоставляет значение для коллекции ObjectPropertyValueCollection.
ObjectPropertyValueCollection	Используется некоторыми преобразователями данных для хранения значений.

Класс	Описание
`ObjectRef<T>`	Содержит ссылку на объект в *контексте объектов*.
`ObjectState`	Позволяет получать флаги состояния объекта.
`PowerOfAttorney`	Системная карточка доверенности.
`PowerOfAttorneyMainInfo`	Представляет секцию "Основная информация" системной карточки доверенности.
`PowerOfAttorneyFNSDOVEL502RevocationData`	Данные отзыва доверенности версии 5.02 для взаимодействия с ФНС.
`FIO`	Фамилия, имя, отчество (ФИОТип). Таблица 4.16.
`ForeignCompanyInfo`	Сведения об иностранной организации (при отзыве основной доверенности) (ИО). Таблица 4.8.
`IdentityCardInfo`	Сведения о документе, удостоверяющем личность физического лица (УдЛичнФЛТип). Таблица 4.15.
`IndividualInfo1`	Сведения о физическом лице (СведФЛТип). Таблица 4.12.
`IndividualInfo`	Сведения о физическом лице, в том числе индивидуальном предпринимателе (СвФизЛицТип). Таблица 4.14.
`OrganizationInfo`	Сведения об организации (СвОргТип). Таблица 4.13.
`PowerOfAttorneyInfo`	Сведения для доверенности, подтверждающей полномочия нескольких уполномоченных представителей (СвПдтвУП). Таблица 4.4.
`PrincipalInfo`	Сведения о доверителе (СвДоверит). Таблица 4.5.

Класс	Описание
PrincipalWithoutPowerOfAttorneyInfo	Сведения о лице, действующем от имени юридического лица без доверенности (СвРукОрг). Таблица 4.7.
RetrustPrincipalInfo	Сведения о доверителе при отзыве доверенности, совершенной (выданной) в порядке передоверия (СвДоверщП). Таблица 4.10.
RevocationDocument	Состав и структура документа (Документ). Таблица 4.2.
RevocationInfo	Сведения заявления об отзыве доверенности (СвЗаяв). Таблица 4.3.
RussianCompanyInfo	Сведения о российской организации (при отзыве основной доверенности) (НПЮЛ). Таблица 4.6.
SeparateSubdivisionManagerInfo	Сведения о руководителе обособленного подразделения (СвРукОП). Таблица 4.9.
SignerInfo	Сведения о физическом лице, подписывающем заявление об отзыве доверенности от имени доверителя (российской организации / иностранной организации) без доверенности, или сведения о физическом лице, подписывающем заявление об отзыве доверенности от своего имени, в том числе при передаче полномочий другому лицу в порядке передоверия (Подписант). Таблица 4.11.
PowerOfAttorneyRepresentative	Представляет секцию "Представитель" системной карточки доверенности.
PowerOfAttorneyRepresentativesPowers	Представляет секцию "Полномочия" системной карточки доверенности.

Класс	Описание
PowerOfAttorneyRevocationApplicantInfo	Информация о лице, отзывающем доверенность.
PowerOfAttorneyRevocationData	Данные отзыва доверенности.
PowerOfAttorneyRevocationInfo	Содержит информацию об отзыве доверенности.
PowerOfAttorneySubsidiaryPowersOfAttorney	Представляет секции "Системные карточки" дочерних доверенностей.
Powers	Представляет полномочия для справочника полномочий МЧД.
PowersCode	Представляет коды полномочий для справочника полномочий МЧД.
PowersGroup	Представляет группы полномочий для справочника полномочий МЧД.
PowersPowerOfAttorneyFormat	Представляет типы доверенности карточки справочник полномочий МЧД.
ServiceFactory	Управляет получением сервиса из фабрики сервисов.
StaticObjectContext	Управляет получением сервиса из фабрики сервисов.

Интерфейсы

Интерфейс	Описание
IContextService	Определяет возможность передачи контекста объектов в класс.
IInitializable	Обозначает в классе признак того, что объект был инициализирован.
ILockManager	Представляет менеджер блокировки.
IObjectContext	Описывает ключевую функциональность класса ObjectContext по осуществлению наиболее общих действий над объектами системы.

Интерфейс	Описание
IObjectRef	Определяет наличие у объекта уникального идентификатора и типа.
IObjectStateService	Представляет интерфейс сервиса состояний объектов.
IServiceFactoryRegistry	Интерфейс реестра фабрик сервисов.

Перечисления

Перечисление	Описание
ObjectAccessControlType	Определяет тип объекта доступа.
ObjectLockStatus	Определяет статус блокировки объекта.
ObjectPropertyOptions	Определяет список флагов, используемых свойствами объектной модели.
ObjectStateFlags	Определяет состояние объекта.
PowerOfAttorneyRegTransferStatuses	Статус передачи доверенности в распределённый реестр ФНС.
PowerOfAttorneyRetrustType	Признак возможности оформления передоверия.
PowerOfAttorneyRevocationApplicantType	Тип заявителя отзыва доверенности.
PowerOfAttorneyRevocationType	Тип заявления на отзыв.

DocsVision.Platform.ObjectModel.Mapping — пространство имён

Пространство имён DocsVision.Platform.ObjectModel.Mapping содержит базовые классы преобразователей данных объектной модели уровня бизнес-логики.

Классы

Класс	Описание
CardMapper	Преобразователь данных для модели карточки.

Класс	Описание
CardRowMapper	Преобразователь данных для модели строки карточки.
DictionaryMapper<T>	Базовый класс преобразователя данных для объектной модели справочника.
DictionaryRowMapper	Преобразователь данных для модели строки справочника.
ObjectMap	Поставляет данные для преобразователя данных.
ObjectMapper<T>	Реализация интерфейса IObjectMapper.
ObjectMapperBase<T>	Базовый класс *преобразователя данных*.
ObjectMapperFactory	Базовый класс *фабрики преобразователей данных*.

Интерфейсы

Интерфейс	Описание
IObjectMapper	Описывает основную функциональность *преобразователь данных*.
IObjectMapperFactory	Интерфейс фабрики преобразователей данных.
IObjectMapperFactoryRegistry	Интерфейс реестра фабрик преобразователей данных.

DictionaryMapper<T> — класс

Базовый класс преобразователя данных для объектной модели справочника.

- **Пространство имён:** DocsVision.Platform.ObjectModel.Mapping
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public abstract class DictionaryMapper<T> : ObjectMapperBase<T> where T : ObjectBase
```

Методы

Имя	Описание
`CreateObject(T)`	При переопределении в производном классе должен создавать экземпляр объекта типа `T`.
`LoadObject(ObjectRef<T>, IDictionary<String, Object>)`	Получает данные карточки из хранилища и загружает их в новый экземпляр карточки.

ObjectMap — класс

Поставляет данные для преобразователя данных.

- **Пространство имён:** `DocsVision.Platform.ObjectModel.Mapping`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public class ObjectMap
```

Конструкторы

Имя	Описание
`ObjectMap()`	Инициализирует новый экземпляр класса ObjectMap.

Свойства

Имя	Описание
`Collections`	Возвращает коллекцию секций.
`Fields`	Возвращает коллекцию полей.
`ObjectTypeId`	Задаёт или возвращает идентификатор регистрируемого объекта.

Имя	Описание
`ParentObjectId`	Задаёт или возвращает идентификатор родительского, по отношению к регистрируемому, объекта.
`References`	Возвращает коллекцию ссылочных полей карточки (для типа `refid`).

Методы

Имя	Описание
`Collection(ObjectProperty, Guid)`	Связывает свойство с секцией.
`Field(ObjectProperty, String)`	Связывает свойство с полем.
`Reference(ObjectProperty, String)`	Связывает свойство со ссылочным полем карточки.

Примеры

Ниже приведён пример части преобразователя данных для карточки "Документ", в котором выполняется привязка идентификатора секции к свойству `Document.MainInfoProperty`

```
protected override ObjectMap GetObjectMap()
{
 ObjectMap objectMap = new ObjectMap();

 objectMap.ObjectTypeId = CardDocument.ID; ①

 objectMap.Collection(Document.MainInfoProperty, CardDocument.MainInfo.ID); ②
 return objectMap;
}
```

① Идентификатор объекта — идентификатор карточки типа "Документ" (в нашем случает он уже статически присвоен объекту `CardDocument.ID`).

② Выполнение связки свойства `Document.MainInfoProperty` с секцией, имеющей идентификатор `CardDocument.MainInfo.ID` (статический, уже присвоен) .

ObjectMapperBase<T> — класс

Базовый класс *преобразователя данных*.

- **Пространство имён:** DocsVision.Platform.ObjectModel.Mapping
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public abstract class ObjectMapperBase<T> : ObjectMapper<T> where T : ObjectBase
```

Конструкторы

Имя	Описание
ObjectMapperBase(ObjectContext)	Инициализирует новый экземпляр класса ObjectMapperBase с помощью указанного значения.

Свойства

Имя	Описание
Mapping	Возвращает ссылку на *карту объектов*, которая содержит коллекцию связей между свойствами объекта и отдельными элементами карточки.
PersistentStore	Возвращает интерфейс хранилища Docsvision, предоставляющий методы создания, поиска и удаления объектов, а также несколько вспомогательных методов.

Методы

Имя	Описание
CreateObject(ObjectInitializationData)	Создаёт новый объект, с переданными параметрами, в хранилище Docsvision .
GetObjectMap	Возвращает *карту объектов*.
GetObjectRef(Object)	Возвращает ссылку на переданный объект. Позволяет получить идентификатор объекта.
GetParentObject(T, Type)	Возвращает родительский объект.

Имя	Описание
`GetStore(ObjectRef< T>)`	Возвращает хранилище данных переданного по ссылке объекта.
`GetStore(T)`	Возвращает хранилище данных переданного объекта.
`GetStore(ObjectRef<T>, Boolean)`	Возвращает хранилище данных переданного по ссылке объекта.
`GetStore(T, Boolean)`	Возвращает хранилище данных переданного объекта.
`LoadObject(ObjectRef<T>)`	Загружает объект, переданный по ссылке, в *преобразователь данных*.
`LoadObject(PropertyStore)`	Загружает объект в *преобразователь данных*.
`LoadObject(ObjectRef<T>, IDictionary<String, Object>)`	Загружает объект, переданный по ссылке, в *преобразователь данных*. Также загружает параметры объекта.
`LoadObjectData(PropertyStore, ObjectInitializationData)`	Выполняет загрузку данных из хранилища в проинициализированное хранилище данных.
`LoadObjectData(ObjectRef<T>, IDictionary<String, Object>, ObjectInitializationData)`	Выполняет загрузку данных из хранилища в проинициализированное хранилище данных. Также загружает параметры объекта.
`RefreshObject(T)`	Выполняет обновление данных в хранилище данных объекта.
`SaveObject(T)`	Выполняет сохранение данных объекта.
`SaveSecurity(T)`	Выполняет сохранение параметров безопасности объекта.
`TransformQuery(QueryObject)`	Выполняет преобразование поискового запроса.

Заметки

Практическая информация по разработке *преобразователя данных* содержится в разделе Разработка преобразователя данных.

ObjectMapperFactory — класс

Базовый класс *фабрики преобразователей данных*.

- **Пространство имён:** DocsVision.Platform.ObjectModel.Mapping
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public abstract class ObjectMapperFactory : IObjectMapperFactory
```

Конструкторы

Имя	Описание
ObjectMapperFactory(ObjectContext)	Инициализирует новый экземпляр класса ObjectMapperFactory с помощью указанного значения.

Методы

Имя	Описание
GetObjectMapper(Type)	Возвращает *преобразователь данных* указанного типа.
RegisterObjectMapper(Type, Type)	Регистрирует новый *преобразователь данных* в *фабрике преобразователей данных*

Примеры

В качестве примера приведена часть класса *SystemCardsMapperFactory* (*фабрика преобразователей данных* библиотеки "Системные карточки")

```
public sealed class SystemCardsMapperFactory : ObjectMapperFactory
{
 public SystemCardsMapperFactory(ObjectContext context) : base(context)
 {
```

```
    // Производится регистрация преобразователя данных для "Карточки папок"
    base.RegisterObjectMapper(typeof(Folders), typeof(FoldersMapper));
  }
 }
```

IObjectMapper — интерфейс

Описывает основную функциональность *преобразователь данных*.

- **Пространство имён:** DocsVision.Platform.ObjectModel.Mapping
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public interface IObjectMapper
```

Свойства

Имя	Описание
MapperType	Возвращает тип *преобразователя данных*.
Mapping	Возвращает карту объектов, определённую в *преобразователе данных*.

Методы

Имя	Описание
CopyObject(ObjectBase, ObjectBase)	При переопределении в производном классе создаёт копию переданного объекта в указанном родительском объекте.
CreateObject(ObjectBase)	При переопределении в производном классе загружает переданную сущность в хранилище объектов (DocsVision.Platform.ObjectModel.Persistence.DocsVisionStore) объектной модели.
DeleteObject(ObjectBase)	При переопределении в производном классе удаляет объект из хранилища.

Имя	Описание
GetObjectLockInfo(IObjectRef)	При переопределении в производном классе предоставляет информацию о блокировке объекта.
GetObjectRef(Object)	При переопределении в производном классе возвращает ссылку на переданный объект.
GetParentObject(ObjectBase, Type)	При переопределении в производном классе возвращает родительский объект.
GetParentObject(ObjectBase, Type)	При переопределении в производном классе возвращает родительский объект указанного типа.
LoadObject(IObjectRef)	При переопределении в производном классе получает из хранилища данные для объекта и создаёт новый экземпляр объекта на основе полученных данных.
LockObject(IObjectRef, Boolean)	При переопределении в производном классе выполняет блокировку объекта.
MoveObject(ObjectBase, ObjectBase)	При переопределении в производном классе перемещает заданный объект к новому родительскому объекту.
ObjectExists(IObjectRef, Boolean)	При переопределении в производном классе проверяет существование объекта.
RefreshObject(ObjectBase)	При переопределении в производном классе выполняет перезагрузку данных для переданного объекта.
SaveObject(ObjectBase)	При переопределении в производном классе выполняет сохранение данных переданного объекта.
SaveSecurity(ObjectBase)	При переопределении в производном классе выполняет сохранение настроек безопасности переданного объекта.

Имя	Описание
`TransformQuery(QueryObject)`	При переопределении в производном классе преобразует переданный запрос в соответствии с картой объектов, загруженной в преобразователь данных.
`UnlockObject(IObjectRef)`	При переопределении в производном классе снимает блокировку с переданного объекта.

IObjectMapperFactoryRegistry — интерфейс

Интерфейс реестра фабрик преобразователей данных.

- **Пространство имён:** `DocsVision.Platform.ObjectModel.Mapping`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public interface IObjectMapperFactoryRegistry
```

Свойства

Имя	Описание
`Factories`	Возвращает коллекцию зарегистрированных фабрик.

Методы

Имя	Описание
`RegisterFactory(Type)`	Регистрирует фабрику указанного типа.

DocsVision.Platform.ObjectModel.Search — пространство имён

Пространство имён DocsVision.Platform.ObjectModel.Search предоставляет типы, используемые для организации поиска.

Классы

Класс	Описание
`Criterion`	Предоставляет критерий поиска для поискового запроса.
`QueryObject`	Класс QueryObject представляет собой контейнер для формирования поискового запроса.

Перечисления

Перечисление	Описание
`CompareOperation`	Определяет доступные операции сравнения, используемые критерием поиска.
`QueryOperation`	Определяет логические операции применимые к критериями поискового запроса.

Criterion — класс

Предоставляет критерий поиска для поискового запроса.

- **Пространство имён:** `DocsVision.Platform.ObjectModel.Search`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public class Criterion
```

Конструкторы

Имя	Описание
`Criterion(String, CompareOperation)`	Инициализирует новый экземпляр класса Criterion с заданными параметрами.
`Criterion(String, Object)`	Инициализирует новый экземпляр класса Criterion с заданными параметрами.

Имя	Описание
`Criterion(String, Object, CompareOperation)`	Инициализирует новый экземпляр класса Criterion с заданными параметрами.

Свойства

Имя	Описание
`FieldName`	Задаёт или возвращает название поля, по которому осуществляется поиск.
`ObjectTypeId`	Задаёт или возвращает типа искомой карточки.
`Operation`	Задаёт или возвращает операцию сравнения значения поля и параметра Value.
`Value`	Задаёт или возвращает значение поля FieldName.

QueryObject — класс

Класс QueryObject представляет собой контейнер для формирования поискового запроса.

- **Пространство имён:** `DocsVision.Platform.ObjectModel.Search`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public class QueryObject
```

Конструкторы

Имя	Описание
`QueryObject()`	Инициализирует новый экземпляр класса QueryObject.
`QueryObject(Criterion)`	Инициализирует новый экземпляр класса QueryObject с заданными параметрами.

Имя	Описание
QueryObject(String, CompareOperation)	Инициализирует новый экземпляр класса QueryObject с заданными параметрами.
QueryObject(String, Object)	Инициализирует новый экземпляр класса QueryObject с заданными параметрами.

Свойства

Имя	Описание
Criteria	Возвращает коллекцию критериев поиска, определённых в контейнере поискового запроса.
ObjectTypeId	Задаёт или возвращает идентификатор типа объекта поиска.
Operation	Задаёт или возвращает логическую операцию, применяемую к критериям поиска.

Методы

Имя	Описание
AddCriterion(Criterion)	Добавляет новый критерий поиска к запросу.
AddCriterion(String, CompareOperation)	Добавляет критерий поиска по значению поля.
AddCriterion(String, Object)	Добавляет критерий поиска по значению поля.
AddCriterion(String, Object, CompareOperation)	Добавляет новый критерий поиска в контейнер поискового запроса.

Примеры

В приведенном примере выполняется поиск строки с именем "SyncMaster 123" в узлах *Конструктора справочников*

```
①

QueryObject query = new QueryObject(BaseUniversalItem.NameProperty.Name, "SyncMaster
123"); ②

BaseUniversalItem item = objectContext.FindObject<BaseUniversalItem>(query); ③
```

① Инициализация контекста объектов.

② Формирование поискового запроса по полю `Name`.

③ Выполнение запроса.

QueryObject.Criteria — свойство

Возвращает коллекцию критериев поиска, определённых в контейнере поискового запроса.

- **Пространство имён:** `DocsVision.Platform.ObjectModel.Search`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public IEnumerable<Criterion> Criteria { get; }
```

Значение свойства

Тип: `System.Collections.Generic.IEnumerable<Criterion>`

Коллекция критериев поиска `Criterion`

QueryObject.ObjectTypeId — свойство

Задаёт или возвращает идентификатор типа объекта поиска.

- **Пространство имён:** `DocsVision.Platform.ObjectModel.Search`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public Guid? ObjectTypeId { get; set; }
```

Значение свойства

Тип: `System.Guid?`

Идентификатор типа карточки/справочника

Заметки

Идентификатор используется для уточнения типа карточки искомого объекта: карточка или справочник. В зависимости от типа карточки, возвращается строка справочника, либо карточка.

QueryObject.Operation — свойство

Задаёт или возвращает логическую операцию, применяемую к критериям поиска.

- **Пространство имён:** DocsVision.Platform.ObjectModel.Search
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public QueryOperation Operation { get; set; }
```

Значение свойства

Тип: QueryOperation

Логическая операция

QueryObject.AddCriterion — метод (String, Object, CompareOperation)

Добавляет новый критерий поиска в контейнер поискового запроса.

- **Пространство имён:** DocsVision.Platform.ObjectModel.Search
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public Criterion AddCriterion(string fieldName, object value, CompareOperation operation)
```

Параметры

fieldName

Тип: System.String

Название поля, по котором производится поиск

value

Тип: `System.Object`

Значение искомого поля

operation

Тип: `CompareOperation`

Оператор сравнения действительного и искомого значений поля

Возвращаемое значение

Тип: `Criterion`

Критерий поиска

CompareOperation — перечисление

Определяет доступные операции сравнения, используемые критерием поиска.

- **Пространство имён:** `DocsVision.Platform.ObjectModel.Search`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public enum CompareOperation
```

Члены

Имя члена	Описание
`Equals`	Равны.
`NotEquals`	Не равны.
`Contains`	Содержится.
`StartsWith`	Начинается с.
`EndsWith`	Заканчивается на
`IsNull`	Пустой элемент.

QueryOperation — перечисление

Определяет логические операции применимые к критериям поискового запроса.

- **Пространство имён:** DocsVision.Platform.ObjectModel.Search
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public enum QueryOperation
```

Члены

Имя члена	Описание
Or	ИЛИ
And	И

ContextService — класс

Базовый класс сервисов использующих контекст объектов.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public abstract class ContextService : IContextService
```

Свойства

Имя	Описание
Context	Возвращает контекст объектов.

Заметки

Данный класс должен быть базовым для всех разрабатываемых сервисов.

ObjectBase — класс

Базовый класс объектов модели уровня бизнес-логики.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public abstract class ObjectBase : IDisposable, INotifyPropertyChanging,
INotifyPropertyChanged, IComparable
```

Методы

Имя	Описание
Dispose	Запускает очистку ресурсов.
Dispose(Boolean)	Запускает очистку ресурсов.
GetObjectId()	Возвращает идентификатор объекта.
GetValue(ObjectProperty)	Получает значение указанного свойства.
GetValue(String)	Получает значение свойства с заданным названием.
HasValue(ObjectProperty)	Проверяет доступность значения указанного свойства.
OnPropertyChanged(ObjectPropertyChangedEventArgs)	Создаёт событие PropertyChanged. Передает старое и новое значение свойства.
OnPropertyChanged(String)	Создаёт событие PropertyChanged.
OnPropertyChanging(ObjectPropertyChangedEventArgs)	Создаёт событие PropertyChanging. Передает старое и новое значение свойства.
ResetToOriginalValues	Сбрасывает значения всех зарегистрированных свойств в начальное значение.
ResetValueChangedFlags	Сбрасывает значение флага того, что свойство было изменено.
SetValue(ObjectProperty, Object)	Задаёт значение указанного свойства.
ValueChanged(ObjectProperty)	Определяет наличие изменений в свойстве.

События

Имя	Описание
PropertyChanged	Возникает после изменения свойства объекта.
PropertyChanging	Возникает при изменении свойства объекта.

ObjectBase.Dispose — метод

Запускает очистку ресурсов.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public void Dispose()
```

Заметки

Передает управление методу ObjectBase.Dispose(Boolean) и запрещает сборщику мусора вызвать Object.Finalize для экземпляра класса.

ObjectBase.Dispose — метод (Boolean)

Запускает очистку ресурсов.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
protected virtual void Dispose(bool disposing)
```

Параметры

disposing

Тип: System.Boolean

Очищать ресурсы

Заметки

При `disposing` равном `true`, метод вызывает `this.events.Dispose()` класса `System.ComponentModel.EventHandlerList`. Желательно в производном классе реализовать необходимую очистку ресурсов.

ObjectBase.GetValue — метод (ObjectProperty)

Получает значение указанного свойства.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public object GetValue(ObjectProperty property)
```

Параметры

property

Тип: `ObjectProperty`

Свойство

Возвращаемое значение

Тип: `System.Object`

Значение свойства

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `property`.
`System.InvalidOperationException`	Ошибка возвращается в случае, если свойство property не зарегистрировано.

ObjectBase.GetValue — метод (String)

Получает значение свойства с заданным названием.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`

- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public object GetValue(string propertyName)
```

Параметры

propertyName

Тип: System.String

Название свойства

Возвращаемое значение

Тип: System.Object

Значение свойства

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр propertyName.
System.InvalidOperationException	Ошибка возвращается в случае, если свойство не зарегистрировано.

ObjectBase.HasValue — метод (ObjectProperty)

Проверяет доступность значения указанного свойства.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public bool HasValue(ObjectProperty property)
```

Параметры

property

Тип: ObjectProperty

Свойство

Возвращаемое значение

Тип: `System.Boolean`

`true` — свойство доступно, при `false` — свойство недоступно, либо не зарегистрировано

ObjectBase.ResetToOriginalValues — метод

Сбрасывает значения всех зарегистрированных свойств в начальное значение.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public void ResetToOriginalValues()
```

Заметки

Начальные значения, метод получает из состояния объекта (свойство `IObjectWithState.ObjectState`), реализованное в классе ObjectBase. Также сбрасывается значение флага Changed — отражает наличие изменений в свойстве.

ObjectBase.ResetValueChangedFlags — метод

Сбрасывает значение флага того, что свойство было изменено.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public void ResetValueChangedFlags()
```

Заметки

Метод сбрасывает флаг `Changed` всех зарегистрированных свойств в значение `false`. Данный флаг проверяется методом `ObjectBase.ValueChanged(ObjectProperty)`.

ObjectBase.SetValue — метод (ObjectProperty, Object)

Задаёт значение указанного свойства.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public void SetValue(ObjectProperty property, object value)
```

Параметры

property

Тип: ObjectProperty

Свойство

value

Тип: System.Object

Значение свойства

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр property.
System.InvalidOperationException	Ошибка возвращается в случае, если свойство не зарегистрировано.

ObjectBase.ValueChanged — метод (ObjectProperty)

Определяет наличие изменений в свойстве.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public bool ValueChanged(ObjectProperty property)
```

Параметры

property

Тип: `ObjectProperty`

Свойство

Возвращаемое значение

Тип: `System.Boolean`

`true` — значение было изменено, иначе — `false`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `property`.
`System.InvalidOperationException`	Ошибка возвращается в случае, если свойство не зарегистрировано.

Заметки

Значение проверяемого методом флага `Changed` устанавливается методом `ObjectBase.SetPropertyValue`.

ObjectBaseSecurity — класс

Представляет объект безопасности. Предоставляет методы установки и получения прав доступа.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
[ClassInterface(ClassInterfaceType.AutoDual)]
[ComVisible(true)]
public sealed class ObjectBaseSecurity : ISecureObject
```

Свойства

Имя	Описание
AcceptSecurityChangesOnly	Задаёт или возвращает признак того, что в случае наличия изменений объекта, следует применять исключительно изменения настроек прав.
FullName	Возвращает полное имя объекта.
Id	Возвращает идентификатор объекта.
Name	Возвращает имя/название объекта.
Parent	Возвращает родительский объект безопасности.
SecurityClassId	Возвращает идентификатор класса безопасности объекта.

Методы

Имя	Описание
AccessCheck(Int32, Int32)	Проверяет наличие указанного доступа и записывает существующий.
AccessCheck(Guid, Int32, Int32)	Проверяет наличие указанного доступа и записывает существующий.
GetAccessControl(AccessControlSections)	Возвращает параметры безопасности для объекта.
GetDefaultValue(ObjectAccessControlType)	Возвращает делегат, получающий базовые настройки безопасности.
SetAccessControl(Byte[], AccessControlSections, AccessControlPropagation)	Устанавливает параметры безопасности для объекта.

ObjectCollection<T> — класс

Основной класс коллекции объектов ObjectBase, поддерживающий функцию оповещения об изменении.

- **Пространство имён:** DocsVision.Platform.ObjectModel

- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public class ObjectCollection<T> : ObjectCollectionBase<T>, IList, ICollection, IList<T>,
ICollection<T>, IEnumerable<T>, IEnumerable where T : DocsVision.Platform.ObjectModel
.ObjectBase
```

Конструкторы

Имя	Описание
ObjectCollection()	Инициализирует экземпляр класса ObjectCollection.
ObjectCollection(ObjectCollectionInitializationData)	Инициализирует экземпляр класса ObjectCollection с помощью указанных значений.

Свойства

Имя	Описание
Item	Задаёт или возвращает элемент коллекции с указанным индексом.
Items	Возвращает все элементы коллекции.

Методы

Имя	Описание
Add(T)	Добавляет элемент коллекции.
AddRange(IEnumerable<T>)	Добавляет несколько элементов в коллекцию.
Clear	Удаляет все элементы коллекции.
Contains(T)	Проверяет вхождение в коллекцию указанного элемента.
CopyTo(T[], Int32)	Копирует элементы коллекции в массив, начиная с указанного индекса.
GetEnumerator	Возвращает перечислитель.

Имя	Описание
IndexOf(T)	Определяет индекс заданного элемента в коллекции ObjectCollection.
Insert(Int32, T)	Вставляет элемент в коллекцию ObjectCollection с заданным индексом.
Remove(T)	Удаляет первое вхождение указанного объекта из списка ObjectCollection.
RemoveAt(Int32)	Удаляет элемент ObjectCollection, расположенный по указанному индексу.
ToArray	Возвращает массив, полученный из элементов коллекции ObjectCollection.

ObjectContext — класс

Класс ObjectContext предоставляет методы доступа к сущностям объектной модели уровня бизнес-логики.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public sealed class ObjectContext : IDisposable, IServiceContainer, IServiceProvider,
IObjectContext, ILockManager
```

Конструкторы

Имя	Описание
ObjectContext(IServiceProvider)	Инициализирует новый экземпляр класса ObjectContext с помощью указанного значения.

Свойства

Имя	Описание
IsClosed	Возвращает признак существования контекста объектов.

Имя	Описание
UserAccount	Возвращает название учетной записи текущего пользователя.

Объектная модель семейства ObjectContext потоконебезопасна (не thread-safe), поэтому важно использовать межпоточную синхронизацию или не использовать её параллельно.

Методы

Имя	Описание
AcceptChanges	Применяет все изменения, которые были сделаны в контексте объектов.
AddObject(ObjectBase)	Добавляет объект в контекст объектов.
AddService<T>(T)	Добавляет указанный сервис в контекст объектов.
CopyObject<T>(T, ObjectCollection<T>, Action<T>)	Копирует указанный объект в нового владельца.
DeleteObject(ObjectBase)	Данный метод используется для удаления заданного объекта.
Dispose	Выполняет очистку ресурсов контекста объектов.
FindObject<T>(QueryObject)	Возвращает первый результат поиска, удовлетворяющий заданным условиям.
FindObjects<T>(QueryObject)	Возвращает первый результат поиска, удовлетворяющий заданным условиям.
GetObject(IObjectRef)	Получает объект по переданной ссылке.
GetObject(IObjectRef, IDictionary<String, Object>)	Получает объект по переданной ссылке. Также передаются дополнительные параметры получения объекта.
GetObject<T>(Object)	Получает объект заданного типа с указанным идентификатором.

Имя	Описание
GetObject<T>(ObjectRef<T>)	Получает объект заданного типа из переданной ссылки.
GetObjectRef<T>(Object)	Получает ссылку на объект заданного типа с указанным идентификатором.
GetObjectRef(ObjectBase)	Получает ссылку на переданный объект.
GetService<T>	Возвращает *сервис*, реализующий переданный интерфейс.
LockObject(IObjectRef, Boolean)	Блокирует объект по переданной ссылке.
LockObject(ObjectBase, Boolean)	Блокирует переданный объект.
RefreshObject(IObjectRef)	Метод обновляет корневой объект.
RollbackChanges	Отменяет все изменения в контексте объектов, которые не были сохранены методом AcceptChanges, SaveObject<T>(T) или SaveObjects<T>(IEnumerable<T>).
RollBackObject(ObjectBase)	Отменяет несохранённые изменения в заданном объекте.
SaveObject(ObjectBase)	Сохраняет изменения выбранного объекта.
SaveObject<T>(T)	Сохраняет изменения выбранного объекта.
SaveObjects<T>(IEnumerable<T>)	Сохраняет изменения в заданной коллекции объектов.
UnlockObject(ObjectBase)	Снимает с объекта собственную блокировку.

События

Имя	Описание
Accepted	Событие возникает после сохранения всех изменений в контексте, вызванного AcceptChanges.

Имя	Описание
AcceptedObject	Событие возникает после сохранения всех изменений в контексте, вызванного SaveObject.
AcceptedObjects	Событие возникает после сохранения всех изменений в контексте, вызванного SaveObjects.
Accepting	Событие возникает до сохранения всех изменений в контексте, вызванного AcceptChanges.
AcceptingObject	Событие возникает до сохранения всех изменений в контексте, вызванного SaveObject.
AcceptingObjects	Событие возникает до сохранения всех изменений в контексте, вызванного SaveObjects.
BeforeRollback	Событие возникает до отмены изменений, вызванной RollbackChanges.
BeforeRollbackObject	Событие возникает до отмены изменений, вызванной RollBackObject.
BeforeRollbackObjects	Событие возникает до отмены изменений, вызванной RollBackObjects.
Disposed	Событие возникает перед очисткой, вызванной методом Dispose или Dispose(true).
RefreshedObject	Событие возникает после обновления, вызванного методом RefreshObject.
Rollback	Событие возникает после отмены изменений, вызванной RollbackChanges.
RollbackObject	Событие возникает после отмены изменений, вызванной RollBackObject.
RollbackObjects	Событие возникает после отмены изменений, вызванной RollBackObjects.

ObjectContext — конструктор (IServiceProvider)

Инициализирует новый экземпляр класса ObjectContext с помощью указанного значения.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public ObjectContext(IServiceProvider parentProvider)
```

Параметры

parentProvider

Тип: System.IServiceProvider

В качестве сервис-провайдера parentProvider может выступать компонент карточки, унаследованный от DocsVision.Platform.WinForms.CardControl.

Заметки

Контекст объектов использует тип System.ComponentModel.Design.ServiceContainer для хранения сервисов. При инициализации контекста, в него будут добавлены несколько базовых сервисов.

ObjectContext.IsClosed — свойство

Возвращает признак существования контекста объектов.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public bool IsClosed
```

Значение свойства

Тип: System.Boolean

true — контекст не существует, иначе — false

ObjectContext.UserAccount — свойство

Возвращает название учетной записи текущего пользователя.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public string UserAccount { get; }
```

Значение свойства

Тип: System.String

Учетная запись

ObjectContext.AcceptChanges — метод

Применяет все изменения, которые были сделаны в контексте объектов.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public void AcceptChanges()
```

Заметки

Метод получает все объекты, у которых установлены флаги состояния (ObjectStateFlags) в значения: Modified, Added, Deleted или DeletedFromParent.

ObjectContext.AddObject — метод (ObjectBase)

Добавляет объект в контекст объектов.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public void AddObject(ObjectBase objectInstance)
```

Параметры

objectInstance

Тип: `ObjectBase`

Объект типа `ObjectBase`

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `objectInstance`.

Заметки

Многие методы и сервисы самостоятельно добавляют созданные объекты в контекст — смотрите заметки к методам и сервисам.

При добавлении объекта в контекст объектов, объект становится доступен в рамках данного контекста. Для сохранения такого объекта в базе данных используйте методы `AcceptChanges` и `SaveObject`.

ObjectContext.AddService<T> — метод (T)

Добавляет указанный сервис в контекст объектов.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public void AddService<T>(T service)
```

Параметры типа

T

Интерфейс, реализуемый сервисом

Параметры

service

Тип: `T`

Экземпляр сервиса

Исключения

Исключение	Условие
System.NotSupportedException	Ошибка возвращается, если интерфейс T не входит в список допустимых интерфейсов: IMetadataProvider, IMetadataManager или IPersistentStore.

Заметки

Данный метод предназначен для первоначальной инициализации контекста объектов (см. Инициализация контекста объектов).

ObjectContext.CopyObject<T> — метод (T, ObjectCollection<T>, Action<T>)

Копирует указанный объект в нового владельца.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public T CopyObject<T>(T objectInstance, ObjectCollection<T> newParentCollection, Action
<T> initAction) where T : ObjectBase
```

Параметры типа

T

Тип копируемого объекта, ограничен классом унаследованным от базового типа ObjectBase

Параметры

objectInstance

Тип: T

Экземпляр копируемого объекта

newParentCollection

Тип: ObjectCollection<T>

Коллекция потомков объекта в который производится копирование. В данную коллекцию будет добавлен копируемый objectInstance

initAction

Тип: `System.Action<T>`

Делегат на метод, который будет вызван после создания копии в целевом объекте `newParentCollection`, но до сохранения изменений методом `ObjectContext.AcceptChanges`

Возвращаемое значение

Тип: `T`

Созданная копия объекта `objectInstance`

Заметки

Данный метод применим исключительно при копировании объектов, имеющих родителей: строки справочников. В остальных случаях используйте метод `CardData.Copy` базового API.

ObjectContext.DeleteObject — метод (ObjectBase)

Данный метод используется для удаления заданного объекта.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public void DeleteObject(ObjectBase objectInstance)
```

Параметры

objectInstance

Тип: `ObjectBase`

Удаляемый объект

Исключения

Исключение	Условие
`System.ArgumentNullException`	Ошибка возвращается в случае, если не задан параметр `objectInstance`.

Примеры

```
①

var document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ②

objectContext.DeleteObject(document); ③

objectContext.AcceptChanges(); ④
```

① Инициализация контекста объектов.

② Получение документа с идентификатором `00000000-0000-0000-0000-000000000000`.

③ Удаление объекта.

④ Сохранение изменений в контексте.

ObjectContext.FindObject<T> — метод (QueryObject)

Возвращает первый результат поиска, удовлетворяющий заданным условиям.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public T FindObject<T>(QueryObject query) where T : ObjectBase
```

Параметры типа

T

Тип объекта поиска, унаследованный от `ObjectBase`

Параметры

query

Тип: `QueryObject`

Поисковый запрос

Возвращаемое значение

Тип: `T`

Объект, являющийся результатом поиска

Примеры

В приведенном примере выполняется поиск сотрудника с названием учетной записи "DOMAIN\\IvanovII"

```
①

QueryObject query = new QueryObject(StaffEmployee.AccountNameProperty.Name,
"DOMAIN\\IvanovII"); ②
StaffEmployee staff = context.FindObject<StaffEmployee>(query); ③
```

① Инициализация контекста объектов.

② Формирование поискового запроса по полю `AccountName`.

③ Выполнение запроса.

Если сотрудника с учетной записью `DOMAIN\IvanovII` нет в справочнике, поиск вернет `null`. Можно проверить, добавив скрипт на кнопку:

```
    ObjectContext context = this.CardControl.ObjectContext;
        QueryObject query = new QueryObject(StaffEmployee.AccountNameProperty.Name,
"DOMAIN\\IvanovII");
        StaffEmployee staff = context.FindObject<StaffEmployee>(query);
        if (staff == null)
            MessageBox.Show("Не найдено");
        else
            MessageBox.Show(staff.DisplayString);
```

ObjectContext.FindObjects — метод (QueryObject)

Возвращает список объектов, полученный в результате выполнения переданного запроса.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public IEnumerable<T> FindObjects<T>(QueryObject query) where T : ObjectBase
```

Параметры типа

T

Тип искомого объекта

Параметры

query

Тип: `QueryObject`

Запрос

Возвращаемое значение

Тип: `System.Collections.Generic.IEnumerable<T>`

Коллекция объектов

Примеры

Ниже приведён пример поиска сотрудников компании с именем Иван

```
①

QueryObject query = new QueryObject(StaffEmployee.FirstNameProperty.Name, "Иван"); ②

IEnumerable<StaffEmployee> employees = objectContext.FindObjects<StaffEmployee>(query);
③
```

① Инициализация контекста объектов.

② Формирование поискового запроса по полю `FirstName`.

③ Выполнение запроса.

ObjectContext.GetObject — метод (IObjectRef, IDictionary<String, Object>)

Получает объект по переданной ссылке. Также передаются дополнительные параметры получения объекта.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public ObjectBase GetObject(IObjectRef objectRef, IDictionary<string, object> parameters)
```

Параметры

objectRef

Тип: `IObjectRef`

Ссылка на объект

parameters

Тип: `System.Collections.Generic.IDictionary<String, Object>`

Дополнительные параметры, используемые при загрузке в контекст объектов отсутствующего объекта

Возвращаемое значение

Тип: `ObjectBase`

Искомый объект

Заметки

Параметр `parameters` может быть использован для изменения данных загружаемой карточки, до её непосредственной загрузки в контекст. Если объект был загружен в контекст ранее, то передача параметра `parameters` не возымеет эффекта.

Примеры

Ниже приведён пример использования перегрузки метода GetObject с целью подмены примечания к карточке Документ

```
①

IObjectRef objectRef = objectContext.GetObjectRef(typeof(Document), new Guid("00000000-
0000-0000-0000-000000000000"));  ②

IDictionary<string, object> properties = new Dictionary<string, object>();
properties.Add(Document.DescriptionProperty.Name, "OverrideDescription"); ③

Document document = (Document)objectContext.GetObject(objectRef, properties); ④

Console.WriteLine(document.Description); ⑤
```

① Инициализация контекста объектов.

② Получение ссылки на объект.

③ Установка дополнительных параметров загрузки.

④ Получение документа.

⑤ Вывод изменённого примечания.

ObjectContext.GetObject<T> — метод (Object)

Получает объект заданного типа с указанным идентификатором.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public T GetObject<T>(object objectId) where T : ObjectBase
```

Параметры типа

T

Тип искомого объекта, унаследованный от базового класса `ObjectBase`

Параметры

objectId

Тип: `System.Object`

Идентификатор искомого объекта

Возвращаемое значение

Тип: `T`

Искомый объект

Заметки

В качестве идентификатора `objectId` должен выступать объект типа `Guid`.

Примеры

```
①

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ②
```

① Инициализация контекста объектов.

② Получение объекта типа Документ (Document) с идентификатором `00000000-0000-0000-0000-000000000000`.

ObjectContext.GetService<T> — метод

Возвращает *сервис*, реализующий переданный интерфейс.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public T GetService<T>()
```

Параметры типа

T

Интерфейс, реализованный *сервисом*

Возвращаемое значение

Тип: `T`

Сервис

Исключения

Исключение	Условие
`System.InvalidOperationException`	Ошибка возвращается в случае, если сервис не зарегистрирован.

Примеры

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②
```

① Инициализация контекста объектов.

② Получение сервиса для работы с документами.

ObjectContext.LockObject — метод (IObjectRef, Boolean)

Блокирует объект по переданной ссылке.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public bool LockObject(IObjectRef objectRef, bool permanently)
```

Параметры

objectRef

Тип: IObjectRef

Ссылка на объект

permanently

Тип: System.Boolean

true — постоянная блокировка; false — временная блокировка

Возвращаемое значение

Тип: System.Boolean

true — блокировка выполнена, иначе — false

ObjectContext.LockObject — метод (ObjectBase, Boolean)

Блокирует переданный объект.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public bool LockObject(ObjectBase objectInstance, bool permanently)
```

Параметры

objectInstance

Тип: ObjectBase

Объект, требующий блокировки

permanently

Тип: `System.Boolean`

`true` — постоянная блокировка; `false` — временная блокировка

Возвращаемое значение

Тип: `System.Boolean`

`true` — блокировка выполнена, иначе — `false`

ObjectContext.RefreshObject — метод (IObjectRef)

Метод обновляет корневой объект.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public void RefreshObject(IObjectRef objectRef)
```

Примеры

Данный пример демонстрирует вызов метода с `ref`:

```
ObjectContext.RefreshObject(ref document);
```

ObjectContext.RollbackChanges — метод

Отменяет все изменения в контексте объектов, которые не были сохранены методом `AcceptChanges`, `SaveObject<T>(T)` или `SaveObjects<T>(IEnumerable<T>)`.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public void RollbackChanges()
```

Заметки

Отменяются изменения объектов, имеющих следующие флаги в состояниях (`ObjectStateFlags`): `ObjectStateFlags.Modified`, `ObjectStateFlags.Added` или `ObjectStateFlags.Deleted`. Перед непосредственной отменой изменений, метод взывает событие `ObjectContext.BeforeRollback`, которое может быть использовано для прекращения процедуры отката.

После вызова `ObjectContext.AcceptChanges` отмена принятых изменений будет невозможна.

Примеры

В качестве примера приведён код получения двух документов, в каждый из которых вносится определённое изменение. Изменения могут быть отменены при проверке флага `roolback`

```
①

Document documentA = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000"));
documentA.Description = "Описание не корректно"; ②

Document documentB = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000001"));
documentB.MainInfo.Name = "Название не корректно"; ③

if (roolback) ④
{

 objectContext.RollbackChanges(); ⑤
}
```

① Инициализация контекста объектов.

② Получение и изменение документа.

③ Флаг разрешающий откат изменений.

④ Отмена всех изменений.

ObjectContext.RollBackObject — метод (ObjectBase)

Отменяет несохранённые изменения в заданном объекте.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`

- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public IObjectRef RollBackObject(ObjectBase objectInstance)
```

Параметры

objectInstance

Тип: ObjectBase

Объект, у которого отменяются изменения

Возвращаемое значение

Тип: IObjectRef

Ссылка на объект в *контексте объектов*

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр objectInstance.

Примеры

```
①

Document document = objectContext.GetObject<Document>(new Guid("00000000-0000-0000-0000-
000000000000")); ②
document.Description = "Описание не корректно";

if (roolback) ③
{

 objectContext.RollBackObject(document); ④
}
```

① Инициализация контекста объектов.

② Получение и изменение документа.

③ Флаг разрешающий откат изменений.

④ Отмена изменений в документе.

ObjectContext.SaveObject — метод (ObjectBase)

Сохраняет изменения выбранного объекта.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public IObjectRef SaveObject(ObjectBase objectInstance)
```

Параметры

objectInstance

Тип: ObjectBase

Сохраняемый объект

Возвращаемое значение

Тип: IObjectRef

Ссылка на сохранённый объект

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр objectInstance.

ObjectContext.SaveObject<T> — метод (T)

Сохраняет изменения выбранного объекта.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public ObjectRef<T> SaveObject<T>(T objectInstance) where T : ObjectBase
```

Параметры типа

T

Тип объекта

Параметры

objectInstance

Тип: T

Сохраняемый объект

Возвращаемое значение

Тип: ObjectRef<T>

Ссылка на объект в *контексте объектов*

Исключения

Исключение	Условие
System.ArgumentNullException	Ошибка возвращается в случае, если не задан параметр objectInstance.

Примеры

Ниже приведён пример создания и сохранения документа

```
①

IDocumentService documentService = objectContext.GetService<IDocumentService>(); ②

Document document = documentService.CreateDocument(); ③

document.Description = "Документ созданный из кода"; ④

objectContext.SaveObject<Document>(document); ⑤
```

① Инициализация контекста объектов.

② Получение сервиса для работы с документами.

③ Создание документа.

④ Изменение описания.

⑤ Сохранение изменений в контексте.

ObjectContext.UnlockObject — метод (ObjectBase)

Снимает с объекта собственную блокировку.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public void UnlockObject(ObjectBase objectInstance)
```

Параметры

objectInstance

Тип: ObjectBase

Объект, с которого снимается блокировка

Примеры

Приведенный далее пример демонстрирует снятие с карточки собственной блокировки.

```
①

BaseCard card = objectContext.GetObject<BaseCard>(cardId); ②

if(objectContext.GetObjectLockInfo(card).Status = ObjectLockStatus.OwnerLocked) ③
{
 objectContext.UnlockObject(card); ④
}
```

① Инициализация контекста объектов.

② Получаем карточку, которую требуется разблокировать.

③ Проверяем наличие блокировки и то, что блокировка установлена владельцем сессии.

④ Снимаем блокировку.

ObjectHelper — класс

Содержит метод получения родительского объекта.

- **Пространство имён:** DocsVision.Platform.ObjectModel

- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public static class ObjectHelper
```

Методы

Имя	Описание
GetParent<T>(Object)	Возвращает объект, который содержит указанный дочерний объект.

ObjectHelper.GetParent<T> — метод (Object)

Возвращает объект, который содержит указанный дочерний объект.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public static T GetParent<T>(object objectInstance)
```

Параметры типа

T

Тип родительского объекта

Параметры

objectInstance

Тип: System.Object

Дочерний объект

Возвращаемое значение

Тип: T

Родительский объект

Примеры

приведён пример использования метода GetObject для получения узла

конструктора справочников, содержащего строку с определённым идентификатором

```
①

BaseUniversalItem baseUniversalItem = objectContext.GetObject<BaseUniversalItem>(new
Guid("00000000-0000-0000-0000-000000000000")); ②

BaseUniversalItemType baseUniversalItemType = ObjectHelper.GetParent
<BaseUniversalItemType>(baseUniversalItem); ③

Guid id = objectContext.GetObjectRef<BaseUniversalItemType>(baseUniversalItemType).Id; ④
```

① Инициализация контекста объектов.

② Получение строки с идентификатором `00000000-0000-0000-0000-000000000000`.

③ Получение узла справочника в котором храниться искомая строка.

④ Получение идентификатора узла.

ObjectLockInfo — класс

Содержит информацию о блокировке объекта.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public class ObjectLockInfo
```

Конструкторы

Имя	Описание
`ObjectLockInfo(ObjectLockStatus)`	Инициализирует новый экземпляр класса `ObjectLockInfo` с помощью указанного значения.
`ObjectLockInfo(ObjectLockStatus, String)`	Инициализирует новый экземпляр класса `ObjectLockInfo` с помощью указанного значения.

Свойства

Имя	Описание
`Owner`	Возвращает владельца блокировки.
`Status`	Возвращает статус блокировки объекта.

ObjectModelException — класс

Представляет общее исключение объектной модели.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public class ObjectModelException : Exception
```

Конструкторы

Имя	Описание
`ObjectModelException()`	Инициализирует новый экземпляр класса ObjectModelException.
`ObjectModelException(String)`	Инициализирует новый экземпляр класса `ObjectModelException` с помощью указанного значения.
`ObjectModelException(String, Exception)`	Инициализирует новый экземпляр класса `ObjectModelException` с помощью указанного значения.

ObjectProperty — класс

Предоставляет методы для работы со свойствами объектной модели.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public sealed class ObjectProperty
```

Свойства

Имя	Описание
`DefaultValue`	Возвращает значение свойства по умолчанию.
`Name`	Возвращает название свойства.
`OwnerType`	Возвращает тип владельца свойства.
`ParentLink`	Возвращает `true`, если свойство ссылается на родителя.
`PropertyType`	Определяет тип хранимого свойства.
`ReadOnly`	Возвращает `true`, если свойство доступно только на чтение.

Методы

Имя	Описание
`GetHashCode`	В качестве хеш-кода возвращается индекс данного свойства в статической коллекции свойств.
`GetRegistered(Type)`	Возвращает массив всех зарегистрированных свойств для указанного типа владельца.
`GetRegistered(Type, Boolean)`	Возвращает массив всех зарегистрированных свойств для указанного типа владельца, с учетом наследования классов.
`Register(String, Type, Type)`	Статический метод регистрации свойств.
`Register(String, Type, Type, ObjectPropertyOptions, Object)`	Регистрирует свойство с указанием дополнительных параметров и значения по умолчанию.

Имя	Описание
ToString	Возвращает строку содержащую название свойства и название класса владельца свойства.

ObjectProperty.Register — метод (String, Type, Type)

Статический метод регистрации свойств.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public static ObjectProperty Register(string name, Type propertyType, Type ownerType)
```

Параметры

name

Тип: System.String

Псевдоним свойства

propertyType

Тип: System.Type

Тип регистрируемого свойства. Может быть как стандартным, например System.Guid, так и типом объектной модели секции карточки.

ownerType

Тип: System.Type

Тип владельца свойства. Как правило, класс, в котором осуществляется регистрация свойства.

Возвращаемое значение

Тип: ObjectProperty

Зарегистрированное свойство

ObjectPropertyValue — класс

Предоставляет значение для коллекции `ObjectPropertyValueCollection`.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public sealed class ObjectPropertyValue
```

Конструкторы

Имя	Описание
`ObjectPropertyValue()`	Инициализирует новый экземпляр класса `ObjectPropertyValue`.
`ObjectPropertyValue(String, Object)`	Инициализирует новый экземпляр класса `ObjectPropertyValue` с помощью указанного значения.

Свойства

Имя	Описание
`Name`	Задаёт или возвращает название свойства.
`Value`	Задаёт или возвращает значение свойства.

ObjectPropertyValueCollection — класс

Используется некоторыми преобразователями данных для хранения значений.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public class ObjectPropertyValueCollection : KeyedCollection<string, ObjectPropertyValue>
```

Конструкторы

Имя	Описание
`ObjectPropertyValueCollection()`	Инициализирует новый экземпляр класса `ObjectPropertyValueCollection`.

Свойства

Имя	Описание
`Add(ObjectProperty, IObjectRef)`	Добавляет свойство в коллекцию свойств.
`Add(ObjectProperty, Object)`	Добавляет свойство в коллекцию свойств.
`Add(ObjectProperty, ObjectPropertyAccessorDelegate)`	Добавляет свойство в коллекцию свойств.
`GetKeyForItem(ObjectPropertyValue)`	Возвращает ключ по значению.

ObjectRef — класс

Содержит ссылку на объект в *контексте объектов*.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public class ObjectRef<T> : IObjectRef where T : ObjectBase
```

Конструкторы

Имя	Описание
`ObjectRef(Guid)`	Создаёт новый экземпляр класса `ObjectRef` с заданным значением.

Свойства

Имя	Описание
`Id`	Возвращает идентификатор объекта.
`Type`	Возвращает тип объекта.

Методы

Имя	Описание
`Equals(Object)`	Сравнивает текущий объект с заданным.
`GetHashCode`	Возвращает хэш объекта. В качестве основы используется идентификатор объекта.
`ToString`	Возвращает строковое представление названия типа объекта и его идентификатора.

ObjectState — класс

Позволяет получать флаги состояния объекта.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public sealed class ObjectState
```

Конструкторы

Имя	Описание
`ObjectState(ObjectContext, ObjectBase, IObjectRef)`	Инициализирует новый экземпляр класса `ObjectState` с помощью указанного значения.

Свойства

Имя	Описание
`Context`	Возвращает контекст объектов.
`Flags`	Возвращает флаги состояния объекта.
`Instance`	Возвращает экземпляр объекта, которым инициализирован экземпляр `ObjectState`.

Имя	Описание
`Reference`	Возвращает ссылку для экземпляра объекта.
`TemporaryLocked`	Возвращает статус временной блокировки.
`Timestamp`	Возвращает отметку, по которой можно определить наличие новой версии объекта.

Методы

Имя	Описание
`ToString`	Возвращает строку сформированную из флагов и ссылки на объект.

ServiceFactory — класс

Управляет получением сервиса из фабрики сервисов.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public abstract class ServiceFactory : IServiceFactory
```

Конструкторы

Имя	Описание
`ServiceFactory()`	Инициализирует новый экземпляр класса `ServiceFactory`.

Методы

Имя	Описание
`GetService(Type)`	Возвращает *сервис* указанного типа.

Примеры

Ниже приведён пример создания фабрики сервисов, в котором переопределяется метод `GetService`, создающий сервис определённого типа

```
public class TestWorkLibServiceFactory : ServiceFactory
{
 protected override object GetService(System.Type serviceType)
 {
  if (serviceType = typeof(ITestWorkService)) ①
  {
   return new TestWorkService(); ②
  }
  return null;
 }
}
```

① Интерфейс сервиса, предоставляемого фабрикой.

② Возвращение экземпляра класса реализующего соответствующий интерфейс.

StaticObjectContext — класс

Предоставляет статические члены создания и получения контекста объектов.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public sealed class StaticObjectContext
```

Свойства

Имя	Описание
`Current`	Задаёт или возвращает статическое значение контекста объектов.
`ObjectContext`	Возвращает контекст объектов.

Методы

Имя	Описание
`Create`	Инициализирует контекст с пустым сервис-провайдером.
`Create(IServiceProvider)`	Инициализирует контекст с указанным сервис-провайдером.

Примеры

Инициализация статического контекста объектов во многом аналогична инициализации контекста объектов, пример которой приведён в примере Инициализация контекста объектов

```
private static void CreateStaticContext()
{
 var sessionContainer = new System.ComponentModel.Design.ServiceContainer(); ①
 sessionContainer.AddService(typeof(DocsVision.Platform.ObjectManager.UserSession),
userSession);

 StaticObjectContext.Create(sessionContainer); ②

 BackOffice.Register(StaticObjectContext.Current); ③

 var serviceFactoryRegistry = StaticObjectContext.Current.GetService
<IServiceFactoryRegistry>();
 serviceFactoryRegistry.RegisterFactory(typeof(BackOfficeServiceFactory));
 serviceFactoryRegistry.RegisterFactory(typeof(SystemCardsServiceFactory)); ④

 StaticObjectContext.Current.AddService<IPersistentStore>(DocsVisionObjectFactory.CreateP
ersistentStore(new SessionProvider(userSession), null)); ⑤

 IMetadataProvider metadataProvider = DocsVisionObjectFactory.CreateMetadataProvider
(userSession); ⑥
 StaticObjectContext.Current.AddService<IMetadataManager>(DocsVisionObjectFactory.CreateM
etadataManager(metadataProvider, userSession));
 StaticObjectContext.Current.AddService<IMetadataProvider>(metadataProvider);
}
```

① Инициализация сервис-провайдера.

② Создание контекста объектов.

③ Регистрация базовых (`SystemCardsMapperFactory`, `BackOfficeMapperFactory`) фабрик преобразователей данных.

④ Регистрация базовых фабрик сервисов.

⑤ Регистрация сервиса для работы с хранилищем Docsvision.

⑥ Регистрация поставщика метаданных карточек.

Сессия пользователя (`userSession`) может быть получена по аналогии с примером Инициализация контекста объектов

ILockManager — интерфейс

Представляет менеджер блокировки.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public interface ILockManager
```

Методы

Имя	Описание
`GetObjectLockInfo(IObjectRef)`	При переопределении в производном классе возвращает информацию о блокировке объекта.
`GetObjectLockInfo(ObjectBase)`	При переопределении в производном классе возвращает информацию о блокировке объекта.
`LockObject(IObjectRef, Boolean)`	Блокирует объект в хранилище, запрещая редактирование другими пользователями.
`LockObject(ObjectBase, Boolean)`	Блокирует объект в хранилище, запрещая редактирование другими пользователями.
`UnlockObject(IObjectRef)`	Снимает блокировку с объекта.
`UnlockObject(ObjectBase)`	Снимает блокировку с объекта.

Заметки

Интерфейс реализован в классе контекста объектов `DocsVision.Platform.ObjectModel.ObjectContext`.

IObjectContext — интерфейс

Описывает ключевую функциональность класса `ObjectContext` по осуществлению наиболее общих действий над объектами системы.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public interface IObjectContext
```

Свойства

Имя	Описание
`IsClosed`	Возвращает состояние контекста объектов.
`UserAccount`	Возвращает имя учетной записи текущего пользователя.

Методы

Имя	Описание
`AcceptChanges`	При переопределении в производном классе сохраняет все произведенные изменения в контексте объектов.
`AddObject(ObjectBase)`	При переопределении в производном классе создаёт новый объект в контексте.
`DeleteObject(ObjectBase)`	При переопределении в производном классе удаляет заданный объект.
`GetObject(IObjectRef)`	При переопределении в производном классе получает объект из переданной ссылки.
`GetObject<T>(Object)`	При переопределении в производном классе получает объект заданного типа с указанным идентификатором.

Имя	Описание
`GetObject<T>(ObjectRef<T>)`	При переопределении в производном классе получает объект заданного типа из переданной ссылки.
`GetObjectRef<T>(Object)`	При переопределении в производном классе получает ссылку на объект заданного типа с указанным идентификатором.
`GetObjectRef(ObjectBase)`	При переопределении в производном классе получает ссылку на объект.
`RollbackChanges`	При переопределении в производном классе отменяет произведенные несохранённые изменения во всех объектах контекста.
`SaveObject<T>(T)`	При переопределении в производном классе сохраняет изменения в указанном объекте.
`SaveObjects<T>(IEnumerable<T>)`	При переопределении в производном классе сохраняет изменения в заданной коллекции объектов.

IObjectContext.IsClosed — свойство

Возвращает состояние контекста объектов.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
bool IsClosed { get; }
```

Значение свойства

Тип: `System.Boolean`

`true` — контекст недоступен, иначе — `false`

Заметки

При попытке прочитать свойство не инициализированного объекта вернет ошибку. При закрытом соединении вернет `true`.

IObjectContext.UserAccount — свойство

Возвращает имя учетной записи текущего пользователя.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
string UserAccount { get; }
```

Значение свойства

Тип: `System.String`

Учетная запись

IObjectContext.AcceptChanges — метод

При переопределении в производном классе сохраняет все произведенные изменения в контексте объектов.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
void AcceptChanges()
```

Заметки

До сохранения изменений невозможно получить идентификатор созданного в контексте объекта.

IObjectContext.AddObject — метод (ObjectBase)

При переопределении в производном классе создаёт новый объект в контексте.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`

- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
void AddObject(ObjectBase objectInstance)
```

Параметры

objectInstance

Тип: ObjectBase

Экземпляр создаваемого объекта

Примеры

В примере создаётся документ с текстом дайджеста "Документ созданный из кода". Созданный документ добавляется в контекст, контекст сохраняется. После сохранения контекста, документ доступен в Windows-клиенте (в разделе *Cards/Базовые объекты/Документ*)

```
ObjectContext objectContext = new ObjectContext(this);

IDocumentService docService = objectContext.GetService<IDocumentService>(); ①

Document document = docService.CreateDocument(); ;
document.Description = "Документ созданный из кода"; ②

objectContext.AddObject(document); ③

objectContext.AcceptChanges(); ④
```

① Получение сервиса для работы с документами.

② Создание документа.

③ Добавление документа в контекст.

④ Сохранение изменений в контексте.

IObjectRef — интерфейс

Определяет наличие у объекта уникального идентификатора и типа.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public interface IObjectRef
```

Свойства

Имя	Описание
Id	Возвращает уникальный идентификатор объекта.
Type	Возвращает тип объекта.

IObjectStateService — интерфейс

Представляет интерфейс сервиса состояний объектов.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public interface IObjectStateService
```

Методы

Имя	Описание
GetModifiedObjects(ObjectBase)	Возвращает массив изменённых объектов контекста.
GetObjectState(IObjectRef)	Возвращает статус для переданной ссылки на объект.
GetObjectState(ObjectBase)	Возвращает статус для заданного объекта.
HasChangedObjects	Получает статус наличия изменённых объектов в контексте.
IsObjectChanged(ObjectBase)	Определяет, является ли переданный объект изменённым.

События

Имя	Описание
ObjectAdded	Событие возникает при добавлении объекта в контекст объектов.
ObjectDeleted	Событие возникает при удалении объекта из контекста объектов.
ObjectModified	Событие возникает при изменении объекта в контексте объектов.

Заметки

События сигнализируют об изменениях произошедших в текущем контексте объектов, т.е. если изменения в карточке произведены в другом контексте, то событие не будет вызвано.

IServiceFactoryRegistry — интерфейс

Интерфейс реестра фабрик сервисов.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public interface IServiceFactoryRegistry
```

Методы

Имя	Описание
RegisterFactory(Type)	При переопределении в производном классе регистрирует фабрику заданного типа.

ObjectAccessControlType — перечисление

Определяет тип объекта доступа.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
public enum ObjectAccessControlType
```

Члены

Имя члена	Описание
`Card`	Карточка.
`Row`	Строка.

ObjectLockStatus — перечисление

Определяет статус блокировки объекта.

- **Пространство имён:** `DocsVision.Platform.ObjectModel`
- **Сборка:** `DocsVision.Platform.ObjectModel.dll`

Синтаксис

```
public enum ObjectLockStatus
```

Члены

Имя члена	Описание
`Free`	Объект не заблокирован.
`Locked`	Объект заблокирован на время сессии.
`CheckedOut`	Объект заблокирован (редактируется).
`OwnerLocked`	Объект временно заблокирован текущим пользователем.
`OwnerCheckedOut`	Объект заблокирован текущим пользователем (редактируется).

ObjectPropertyOptions — перечисление

Определяет список флагов, используемых свойствами объектной модели.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
[Flags]
public enum ObjectPropertyOptions
```

Члены

Имя члена	Описание
None	Не задан. Значение 0x00.
ReadOnly	Только чтение. Значение 0x01.
ParentLink	Свойство ссылается на родителя. Значение 0x02.

ObjectStateFlags — перечисление

Определяет состояние объекта.

Это перечисление имеет атрибут FlagsAttribute, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** DocsVision.Platform.ObjectModel
- **Сборка:** DocsVision.Platform.ObjectModel.dll

Синтаксис

```
[Flags]
public enum ObjectStateFlags
```

Члены

Имя члена	Описание
None	Флаг не установлен. Значение 0x00.
Modified	Объект был модифицирован. Значение 0x01.
Added	Объект был добавлен. Значение 0x02.

Имя члена	Описание
`Deleted`	Объект был удалён. Значение `0x04`.
`Attached`	Объект был прикреплен к другому. Значение `0x08`.
`DeletedFromParent`	Объект был удалён у родителя. Значение `0x10`.

Заметки

Данные флаги проверяются при сохранении контекста объектов, для проверки необходимости сохранения изменений у конкретного объекта.

DocsVision.Platform.Security — пространство имён

В пространстве имён `DocsVision.Platform.Security` содержаться объекты связанные с настройкой и реализацией модели безопасности и криптографии Docsvision.

Пространства имён

Пространство имён	Описание
`AccessControl`	Содержит классы связанные с реализацией модели безопасности Docsvision.

DocsVision.Platform.Security.AccessControl — пространство имён

Содержит классы связанные с реализацией модели безопасности Docsvision.

Классы

Класс	Описание
`CardDataAccessRule`	Представляет разрешение на карточку.
`CardDataAuditRule`	Описывает правило аудита доступа к карточке.
`CardDataSecurity`	Представляет описатель прав карточки.
`DVObjectSecurity`	Базовый класс для работы с правами доступа к объектам.
`FileDataAccessRule`	Описывает правило доступа к простому файлу.

Класс	Описание
FileDataAuditRule	Описывает правило аудита доступа к простому файлу.
FileDataSecurity	Реализует описатель прав на файл.
ReportAccessRule	Описывает правило доступа к хранимой процедуре.
ReportAuditRule	Описывает правило аудита доступа к хранимой процедуре.
ReportSecurity	Права на хранимую процедуру.
SecurityContext	Содержит информацию о контексте безопасности пользователя, прошедшего проверку подлинности.
SecurityManager	Реализует менеджер управления безопасностью.

Структуры

Структура	Описание
GenericMapping	Обеспечивает сопоставление общих и специфических прав доступа.

Интерфейсы

Интерфейс	Описание
IPickerExtension	Позволяет использовать расширение пикер (интерфейс выбора учетных записей), осуществлять поиск и вызывать диалог выбора учетных записей.
IPickerExtensionProvider	Позволяет выбирать расширение пикер.
ISecureObject	Позволяет получить доступ к параметрам безопасности объекта.
ISecurityContextExtension	Позволяет работать с контекстом безопасности.

Перечисления

Перечисление	Описание
AccessControlPropagation	Определяет, каким образом происходит наследование прав доступа.
CardDataRights	Определяет возможные права на карточку.
FileDataRights	Определяет возможные права на файл.
MandatoryAccessLevel	Уровни доступа мандатной модели доступа.
MandatoryAccessRights	Права доступа для участников с уровнем мандата ниже требуемого.
ReportRights	Определяет возможные права на хранимую процедуру.
SecureObjectType	Определяет тип объекта безопасности.

CardDataAccessRule — класс

Представляет разрешение на карточку. В текущей версии Docsvision отсутствует System.Security.Principal.NTAccount, как и API, связанный с дискреционной безопасностью, т.к. это исключительно Windows-специфический код. GetAccessRules можно использовать, но третьим параметром нужно передавать тип DocsVision.Platform.Security.AccessControl.ISecurityIdentifier (это интерфейс над Windows и Linux-специфичными реализациями дискреционной безопасности). В dv6 вся дискреционная безопасность стала чуть более абстрактной по сравнению с Docsvision 5 из-за необходимости поддержания как Windows-, так и Linux-реализаций.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
internal sealed class CardDataAccessRule : ObjectAccessRule, ICardDataAccessRule
```

Конструкторы

Имя	Описание
`CardDataAccessRule(String, CardDataRights, AccessControlType)`	Инициализирует новый экземпляр класса `CardDataAccessRule` с требуемыми правами на указанного пользователя.

Свойства

Имя	Описание
`AccessRights`	Возвращает объект описывающий право на карточку.

CardDataSecurity — класс

Представляет описатель прав карточки.

- **Пространство имён:** `DocsVision.Platform.Security.AccessControl`
- **Сборка:** `DocsVision.Platform.dll`

Синтаксис

```
internal sealed class CardDataSecurity : DVObjectSecurity, ICardDataSecurity
```

Конструкторы

Имя	Описание
`CardDataSecurity()`	Инициализирует новый экземпляр класса CardDataSecurity.

Свойства

Имя	Описание
`AccessRightType`	Возвращает тип объекта-описателя прав.
`AccessRuleType`	Возвращает тип объекта-разрешения.
`AuditRuleType`	Возвращает тип объекта аудита.

Методы

Имя	Описание
AccessRuleFactory(IdentityReference, Int32, Boolean, InheritanceFlags, PropagationFlags, AccessControlType)	Создание нового разрешения на объект.
AddAccessRule(CardDataAccessRule)	Добавляет новое разрешение на карточку.
AddAuditRule(CardDataAuditRule)	Добавляет аудит на объект.
AuditRuleFactory(IdentityReference, Int32, Boolean, InheritanceFlags, PropagationFlags, AuditFlags)	Создание нового аудита.
RemoveAccessRule(CardDataAccessRule)	Удаляет разрешение из описателя прав.
RemoveAccessRuleAll(CardDataAccessRule)	Удаляет все разрешения для данного субъекта из описателя прав.
RemoveAccessRuleSpecific(CardDataAccessRule)	Удаляет специфическое разрешение из описателя прав (учитываются все признаки разрешения).
RemoveAuditRule(CardDataAuditRule)	Удаляет конкретный аудит из описателя прав.
RemoveAuditRuleAll(CardDataAuditRule)	Удаляет все правила аудита для данного субъекта из описателя прав.
RemoveAuditRuleSpecific(CardDataAuditRule)	Удаляет специфический аудит из описателя прав (учитываются все признаки аудита).
ResetAccessRule(CardDataAccessRule)	Удаляет все не унаследованные разрешения объекта, и добавляет одно новое — переданное в качестве параметра
SetAccessRule(CardDataAccessRule)	Удаляет из описателя прав все разрешения для данного субъекта, и добавляет вместо них одно новое — переданное в качестве параметра.

Имя	Описание
SetAuditRule(CardDataAuditRule)	Удаляет из описателя прав все аудиты для данного субъекта, и добавляет вместо них одно новое — переданное в качестве параметра.

CardDataSecurity — конструктор

Инициализирует новый экземпляр класса CardDataSecurity.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public CardDataSecurity()
```

CardDataSecurity.AddAccessRule — метод (CardDataAccessRule)

Добавляет новое разрешение на карточку.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public void AddAccessRule(CardDataAccessRule rule)
```

Параметры

rule

Тип: CardDataAccessRule

Правило доступа

CardDataSecurity.RemoveAccessRuleSpecific — метод (CardDataAccessRule)

Удаляет специфическое разрешение из описателя прав (учитываются все признаки разрешения).

- **Пространство имён:** DocsVision.Platform.Security.AccessControl

- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public void RemoveAccessRule(CardDataAccessRule rule)
```

Параметры

rule

Тип: CardDataAccessRule

Правило доступа

CardDataSecurity.SetAccessRule — метод (CardDataAccessRule)

Удаляет из описателя прав все разрешения для данного субъекта, и добавляет вместо них одно новое — переданное в качестве параметра.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public void SetAccessRule(CardDataAccessRule rule)
```

Параметры

rule

Тип: CardDataAccessRule

Правило доступа

DVObjectSecurity — класс

Базовый класс для работы с правами доступа к объектам.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public abstract class DVObjectSecurity : DirectoryObjectSecurity
```

Свойства

Имя	Описание
ObjectType	Возвращает тип объекта безопасности.

DVObjectSecurity.ObjectType — свойство

Возвращает тип объекта безопасности.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public SecureObjectType ObjectType { get; }
```

Значение свойства

Тип: SecureObjectType

Тип объекта безопасности

ReportSecurity — класс

Права на хранимую процедуру.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public sealed class ReportSecurity : DVObjectSecurity
```

Свойства

Имя	Описание
AccessRightType	Возвращает тип объекта-описателя прав.
AccessRuleType	Возвращает тип объекта-разрешения.
AuditRuleType	Возвращает тип объекта аудита.

Методы

Имя	Описание
AddAccessRule(ReportAccessRule)	Добавляет новое разрешение на хранимую процедуру.

GenericMapping — структура

Обеспечивает сопоставление общих и специфических прав доступа.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public struct GenericMapping : IEquatable<GenericMapping>
```

Методы

Имя	Описание
Equals(GenericMapping)	Сравнивает значение текущих прав доступа и заданных.

Операторы

Имя	Описание
Equality	Определяет, равняются ли значения двух объектов содержащих права доступа.
Inequality	Определяет, различаются ли значения двух объектов содержащих права доступа.

Поля

Имя	Описание
GenericAll	Значение полного права доступа.
GenericExecute	Значение права доступа на исполнение.

Имя	Описание
GenericRead	Значение права доступа на чтение.
GenericWrite	Значение права доступа на запись.

Заметки

Значение GenericAll, GenericExecute, GenericRead и GenericWrite может различаться для разных типов объектов, на этом и основана процедура сравнения.

GenericMapping.Equals — метод (GenericMapping)

Сравнивает значение текущих прав доступа и заданных.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public bool Equals(GenericMapping other)
```

Параметры

other

Тип: GenericMapping

Объект для сравнения

Возвращаемое значение

Тип: Boolean

true — уровни доступа эквивалентны; false — значения различаются

Заметки

Метод сравнивает значение полей двух структур.

GenericMapping.Equality — оператор

Определяет, равняются ли значения двух объектов содержащих права доступа.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public static bool operator ==(
        GenericMapping left,
        GenericMapping right)
```

Параметры

left

Тип: `GenericMapping`

Первый объект

right

Тип: `GenericMapping`

Второй объект

Возвращаемое значение

Тип: `Boolean`

`true` — уровни доступа эквивалентны; `false` — значения различаются

GenericMapping.Inequality — оператор

Определяет, различаются ли значения двух объектов содержащих права доступа.

- **Пространство имён:** `DocsVision.Platform.Security.AccessControl`
- **Сборка:** `DocsVision.Platform.dll`

Синтаксис

```
public static bool operator !=(
        GenericMapping left,
        GenericMapping right)
```

Параметры

left

Тип: `GenericMapping`

Первый объект

right

Тип: `GenericMapping`

Второй объект

Возвращаемое значение

Тип: `Boolean`

`true` — значения различаются; `false` — значения эквивалентны

GenericMapping.GenericAll — поле

Значение полного права доступа.

- **Пространство имён:** `DocsVision.Platform.Security.AccessControl`
- **Сборка:** `DocsVision.Platform.dll`

Синтаксис

```
public int GenericAll
```

GenericMapping.GenericExecute — поле

Значение права доступа на исполнение.

- **Пространство имён:** `DocsVision.Platform.Security.AccessControl`
- **Сборка:** `DocsVision.Platform.dll`

Синтаксис

```
public int GenericExecute
```

GenericMapping.GenericRead — поле

Значение права доступа на чтение.

- **Пространство имён:** `DocsVision.Platform.Security.AccessControl`
- **Сборка:** `DocsVision.Platform.dll`

Синтаксис

```
public int GenericRead
```

GenericMapping.GenericWrite — поле

Значение права доступа на запись.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public int GenericWrite
```

ISecureObject — интерфейс

Позволяет получить доступ к параметрам безопасности объекта.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public interface ISecureObject
```

Свойства

Имя	Описание
FullName	Возвращает полное имя объекта.
Id	Возвращает идентификатор объекта.
Name	Возвращает имя/название объекта.
Parent	Возвращает родительский объект безопасности.
SecurityClassId	Возвращает идентификатор класса безопасности объекта.

Методы

Имя	Описание
`AccessCheck(Int32, Int32)`	Проверяет наличие заданных прав и позволяет получить существующие права.
`AccessCheck(Guid, Int32, Int32)`	Проверяет наличие заданных прав для указанного типа объектов и позволяет получить существующие права.
`GetAccessControl(AccessControlSections)`	Возвращает параметры безопасности для объекта.
`SetAccessControl(Byte[], AccessControlSections, AccessControlPropagation)`	Устанавливает параметры безопасности для объекта.

AccessControlPropagation — перечисление

Определяет, каким образом происходит наследование прав доступа.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.Security.AccessControl`
- **Сборка:** `DocsVision.Platform.dll`

Синтаксис

```
[Flags]
public enum AccessControlPropagation
```

Члены

Имя члена	Описание
`None`	Не учитывать права родителя. Значение `0x0`.
`OwnerRecurse`	Рекурсивное наследование прав.
`ResetDaclTree`	Не учитывать список `DACL`.
`ResetSaclTree`	Не учитывать список `SACL`.
`All`	Учитывать все права родителя.

Заметки

На текущий момент процедура аудита не реализована.

CardDataRights — перечисление

Определяет возможные права на карточку.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.Security.AccessControl`
- **Сборка:** `DocsVision.Platform.dll`

Синтаксис

```
[Flags]
public enum CardDataRights
```

Члены

Имя члена	Описание
`GenericRead`	Простое чтение. Значение `0x80000000`.
`GenericWrite`	Простая запись. Значение `0x40000000`.
`GenericExecute`	Простое выполнение. Значение `0x20000000`.
`GenericAll`	Чтение, запись и выполнение. Значение `0x10000000`.
`Delete`	Удаление. Значение `0x00010000`.
`ReadPermissions`	Чтение прав. Значение `0x00020000`.
`ChangePermissions`	Изменение прав. Значение `0x00040000`.
`TakeOwnership`	Назначение владельца. Значение `0x00080000`.
`StandardRightsAll`	Стандартные права на объект. Значение `0x000F0000`.
`ListChildObjects`	Получение списка дочерних объектов. Значение `0x00000001`.

Имя члена	Описание
`CreateChildObjects`	Создание дочерних объектов. Значение `0x00000002`.
`DeleteChildObjects`	Удаление дочерних объектов. Значение `0x00000004`.
`Copy`	Копирование. Значение `0x00000008`.
`ReadAttributes`	Чтение атрибутов. Значение `0x00000010`.
`WriteAttributes`	Запись атрибутов. Значение `0x00000020`.
`PerformOperations`	Выполнение списка операций. Значение `0x00000100`.
`SpecificRightsAll`	Все существующие права. Значение `0x0000013F`.
`Read`	Чтение. Значение `0x00020019`.
`Modify`	Изменение. Значение `0x00000026`.
`Own`	Назначить себя владельцем. Значение `0x000E0000`.
`FullControl`	Полный доступ. Значение `0x000F013F`.

FileDataRights — перечисление

Определяет возможные права на файл.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.Security.AccessControl`
- **Сборка:** `DocsVision.Platform.dll`

Синтаксис

```
[Flags]
public enum FileDataRights
```

Члены

Имя члена	Описание
GenericRead	Простое чтение. Значение 0x80000000.
GenericWrite	Простая запись. Значение 0x40000000.
GenericExecute	Простое выполнение. Значение 0x20000000.
GenericAll	Чтение, запись и выполнение. Значение 0x10000000.
Delete	Удаление. Значение 0x00010000.
ReadPermissions	Чтение прав. Значение 0x00020000.
ChangePermissions	Изменение прав. Значение 0x00040000.
TakeOwnership	Назначение владельца. Значение 0x00080000.
StandardRightsAll	Стандартные права на объект. Значение 0x000F0000.
ReadData	Чтение. Значение 0x00000001.
WriteData	Запись. Значение 0x00000002.
Copy	Копирование. Значение 0x00000008.
ReadAttributes	Чтение атрибутов. Значение 0x00000010.
WriteAttributes	Запись атрибутов. Значение 0x00000020.
SpecificRightsAll	Все существующие права. Значение 0x0000003B.
Read	Чтение. Значение 0x00020019.
Modify	Изменение. Значение 0x00000022.
Own	Назначить себя владельцем. Значение 0x000E0000.
FullControl	Полный доступ. Значение 0x000F003B.

MandatoryAccessLevel — перечисление

Уровни доступа мандатной модели доступа.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl

- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public enum MandatoryAccessLevel
```

Члены

Имя члена	Описание
Untrusted	Не задан.
Low	Низкий.
Medium	Средний.
High	Высокий.
System	Доступен только для процессов платформы.

MandatoryAccessRights — перечисление

Права доступа для участников с уровнем мандата ниже требуемого.

Это перечисление имеет атрибут FlagsAttribute, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
[Flags]
public enum MandatoryAccessRights
```

Члены

Имя члена	Описание
None	Не указан.
NoWriteUp	Запрещен доступ на запись.
NoReadUp	Запрещен доступ на чтение.

Имя члена	Описание
NoExecuteUp	Запрещен доступ на выполнение.
All	Доступ запрещен.

SecureObjectType — перечисление

Определяет тип объекта безопасности.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
public enum SecureObjectType
```

Члены

Имя члена	Описание
Row	Строка.
Section	Секция. Не рекомендуется.
Card	Карточка.
File	Файл.
Object	Настраиваемый объект.
Report	Хранимая процедура.
CardType	Тип карточки.

ReportRights — перечисление

Определяет возможные права на хранимую процедуру.

- **Пространство имён:** DocsVision.Platform.Security.AccessControl
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
[Flags]
public enum ReportRights
```

Члены

Имя члена	Описание
`GenericRead`	Простое чтение. Значение `0x80000000`.
`GenericWrite`	Простая запись. Значение `0x40000000`.
`GenericExecute`	Простое выполнение. Значение `0x20000000`.
`GenericAll`	Чтение, запись и выполнение. Значение `0x10000000`.
`ReadPermissions`	Чтение прав. Значение `0x00020000`.
`ChangePermissions`	Изменение прав. Значение `0x00040000`.
`TakeOwnership`	Назначение владельца. Значение `0x00080000`.
`StandardRightsAll`	Стандартные права на объект. Значение `0x000E0000`.
`ReadData`	Чтение. Значение `0x00000001`.
`ReadAttributes`	Чтение атрибутов. Значение `0x00000010`.
`SpecificRightsAll`	Все существующие права. Значение `0x00000011`.
`Read`	Чтение. Значение `0x00020011`.
`Own`	Назначить себя владельцем. Значение `0x000E0000`.
`FullControl`	Полный доступ. Значение `0x000E0011`.

DocsVision.Platform.StorageServer — пространство имён

Пространство имён DocsVision.Platform.StorageServer содержит основные интерфейсы сервера приложений Docsvision, набор прокси-классов для интерфейса сервера IStorageServer, набор генераторов и интерфейсов к базе данных, а также реализацию логики сервера приложений.

Пространства имён

Пространство имён	Описание
Cursors	Пространство имён DocsVision.Platform.StorageServer.Cursors содержит класс позволяющий создать программно заполняемый серверный курсор.
Extensibility	Пространство имён DocsVision.Platform.StorageServer.Extensibility содержит классы, позволяющие реализовать серверное расширение.

Интерфейсы

Интерфейс	Описание
IBinaryStorage	Определяет методы провайдера к внешнему хранилищу.
IBinaryStorageRule	Определяет методы компонента, реализующий правило добавления в хранилище.
IEnumerableBinaryStorage	Определяет метод перечисления идентификаторов файлов в разделе хранилища.
IExtensionInitialize	Определяет метод загрузки провайдера к внешнему хранилищу.
IStreamedBinaryStorage	Определяет методы прямого доступа к потокам чтения/записи в провайдере к внешнему хранилищу.

Перечисления

Перечисление	Описание
UserRoles	Группы пользователей сервера Docsvision.
StoragePartitionType	Определяет возможные типы разделов во внешнем хранилище.

ITokenAuthenticator — интерфейс

Контейнер для токена аутентификации с возможностью создания токена

- **Пространство имён:** DocsVision.Platform.Authentication
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public interface ITokenAuthenticator : ITokenContainer
```

Методы

Имя	Описание
Authenticate()	Выполняет аутентификацию и создаёт токен

ITokenContainer — интерфейс

Представляет контейнер для токена аутентификации

- **Пространство имён:** DocsVision.Platform.Authentication
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public interface ITokenContainer
```

Свойства

Имя	Описание
AuthenticationToken	Токен аутентификации **Тип**: System.String
AuthenticationProviderId	Идентификатор провайдера аутентификации **Тип**: System.Guid

Методы

Имя	Описание
`UpdateToken(string)`	Обновляет токен *Параметры:* • `token` — токен

Cursors — пространство имён

Пространство имён `DocsVision.Platform.StorageServer.Cursors` содержит класс позволяющий создать программно заполняемый серверный курсор.

Классы

Класс	Описание
`CursorBuilder`	Представляет программно-заполняемый курсор.

CursorBuilder — класс

Представляет программно-заполняемый курсор.

- **Пространство имён:** `DocsVision.Platform.StorageServer.Cursors`
- **Сборка:** `DocsVision.Platform.StorageServer.Runtime.dll`

Синтаксис

```
public sealed class CursorBuilder : IDisposable
```

Свойства

Имя	Описание
`RowCount`	Возвращает количество строк.

Extensibility — пространство имён

Пространство имён `DocsVision.Platform.StorageServer.Extensibility` содержит классы, позволяющие реализовать серверное расширение.

Классы

Класс	Описание
ExtensionMethodAttribute	Атрибут методов серверного расширения.
StorageServerExtension	Базовый класс для создания серверного расширения.

ExtensionMethodAttribute — класс

Атрибут методов серверного расширения.

- **Пространство имён:** DocsVision.Platform.StorageServer.Extensibility
- **Сборка:** DocsVision.Platform.StorageServer.Runtime.dll

Синтаксис

```
[AttributeUsage(AttributeTargets.Method)]
public sealed class ExtensionMethodAttribute : Attribute
```

Конструкторы

Имя	Описание
ExtensionMethodAttribute()	Инициализирует новый экземпляр класса ExtensionMethodAttribute.
ExtensionMethodAttribute(UserRoles)	Инициализирует новый экземпляр класса ExtensionMethodAttribute с разрешённой пользовательской ролью.

Свойства

Имя	Описание
AllowedUserRoles	Задаёт или возвращает группу пользователей.

ExtensionMethodAttribute — конструктор (UserRoles)

Инициализирует новый экземпляр класса ExtensionMethodAttribute с разрешённой пользовательской ролью.

- **Пространство имён:** DocsVision.Platform.StorageServer.Extensibility

- **Сборка:** DocsVision.Platform.StorageServer.Runtime.dll

Синтаксис

```
public ExtensionMethodAttribute(UserRoles allowedUserRoles)
```

Параметры

allowedUserRoles

Тип: UserRoles

Группа пользователя

Заметки

allowedUserRoles позволяет ограничить возможность вызова серверного расширения только пользователями водящими в указанную группу.

StorageServerExtension — класс

Базовый класс для создания серверного расширения.

- **Пространство имён:** DocsVision.Platform.StorageServer.Extensibility
- **Сборка:** DocsVision.Platform.StorageServer.Runtime.dll

Синтаксис

```
public abstract class StorageServerExtension : IServiceProvider, IDisposable
```

Свойства

Имя	Описание
Application	Возвращает объект типа IApplication.
DbRequest	Возвращает объект типа IDbRequest.
Request	Возвращает объект, содержащий запрос к сессии.

Методы

Имя	Описание
GetCursorBuilder	Создаёт и возвращает программно-заполняемый серверный курсор
ExecuteCursorCommand(DatabaseCommand)	Запускает сформированный SQL-запрос и возвращает серверный курсор с данными.

StorageServerExtension.GetCursorBuilder — метод

Создаёт и возвращает программно-заполняемый серверный курсор.

- **Пространство имён:** DocsVision.Platform.StorageServer.Extensibility
- **Сборка:** DocsVision.Platform.StorageServer.Runtime.dll

Синтаксис

```
protected CursorBuilder GetCursorBuilder()
```

Возвращаемое значение

Тип: CursorBuilder

Заполняемый курсор

Заметки

Данный метод инициализирует объект для программного заполнения серверного курсора.

StorageServerExtension.ExecuteCursorCommand — метод (DatabaseCommand)

Запускает сформированный SQL-запрос и возвращает серверный курсор с данными.

- **Пространство имён:** DocsVision.Platform.StorageServer.Extensibility
- **Сборка:** DocsVision.Platform.StorageServer.Runtime.dll

Синтаксис

```
protected CursorInfo ExecuteCursorCommand(DatabaseCommand command)
```

Параметры

command

Тип: DatabaseCommand

Запрос к базе данных

Возвращаемое значение

Тип: CursorInfo

Серверный курсор

Files — пространство имён

Пространство имён DocsVision.Platform.StorageServer.Files содержит классы и интерфейсы, содержащие информацию о файле.

Интерфейсы

Интерфейс	Описание
IFileInfo	Определяет набор информации о файле.

IFileInfo — интерфейс

Определяет набор информации о файле.

- **Пространство имён:** DocsVision.Platform.StorageServer.Files
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
public interface IFileInfo
```

Свойства

Имя	Описание
ArchiveState	Задаёт или возвращает статус архивирования файла.
BinaryId	Задаёт или возвращает идентификатор бинарных данных файла.

Имя	Описание
`Created`	Возвращает дату создания файла.
`Encrypted`	Возвращает статус шифрования.
`ExtAttributes`	Возвращает флаги дополнительных атрибутов файла.
`FileId`	Задаёт или возвращает идентификатор файла.
`LastChanged`	Возвращает дату изменения файла.
`Name`	Задаёт или возвращает название файла.
`OfflineState`	Задаёт или возвращает статус оперативности файла.
`OwnerCardId`	Задаёт или возвращает идентификатор родительской карточки.
`Signed`	Возвращает статус подписания.
`Size`	Задаёт или возвращает размер файла.
`StdAttributes`	Задаёт или возвращает флаги стандартные атрибуты файла.

CursorInfo — структура

Предоставляет хранилище данных для серверного курсора.

- **Пространство имён:** `DocsVision.Platform.StorageServer`
- **Сборка:** `DocsVision.Platform.StorageServer.dll`

Синтаксис

```
[ComVisible(true)]
[DataContract(Namespace = "http://schemas.docsvision.com/Platform/2009-02-
03/StorageServer/")]
public struct CursorInfo
```

Поля

Имя	Описание
FirstPage	Первая страница.
Id	Идентификатор курсора
PageCount	Количество страницы
RowCount	Количество строк.
UsingCompression	Использовать сжатие данных.

IBinaryStorage — интерфейс

Определяет методы провайдера к внешнему хранилищу.

- **Пространство имён:** DocsVision.Platform.StorageServer
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
public interface IBinaryStorage
```

Свойства

Имя	Описание
IsIntegratedStorage	Флаг, указывающий является ли данное хранилище интегрированным в Docsvision. **Должен всегда возвращать** false.
StorageId	Задаёт или возвращает уникальный идентификатор хранилища.
SupportedPartions	Задаёт или возвращает типы разделов, поддерживаемых хранилищем.

Методы

Имя	Описание
CanSafelyMove(String, StoragePartitionType)	Проверяет возможность перемещения файла в другой раздел данного хранилища без изменения данных.

Имя	Описание
`Create(IFileInfo, StoragePartitionType)`	Создаёт файл во внешнем хранилище.
`CreateFrom(IFileInfo, String, StoragePartitionType)`	Создаёт файл во внешнем хранилище и загружает в него данные другого файла.
`Delete(String)`	Удаляет файл с указанным идентификатором из внешнего хранилища.
`Read(String, Stream)`	Считывает данные файла из внешнего хранилища в поток.
`Read(String, Stream, Int64, Int32)`	Считывает данные файла из внешнего хранилища в поток начиная с заданной позиции.
`SafeMove(String, StoragePartitionType)`	Перемещает файл в новый раздел внешнего хранилища.
`Write(String, Stream)`	Записывает данные из потока в файл во внешнем хранилище.
`Write(String, Stream, Int64)`	Записывает данные из потока в файл во внешнем хранилище начиная с заданной позиции.

Заметки

Список возможных ошибок, которые могут возвращаться в реализации интерфейса `IBinaryStorage`:

- `DocsVision.Platform.StorageServer.BinaryNotFoundException` — указанный файл не существует в хранилище.
- `DocsVision.Platform.StorageServer.InvalidBinaryIdException` — переданный идентификатор файла имеет некорректный формат.
- `System.NotSupportedException` — метод не поддерживается в данном хранилище.
- `System.ArgumentNullException` — при вызове метода передан нулевой/пустой обязательный аргумент.
- `DocsVision.Platform.StorageServer.BinaryStorageException` — прочая ошибка в хранилище, не относящаяся к другим типам ошибок.

IBinaryStorage.SupportedPartions — свойство

Задаёт или возвращает типы разделов, поддерживаемых хранилищем.

- **Пространство имён:** DocsVision.Platform.StorageServer
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
StoragePartitionType SupportedPartions { get; set; }
```

Значение свойства

Тип: StoragePartitionType

Типы поддерживаемых разделов

Заметки

Значение свойства SupportedPartions может быть задано в виде набора флагов StoragePartitionType, если тип хранилища содержит несколько разделов:

```
SupportedPartions = StoragePartitionType.Primary` | StoragePartitionType.Archive;
```

IBinaryStorage.CanSafelyMove — метод (String, StoragePartitionType)

Проверяет возможность перемещения файла в другой раздел данного хранилища без изменения данных.

- **Пространство имён:** DocsVision.Platform.StorageServer
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
bool CanSafelyMove(string id, StoragePartitionType toPartition);
```

Параметры

id

Тип: System.String

Идентификатор файла в хранилище

toPartition

Тип: StoragePartitionType

Раздел хранилища, возможность перемещения в который проверяется

Возвращаемое значение

Тип: System.Boolean

true — перемещение возможно, false — перемещение невозможно

IBinaryStorage.Create — метод (IFileInfo, StoragePartitionType)

Формирует во внешнем хранилище путь к добавляемому файлу.

- **Пространство имён:** DocsVision.Platform.StorageServer
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
string Create(IFileInfo fileInfo, StoragePartitionType onPartition =
StoragePartitionType.Primary)
```

Параметры

fileInfo

Тип: IFileInfo

Информация о файле

onPartition

Тип: StoragePartitionType

Раздела хранилища, в который записывается файл

Возвращаемое значение

Тип: System.String

Уникальный идентификатор файла в хранилище

Заметки

Метод Create не записывает бинарные данные во внешнее хранилище, а только возвращает уникальный идентификатор для файла в данном хранилище. По

полученному идентификатору Docsvision будет получать файл из хранилища. Способ формирования идентификатора определяется программистом. Запись бинарных данных в хранилище осуществляет метод `Write`.

IBinaryStorage.CreateFrom — метод (IFileInfo, String, StoragePartitionType)

Формирует во внешнем хранилище путь к добавляемому файлу.

- **Пространство имён:** `DocsVision.Platform.StorageServer`
- **Сборка:** `DocsVision.Platform.StorageServer.dll`

Синтаксис

```
string CreateFrom(IFileInfo fileInfo, string sourceId, StoragePartitionType onPartition =
StoragePartitionType.Primary)
```

Параметры

fileInfo

Тип: `IFileInfo`

Информация о файле

sourceId

Тип: `System.String`

Идентификатор файла, данные которого копируются в новый файл, во внешнем хранилище

onPartition

Тип: `StoragePartitionType`

Раздела хранилища, в который записывается файл

Возвращаемое значение

Тип: `System.String`

Уникальный идентификатор файла в хранилище

Заметки

Метод `CreateFrom` должен создать новый файл во внешнем хранилище в разделе `onPartition`, и записать в него данные файла `sourceId`.

IBinaryStorage.Read — метод (String, Stream, Int64, Int32)

Считывает данные файла из внешнего хранилища в поток начиная с заданной позиции.

- **Пространство имён:** DocsVision.Platform.StorageServer
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
long Read(string id, Stream destStream, long position, int maxBytes)
```

Параметры

id

Тип: System.String

Идентификатор файла в хранилище

destStream

Тип: System.IO.Stream

Поток для записи данных файла

position

Тип: System.Int64

Позиция в файле, с которой считывать данные

maxBytes

Тип: System.Int32

Максимальное количество считываемых байт

Возвращаемое значение

Тип: System.Int64

Позиция в файле, на которой завершилось чтение

IBinaryStorage.SafeMove — метод (String, StoragePartitionType)

Перемещает файл в новый раздел внешнего хранилища.

- **Пространство имён:** DocsVision.Platform.StorageServer
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
string SafeMove(string id, StoragePartitionType toPartition)
```

Параметры

id

Тип: System.String

Идентификатор файла в хранилище

toPartition

Тип: StoragePartitionType

Раздел, в который перемещается файл

Возвращаемое значение

Тип: System.String

Идентификатор файла после перемещения

IBinaryStorage.Write — метод (String, Stream, Int64)

Записывает данные из потока в файл во внешнем хранилище начиная с заданной позиции.

- **Пространство имён:** DocsVision.Platform.StorageServer
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
long Write(string id, Stream sourceStream, long position)
```

Параметры

id

Тип: System.String

Идентификатор файла

sourceStream

Тип: `System.IO.Stream`

Поток с данными файла

position

Тип: `System.Int64`

Позиция в файле в хранилище, с которой должна выполняться запись

Возвращаемое значение

Тип: `System.Int64`

Количество записанных байт

IBinaryStorageRule — интерфейс

Определяет методы компонента, реализующий правило добавления в хранилище.

- **Пространство имён:** `DocsVision.Platform.StorageServer`
- **Сборка:** `DocsVision.Platform.StorageServer.dll`

Синтаксис

```
public interface IBinaryStorageRule
```

Свойства

Имя	Описание
`GroupId`	Задаёт или возвращает идентификатор группы хранилищ, для которой используется данное правило.

Методы

Имя	Описание
`IsRuleSatisfied(IFileInfo)`	Проверяет возможность перемещения файла в группу хранилищ.

IBinaryStorageRule.IsRuleSatisfied — метод (IFileInfo)

Проверяет возможность перемещения файла в группу хранилищ.

- **Пространство имён:** DocsVision.Platform.StorageServer
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
bool IsRuleSatisfied(IFileInfo fileInfo)
```

Параметры

fileInfo

Тип: IFileInfo

Информация о файле

Возвращаемое значение

Тип: System.Boolean

Должен вернуть `true`, если файл `fileInfo` удовлетворяет условиям данного правила, и `false` — если не удовлетворяет.

IEnumerableBinaryStorage — интерфейс

Определяет метод перечисления идентификаторов файлов в разделе хранилища.

- **Пространство имён:** DocsVision.Platform.StorageServer
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
public interface IEnumerableBinaryStorage
```

Методы

Имя	Описание
GetEnumerable(StoragePartitionType)	Возвращает перечисления идентификаторов файлов в указанном разделе хранилища.

IExtensionInitialize — интерфейс

Определяет метод загрузки провайдера к внешнему хранилищу.

- **Пространство имён:** DocsVision.Platform.StorageServer
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
public interface IExtensionInitialize
```

Методы

Имя	Описание
Initialize(IServiceProvider, String)	Инициализирует провайдер к хранилищу.

IExtensionInitialize.Initialize — метод (IServiceProvider, String)

Инициализирует провайдер к хранилищу.

- **Пространство имён:** DocsVision.Platform.StorageServer
- **Сборка:** DocsVision.Platform.StorageServer.dll

Синтаксис

```
void Initialize(IServiceProvider serviceProvider, string settings)
```

Параметры

serviceProvider

Тип: System.IServiceProvider

Провайдер к сервисам Docsvision

settings

Тип: System.String

Сериализованные в строку настройки провайдера к хранилищу

Заметки

Сериализованные настройки

IStreamedBinaryStorage — интерфейс

Определяет методы прямого доступа к потокам чтения/записи в провайдере к внешнему хранилищу.

- **Пространство имён:** `DocsVision.Platform.StorageServer`
- **Сборка:** `DocsVision.Platform.StorageServer.dll`

Синтаксис

```
public interface IStreamedBinaryStorage
```

Методы

Имя	Описание
`OpenReadStream(String)`	Возвращает поток чтения файла из внешнего хранилища.
`OpenWriteStream(String)`	Возвращает поток записи файла во внешнем хранилище.

StoragePartitionType — перечисление

Определяет возможные типы разделов во внешнем хранилище.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.StorageServer`
- **Сборка:** `DocsVision.Platform.StorageServer.dll`

Синтаксис

```
[Flags]
public enum StoragePartitionType
```

Члены

Имя члена	Описание
`Default`	Раздел без назначения. Значение `0x0`.

Имя члена	Описание
`Primary`	Раздел для хранения основных файлов. Значение `0x1`.
`Archive`	Раздел для хранения архивных файлов. Значение `0x2`.
`Temp`	Раздел для хранения временных файлов. Значение `0x4`.

UserRoles — перечисление

Группы пользователей сервера Docsvision.

Это перечисление имеет атрибут `FlagsAttribute`, поддерживающий побитовое соединение составляющих его значений.

- **Пространство имён:** `DocsVision.Platform.StorageServer`
- **Сборка:** `DocsVision.Platform.StorageServer.Runtime.dll`

Синтаксис

```
[Flags]
public enum UserRoles
```

Члены

Имя члена	Описание
`None`	Не задана. Значение `0x00`.
`User`	Пользователь Docsvision. Значение `0x01`.
`PowerUser`	Привилегированный пользователь Docsvision. Значение `0x02`.
`Administrator`	Администратор Docsvision. Значение `0x04`.
`SecurityAdministrator`	Администратор безопасности Docsvision. Значение `0x08`.
`ArchiveOperator`	Администратор архива Docsvision. Значение `0x10`.

Имя члена	Описание
`SearchQueryCreator`	Создатель поисковых запросов Docsvision. Значение `0x20`.
`WorkflowProcessCreators`	Создатель бизнес-процесса Docsvision. Значение `0x40`.
`All`	Все пользователи. Значение `0x7F`.

Заметки

Приведенные группы добавляются в список локальных пользователей и групп Windows при установке серверной части Docsvision.

DocsVision.Platform.SystemCards — пространство имён

Пространство имён `DocsVision.Platform.SystemCards` содержит реализацию объектов бизнес-логики для системных карточек.

Пространства имён

Пространство имён	Описание
`DocsVision.Platform.SystemCards.ObjectModel`	Пространство имён `DocsVision.Platform.SystemCards.ObjectModel` содержит реализацию объектной модели высокого уровня для системных карточек.

DocsVision.Platform.SystemCards.ObjectModel — пространство имён

Пространство имён `DocsVision.Platform.SystemCards.ObjectModel` содержит реализацию объектной модели высокого уровня для системных карточек.

Пространства имён

Пространство имён	Описание
`DocsVision.Platform.SystemCards.ObjectModel.Mapping`	Содержит фабрики преобразования данных для системных карточек.
`DocsVision.Platform.SystemCards.ObjectModel.Services`	Содержит фабрики сервисов для системных карточек.

Классы

Класс	Описание
`FoldersFolder`	Объектная модель папки.
`SavedViewsView`	Представляет объектную модель *Карточки сохранённых представлений*.

DocsVision.Platform.SystemCards.ObjectModel.Mapping — пространство имён

Содержит фабрики преобразования данных для системных карточек.

Классы

Класс	Описание
`SystemCardsMapperFactory`	Фабрика преобразователей данных для системных карточек.

SystemCardsMapperFactory — класс

Фабрика преобразователей данных для системных карточек.

- **Пространство имён:** `DocsVision.Platform.SystemCards.ObjectModel.Mapping`
- **Сборка:** `DocsVision.Platform.SystemCards.ObjectModel.dll`

Синтаксис

```
public sealed class SystemCardsMapperFactory : ObjectMapperFactory
```

Конструкторы

Имя	Описание
`SystemCardsMapperFactory(ObjectContext)`	Инициализирует новый экземпляр класса `SystemCardsMapperFactory`.

DocsVision.Platform.SystemCards.ObjectModel.Services — пространство имён

Содержит фабрики сервисов для системных карточек.

Классы

Класс	Описание
SystemCardsServiceFactory	Фабрика сервисов для системных карточек.

Интерфейсы

Интерфейс	Описание
IMandatoryAccessService	Сервис для работы с мандатной моделью доступа.

SystemCardsServiceFactory — класс

Фабрика сервисов для системных карточек.

- **Пространство имён:** DocsVision.Platform.SystemCards.ObjectModel.Services
- **Сборка:** DocsVision.Platform.SystemCards.ObjectModel.dll

Синтаксис

```
public class SystemCardsServiceFactory : ServiceFactory
```

Конструкторы

Имя	Описание
SystemCardsServiceFactory()	Инициализирует новый экземпляр класса SystemCardsServiceFactory.

Методы

Имя	Описание
GetService(Type)	Возвращает объект обслуживания указанного типа.

FoldersFolder — класс

Объектная модель папки.

- **Пространство имён:** DocsVision.Platform.SystemCards.ObjectModel
- **Сборка:** DocsVision.Platform.SystemCards.ObjectModel.dll

Синтаксис

```
public sealed class FoldersFolder : ObjectBase
```

Конструкторы

Имя	Описание
FoldersFolder()	Инициализирует новый экземпляр класса FoldersFolder.

Свойства

Имя	Описание
DefaultView	Задаёт или возвращает представление по умолчанию.
FolderFlags	Задаёт или возвращает дополнительные флаги.
Folders	Возвращает коллекцию подчиненных папок.
FolderType	Задаёт или возвращает тип папки.
Name	Задаёт или возвращает название папки.
Parent	Возвращает родительскую папку.
Reference	Задаёт или возвращает ассоциированный идентификатор.
Security	Возвращает объект безопасности, связанный с папкой.

Поля

Имя	Описание
DefaultViewProperty	Определяет свойство "Представление по умолчанию".
FlagsProperty	Определяет свойство "Флаги".
FoldersProperty	Определяет свойство "Папки".

Имя	Описание
NameProperty	Определяет свойство "Название папки".
ReferenceProperty	Определяет свойство "Ассоциированный идентификатор".
SecurityProperty	Определяет свойство "Безопасность".
TypeProperty	Определяет свойство "Тип папки".

SavedViewsView — класс

Представляет объектную модель *Карточки сохранённых представлений*.

- **Пространство имён:** DocsVision.Platform.SystemCards.ObjectModel
- **Сборка:** DocsVision.Platform.SystemCards.ObjectModel.dll

Синтаксис

```
public sealed class SavedViewsView : ObjectBase
```

Конструкторы

Имя	Описание
SavedViewsView	Инициализирует новый экземпляр класса SavedViewsView.

Свойства

Имя	Описание
Hidden	Задаёт или возвращает признак скрытого представления.
Name	Задаёт или возвращает название представления.
Text	Задаёт или возвращает описание.

Поля

Имя	Описание
HiddenProperty	Определяет свойство "Скрытое представление".
NameProperty	Определяет свойство "Название представления".
TextProperty	Определяет свойство "XML описание".

Заметки

Представление может быть создано в *Рабочем месте администратора* (см. документ "Справка Docsvision 4.5").

Примеры

Представление может быть получено по уникальному идентификатору

```
①

SavedViewsView savedViewsView = objectContext.GetObject<SavedViewsView>(new Guid
("00000000-0000-0000-0000-000000000000"));
```

① Инициализация контекста объектов.

DocsVision.Platform.WinForms — пространство имён

Пространство имён с набором базовых классов и элементов управления WinForms.

Пространства имён

Пространство имён	Описание
Controls	Пространство имён Controls содержит коллекцию элементов управления WinForms, используемых в карточках Docsvision.
DataSource	Коллекция источников данных для карточек.

Пространство имён	Описание
SystemDialogs	Пространство имён SystemDialogs содержит формы различных диалоговых окон, используемых Windows-клиентом.

Классы

Класс	Описание
CardActionEventArgs	Содержит данные для события CardAction.
CardActivatedEventArgs	Содержит данные для события CardActivated.
CardActivatingEventArgs	Содержит данные для события CardActivating.
CardClosingEventArgs	Содержит данные для события CardClosing.
CardControl	Базовый класс для компонента карточки.
CardLibrary	Базовый класс для библиотеки карточек
NavCommandContext	Представляет контекст команд.
NavExtension	Базовый класс для расширения Windows-клиента.
NavPropertyPageControl	Базовый класс для элемента управления страницы свойств Windows-клиента.

DocsVision.Platform.WinForms.Controls — пространство имён

Пространство имён Controls содержит коллекцию элементов управления WinForms, используемых в карточках Docsvision.

Интерфейсы

Интерфейс	Описание
IExtensionPropertiesControl	Определяет методы и свойства страницы настройки.

IExtensionPropertiesControl — интерфейс

Определяет методы и свойства страницы настройки.

- **Пространство имён:** DocsVision.Platform.WinForms.Controls
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public interface IExtensionPropertiesControl
```

Свойства

Имя	Описание
Settings	Возвращает настройки, сериализованные в строку.

Методы

Имя	Описание
Initialize(String)	Загружает сериализованные настройки в компонент.
Save	Сохраняет настройки в Settings. При наличии ошибок должен вернуть false, иначе — true.

События

Имя	Описание
OnPropertiesChanged	Вызывается при изменении настроек с помощью графического интерфейса.

DocsVision.Platform.WinForms.SystemDialogs — пространство имён

Пространство имён SystemDialogs содержит формы различных диалоговых окон, используемых Windows-клиентом.

Пространства имён

Пространство имён	Описание
ExportDialog	Пространство имён ExportDialog содержит форму экспорта и печати данных карточки, а также элементы управления для данной формы.
LogSettings	Предоставляет дополнительные классы, используемые журналом отображения событий.
LogViewer	Содержит различные формы, предоставляющие интерфейс журналов событий с поиском.
ParametersDialog	Пространство имён ParametersDialog содержит классы предоставляющие интерфейсу пользователя форму настройки различных параметров.
ScanDialog	Пространство имён ScanDialog содержит классы и интерфейсы, отвечающие за работу пользователя со сканером из среды Windows-клиента.

ExportDialog — пространство имён

Пространство имён ExportDialog содержит форму экспорта и печати данных карточки, а также элементы управления для данной формы.

Классы

Класс	Описание
ExportDialog	Предоставляет классу ExportForm элементы управления.
ExportForm	Класс ExportForm представляет форму экспорта и печати содержимого карточки, позволяет отправлять содержимое по электронной почте.

ExportForm — класс

Класс `ExportForm` представляет форму экспорта и печати содержимого карточки, позволяет отправлять содержимое по электронной почте.

- **Пространство имён:** `DocsVision.Platform.WinForms.SystemDialogs.ExportDialog`
- **Сборка:** `DocsVision.Platform.WinForms.SystemDialogs.dll`

Синтаксис

```
public class ExportForm : Form
```

Конструкторы

Имя	Описание
`ExportForm()`	Инициализирует новый экземпляр класса `ExportForm`.

Свойства

Имя	Описание
`Caption`	Задаёт или возвращает тему письма, отправляемого из формы `ExportForm`.
`CardData`	Задаёт или возвращает содержимое карточки, для которой вызывается диалог экспорта и печати.
`CardHost`	Задаёт или возвращает контейнер карточки.
`CardTypeId`	Задаёт или возвращает идентификатор типа карточки. Данный параметр используется при формировании списка доступных преобразований печати и экспорта.

Имя	Описание
`CardXml`	Задаёт или возвращает содержимое произвольного XML-документа, предназначенного для вывода в диалог печати и экспорта. Используется вместо XML-документа, полученного из `CardData`.
`Session`	Задаёт или возвращает пользовательскую сессию.
`TemplateAlias`	Задаёт или возвращает псевдоним преобразования, которое будет выбрано по умолчанию.

Заметки

Данный класс предоставляет два варианта использования:

1. Используется `CardData` — диалог вызывается для существующей карточки. Параметры `CardXml`, `CardTypeId`, `Caption` указывать не требуется.
2. Используется `CardXml` — диалог вызывается для печати переданной Xml-строки. Параметр `CardData` указывать не требуется. Данный параметр доступен начиная с версии платформы 5.4.

При печати собственного XML-документа список доступных преобразований печати и экспорта будет ограничен преобразованиями имеющими тип `Custom` (см. раздел Раздел 'Transformations').

Не рекомендуется задавать значения `CardData` и `CardXml` одновременно, так как при этом возможно формирование некорректного списка преобразований.

CardActionEventArgs — класс

Содержит данные для события `CardAction`.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
public class CardActionEventArgs : EventArgs
```

Конструкторы

Имя	Описание
`CardActionEventArgs(Guid, Guid)`	Инициализирует объект CardActionEventArgs.

Свойства

Имя	Описание
`ActionId`	Идентификатор задействованного метода.
`ModeId`	Идентификатор текущего режима открытия карточки.

CardActionEventArgs — конструктор (Guid, Guid)

Инициализирует новый экземпляр класса `CardActionEventArgs` с помощью указанного значения.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
public CardActionEventArgs(Guid actionId, Guid modeId);
```

Параметры

actionId

Тип: `Guid`

Идентификатор задействованного метода.

modeId

Тип: `Guid`

Идентификатор текущего режима открытия карточки.

CardActivatedEventArgs — класс

Содержит данные события `CardActivated`.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public class CardActivatedEventArgs : EventArgs
```

Конструкторы

Имя	Описание
CardActivatedEventArgs(ActivateMode, ActivateFlags, Object)	Инициализирует новый экземпляр класса CardActivatedEventArgs с помощью указанных значений.

Свойства

Имя	Описание
ActivateFlags	Возвращает дополнительные флаги активации карточки.
ActivateMode	Возвращает режим активации карточки.
ActivateParams	Возвращает дополнительные флаги активации карточки.

CardActivatedEventArgs — конструктор (ActivateMode, ActivateFlags, Object)

Инициализирует новый экземпляр класса CardActivatedEventArgs с помощью указанных значений.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public CardActivatedEventArgs(DocsVision.Platform.CardHost.ActivateMode activateMode,
DocsVision.Platform.CardHost.ActivateFlags activateFlags, object activateParams)
```

Параметры

activateMode

Тип: `ActivateMode`

Режим активации карточки.

activateFlags

Тип: `ActivateFlags`

Дополнительные флаги активации карточки.

activateParams

Тип: `Object`

Дополнительные флаги активации карточки.

CardActivatingEventArgs — класс

Предоставляет данные для события `CardActivating`.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
public class CardActivatingEventArgs : EventArgs
```

Конструкторы

Имя	Описание
`CardActivatingEventArgs(DocsVision.Platform.CardHost.ActivateMode, DocsVision.Platform.CardHost.ActivateFlags, object)`	Инициализирует новый экземпляр класса CardActivatingEventArgs с помощью указанных значений.

Свойства

Имя	Описание
`ActivateFlags`	Возвращает дополнительные флаги активации карточки.

Имя	Описание
ActivateMode	Возвращает режим активации карточки.
ActivateParams	Возвращает дополнительные флаги активации карточки.

CardActivatingEventArgs — конструктор (ActivateMode, ActivateFlags, Object)

Инициализирует новый экземпляр класса CardActivatingEventArgs с помощью указанных значений.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public CardActivatingEventArgs(DocsVision.Platform.CardHost.ActivateMode activateMode,
DocsVision.Platform.CardHost.ActivateFlags activateFlags, object activateParams)
```

Параметры

activateMode

IBinaryStorage.Write — метод (String, Stream, Int64)ActivateMode

Режим активации карточки.

activateFlags

Тип: ActivateFlags

Дополнительные флаги активации карточки.

activateParams

Тип: Object

Дополнительные флаги активации карточки.

CardClosingEventArgs — класс

Предоставляет данные для события CardClosing.

- **Пространство имён:** DocsVision.Platform.WinForms

- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public class CardClosingEventArgs : EventArgs
```

Конструкторы

Имя	Описание
CardClosingEventArgs()	Инициализирует новый экземпляр класса CardClosingEventArgs.

Свойства

Имя	Описание
ActionFlags	Флаги определяющие действия, которые должны быть исполнены при закрытии карточки.

CardClosingEventArgs — конструктор

Инициализирует новый экземпляр класса CardClosingEventArgs.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public CardClosingEventArgs()
```

CardControl — класс

Базовый класс для компонента карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[ClassInterface(ClassInterfaceType.None)]
[ToolboxItem(false)]
```

```
[DefaultEvent("CardActivated")]
[Designer(typeof(CardControlDesigner), typeof(IRootDesigner))]
[Guid("0716E023-6A9A-4D73-9662-A2CDEFCD851A")]
[ComVisible(true)]
[ComDefaultInterface(typeof(ICardUI))]
public class CardControl : System.Windows.Forms.ContainerControl, IServiceProvider
```

Конструкторы

Имя	Описание
`CardControl()`	Инициализирует новый экземпляр класса `CardControl`.

Свойства

Имя	Описание
`AcceptButton`	Возвращает или задаёт компонент управления `AcceptButton`.
`ActivateFlags`	Возвращает флаги активации карточки.
`ActivateMode`	Возвращает режим активации карточки.
`ActivateParams`	Возвращает дополнительные параметры активации.
`CancelButton`	Возвращает или задаёт компонент управления `CancelButton`.
`CardData`	Совокупность всех данных секций и атрибутов конкретного экземпляра карточки.
`CardFrame`	Ссылка на окно карточки.
`CardHost`	Ссылка на объект контейнера карточки.
`DialogResult`	Возвращает или задаёт результат диалога для формы.
`FolderId`	Возвращает идентификатор папки, в которой лежит карточка.

Имя	Описание
KeyPreview	Признак получения события нажатия клавиши формой карточки, до передачи его элементу управления имеющему фокус.
ModeId	Идентификатор режима открытия карточки.
Session	Возвращает текущую сессию пользователя.
ShortcutId	Возвращает идентификатор ярлыка.

Методы

Имя	Описание
Close	Закрывает окно карточки.
Dispose(Boolean)	Освобождает неуправляемые ресурсы.
GetActiveCard	Возвращает из стека карточек ссылку на корневую карточку.
GetActiveCards	Возвращает массив ссылок на активные карточки.
GetParentCard(Control)	Возвращает ссылку на родительскую карточку указанного элемента управления.
GetService(Type)	Возвращает сервис указанного типа.
OnCardAction(CardActionEventArgs)	Создаёт событие CardAction. Вызывается до открытия карточки.
OnCardActivated(CardActionEventArgs)	Создаёт событие CardActivated. Вызывается после открытия карточки.
OnCardActivating(CardActivatingEventArgs)	Создаёт событие CardActivating. Вызывается до открытия карточки.
OnCardClosed(EventArgs)	Создаёт событие CardClosed. Вызывается после открытия карточки.

Имя	Описание
OnCardClosing(CardClosingEventArgs)	Создаёт событие CardClosing. Вызывается до открытия карточки.
OnCardInitialized(EventArgs)	Создаёт событие CardInitialized. Вызывается при первичной инициализации карточки.
OnCardLoaded(EventArgs)	Создаёт событие CardLoaded. Вызывается после загрузки карточки.
OnCardUnloaded(EventArgs)	Создаёт событие CardUnloaded. Вызывается перед выгрузкой карточки.
ProcessDialogKey(Keys)	Признак обработки клавиши элементом управления.
ProcessKeyPreview	Выполняет предварительный просмотр сообщения клавиатуры.
ProcessTabKey	Выбирает следующий доступный элемент управления и активизирует его.
ShowMessage(String)	Отображает окно сообщения с заданным текстом.
ShowMessage(String, String)	Отображает окно сообщения с заданным текстом и заголовком окна.
ShowMessage(String, String, String)	Отображает окно сообщения с заданным текстом, заголовком окна и полем дополнительной информации.
ShowMessage(String, String, MessageType, MessageButtons)	Отображает окно сообщения с заданным текстом и заголовком окна. Также передается список отображаемых кнопок и тип сообщения.

Имя	Описание
ShowMessage(String, String, String, MessageType, MessageButtons)	Отображает окно сообщения с заданным текстом, заголовком окна и полем дополнительной информации. Также передается список отображаемых кнопок и тип сообщения.
UpdateDefaultButton	Обновляет сведения о том, какая кнопка является кнопкой по умолчанию.

События

Имя	Описание
CardAction	Событие, которое инициируется при активации пользователем одного из методов карточки.
CardActivated	Событие инициируется после активации компонента карточки Windows-клиентом.
CardActivating	Событие инициируется до активации компонента карточки Windows-клиентом.
CardClosed	Событие возникает после закрытия пользовательского интерфейса карточки.
CardClosing	Событие возникает до закрытия пользовательского интерфейса карточки.
CardInitialized	Событие инициируется после создания компонента карточки, и передачи ей актуальных данных.
CardLoaded	Событие возникает после загрузки компонента карточки в память.

Имя	Описание
`CardUnloaded`	Событие инициируется при выгрузке компонента карточки из памяти.

CardControl — конструктор

Инициализирует новый экземпляр класса `CardControl`.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
public CardControl()
```

CardControl.CardInitialized — событие

Событие инициируется после создания компонента карточки, и передачи ей актуальных данных.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
[ResCategory("Behavior")]
public event EventHandler CardInitialized
```

Заметки

Событие запускается при инициализации объектов `CardData` и `UserSession`.

CardControl.AcceptButton — свойство

Возвращает или задаёт компонент управления `AcceptButton`.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
[Browsable(true)]
```

```
[ResCategory("Behavior")]
public System.Windows.Forms.IButtonControl AcceptButton { get; set; }
```

Значение свойства

Тип: IButtonControl

Компонент управления AcceptButton

CardControl.ActivateFlags — свойство

Возвращает флаги активации карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
[Browsable(false)]
public DocsVision.Platform.CardHost.ActivateFlags ActivateFlags { get; }
```

Значение свойства

Тип: ActivateFlags

Флаги активации

CardControl.ActivateMode — свойство

Возвращает режим активации карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
[Browsable(false)]
public DocsVision.Platform.CardHost.ActivateMode ActivateMode { get; }
```

Значение свойства

Тип: ActivateMode

Режим активации

CardControl.ActivateParams — свойство

Возвращает дополнительные параметры активации карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[Browsable(false)]
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
public object ActivateParams { get; }
```

Значение свойства

Тип:http://msdn.microsoft.com/ru-ru/library/system.object.aspx[Object]`

Параметры активации

Заметки

Вызываемая карточка трактует эти параметры по своему усмотрению. Описание параметров активации приведено в разделе Параметры активации справочников.

CardControl.CancelButton — свойство

Возвращает или задаёт компонент управления CancelButton.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[Browsable(true)]
[ResCategory("Behavior")]
public System.Windows.Forms.IButtonControl CancelButton { get; set; }
```

Значение свойства

Тип: Object

Компонент управления CancelButton

CardControl.CardData — свойство

Совокупность всех данных секций и атрибутов конкретного экземпляра карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[Browsable(false)]
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
public CardData CardData { get; }
```

Значение свойства

Тип: CardData

Данные карточки

CardControl.CardFrame — свойство

Ссылка на окно карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
[Browsable(false)]
public DocsVision.Platform.CardHost.ICardFrame CardFrame { get; }
```

Значение свойства

Тип: ICardFrame

Окно карточки

Заметки

Этот объект позволяет выполнять действия с окном: закрыть окно, задать заголовок окна, открыть карточку в новом окне и пр.

CardControl.CardHost — свойство

Ссылка на объект контейнера карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[Browsable(false)]
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
public DocsVision.Platform.CardHost.ICardHost CardHost { get; }
```

Значение свойства

Тип: ICardHost

Контейнер карточки

CardControl.DialogResult — свойство

Возвращает или задаёт результат диалога для формы.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[Browsable(false)]
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
[ResCategory("Behavior")]
public System.Windows.Forms.DialogResult DialogResult { get; set; }
```

Значение свойства

Тип: Form.DialogResult

Результат выбора

CardControl.FolderId — свойство

Возвращает идентификатор папки, в которой лежит карточка.

- **Пространство имён:** DocsVision.Platform.WinForms

- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
[Browsable(false)]
public Guid FolderId { get; }
```

Значение свойства

Тип: Guid

Идентификатор папки

CardControl.KeyPreview — свойство

Признак получения события нажатия клавиши формой карточки, до передачи его элементу управления имеющему фокус.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[ResCategory("Behavior")]
[Browsable(true)]
[DefaultValue(false)]
public bool KeyPreview { get; set; }
```

Значение свойства

Тип: Boolean

Признак контроля

Заметки

Возвращает или задаёт значение, указывающее, будет ли форма карточки получать события клавиш до того, как событие передастся элементу управления, имеющему фокус. Значение true, если форма карточки получит все события клавиш; значение false, если выбранный в настоящее время элемент управления получает события клавиш.

CardControl.ModeId — свойство

Идентификатор режима открытия карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
[Browsable(false)]
public Guid ModeId { get; }
```

Значение свойства

Тип: Guid

Режим открытия

CardControl.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
[Browsable(false)]
public UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Сессия

CardControl.ShortcutId — свойство

Возвращает идентификатор ярлыка.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[Browsable(false)]
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
public Guid ShortcutId { get; }
```

Значение свойства

Тип: Guid

Идентификатор ярлыка.

Заметки

CardControl.Close — метод

Закрывает окно карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public void Close()
```

CardControl.Dispose — метод (Boolean)

Освобождает неуправляемые ресурсы.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected override void Dispose(bool disposing)
```

Параметры

disposing

Тип: Boolean

Значение true позволяет освободить управляемые и неуправляемые ресурсы; значение false позволяет освободить только неуправляемые ресурсы.

Заметки

Освобождает ресурсы объектов `CardFrame` и `CardHost`.

CardControl.GetActiveCard — метод

Возвращает из стека карточек ссылку на корневую карточку.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
public static CardControl GetActiveCard()
```

Возвращаемое значение

Тип: `CardControl`

Элемент управления карточки

Заметки

Вместо GetActiveCard рекомендуется использовать метод `GetParentCard`.

CardControl.GetParentCard — метод (Control)

Возвращает ссылку на родительскую карточку указанного элемента управления.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
public static CardControl GetParentCard(System.Windows.Forms.Control control)
```

Параметры

control

Тип: `Control`

Элемент управления

Возвращаемое значение

Тип: `CardControl`

Элемент управления карточки

CardControl.GetActiveCards — метод

Возвращает массив ссылок на активные карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public static CardControl[] GetActiveCards()
```

Возвращаемое значение

Тип: CardControl[]

Массив из элементов управления карточками

CardControl.GetService — метод (Type)

Возвращает сервис указанного типа.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected override object GetService(Type serviceType)
```

Параметры

serviceType

Тип: Type

Специфический тип искомого сервиса

Возвращаемое значение

Тип: Object

Объект содержащий искомый сервис

Заметки

Если искомый сервис отсутствует, то будет возращен null.

CardControl.OnCardAction — метод (CardActionEventArgs)

Создаёт событие CardAction.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnCardAction(CardActionEventArgs e)
```

Параметры

e

Тип: CardActionEventArgs

Аргументы события.

CardControl.OnCardActivated — метод (CardActivatedEventArgs)

Создаёт событие CardActivated.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnCardActivated(CardActivatedEventArgs e)
```

Параметры

e

Тип: CardActivatedEventArgs

Аргументы события

CardControl.OnCardActivating — метод (CardActivatingEventArgs)

Создаёт событие CardActivating.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnCardActivating(CardActivatingEventArgs e)
```

Параметры

e

Тип: CardActivatingEventArgs

Аргументы события

CardControl.OnCardClosed — метод (EventArgs)

Создаёт событие CardClosed.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnCardClosed(EventArgs e)
```

Параметры

e

Тип: EventArgs

Аргументы события

CardControl.OnCardClosing — метод (CardClosingEventArgs)

Создаёт событие CardClosing.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnCardClosing(CardClosingEventArgs e)
```

Параметры

e

Тип: CardClosingEventArgs

Аргументы события

CardControl.OnCardInitialized — метод (EventArgs)

Создаёт событиеCardInitialized.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnCardInitialized(EventArgs e)
```

Параметры

e

Тип: EventArgs

Данные события.

CardControl.OnCardLoaded — метод (EventArgs)

Создаёт событие CardLoaded.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnCardLoaded(EventArgs e)
```

Параметры

e

Тип: EventArgs

Аргументы события

CardControl.OnCardUnloaded — метод (EventArgs)

Создаёт событие `CardUnloaded`.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
protected virtual void OnCardUnloaded(EventArgs e)
```

Параметры

e

Тип: `EventArgs`

Аргументы события

CardControl.ProcessDialogKey — метод (Keys)

Признак обработки клавиши элементом управления.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
protected override bool ProcessDialogKey(System.Windows.Forms.Keys keyData)
```

Параметры

keyData

Тип: `Keys`

Одно из значений `Keys`, представляющее обрабатываемую клавишу.

Возвращаемое значение

Тип: `Boolean`

Значение `true`, если клавиша была обработана элементом управления; в противном случае — значение `false`.

CardControl.ProcessKeyPreview — метод (Message)

Выполняет предварительный просмотр сообщения клавиатуры.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected override bool ProcessKeyPreview(ref System.Windows.Forms.Message m)
```

Параметры

m

Тип: Message

Сообщение окна для обработки

Возвращаемое значение

Тип: System.Boolean

Значение true, если сообщение было обработано элементом управления; в противном случае — значение false.

Заметки

Аналогичен методу Control.ProcessKeyPreview

CardControl.ShowMessage — метод (String, String, String, MessageType, MessageButtons)

Отображает окно сообщения с заданным текстом, заголовком окна и полем дополнительной информации. Также передается список отображаемых кнопок и тип сообщения.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public MessageResult ShowMessage(string text, string caption, string details, MessageType
messageType, MessageButtons buttons);
```

Параметры

text

Тип: `String`

Содержимое сообщения.

caption

Тип: `String`

Заголовок окна сообщения.

messageType

Тип: `MessageType`

Тип сообщение. Влияет на оформление окна.

buttons

Тип: `MessageButtons`

Кнопки, отображаемые в окне сообщения.

Возвращаемое значение

Тип: `MessageResult`

Кнопка, которая была нажата пользователем при закрытии сообщения.

CardControl.CardClosing — событие

Событие возникает до закрытия пользовательского интерфейса карточки.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
public event EventHandler<CardClosingEventArgs> CardClosing
```

Заметки

Подписавшись на событие можно провести сохранение формы карточки без запроса к пользователю.

CardLibrary — класс

Содержит сведения о библиотеке карточек.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[ComDefaultInterface(typeof(ICardLibraryInfo))]
[Guid("A7AC1B75-1DA5-4D9B-B509-601EBBFE3E7F")]
[ComVisible(true)]
[ClassInterface(ClassInterfaceType.None)]
[ToolboxItem(false)]
public class CardLibrary : Component, NativeMethods.ICardLibraryInfo, NativeMethods
.ICardLibraryInfo2
```

Конструкторы

Имя	Описание
CardLibrary	Инициализирует новый экземпляр класса CardLibrary.

Методы

Имя	Описание
GetCardDefinition(Guid)	Возвращает XML-описание карточки с указанным идентификатором.
GetCardIcon(Guid)	Возвращает иконку типа карточки с указанным идентификатором.
GetFileVersion()	Возвращает версию компонента библиотеки карточек.
GetIcon	Возвращает иконку библиотеки карточек.
GetLibraryDefinition	Возвращает XML-описание библиотеки карточек.
GetVersion	Возвращает версию библиотеки карточек.

Примечание

Обратите внимание, если `CardLibrary` наследуется в WinForms-компоненте, то пространство имён для подключения сборки — `DocsVision.Platform.WinForms`, если наследуется в WPF-компоненте — пространство имён для подключения сборки — `DocsVision.Platform.WPF`.

CardLibrary — конструктор

Инициализирует новый экземпляр класса `CardLibrary`.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
public CardLibrary()
```

CardLibary.GetCardDefinition — метод (Guid)

Возвращает XML-описание карточки с указанным идентификатором.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
public virtual string GetCardDefinition(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: `Guid`

Идентификатор карточки

Возвращаемое значение

Тип: `String`

Описание карточки

CardLibary.GetCardIcon — метод (Guid)

Возвращает иконку типа карточки с указанным идентификатором.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public virtual System.Drawing.Icon GetCardIcon(Guid cardTypeId)
```

Параметры

cardTypeId

Тип: Guid

Идентификатор карточки

Возвращаемое значение

Тип: System.Drawing.Icon

Иконка указанной карточки

CardLibary.GetFileVersion — метод

Возвращает версию компонента библиотеки карточек.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public virtual Version GetFileVersion()
```

Возвращаемое значение

Тип: System.Version

Версия компонента

Примечание

Метод GetFileVersion должен возвращать версию сборки, содержащей реализацию CardLibary.

CardLibary.GetIcon — метод

Возвращает иконку библиотеки карточек.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public virtual System.Drawing.Icon GetIcon()
```

Возвращаемое значение

Тип: System.Drawing.Icon

Иконка библиотеки

CardLibary.GetLibraryDefinition — метод

Возвращает XML-описание библиотеки карточек.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual string GetLibraryDefinition()
```

Возвращаемое значение

Тип: System.String

Схема библиотеки карточек.

Примеры

Схема карточки должна быть добавлена в компонент в виде ресурса, который может быть возвращен методом GetLibraryDefinition:

```
public override string GetCardDefinition(Guid cardTypeID)
{
   return Resources.SomeCardLibXml;
}
```

CardLibary.GetVersion — метод

Возвращает версию библиотеки карточек.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public virtual Version GetVersion()
```

Возвращаемое значение

Тип: System.Version

Версия библиотеки

Примечание

Метод GetVersion возвращает версию из описания библиотеки карточек (используется метод GetLibraryDefinition). Если получить описание стандартным способом не получается, будет возвращена версия файла, содержащего реализацию CardLibary.

NavCommandContext — класс

Предоставляет контекст выполнения для команды.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public class NavCommandContext
```

Конструкторы

Имя	Описание
NavCommandContext()	Инициализирует новый экземпляр класса NavCommandContext.

Свойства

Имя	Описание
FolderId	Возвращает или задаёт идентификатор папки.

Имя	Описание
FolderType	Возвращает или задаёт тип папки.
Selection	Возвращает или задаёт массив идентификаторов выделенных объектов.

Заметки

Контекст выполнения используется методами отвечающими за работу с командами расширения для достижения вариативности отображения одной команды для разных типов папок, а также выбранных карточек и ссылок.

NavCommandContext — конструктор

Инициализирует новый экземпляр класса NavCommandContext.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public NavCommandContext()
```

NavCommandContext.FolderId — свойство

Возвращает или задаёт идентификатор папки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public Guid FolderId { get; set; }
```

Значение свойства

Тип: System.Guid

Идентификатор папки

NavCommandContext.FolderType — свойство

Возвращает или задаёт тип папки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public NavFolderControlType FolderType { get; set; }
```

Значение свойства

Тип: NavFolderControlType

NavCommandContext.Selection — свойство

Возвращает или задаёт массив идентификаторов выделенных объектов.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public Guid[] Selection { get; set; }
```

Значение свойства

Тип: System.Guid[]`

Массив идентификаторов выбранных объектов

NavPropertyPageControl — класс

Базовый класс для элемента управления страницы свойств Windows-клиента.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[ClassInterface(ClassInterfaceType.None)]
[ComVisible(true)]
[DefaultEvent("PageActivated")]
[Guid("D2329F03-D431-4169-B927-8670F72732B2")]
```

```
[ToolboxItem(false)]
public class NavPropertyPageControl : ContainerControl, IServiceProvider, IPropertyPageUI
```

Конструкторы

Имя	Описание
NavPropertyPageControl	Инициализирует новый экземпляр класса NavPropertyPageControl.

Свойства

Имя	Описание
ActivateParams	Возвращает параметры активации страницы.
CardHost	Возвращает ссылку на контейнер карточки.
PageFrame	Возвращает ссылку на окно текущей страницы свойств.
Session	Возвращает текущую сессию пользователя.

Методы

Имя	Описание
Dispose(Boolean)	Освобождает ресурсы.
GetService(Type)	Получает сервис заданного типа.
OnPageActivated(EventArgs)	Создаёт событие PageActivated.
OnPageApplied(EventArgs)	Создаёт событие PageApplied.
OnPageClosed(EventArgs)	Создаёт событие PageClosed.
OnPageDeactivated(EventArgs)	Создаёт событие PageDeactivated.
OnPageInitialized(EventArgs)	Создаёт событие PageInitialized.

События

Имя	Описание
PageActivated	Событие вызывается при активации страницы свойства.
PageApplied	Событие вызывается при необходимости сохранить изменения.
PageClosed	Событие вызывается при закрытии страницы свойств.
PageDeactivated	Событие вызывается при деактивации страницы.
PageInitialized	Событие вызывается при инициализации страницы свойств.

Заметки

Объект типа NavPropertyPageControl используется расширением страниц свойств Windows-клиента. Объект унаследованный от данного класса должен иметь идентификатор COM-интерфейса.

Примеры

```
[ComVisible(true)]
[Guid("572860E1-E4C6-4120-B3DC-78C0A03F7445")]
[ClassInterface(ClassInterfaceType.None)]
public sealed partial class TestPropertyPage : NavPropertyPageControl
```

NavPropertyPageControl — конструктор

Инициализирует новый экземпляр класса NavPropertyPageControl.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public NavPropertyPageControl()
```

NavPropertyPageControl.ActivateParams — свойство

Возвращает параметры активации страницы свойств карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public object ActivateParams { get; }
```

Значение свойства

Тип: System.Object

Параметры активации

Заметки

Допустимые параметры для разных типов страниц свойств приведены в приложении Параметры активации страницы свойств карточки.

NavPropertyPageControl.CardHost — свойство

Возвращает ссылку на контейнер карточки.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[Browsable(false)]
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
public ICardHost CardHost { get; }
```

Значение свойства

Тип: ICardHost

Контейнер карточки

NavPropertyPageControl.PageFrame — свойство

Возвращает ссылку на окно текущей страницы свойств.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public IPropertyPageFrame PageFrame { get; }
```

Значение свойства

Тип: IPropertyPageFrame

Окно страницы

NavPropertyPageControl.Session — свойство

Возвращает текущую сессию пользователя.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[Browsable(false)]
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
public UserSession Session { get; }
```

Значение свойства

Тип: UserSession

Сессия пользователя

NavPropertyPageControl.Dispose — метод (Boolean)

Освобождает ресурсы.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected override void Dispose(bool disposing)
```

Параметры

disposing

Тип: System.Boolean

`true` — освобождать неуправляемые ресурсы, `false` — освобождать только управляемые ресурсы

NavPropertyPageControl.GetService — метод (Type)

Возвращает объект обслуживания указанного типа.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected override object GetService(Type serviceType)
```

Параметры

serviceType

Тип: System.Type

Тип сервиса

Возвращаемое значение

Тип: System.Object

Объект обслуживания типа `serviceType`

NavPropertyPageControl.OnPageActivated — метод (EventArgs)

Создаёт событие `PageActivated`.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnPageActivated(EventArgs e)
```

Параметры

e

Тип: System.EventArgs

Аргументы события

NavPropertyPageControl.OnPageApplied — метод (EventArgs)

Создаёт событие PageApplied.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnPageApplied(EventArgs e)
```

Параметры

e

Тип: System.EventArgs

Аргументы события

NavPropertyPageControl.OnPageClosed — метод (EventArgs)

Создаёт событие PageClosed.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnPageClosed(EventArgs e)
```

Параметры

e

Тип: System.EventArgs

Аргументы события

NavPropertyPageControl.OnPageDeactivated — метод (EventArgs)

Создаёт событие PageDeactivated.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnPageDeactivated(EventArgs e)
```

Параметры

e

Тип: System.EventArgs

Аргументы события

NavPropertyPageControl.OnPageInitialized — метод (EventArgs)

Создаёт событие PageInitialized.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnPageInitialized(EventArgs e)
```

Параметры

e

Тип: System.EventArgs

Аргументы события

NavPropertyPageControl.PageActivated — событие

Событие вызывается при активации страницы свойства.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public event EventHandler PageActivated
```

Заметки

Данное событие происходит при получении фокуса элементом управления.

NavPropertyPageControl.PageApplied — событие

Событие вызывается при необходимости сохранить изменения.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public event EventHandler PageApplied
```

Заметки

Данное событие происходит при нажатии пользователем кнопки **OK** или **Apply**.

NavPropertyPageControl.PageClosed — событие

Событие вызывается при закрытии страницы свойств.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public event EventHandler PageClosed
```

NavPropertyPageControl.PageDeactivated — событие

Событие вызывается при деактивации страницы.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public event EventHandler PageDeactivated
```

Заметки

Происходит когда элемент управления теряет фокус.

NavPropertyPageControl.PageInitialized — событие

Событие вызывается при инициализации страницы свойств.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public event EventHandler PageInitialized
```

NavExtension — класс

Базовый класс для расширения Windows-клиента.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
[ClassInterface(ClassInterfaceType.None)]
[ComVisible(true)]
[DefaultEvent("CardLoaded")]
[Guid("83458583-6D7E-41EE-B4D6-A3FD5D661CC9")]
[ToolboxItem(false)]
public class NavExtension : CardControl, INavExtension, INavCardSubTypesExtension,
INavDragDropExtension, INavFolderExtTypesExtension, INavReportExtension,
INavCommandExtension, INavStandardCommandExtension, INavEventExtension,
INavPickerExtension, INavControlExtension, INavPropPagesExtension,
INavCardCreatorExtension, INavCardCreatorEntryContainerExtension
```

Конструкторы

Имя	Описание
NavExtension()	Инициализирует новый экземпляр класса NavExtension.

Свойства

Имя	Описание
Commands	Возвращает коллекцию команд расширения.
PropertyPages	Возвращает коллекцию страниц свойств карточек и папок.

Имя	Описание
StandardCommands	Возвращает коллекцию стандартных команд поддерживаемых расширением.
SupportedTypes	Возвращает типы расширений реализуемые карточкой расширения.

Методы

Имя	Описание
CreateCommands	Возвращает информацию о командах расширения Windows-клиента.
CreatePropertyPages	Создаёт и возвращает коллекцию страниц свойств.
CreateReport(Object, InfoRowCollection, Guid)	Запускает механизм экспорта.
GetExtensionName(NavExtensionTypes)	Возвращает название заданного расширения.
InvokeCommand(NavCommand, NavCommandContext)	Вызов указанной команды расширения.
InvokeStandardCommand(NavStandardCommand, NavCommandContext, Object)	Вызов стандартной команды.
LookupAccounts(String[], SecurityIdentifier[], String[], NavPickerAccountTypes[])	Формирует коллекцию идентификаторов безопасности SID из коллекции имён учетных записей.
LookupCardTypes(Guid[], Guid[])	Сопоставляет заданной коллекции карточек их подтипы.
LookupNames(NavPickerNameFormat, NavPickerNameFormat, String[], String[])	Формирует по заданным учетным данным коллекцию учетных данных заданного формата.
LookupSids(SecurityIdentifier[], String[], String[], NavPickerAccountTypes[])	Формирует коллекцию учетных записей из заданной коллекции идентификаторов безопасности SID.

Имя	Описание
OnDragEnter(DragEventArgs)	Создаёт событие при перетаскивания объекта, когда курсор входит в пределы контейнера карточки.
OnDragLeave(DragEventArgs)	Создаёт событие, когда указатель выходит за границы контейнера карточки, в режиме перетаскивания.
OnDrop(DragEventArgs, Folder)	Создаёт событие завершения перетаскивания, при отпускании объекта в пределах контейнера карточки.
OnShutdown	Вызывается при завершении работы Windows-клиента.
OnStartup	Вызывается при загрузке Windows-клиента.
PickAccounts(NavPickerAccountTypes)	Отображает интерфейс выбора учетных записей.
PopulateCardTypes(INavCardTypeCollection)	Формирует коллекцию всех доступных подтипов карточек.
PopulateFolderTypes(INavFolderTypeCollection)	Формирует коллекцию описаний типов папок.
QueryCommandStatus(NavCommand, NavCommandContext)	Возвращает статус заданной команды.
QueryCommandStatusEx(NavCommand[], NavCommandContext)	Возвращает статусы заданной коллекции команд.
QueryFolderControl(NavFolderControlType, Guid)	Возвращает для заданной папки режим работы с Windows-клиентом.
QueryStandardCommandStatus(NavStandardCommand, NavCommandContext, Object)	Возвращает статус заданной стандартной команды.
QueryStandardCommandStatusEx(NavStandardCommand[], NavCommandContext, Object)	Возвращает статусы заданной коллекции стандартных команд.
Reset	Перезагружает расширение.

Примеры

```
[ComVisible(true)]
[Guid("6A0676EE-1DAF-4A59-B5EB-E0B5B4C175EE")]
[ClassInterface(ClassInterfaceType.None)]
public sealed partial class TestExtension : NavExtension
{
    public TestExtension() { }

    protected override NavExtensionTypes SupportedTypes
    {
        get
        {
            return NavExtensionTypes.All;
        }
    }
    protected override string GetExtensionName(NavExtensionTypes extensionType)
    {
        return "TextExtension";
    }
    //...
}
```

NavExtension — конструктор

Инициализирует новый экземпляр класса NavExtension.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
public NavExtension()
```

NavExtension.CreateCommands — метод

Возвращает информацию о командах расширения Windows-клиента.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual IEnumerable<NavCommand> CreateCommands()
```

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<NavCommand>

Коллекция команд расширения типа NavCommand.

NavExtension.CreatePropertyPages — метод

Создаёт и возвращает коллекцию страниц свойств.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual IEnumerable<NavPropertyPage> CreatePropertyPages()
```

Возвращаемое значение

Тип: System.Collections.Generic.IEnumerable<NavPropertyPage>

Коллекция страниц свойств типа NavPropertyPage

NavExtension.CreateReport — метод (Object, InfoRowCollection, Guid)

Запускает механизм экспорта.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void CreateReport(object grid, InfoRowCollection infoRowCollection,
Guid folderId)
```

Параметры

grid

Тип: System.Object

Ссылка на элемент управления JanusGrid, используемый для отображения данных представления в Windows-клиенте.

rowCollection

Тип: `InfoRowCollection`

Данные текущего представления

folderId

Тип: `System.Guid`

Идентификатор текущей папки

NavExtension.InvokeCommand — метод (NavCommand, NavCommandContext)

Вызов указанной команды расширения.

- **Пространство имён:** `DocsVision.Platform.WinForms`
- **Сборка:** `DocsVision.Platform.WinForms.dll`

Синтаксис

```
protected virtual void InvokeCommand(NavCommand command, NavCommandContext context)
```

Параметры

command

Тип: `NavCommand`

Вызываемая команда

context

Тип: `NavCommandContext`

Вызываемая команда

Примеры

```
protected override void InvokeCommand(NavCommand command, NavCommandContext context)
{
    MessageBox.Show(command.Description);
    base.InvokeCommand(command, context);
}
```

NavExtension.LookupAccounts — метод (String[], SecurityIdentifier[], String[], NavPickerAccountTypes[])

Формирует коллекцию идентификаторов безопасности SID из коллекции имён учетных записей.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void LookupAccounts(string[] accountNames, SecurityIdentifier[]
accountSids, string[] commonNames, NavPickerAccountTypes[] accountTypes)
```

Параметры

accountNames

Тип: System.String[]`

Массив имён учетных записей.

accountSids

Тип: System.Security.Principal.SecurityIdentifier[]

Массив идентификаторов безопасности SID.

commonNames

Тип: System.String[]`

Массив полных имена учетных записей.

accountTypes

Тип: NavPickerAccountTypes[]

Массив типов учетных записей.

NavExtension.LookupCardTypes — метод (Guid[], Guid[])

Сопоставляет заданной коллекции карточек их подтипы.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void LookupCardTypes(Guid[] cardIds, Guid[] cardTypeIds)
```

Параметры

cardIds

Тип: System.Guid[]`

Массив идентификаторов карточек

cardTypeIds

Тип: System.Guid[]`

Массив подтипов карточек

NavExtension.LookupNames — метод (NavPickerNameFormat, NavPickerNameFormat, String[], String[])

Формирует по заданным учетным данным коллекцию учетных данных заданного формата.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void LookupNames(NavPickerNameFormat formatOffered, NavPickerNameFormat
formatDesired, string[] accountNames, string[] resultNames)
```

Параметры

formatOffered

Тип: NavPickerNameFormat

Формат входящих учетных данных.

formatDesired

Тип: NavPickerNameFormat

Формат итоговых учетных данных.

accountNames

Тип: System.String[]`

Массив имён учетных записей.

resultNames

Тип: System.String[]`

Массив сформированных данных.

NavExtension.LookupSids — метод (SecurityIdentifier[], String[], String[], NavPickerAccountTypes[])

Формирует коллекцию учетных записей из заданной коллекции идентификаторов безопасности SID.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void LookupSids(SecurityIdentifier[] accountSids, string[]
accountNames, string[] commonNames, NavPickerAccountTypes[] accountTypes)
```

Параметры

accountSids

Тип: System.Security.Principal.SecurityIdentifier[]

Массив идентификаторов безопасности SID.

accountNames

Тип: System.String[]`

Массив имён учетных записей.

commonNames

Тип: System.String[]`

Массив полных имена учетных записей.

accountTypes

Тип: NavPickerAccountTypes[]

Массив типов учетных записей.

NavExtension.OnStartup — метод

Вызывается при загрузке Windows-клиента.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnStartup()
```

NavExtension.OnShutdown — метод

Вызывается при завершении работы Windows-клиента.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void OnShutdown()
```

NavExtension.PickAccounts — метод (NavPickerAccountTypes)

Отображает интерфейс выбора учетных записей.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual SecurityIdentifier[] PickAccounts(NavPickerAccountTypes accountTypes)
```

Параметры

accountTypes

Тип: NavPickerAccountTypes

Тип учетной записи.

Возвращаемое значение

Тип: System.Security.Principal.SecurityIdentifier[]

Массив выбранных учетных записей

Заметки

Методы вызывается по запросу пользователя и позволяет отобразить собственный интерфейс выбора учетных записей.

NavExtension.PopulateCardTypes — метод (INavCardTypeCollection)

Формирует коллекцию всех доступных подтипов карточек.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void PopulateCardTypes(INavCardTypeCollection cardTypes)
```

Параметры

cardTypes

Тип: INavCardTypeCollection

Коллекция типов карточек

NavExtension.PopulateFolderTypes — метод (INavFolderTypeCollection)

Формирует коллекцию описаний типов папок.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual void PopulateFolderTypes(INavFolderTypeCollection folderTypes)
```

Параметры

folderTypes

Тип: INavFolderTypeCollection

Коллекция типов папок

NavExtension.QueryCommandStatus — метод (NavCommand, NavCommandContext)

Возвращает статус заданной команды.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual NavCommandStatus QueryCommandStatus(NavCommand command,
NavCommandContext context)
```

Параметры

command

Тип: NavCommand

Тип команды

context

Тип: NavCommandContext

Контекст команды

Возвращаемое значение

Тип: NavCommandStatus

Статус команды

Заметки

Результат работы данного метода используется Windows-клиентом при формировании интерфейса.

NavExtension.QueryFolderControl — метод (NavFolderControlType, Guid)

Возвращает для заданной папки режим работы с Windows-клиентом.

- **Пространство имён:** DocsVision.Platform.WinForms
- **Сборка:** DocsVision.Platform.WinForms.dll

Синтаксис

```
protected virtual NavFolderControlFlags QueryFolderControl(NavFolderControlType
```

```
folderType, Guid folderId)
```

Параметры

folderType

Тип: `NavFolderControlType`

Тип папки

folderId

Тип: `System.Guid`

Идентификатор папки

Возвращаемое значение

Тип: `NavFolderControlFlags`

Действие Windows-клиента

DocsVision.Platform.Wpf — пространство имён

Пространство имён `DocsVision.Platform.Wpf` содержит реализации некоторых базовых классов из `DocsVision.Platform.WinForms` для разработки пользовательских интерфейсов с использованием WPF.

Классы

Класс	Описание
`NavExtension`	Базовый класс расширения Windows-клиента для реализации пользовательского интерфейса (если предусмотрен) с использованием WPF.

DocsVision.Platform.Wpf.NavExtension — класс

Базовый класс расширения Windows-клиента для реализации пользовательского интерфейса (если предусмотрен) с использованием WPF.

- **Пространство имён:** `DocsVision.Platform.Wpf`
- **Сборка:** `DocsVision.Platform.Wpf.dll`

Синтаксис

```
[ClassInterface(ClassInterfaceType.None)]
[ComVisible(true)]
[DefaultEvent("CardLoaded")]
[Guid("2BE11F4F-2F18-40C0-814D-E99DFD13AC79")]
[ToolboxItem(false)]
public class NavExtension : CardControl, INavExtension, INavCardSubTypesExtension,
INavDragDropExtension,
            INavFolderExtTypesExtension, INavReportExtension, INavCommandExtension,
INavStandardCommandExtension,
            INavEventExtension, INavPickerExtension, INavControlExtension,
INavPropPagesExtension,
            INavCardCreatorExtension, INavCardCreatorEntryContainerExtension,
INavExtensionInitialize
```

Конструкторы

Имя	Описание
NavExtension	Инициализирует новый экземпляр класса NavExtension.

Свойства

Имя	Описание
SupportedTypes	Возвращает список элементов Windows-клиента, для которых предназначено данное расширение.

NavExtension.SupportedTypes — свойство

Возвращает список элементов Windows-клиента, для которых предназначено данное расширение.

- **Пространство имён:** DocsVision.Platform.Wpf
- **Сборка:** DocsVision.Platform.Wpf.dll

Синтаксис

```
protected virtual NavExtensionTypes SupportedTypes { get; }
```

Значение свойства

Тип: NavExtensionTypes

Тип расширения

Заметки

Свойство SupportedTypes должно быть переопределено в производном классе. В переопределении необходимо перечислить все типы, которые поддерживает расширение.

AssemblyResolver — класс

Класс AssemblyResolver реализует функцию загрузки сборок Docsvision из каталога установки Windows-клиента в runtime программы.

- **Пространство имён:** DocsVision.Platform
- **Сборка:** DocsVision.Platform.dll

Синтаксис

```
[ComVisible(true)]
[Guid("230DA85B-DC28-479E-B0B8-62E05C8E3065")]
public class AssemblyResolver
```

Конструкторы

Имя	Описание
AssemblyResolver()	Инициализирует новый экземпляр класса AssemblyResolver.

Методы

Имя	Описание
AddPath(String)	Регистрирует указанный каталог в список каталогов, из которых могут быть загружены требуемые сборки.
RemovePath(String)	Разрегистрирует указанный каталог со сборками.

Имя	Описание
`StartResolve`	Активирует функцию предоставления сборок из зарегистрированных каталогов со сборками. Автоматически вызывается из конструктора класса `AssemblyResolver`.
`StopResolve`	Останавливает функцию предоставления сборок из зарегистрированных каталогов.

Заметки

Функциональность, реализуемую классом `AssemblyResolver`, позволяет получать сущности из клиентских сборок Docsvision без необходимости загрузки этих сборок в каталог программы и GAC. Windows-клиент должен быть установлен.

Метод в `StartResolve` подписывается на событие `AppDomain.CurrentDomain.AssemblyResolve` и при необходимости предоставляет сборки, размещенные в каталоге клиента Docsvision и зарегистрированных каталогах.

Конструктор класса `AssemblyResolver` автоматически вызывает метод `StartResolve`, поэтому отдельно вызывать его не требуется.

В каталоги со сборками по умолчанию добавляется каталог установки Windows-клиента.

Для использования типов API Docsvision в программе потребуется разместить соответствующие сборки Docsvision в каталоге программы.

Примеры

Следующие код демонстрирует пример использования `AssemblyResolver`.

```
using System;
using System.IO;
using System.Reflection;
using Microsoft.Win32;

namespace ObjectContextSample
{
```

```
    internal static class ResolverInstance
    {
        private const string ClientRootKey = @"SOFTWARE\DocsVision\Platform\Client";
        private const string ClientBasePathValue = "BasePath";
        private static readonly Guid ResolverClsId = new Guid("230DA85B-DC28-479E-B0B8-
62E05C8E3065");
        private const string DocsVisionPlatformAssembly = "DocsVision.Platform.dll";
        private const string AssemblyResolverTypeName =
"DocsVision.Platform.AssemblyResolver";

        private static object resolver;

        public static void CreateResolver()
        {
            if (resolver != null)
                return;

            try
            {
                resolver = Activator.CreateInstance(Type.GetTypeFromCLSID(
ResolverClsId)); ①
            }
            catch
            {
            }

            if (resolver != null)
                return;

            string basePath = GetResolverBasePath();
            if (string.IsNullOrEmpty(basePath))
                return; ②

            try
            {
                Assembly resolverAssembly = Assembly.LoadFrom(Path.Combine(basePath,
DocsVisionPlatformAssembly));
                resolver = resolverAssembly.CreateInstance(AssemblyResolverTypeName);
            }
            catch
            {
            }
        }

        private static string GetResolverBasePath()
        {
            try
            {
```

```
                using (RegistryKey regSettings = Registry.LocalMachine.OpenSubKey
(ClientRootKey, false))
                {
                    if (regSettings != null)
                    {
                        string basePath = (string) regSettings.GetValue
(ClientBasePathValue);
                        if (!string.IsNullOrEmpty(basePath))
                            return basePath;
                    }
                }

            }
            catch
            {
            }

            try
            {
                using (RegistryKey regSettings = Registry.CurrentUser.OpenSubKey
(ClientRootKey, false))
                {
                    if (regSettings != null)
                    {
                        string basePath = (string) regSettings.GetValue
(ClientBasePathValue);
                        if (!string.IsNullOrEmpty(basePath))
                            return basePath;
                    }
                }
            }
            catch
            {
            }

            return null;
        }
    }
}
```

① Создаем по CLSID.

② Если не смогли, ищем по пути из реестра.

DocsVision.Platform.Authentication.OpenIDConnect — пространство имён

Содержит типы объектной модели расширения аутентификации OpenID Connect

Классы

Класс	Описание
`OpenIDConnectConfigurationCreator`	Создаёт конфигурацию коннектора к OpenID
`OpenIDConnectConfigurationReader`	Читает конфигурацию коннектора к OpenID
`OpenIDConnectConnectConfiguration`	Конфигурация коннектора к OpenID
`OpenIDConnectRootAuthenticationExtension`	Расширение аутентификации и управления ролями для OpenID
`OpenIDConnectSettings`	Описывает настройки подключения для расширения OpenID
`OpenIDConnectTokenHandlerExtension`	Класс для работы с токеном и пользователями OpenIDConnect
`OpenIDConnectUriWithParams`	Адрес получения параметров OpenID Connect
`OpenIDConnectWebAuthenticationExtension`	Представляет web-расширение аутентификации OpenIDConnect

OpenIDConnectConfigurationCreator — класс

Создаёт конфигурацию коннектора к OpenID

- **Пространство имён:** `DocsVision.Platform.Authentication.OpenIDConnect`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public class OpenIDConnectConfigurationCreator
```

Свойства

Имя	Описание
`ConfigurationString`	Строка конфигурации **Тип**: `System.String`

Методы

Имя	Описание
`GetConfiguration()`	Получает конфигурацию коннектора к OpenID **Возвращаемое значение**: `OpenIDConnectConnectConfiguration`

Конструкторы

Имя	Описание
`OpenIDConnectConfigurationCreator(string)`	Создаёт экземпляр класса `OpenIDConnectConfigurationCreator`. *Параметры:* • `configString` — строка конфигурации коннектора к OpenID

OpenIDConnectConfigurationReader — класс

Читает конфигурацию коннектора к OpenID

- **Пространство имён:** `DocsVision.Platform.Authentication.OpenIDConnect`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public class OpenIDConnectConfigurationCreator
```

Свойства

Имя	Описание
`ConfigurationUrl`	URL конфигурации **Тип**: `System.String`

Методы

Имя	Описание
`GetConfiguration()`	Получает конфигурацию коннектора к OpenID **Возвращаемое значение**: `OpenIDConnectConnectConfiguration`

Конструкторы

Имя	Описание
`OpenIDConnectConfigurationReader(string)`	Создаёт экземпляр класса `OpenIDConnectConfigurationReader`. *Параметры:* • `wellKnownConfigurationUrl` — URL публичной конфигурации OpenID

OpenIDConnectConnectConfiguration-- класс

Конфигурация коннектора к OpenID

- **Пространство имён:** `DocsVision.Platform.Authentication.OpenIDConnect`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public class OpenIDConnectConnectConfiguration
```

Свойства

Имя	Описание
Issuer	Идентификатор стороны, генерирующей токен **Тип**: System.String
AuthorizationEndpoint	URL для получения авторизационного кода **Тип**: System.String
TokenEndpoint	URL для получения маркера доступа **Тип**: System.String
UserInfoEndpoint	URL для получения данных пользователя (по токену) **Тип**: System.String
RevocationEndpoint	URL для отзыва токенов **Тип**: System.String
JwksUri	URL для получения публичных ключей провайдера (JSON Web Key Set URI) **Тип**: System.String
DeviceAuthorizationEndpoint	URL страницы авторизации для устройств **Тип**: System.String

Имя	Описание
`EndSessionEndpoint`	URL для выхода из учётной записи **Тип**: `System.String`
`CertificateThumbprint`	Отпечаток сертификата для работы коннектора **Тип**: `System.String`
`CertificatePassword`	Пароль сертификата для работы коннектора. Пароль обычно устанавливается при импорте файла `.pfx` в Linux. Если пароль на сертификат не установлен, этот параметр можно удалить. **Тип**: `System.String`
`CertificateHash`	Хэш сертификата получаемый через утилиту cpverify **Тип**: `System.String`
`ExtCertificateThumbprint`	Отпечаток сертификата площадки. Можно посмотреть при выполнении `sudo /opt/cprocsp/bin/amd64/certmgr -list` **Тип**: `System.String`
`RequestUriParameterSupported`	Поддерживается ли параметр запроса URI **Тип**: `System.Boolean`

Имя	Описание
HttpLogoutSupported	Поддерживается ли выход из системы напрямую по HTTP (бэк) **Тип**: System.Boolean
FrontChannelLogoutSupported	Поддерживается ли выход из системы по фронтовому каналу (через браузер пользователя) **Тип**: System.Boolean
ResponseTypeSupported	Поддерживаемые типы ответа **Тип**: System.Collections.Generic.IEnumerable
SubjectTypesSupported	Поддерживаемые типы субъектов **Тип**: System.Collections.Generic.IEnumerable<System.String>
ScopesSupported	Список поддерживаемых областей доступа, т.е. запрашиваемых прав **Тип**: System.Collections.Generic.IEnumerable<System.String>
TokenEndpointAuthMethodsSupported	Список поддерживаемых методов аутентификации при запросе токенов **Тип**: System.Collections.Generic.IEnumerable<System.String>

Имя	Описание
`ClaimsSupported`	Список поддерживаемых данных пользователя для запроса при авторизации **Тип**: `System.Collections.Generic.IEnumerable<System.String>`
`GrantTypesSupported`	Список поддерживаемых типов авторизации **Тип**: `System.Collections.Generic.IEnumerable<System.String>`
`CodeChallengeMethodsSupported`	Список поддерживаемых методов защиты PKCE **Тип**: `System.Collections.Generic.IEnumerable<System.String>`
`IdTokenSigningAlgValuesSupported`	Список поддерживаемых алгоритмов подписи токена **Тип**: `System.Collections.Generic.IEnumerable<System.String>`
`ResponseModesSupported`	Список поддерживаемых режимов ответа **Тип**: `System.Collections.Generic.IEnumerable<System.String>`

OpenIDConnectRootAuthenticationExtension — класс

Расширение аутентификации и управления ролями для OpenID

- **Пространство имён:** DocsVision.Platform.Authentication.OpenIDConnect
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public class OpenIDConnectRootAuthenticationExtension : BaseRootAuthenticationExtension
```

Свойства

Имя	Описание
Name	Базовое расширение аутентификации OpenID **Тип**: System.String
Description	Описание: базовое расширение аутентификации OpenID **Тип**: System.String
SupportedTypes	Поддерживаемые типы расширений аутентификации **Тип**: AuthenticationExtensionType **Возвращаемое значение**: комбинация типов расширения аутентификации

Методы

Имя	Описание
GetLogo()	Получает логотип **Тип**: System.Byte[] **Возвращаемое значение**: Logo — логотип

Имя	Описание
`GetExtensionType(AuthenticationExtensionType)`	Получает тип расширения *Параметры:* • `extensionType` — тип расширения
`GetSettings<T>(AuthenticationExtensionType)`	Получает настройки определённого типа *Параметры:* • `extensionType` — тип расширения **Возвращаемое значение**: `T` — настройки определённого типа
`GetSettingsType<T>(AuthenticationExtensionType)`	Получает тип настроек *Параметры:* • `extensionType` — тип расширения **Возвращаемое значение**: `Type` — тип настроек

OpenIDConnectSettings — класс

Описывает настройки подключения для расширения OpenID

- **Пространство имён:** `DocsVision.Platform.Authentication.OpenIDConnect`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public class OpenIDConnectSettings
```

Свойства

Имя	Описание
`WellKnownConfigurationUrl`	URL публичной конфигурации OpenID **Тип**: `System.String`
`ClientId`	Идентификатор клиента приложения, в котором производится привязка пользователей **Тип**: `System.String`
`Secret`	Секрет приложения **Тип**: `System.String`
`AccountNameClaim`	Имя пользователя для запроса при авторизации **Тип**: `System.String`
`LoginHintClaim`	Подсказка к логину из информации для входа пользователя **Тип**: `System.String`
`UniqueIdClaim`	Неизменяемый идентификатор пользователя, запрашиваемый при авторизации **Тип**: `System.String`
`TraceErrors`	Ошибки отладки **Тип**: `System.Boolean`
`TracePI`	Позволяется ли включать персональную информацию в отладке **Тип**: `System.Boolean`

OpenIDConnectTokenHandlerExtension — класс

Класс для работы с токеном и пользователями OpenIDConnect

- **Пространство имён:** DocsVision.Platform.Authentication.OpenIDConnect
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public class OpenIDConnectTokenHandlerExtension : BaseAuthenticationExtension,
ITokenHandlerExtension
```

Свойства

Имя	Описание
Name	Расширение валидации токена OpenIDConnect
Type	Тип расширения аутентификации **Тип**: AuthenticationExtensionType
Settings	Настройки OpenIDConnect **Тип**: OpenIDConnectSettings
Configuration	Конфигурация подключения OpenIDConnect **Тип**: OpenIDConnectConnectConfiguration

Методы

Имя	Описание
`ValidateToken(string)`	Проверяет действительность токена *Параметры:* • `token` — токен авторизации **Возвращаемое значение**: `IPrincipal` — аутентифицированный пользователь
`GetPrincipalName(IPrincipal)`	Получает информацию об авторизованном пользователе *Параметры:* • `principal` — объект с информацией об авторизованном пользователе **Возвращаемое значение**: `System.String` — имя пользователя из объекта аутентификации
`GetLoginHint(IPrincipal)`	Получает подсказку для логина пользователя, которую можно использовать при следующем входе в систему *Параметры:* • `principal` — объект с информацией об авторизованном пользователе **Возвращаемое значение**: `System.String` — подсказка для логина пользователя

Имя	Описание
`GetPrincipalUniqueID(IPrincipal)`	Получает уникальный идентификатор авторизованного пользователя *Параметры:* • `principal` — объект с информацией об авторизованном пользователе **Возвращаемое значение**: `System.String` — строковый идентификатор авторизованного пользователя
`GetPrincipalRoles(IPrincipal)`	Получает роли (группы) пользователей *Параметры:* • `principal` — объект с информацией об авторизованном пользователе **Возвращаемое значение**: `UserRoles` — группы пользователей сервера Docsvision

OpenIDConnectUriWithParams — класс

Адрес получения параметров OpenID Connect

- **Пространство имён:** `DocsVision.Platform.Authentication.OpenIDConnect`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public class OpenIDConnectUriWithParams
```

Свойства

Имя	Описание
`Uri`	URL для запроса параметров **Тип**: `System.String`
`RequestedParams`	Запрашиваемые параметры **Тип**: List<KeyValuePair<string, string>> — список параметров, состоящий из пар "ключ и значение"

Конструкторы

Имя	Описание
`OpenIDConnectUriWithParams(string, List<KeyValuePair<string, string>>)`	Создаёт экземпляр класса `OpenIDConnectUriWithParams`. *Параметры:* • `uri` — URL для запроса параметров • `requestedParams` — параметры запроса

OpenIDConnectWebAuthenticationExtension — класс

Представляет web-расширение аутентификации OpenIDConnect

- **Пространство имён:** `DocsVision.Platform.Authentication.OpenIDConnect`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public class OpenIDConnectWebAuthenticationExtension : BaseWebAuthenticationExtension
```

Свойства

Имя	Описание
`Name`	Расширение аутентификации OpenIDConnect
`Settings`	Настройки OpenIDConnect **Тип**: `OpenIDConnectSettings`
`Configuration`	Конфигурация подключения OpenIDConnect **Тип**: `OpenIDConnectConnectConfiguration`

Методы

Имя	Описание
`GetLoginUrl(string, string, string)`	Получает URL страницы входа *Параметры:* • `redirectUrl` — страница, на которую пользователь будет перенаправлен • `responseType` — тип ответа • `responseMode` — режим ответа **Возвращаемое значение**: `System.String` — URL страницы входа

Имя	Описание
`GetLogoutUrl(string, string, string)`	Получает URL страницы выхода *Параметры:* • `tokenId` — идентификатор токена • `redirectUri` — страница для переадресации • `logoutHint` — подсказка для для логина пользователя при выходе из системы **Возвращаемое значение**: `System.String` — URL страницы выхода
`FeatureName()`	Получает имя опции лицензии **Возвращаемое значение**: `System.String` — имя опции лицензии
`FeatureId()`	Идентификатор опции лицензии **Возвращаемое значение**: `System.Guid` — идентификатор опции лицензии

Имя	Описание
`FindLoginType(string)`	Определяет тип логина *Параметры:* • `value` — тип логина **Возвращаемое значение**: `AuthenticationLoginType` — перечисление: • `Phone = 0` • `Snils = 1` • `Email = 2` • `Custom = 3` • `Upn = 4` — уникальный идентификатор пользователя в домене (УЗ в формате UPN)
`GetTimeOutForExtension()`	Получает таймаут для расширения **Возвращаемое значение**: `System.Int32` — таймаут для расширения
`AddLoginHint(string)`	Добавляет подсказку для логина пользователя *Параметры:* • `login_hint` — подсказка для логина пользователя **Возвращаемое значение**: `System.String` — подсказка для логина пользователя

Имя	Описание
`AddStateGuid()`	Определяет, нужно ли добавлять идентификатор состояния в запрос аутентификации. **Возвращаемое значение**: `System.Boolean` — признак наличия идентификатора
`GetLogoutHintPart(string)`	Получает часть URL-адреса, содержащую параметр подсказки для логина пользователя при выходе из системы *Параметры:* • `logoutHint` — подсказка для логина пользователя при выходе из системы **Возвращаемое значение**: `System.String` — подсказка для логина
`GetReturnUrl(string)`	Получает адрес для перенаправления пользователя после успешной аутентификации *Параметры:* • `defaultUri` — адрес для возврата **Возвращаемое значение**: `System.String` — адрес для возврата

Имя	Описание
`GetTokenFromCode(string[])`	Асинхронный метод, получающий токен из кода *Параметры:* • `args` — параметры запроса для получения токена **Возвращаемое значение**: `Task<ExtensionTokenModel>` — класс, представляющий модель токена расширения: • `access_token` — получить токен • `refresh_token` — обновить • `id_token` — идентификатор • `state` — состояние • `session_state` — состояние сессии • `token_type` — тип токена • `expires_in` — когда истекает срок действия
`GetAccessCodeUrl(string, string)`	Получает URL кода доступа *Параметры:* • `redirectUrl` — адрес для перенаправления пользователя • `code` — код доступа **Возвращаемое значение**: `System.String` — код доступа

Имя	Описание
`GetAccessTokenUrl(string, string)`	Получает URL токена доступа *Параметры:* • `redirectUrl` — адрес для перенаправления пользователя • `code` — код доступа **Возвращаемое значение**: `System.String` — URL токена доступа

Все методы являются виртуальными и могут быть переопределены в классах-наследниках.

DocsVision.Platform.Authentication.Settings — пространство имён

Содержит глобальные типы объектной модели расширений аутентификации

Классы

Класс	Описание
AuthenticationExtensionSettings	Настройки расширения аутентификации
AuthenticationSettings	Представляет общие настройки расширений аутентификации
AuthenticationExtensionsSettingsReader	Предоставляет глобальные методы чтения и записи настроек аутентификации, всех экземпляров расширений для всех тенантов
BaseAuthenticationExtensionSettings	Настройки для web-расширения аутентификации
AuthenticationExtensionsSettingsRestReader	Предоставляет методы чтения и записи для настроек аутентификации, всех экземпляров расширений для всех тенантов, работает через REST API
AuthenticationExtensionsSettings	Настройки расширений аутентификации и их привязки к тенантам
AuthenticationExtensionsSettingsRegistryReader	Предоставляет методы чтения и записи для настроек аутентификации, всех экземпляров расширений для всех тенантов, работает с реестром Windows
TenantAuthenticationSettings	Настройки аутентификации для тенанта

Интерфейсы

Интерфейс	Описание
IAuthenticationExtensionsSettingsReader	Интерфейс для чтения настроек расширений аутентификации

AuthenticationExtensionSettings — класс

Настройки расширения аутентификации

- **Пространство имён:** DocsVision.Platform.Authentication.Settings
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public class AuthenticationExtensionSettings
```

Свойства

Имя	Описание
Id	Идентификатор расширения **Тип**: System.Guid
Name	Имя расширения **Тип**: System.String
TypeName	Имя типа реализации расширения **Тип**: System.String
Settings	Настройки расширения **Тип**: System.String

AuthenticationExtensionsSettings — класс

Настройки расширений аутентификации и их привязки к тенантам

- **Пространство имён:** DocsVision.Platform.Authentication.Settings
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public class AuthenticationExtensionsSettings
```

Свойства

Имя	Описание
Extensions	Расширения **Тип**: List<AuthenticationExtensionSettings>
Tenants	Тенанты **Тип**: List<TenantAuthenticationSettings>

AuthenticationExtensionsSettingsReader — класс

Предоставляет глобальные методы чтения и записи настроек аутентификации, всех экземпляров расширений для всех тенантов

- **Пространство имён:** DocsVision.Platform.Authentication.Settings
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public class AuthenticationExtensionsSettingsReader :
IAuthenticationExtensionsSettingsReader
```

Методы

Имя	Описание
`GetAuthenticationExtensionSettings(Guid)`	Получает настройки конкретного расширения аутентификации по его GUID *Параметры:* • `id` — идентификатор расширения аутентификации **Возвращаемое значение**: `AuthenticationExtensionSettings` — настройки расширения аутентификации
`GetSettings()`	Получает глобальные настройки всех расширений аутентификации **Возвращаемое значение**: `AuthenticationExtensionSettings` — настройки расширения аутентификации
`GetTenantAuthenticationExtensionSettings(string)`	Получает настройки расширения аутентификации для конкретного тенанта *Параметры:* • `tenant` — тенант **Возвращаемое значение**: `IEnumerable<AuthenticationExtensionSettings>` — список настроек расширения аутентификации

Имя	Описание
`SetAuthenticationExtensionSettings(AuthenticationExtensionSettings)`	Задаёт расширения аутентификации для тенанта *Параметры:* • `settings` — настройки расширения аутентификации
`SetTenantAuthenticationExtensions(TenantAuthenticationSettings)`	Задаёт расширение аутентификации для тенанта *Параметры:* • `settings` — настройки расширения аутентификации

Конструкторы

Имя	Описание
`AuthenticationExtensionsSettingsReader(IConfiguration)`	Создаёт экземпляр класса `AuthenticationExtensionsSettingsReader` *Параметры:* • `configuration` — конфигурация расширения аутентификации

AuthenticationExtensionsSettingsRegistryReader — класс

Предоставляет методы чтения и записи для настроек аутентификации, всех экземпляров расширений для всех тенантов, работает с реестром Windows

- **Пространство имён:** `DocsVision.Platform.Authentication.Settings`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public class AuthenticationExtensionsSettingsRegistryReader :
IAuthenticationExtensionsSettingsReader
```

Методы

Имя	Описание
`GetAuthenticationExtensionSettings(Guid)`	Получает настройки конкретного расширения аутентификации по его GUID *Параметры:* • `id` — идентификатор расширения аутентификации **Возвращаемое значение**: `AuthenticationExtensionSettings` — настройки расширения аутентификации
`GetSettings()`	Получает глобальные настройки всех расширений аутентификации **Возвращаемое значение**: `AuthenticationExtensionSettings` — настройки расширения аутентификации
`GetTenantAuthenticationExtensionSettings(string)`	Получает настройки расширения аутентификации для конкретного тенанта *Параметры:* • `tenant` — тенант **Возвращаемое значение**: `IEnumerable<AuthenticationExtensionSettings>` — список настроек расширения аутентификации

Имя	Описание
`SetAuthenticationExtensionSettings(AuthenticationExtensionSettings)`	Задаёт настройки для конкретного экземпляра расширения аутентификации *Параметры:* • `settings` — настройки аутентификации
`SetTenantAuthenticationExtensions(TenantAuthenticationSettings)`	Задаёт расширения аутентификации для тенанта *Параметры:* • `settings` — настройки аутентификации

Конструкторы

Имя	Описание
`AuthenticationExtensionsSettingsRegistryReader()`	Создаёт новый экземпляр класса `AuthenticationExtensionsSettingsRegistryReader` без параметров
`AuthenticationExtensionsSettingsRegistryReader(SignedLicense)`	Создаёт новый экземпляр класса `AuthenticationExtensionsSettingsRegistryReader` с указанной лицензией *Параметры:* • `_license` — лицензия

AuthenticationExtensionsSettingsRestReader — класс

Предоставляет методы чтения и записи для настроек аутентификации, всех экземпляров расширений для всех тенантов, работает через REST API

- **Пространство имён:** `DocsVision.Platform.Authentication.Settings`

- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public class AuthenticationExtensionsSettingsRestReader :
IAuthenticationExtensionsSettingsReader
```

Методы

Имя	Описание
GetAuthenticationExtensionSettings(Guid)	Получает настройки конкретного расширения аутентификации по его GUID *Параметры:* • id — идентификатор расширения аутентификации **Возвращаемое значение**: AuthenticationExtensionSettings — настройки расширения аутентификации
GetSettings()	Получает глобальные настройки всех расширений аутентификации **Возвращаемое значение**: AuthenticationExtensionSettings — настройки расширения аутентификации

Имя	Описание
`IEnumerable<AuthenticationExtensionSettings> GetTenantAuthenticationExtensionSettings(string tenant)`	Получает настройки расширения аутентификации для конкретного тенанта *Параметры:* • `tenant` — тенант **Возвращаемое значение**: `IEnumerable<AuthenticationExtensionSettings>` — список настроек расширения аутентификации
`SetAuthenticationExtensionSettings(AuthenticationExtensionSettings)`	Задаёт настройки для конкретного экземпляра расширения аутентификации *Параметры:* • `settings` — настройки аутентификации
`SetTenantAuthenticationExtensions(TenantAuthenticationSettings)`	Задаёт расширения аутентификации для тенанта *Параметры:* • `settings` — настройки аутентификации

Конструкторы

Имя	Описание
`AuthenticationExtensionsSettingsRestReader(string)`	Создаёт новый экземпляр класса `AuthenticationExtensionsSettingsRestReader` .Параметры: * `serverUrl` — URL сервера

AuthenticationSettings — класс

Представляет общие настройки расширений аутентификации

- **Пространство имён:** DocsVision.Platform.Authentication.Settings
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public class AuthenticationSettings
```

Свойства

Имя	Описание
ProviderName	Имя провайдера **Тип**: System.String
Logo	Логотип провайдера **Тип**: System.String
Settings	Сериализованные общие настройки **Тип**: System.String
Extensions	Список зарегистрированных расширений аутентификации **Тип**: List<AuthenticationExtensionSettings>

BaseAuthenticationExtensionSettings — класс

Настройки для web-расширения аутентификации

- **Пространство имён:** DocsVision.Platform.Authentication.Settings
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public class BaseAuthenticationExtensionSettings
```

Свойства

Имя	Описание
ProviderUrl	URL провайдера аутентификации **Тип**: System.String
AuthenticationHeader	HTTP-заголовок для передачи токена **Тип**: System.String
TenantId	Идентификатор тенанта, при необходимости **Тип**: System.String
ClientId	Идентификатор клиента, при необходимости **Тип**: System.String
RefreshTokenUrl	URL для обновления токена **Тип**: System.String
SigningKeysInfo	Информация для получения ключей подписания токена **Тип**: System.String

IAuthenticationExtensionsSettingsReader — интерфейс

Интерфейс для чтения настроек расширений аутентификации

- **Пространство имён:** DocsVision.Platform.Authentication.Settings

- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public interface IAuthenticationExtensionsSettingsReader
```

Методы

Имя	Описание
GetSettings()	Получает все настройки расширения по GUID **Возвращаемое значение**: AuthenticationExtensionSettings — настройки расширения аутентификации
GetAuthenticationExtensionSettings(Guid)	Получает настройки расширения аутентификации *Параметры:* • id — идентификатор расширения аутентификации **Возвращаемое значение**: AuthenticationExtensionSettings — настройки расширения аутентификации
SetAuthenticationExtensionSettings(AuthenticationExtensionSettings)	Задаёт настройки расширения аутентификации *Параметры:* • settings — настройки расширения аутентификации

Имя	Описание
`GetTenantAuthenticationExtensionSettings(string)`	Получает настройки расширения аутентификации для конкретного тенанта *Параметры:* • `tenant` — тенант **Возвращаемое значение**: `IEnumerable<AuthenticationExtensionSettings>` — список настроек расширения аутентификации
`SetTenantAuthenticationExtensions(TenantAuthenticationSettings)`	Задаёт расширения аутентификации для тенанта *Параметры:* • `settings` — настройки расширения аутентификации

TenantAuthenticationSettings — класс

Настройки аутентификации для тенанта

- **Пространство имён:** `DocsVision.Platform.Authentication.Settings`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public class TenantAuthenticationSettings
```

Свойства

Имя	Описание
`Tenant`	Тенант **Тип**: `System.String`

Имя	Описание
`AuthenticationExtensions`	Расширение аутентификации **Тип**: `List<System.Guid>`

DocsVision.Platform.Authentication — пространство имён

Содержит базовые типы объектной модели расширений аутентификации

Классы

Класс	Описание
AuthenticationExtensionFactory	Фабрика расширений аутентификации
AuthenticationManager	Класс управления аутентификацией
BaseAuthenticationExtension	Базовый класс для расширений аутентификации
BaseRootAuthenticationExtension	Базовый класс для фабрики расширений аутентификации
BaseTokenAuthenticator	Базовый абстрактный аутентификатор по токенам
BaseTokenContainer	Базовый контейнер для токенов
BaseWebAuthenticationExtension	Базовый абстрактный класс web-расширения аутентификации
InMemoryTokenCache	Кэш токена в памяти

Интерфейсы

Интерфейс	Описание
IAuthenticationExtension	Предоставляет методы для провайдеров аутентификации, работающих по схеме
IBasicAuthenticationExtension	Предоставляет методы для провайдеров базовой аутентификации по логину и паролю
IGroupMembershipChecker	Предоставляет методы для проверки наличия пользователя в группе
IGroupMembershipProvider	Предоставляет методы для получения групп пользователей
IRootAuthenticationExtension	Предоставляет свойства и методы для регистрации базового расширения аутентификации
ITokenCache	Предоставляет методы для кэширования токена
ITokenHandlerExtension	Предоставляет методы для работы с токеном
IUserRolesProviderExtension	Предоставляет методы для получения пользовательских ролей

Перечисления

Перечисление	Описание
AuthenticationExtensionType	Типы расширений аутентификации

AuthenticationExtensionFactory — класс

Фабрика расширений аутентификации

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public class AuthenticationExtensionFactory
```

Свойства

Имя	Описание
`ServerAddress`	Адрес сервера **Тип**: `System.String` — URL
`LicenseData`	Данные лицензии **Тип**: `SignedLicense` — подписанная лицензия

Методы

Имя	Описание
`CreateAuthenticationExtension<T>(Guid, AuthenticationExtensionType, string)`	Создаёт расширение аутентификации конкретного типа *Параметры:* • `extensionInstanceId` — идентификатор экземпляра расширения • `extensionType` — тип расширения аутентификации • `extensionLocalSettings` — локальные настройки расширения **Возвращаемое значение**: `T` — расширение аутентификации
`CreateAuthenticationExtension(Guid, AuthenticationExtensionType, string)`	Создаёт расширение аутентификации *Параметры:* • `extensionInstanceId` — идентификатор экземпляра расширения • `extensionType` — тип расширения аутентификации • `extensionLocalSettings` — локальные настройки расширения **Возвращаемое значение**: `IAuthenticationExtension` — расширение аутентификации

Имя	Описание
`GetAuthenticationExtensionsSettings(string)`	Получает настройки расширения аутентификации *Параметры:* • `tenantName` — имя тенанта **Возвращаемое значение**: `IEnumerable<AuthenticationExtensionSettings>` —
`CreateRootAuthenticationExtensions(string, AuthenticationExtensionType)`	Создаёт базовое расширение аутентификации *Параметры:* • `tenantName` — имя тенанта • `supportedExtensionType` — тип поддерживаемого расширения аутентификации **Возвращаемое значение**: `IEnumerable<IRootAuthenticationExtension>` — базовое расширение аутентификации
`CreateAuthenticationExtensions<T>(AuthenticationExtensionType)`	Тип расширения аутентификации *Параметры:* • `supportedExtensionType` — тип поддерживаемого расширения аутентификации **Возвращаемое значение**: `IEnumerable<T>` — список расширений аутентификации типа `T`

Конструкторы

Имя	Описание
`AuthenticationExtensionFactory(IConfiguration)`	Создаёт экземпляр класса `AuthenticationExtensionFactory` *Параметры:* • `configuration` — конфигурация расширения аутентификации
`AuthenticationExtensionFactory(IConfiguration, string, string)`	Создаёт экземпляр класса `AuthenticationExtensionFactory` *Параметры:* • `configuration` — конфигурация расширения аутентификации • `serverAddress` — адрес сервера • `licenseData` — данные лицензии

AuthenticationExtensionType — перечисление

Типы расширений аутентификации

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public enum AuthenticationExtensionType
```

Члены

Имя	Описание
`Root = 0`	Базовое расширение

Имя	Описание
`DesktopClient = 1`	Расширение для интерактивных десктопных клиентов
`WebClient = 2`	Расширение для интерактивных web-клиентов
`ServiceClient = 4`	Расширение для сервисных неинтерактивных приложений
`TokenHandler = 8`	Расширение для пользовательской логики проверки токена
`WebApplicationInitialize = 16`	Расширение для инициализации web-приложения
`BasicAuthentication = 32`	Расширение для базовой аутентификации
`UserRolesProvider = 64`	Расширение, предоставляющее пользовательские роли

AuthenticationManager — класс

Класс управления аутентификацией

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public class AuthenticationManager
```

Методы

Имя	Описание
`CreateAuthenticationExtensions<T>(AuthenticationExtensionType)`	Создаёт расширения аутентификации заданных типов *Параметры:* • `supportedExtensionType` — тип поддерживаемого расширения аутентификации **Возвращаемое значение**: `IEnumerable<T>` — тип расширения аутентификации
`GetAuthenticationExtensions<T>(AuthenticationExtensionType)`	Получает расширения аутентификации заданных типов *Параметры:* • `supportedExtensionType` — тип поддерживаемого расширения аутентификации **Возвращаемое значение**: `IEnumerable<T>` — тип расширения аутентификации

Имя	Описание
`TryGetPrincipal(Guid, string)`	Пытается получить объект с информацией об авторизованном пользователе *Параметры:* • `authProviderId` — идентификатор провайдера аутентификации • `token` — токен **Возвращаемое значение**: `IPrincipal` — объект с информацией об авторизованном пользователе

BaseAuthenticationExtension — класс

Базовый класс для расширений аутентификации

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public abstract class BaseAuthenticationExtension : IAuthenticationExtension
```

Свойства

Имя	Описание
`InstanceId`	Идентификатор расширения аутентификации **Тип**: `System.Guid`
`Name`	Наименование расширения **Тип**: `System.String`

Имя	Описание
`Type`	Тип расширения аутентификации **Тип**: `AuthenticationExtensionType`
`CommonSettings`	Общие глобальные настройки аутентификации **Тип**: `AuthenticationSettings`
`ExtensionSettings`	Специфичные настройки расширения **Тип**: `System.String`
`LocalSettings`	Специфичные локальные настройки **Тип**: `System.String`

Методы

Имя	Описание
`Initialize(Guid, string, string, string)`	Инициализирует расширение аутентификации *Параметры:* • `instanceId` — идентификатор конкретного экземпляра расширения • `commonSettings` — общие глобальные настройки аутентификации • `extensionSettings` — специфичные настройки расширения • `localSettings` — специфичные локальные настройки

BaseRootAuthenticationExtension — класс

Базовый класс для фабрики расширений аутентификации

- **Пространство имён:** DocsVision.Platform.Authentication
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public abstract class BaseRootAuthenticationExtension : BaseAuthenticationExtension,
IRootAuthenticationExtension
```

Свойства

Имя	Описание
Type	Тип расширения аутентификации **Тип**: AuthenticationExtensionType
Description	Описание расширения **Тип**: System.String
Logo	Логотип расширения **Тип**: System.Byte[]
SupportedTypes	Поддерживаемые типы расширений **Тип**: AuthenticationExtensionType

Методы

Имя	Описание
`GetExtensionType(AuthenticationExtensionType)`	Получает тип расширения аутентификации *Параметры:* • `extensionType` — тип расширения аутентификации **Возвращаемое значение**: `Type` — тип расширения аутентификации
`GetSettings<T>(AuthenticationExtensionType)`	Получает настройки конкретного типа *Параметры:* • `extensionType` — тип расширения аутентификации **Возвращаемое значение**: `T` — настройки конкретного типа
`GetSettingsType<T>(AuthenticationExtensionType)`	Получает тип настроек *Параметры:* • `extensionType` — тип расширения аутентификации **Возвращаемое значение**: `Type` — определённый тип настроек

BaseTokenAuthenticator — класс

Базовый абстрактный аутентификатор по токенам

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public abstract class BaseTokenAuthenticator : BaseTokenContainer, ITokenAuthenticator,
DocsVision.Platform.Interop.ITokenAuthenticator
```

Методы

Имя	Описание
`Authenticate()`	Выполняет аутентификацию через интерфейс `ITokenAuthenticator`
`DocsVision.Platform.Interop.ITokenAuthenticator.Authenticate()`	Выполняет аутентификацию через интерфейс `DocsVision.Platform.Interop.ITokenAuthenticator`

BaseTokenContainer — класс

Базовый контейнер для токенов

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public class BaseTokenContainer : ITokenContainer, DocsVision.Platform.Interop
.ITokenContainer
```

Свойства

Имя	Описание
`AuthenticationProviderId`	Идентификатор провайдера расширения аутентификации для интерфейса `ITokenContainer`
`AuthenticationToken`	Токен аутентификации для интерфейса `ITokenContainer`

Имя	Описание
`Interop.ITokenContainer.AuthenticationToken`	Токен аутентификации для интерфейса `DocsVision.Platform.Interop.ITokenContainer`
`Interop.ITokenContainer.AuthenticationProviderID`	Идентификатор провайдера расширения аутентификации для интерфейса `DocsVision.Platform.Interop.ITokenContainer`

Конструкторы

Имя	Описание
`BaseTokenContainer(Guid, string)`	Инициализирует новый экземпляр класса `BaseTokenContainer` *Параметры:* • `providerInstanceId` — идентификатор экземпляра провайдера • `token` — токен

Методы

Имя	Описание
`UpdateToken(string token)`	Обновляет токен *Параметры:* • `token` — токен

BaseWebAuthenticationExtension — класс

Базовый абстрактный класс web-расширения аутентификации

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public abstract class BaseWebAuthenticationExtension : BaseAuthenticationExtension
```

Свойства

Имя	Описание
`Type`	Тип расширения аутентификации **Тип**: `AuthenticationExtensionType`
`SupportsAccountCheck`	Поддерживается ли проверка существования учётной записи **Тип**: `System.Boolean`

Методы

Имя	Описание
`CheckAccount(string)`	Проверяет существование учётной записи *Параметры:* • `accountName` — имя учётной записи
`GetTokenContainer(string)`	Получает контейнер токена *Параметры:* • `token` — токен **Возвращаемое значение**: `ITokenContainer` — контейнер токена
`UpdateTokenContainer(ITokenContainer, string)`	Обновляет контейнер токена *Параметры:* • `container` — контейнер токена • `token` — токен

IAuthenticationExtension — интерфейс

Предоставляет методы для провайдеров аутентификации, работающих по схеме

- **Пространство имён:** DocsVision.Platform.Authentication
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public interface IAuthenticationExtension
```

Свойства

Имя	Описание
InstanceId	Идентификатор экземпляра провайдера **Тип**: System.Guid
Name	Имя расширения **Тип**: System.String
Type	Тип расширения аутентификации **Тип**: AuthenticationExtensionType
CommonSettings	Общие глобальные настройки аутентификации **Тип**: AuthenticationSettings
ExtensionSettings	Специфичные настройки расширения **Тип**: System.String
LocalSettings	Специфичные локальные настройки **Тип**: System.String

Методы

Имя	Описание
`Initialize(Guid, string, string, string)`	Инициализирует интерфейс с заданными параметрами *Параметры:* • `instanceId` — идентификатор экземпляра провайдера • `commonSettings` — общие глобальные настройки аутентификации • `extensionSettings` — специфичные настройки расширения • `localSettings` — специфичные локальные настройки

IBasicAuthenticationExtension — интерфейс

Предоставляет методы для провайдеров базовой аутентификации по логину и паролю

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public interface IBasicAuthenticationExtension
```

Методы

Имя	Описание
`Login(string, string)`	Выполняет логин пользователя *Параметры:* • `accountName` — имя учётной записи • `password` — пароль учётной записи **Возвращаемое значение**: `IPrincipal` — аутентифицированный пользователь
`UserExists(string)`	Проверяет существование УЗ *Параметры:* • `accountName` — имя учётной записи **Возвращаемое значение**: `System.Boolean` — существует ли пользователь
`TryGetSID(string)`	Получает идентификатор безопасности пользователя (SID) из LDAP *Параметры:* • `accountName` — имя учётной записи **Возвращаемое значение**: `System.String` — идентификатор безопасности пользователя (SID) из LDAP

Имя	Описание
`GetPrincipalName(IPrincipal)`	Получает имя учетной записи *Параметры:* • `principal` — объект с информацией об авторизованном пользователе **Возвращаемое значение**: `System.String` — имя пользователя из объекта аутентификации

IGroupMembershipChecker — интерфейс

Предоставляет методы для проверки наличия пользователя в группе

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public interface IGroupMembershipChecker
```

Методы

Имя	Описание
`IsInGroup(string)`	Проверяет, состоит ли пользователь в группе *Параметры:* • `groupName` — название группы **Возвращаемое значение**: `System.Boolean` — состоит ли пользователь в группе

IGroupMembershipProvider — интерфейс

Предоставляет методы для получения групп пользователей

- **Пространство имён:** DocsVision.Platform.Authentication
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public interface IGroupMembershipProvider
```

Методы

Имя	Описание
GetGroups(string, string)	Получает группы *Параметры:* • token — токен • upn — уникальный идентификатор пользователя в домене (УЗ в формате UPN) **Возвращаемое значение**: IEnumerable<System.String> — список групп пользователя

InMemoryTokenCache — класс

Кэш токена в памяти

- **Пространство имён:** DocsVision.Platform.Authentication
- **Сборка:** DocsVision.Authentication.dll

Синтаксис

```
public class InMemoryTokenCache : ITokenCache
```

Методы

Имя	Описание
`GetToken(Guid, string)`	Получает токен *Параметры:* • `providerInstanceId` — идентификатор экземпляра провайдера • `tokenId` — идентификатор токена **Возвращаемое значение**: `System.String` — токен расширения аутентификации
`StoreToken(Guid, string, string)`	Помещает токен в хранилище *Параметры:* • `providerInstanceId` — идентификатор экземпляра провайдера • `tokenId` — идентификатор токена • `token` — токен

IRootAuthenticationExtension — интерфейс

Предоставляет свойства и методы для регистрации базового расширения аутентификации

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public interface IRootAuthenticationExtension : IAuthenticationExtension
```

Свойства

Имя	Описание
`Description`	Описание расширения **Тип**: `System.String`
`Logo`	Логотип расширения **Тип**: `System.Byte[]`
`SupportedTypes`	Поддерживаемые типы расширений аутентификации **Тип**: `AuthenticationExtensionType`

Методы

Имя	Описание
`GetSettingsType<T>(AuthenticationExtensionType)`	Получает тип настроек *Параметры:* • `extensionType` — тип расширения аутентификации **Возвращаемое значение**: `Type`-- тип настроек
`GetSettings<T>(AuthenticationExtensionType)`	Получает настройки определённого типа *Параметры:* • `extensionType` — тип расширения аутентификации **Возвращаемое значение**: `T` — настройки определённого типа

Имя	Описание
`GetExtensionType(AuthenticationExtensionType)`	Получает тип расширения *Параметры:* • `extensionType` — тип расширения аутентификации **Возвращаемое значение**: `Type`-- тип расширения

ITokenCache — интерфейс

Предоставляет методы для кэширования токена

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public interface ITokenCache
```

Методы

Имя	Описание
`StoreToken(Guid, string, string)`	Помещает токен в хранилище *Параметры:* • `providerInstanceId` — идентификатор экземпляра провайдера • `tokenId` — идентификатор токена • `token` — токен

Имя	Описание
`GetToken(Guid, string)`	Получает токен *Параметры:* • `providerInstanceId` — идентификатор экземпляра провайдера • `tokenId` — идентификатор токена **Возвращаемое значение**: `System.String` — токен расширения аутентификации

ITokenHandlerExtension — интерфейс

Предоставляет методы для работы с токеном

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public interface ITokenHandlerExtension
```

Методы

Имя	Описание
`ValidateToken(string)`	Проверяет действительность токена *Параметры:* • `token` — токен авторизации **Возвращаемое значение**: `IPrincipal` — аутентифицированный пользователь

Имя	Описание
`GetPrincipalName(IPrincipal)`	Получает имя учетной записи *Параметры:* • `principal` — объект с информацией об авторизованном пользователе **Возвращаемое значение**: `System.String` — группы пользователей сервера Docsvision
`GetLoginHint(IPrincipal)`	Получает подсказку для логина пользователя, которую можно использовать при следующем входе в систему *Параметры:* • `principal` — объект с информацией об авторизованном пользователе **Возвращаемое значение**: `System.String` — подсказка для логина пользователя

Имя	Описание
`GetPrincipalUniqueID(IPrincipal)`	Получает уникальный идентификатор авторизованного пользователя *Параметры:* • `principal` — объект с информацией об авторизованном пользователе **Возвращаемое значение**: `System.String` — строковый идентификатор авторизованного пользователя
`GetPrincipalRoles(IPrincipal)`	Получает роли (группы) пользователей *Параметры:* • `principal` — объект с информацией об авторизованном пользователе **Возвращаемое значение**: `UserRoles` — группы пользователей сервера Docsvision

IUserRolesProviderExtension — интерфейс

Предоставляет методы для получения пользовательских ролей

- **Пространство имён:** `DocsVision.Platform.Authentication`
- **Сборка:** `DocsVision.Authentication.dll`

Синтаксис

```
public interface IUserRolesProviderExtension
```

Методы

Имя	Описание
`GetRoles(IPrincipal)`	Получает роли пользователя *Параметры:* • `principal` — объект с информацией об авторизованном пользователе **Возвращаемое значение**: `UserRoles` — группы пользователей сервера Docsvision

DocsVision.Workflow — пространство имён

Пространство имён `DocsVision.Workflow` реализует доступ к подсистеме управления бизнес-процессами, а также содержит средства организации взаимодействия платформы и различных внешних систем.

Пространства имён

Пространство имён	Описание
`DocsVision.Workflow.Functions`	В пространстве имён DocsVision.Workflow.Functions сосредоточены объекты, отвечающие за предоставление доступ к базовым функции приложения Управление процессами, функциям взаимодействия с объектами Docsvision, файловой системой и с Microsoft Exchange.

Пространство имён	Описание
DocsVision.Workflow.Gates	Данное пространство имён содержит интерфейсы, используемые при создании шлюза Workflow, методы работы со шлюзами, а также несколько готовых объектов данного типа: шлюз к базовым типам, к Docsvision, шлюз в файловую систему и в Microsoft Exchange.
DocsVision.Workflow.Objects	Пространство имён DocsVision.Workflow.Objects содержит описания структур данных карточек Workflow (карточка процесса, справочники).
DocsVision.Workflow.ObjectsUI	Пространство имён DocsVision.Workflow.ObjectsUI содержит описания элементов пользовательского интерфейса, взаимодействующих с реализуемыми объектами Workflow.
DocsVision.Workflow.Runtime	Пространство имён DocsVision.Workflow.Runtime содержит различные типы, используемые при запуске и обработке бизнес-процессов
DocsVision.Workflow.Tracing	Предоставляет средства для формирования отладочной информации.

DocsVision.Workflow.Gates — пространство имён

Данное пространство имён содержит интерфейсы, используемые при создании шлюза Workflow, методы работы со шлюзами, а также несколько готовых объектов данного типа: шлюз к базовым типам, к Docsvision, шлюз в файловую систему и в Microsoft Exchange.

Классы

Класс	Описание
BasicGate	Содержит методы и свойства шлюза к простым типам.
DVGate	Содержит методы и свойства шлюза к Docsvision.
ExGate	Содержит методы и свойства шлюза к почте.
FSGate	Содержит методы и свойства шлюза к файловой системе.
FSSubscriptionInfo	Предоставляет значения для фильтра при подписке на события шлюза файловой системы.
MailFilter	Предоставляет значения для фильтра почтовых сообщений.
MessageChannel	Предоставляет методы для работы с сообщениями шлюза.

Интерфейсы

Интерфейс	Описание
ICollectionalProperty	Данный интерфейс описывает способ взаимодействия с коллекционным свойством стандартной переменной шлюза.
IGate	Данный интерфейс реализуется серверными компонентами шлюзов. При помощи методов, представленных в данном интерфейсе, сервис СУБП работает с переменными шлюза.
IGateMessageSource	Определяет метод инициализации механизма подписки на события шлюза, а также свойство, через которое может быть получен канал подписки на события.

Интерфейс	Описание
IGateVariable	Данный интерфейс описывает свойства стандартной переменной шлюза, единые для всех типов переменных. Все переменные, уникальные для конкретного шлюза, должны реализовывать этот интерфейс.
IGateVariableCustomProperties	Данный интерфейс описывает методы для работы с пользовательскими свойствами объекта шлюза, поддерживающего обработку подобных свойств.
ILicensableObject	Определяет поддержку шлюзом лицензирования, предоставляет тип лицензии.
ILockableObject	Данный интерфейс описывает методы и свойства объекта шлюза, поддерживающего блокировку доступа к данным объекта.
ISubscriptionChannel	Данный интерфейс добавляет методы подписки на события, и реализуется для соответствующего шлюза Workflow.

Перечисления

Перечисление	Описание
DocsVisionSubscriptionType	Типы подписок на события, поддерживаемые шлюзом к Docsvision.
DVVariableType	Типы переменных шлюза к Docsvision.
FSSubscriptionType	Типы подписок на события, поддерживаемые шлюзом к файловой системе.
MailSubscriptionType	Типы подписок на события, поддерживаемые шлюзом к почте.

BasicGate — класс

Содержит методы и свойства шлюза к простым типам.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public class BasicGate : IGate
```

Конструкторы

Имя	Описание
BasicGate()	Инициализирует новый экземпляр класса BasicGate.

Свойства

Имя	Описание
Data	Задаёт или возвращает конфигурацию шлюза.
ExecutionMode	Возвращает поддерживаемый режим исполнения процесса.
ID	Возвращает уникальный идентификатор шлюза.

Методы

Имя	Описание
AddVariable(Int32, String)	Добавляет переменную шлюза. Не поддерживается шлюзом BasicGate.
AddVariable(Int64, String)	Добавляет переменную шлюза. Не поддерживается шлюзом BasicGate.
CompareVariables(Int32, String, String)	Сравнивает две однотипных переменных шлюза.

Имя	Описание
`CompareVariables(Int64, String, String)`	Сравнивает две однотипных переменных шлюза.
`ConcatenateStrings(Array)`	Последовательно объединяет строки из строкового массива.
`CopyVariable(Int32, String, String)`	Копирует переменную шлюза. Не поддерживается шлюзом `BasicGate`.
`CopyVariable(Int64, String, String)`	Копирует переменную шлюза. Не поддерживается шлюзом `BasicGate`.
`DeleteVariable(Int32, String)`	Удаляет переменную шлюза. Не поддерживается шлюзом `BasicGate`.
`DeleteVariable(Int64, String)`	Удаляет переменную шлюза. Не поддерживается шлюзом `BasicGate`.
`GetCurrentDate`	Возвращает текущую дату.
`GetEnumDisplayValue(ProcessVariable)`	Возвращает отображаемое значение переменной с типом "Перечисление".
`GetVariable(Int32, String)`	Возвращает переменную шлюза. Не поддерживается шлюзом BasicGate.
`GetVariable(Int64, String)`	Возвращает переменную шлюза. Не поддерживается шлюзом `BasicGate`.
`GetVariable(Int32, String, Object)`	Возвращает переменную шлюза. Не поддерживается шлюзом `BasicGate`.
`GetVariableInternalReference(Int32, String)`	Возвращает ссылку на переменную шлюза во внутреннем формате. Не поддерживается шлюзом `BasicGate`.
`GetVariableReference(Int32, String)`	Возвращает внешнюю ссылку на заданную переменную шлюза. Не поддерживается шлюзом `BasicGate`.

Поля

Имя	Описание
`GateId`	Предоставляет идентификатор шлюза.

Имя	Описание
GateName	Предоставляет название шлюза.

BasicGate.GetEnumDisplayValue — метод (ProcessVariable)

Возвращает отображаемое значение переменной с типом "Перечисление".

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public string GetEnumDisplayValue(ProcessVariable Variable)
```

Параметры

Variable

Тип: ProcessVariable

Переменная процесса, содержащая значение типа "Перечисление"

Возвращаемое значение

Тип: System.String

Отображаемое значение переменной

DVGate — класс

Содержит методы и свойства шлюза к Docsvision.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public class DVGate : IGate, IDisposable, IGateMessageSource
```

Конструкторы

Имя	Описание
DVGate()	Инициализирует новый экземпляр класса DVGate.

Свойства

Имя	Описание
BaseURL	Возвращает URL к корню WEB-сервера Docsvision.
Data	Задаёт или возвращает конфигурацию шлюза.
ExecutionMode	Возвращает поддерживаемый режим исполнения процесса.
FoldersCard	Возвращает карточку папок.
ID	Возвращает уникальный идентификатор шлюза.
MessageChannel	Возвращает канал сообщений шлюза.
RefPartners	Возвращает справочник контрагентов.
RefStaff	Возвращает справочник сотрудников.
RefUniversal	Возвращает универсальный справочников (Делопроизводство).

Имя	Описание
Session	Задаёт или возвращает пользовательскую сессию.
SubscriptionChannel	Возвращает канал подписки на события шлюза.
UserSession	Задаёт или возвращает пользовательскую сессию.
WorkflowUserId	Возвращает идентификатор пользователя Workflow.

Методы

Имя	Описание
AddVariable(Int32, String)	Добавляет новую переменную шлюза.
AddVariable(Int64, String)	Добавляет новую переменную шлюза.
CompareVariables(Int32, String, String)	Сравнивает значения двух однотипных переменных шлюза. Выполняется простое сравнение двух строковых переменных.
CompareVariables(Int64, String, String)	Сравнивает значения двух однотипных переменных шлюза. Выполняется простое сравнение двух строковых переменных.
CopyCard(DVCard, Boolean)	Создаёт копию переданной карточки.
CopyCards(Array, Boolean, ProcessVariable)	Создаёт копии нескольких карточек.
CopyVariable(Int32, String, String)	Создаёт копию переменной шлюза.
CopyVariable(Int64, String, String)	Создаёт копию переменной шлюза.
CreateCard(DVCardType)	Создаёт новую карточку указанного типа.
CreateCard(String)	Создаёт новую карточку указанного типа.

Имя	Описание
CreateCardInFolder(DVCardType, DVFolder)	Создаёт новую карточку указанного типа и добавляет ярлык для её открытия в папку.
CreateCardInFolder(String, DVFolder)	Создаёт новую карточку указанного типа и ярлык на неё.
CreateDVFileFromExAttachment(ExAttachment)	Создаёт карточку версионного файла на основе вложения письма переданного шлюзом к Exchange.
CreateDVFileFromExAttachmentInFolder(ExAttachment, DVFolder, ProcessVariable)	Создаёт карточку версионного файла на основе вложения письма переданного шлюзом к Exchange.
CreateDVFileFromFSFile(FSFile)	Создаёт карточку версионного файла на основе файла, полученного из шлюза к файловой системе.
CreateDVFileFromFSFileInFolder(FSFile, DVFolder, ProcessVariable)	Создаёт карточку версионного файла на основе файла, полученного из шлюза к файловой системе, и ярлык на карточку.
CreateFolder(DVFolder, String, DVCardRow)	Создаёт новую папку указанного типа.
CreateMessage(String, String, DVCard, DVPrincipal)	Создаёт карточку "Сообщение" с переданными параметрами.
CreateMessage(String, String, DVCard, DVPrincipal, DVPrincipal)	Создаёт карточку "Сообщение" с переданными параметрами.
CreateMessage(String, String, DVCard, String, DVPrincipal)	Создаёт карточку "Сообщение" с переданными параметрами.
CreateMessageFrom(String, String, DVCard, DVPrincipal, DVPrincipal)	Создаёт карточку "Сообщение" с переданными параметрами.
CreateMessageInFolder(String, String, DVCard, DVFolder, DVPrincipal)	Создаёт карточку "Сообщение" с переданными параметрами.
DeleteCard(DVCard, Boolean)	Удаляет карточку.
DeleteFolder(DVFolder, Boolean)	Удаляет папку.
DeleteShortcut(DVShortcut, Boolean)	Удаляет ярлык на карточку.

Имя	Описание
DeleteVariable(Int32, String)	Удаляет переменную шлюза.
GetCard(Guid, Boolean)	Получает карточку.
GetCardHardLink(String)	Получает сильный ярлык на карточку.
GetCardSecurable(Guid)	Возвращает описатель прав для карточки.
GetCardTypeByID(Guid)	Получает тип карточки по его идентификатору.
GetCurrentProcess(ProcessVariable)	Получает текущий бизнес-процесс.
GetDocStateByName(DVCardRow, String)	Получает строку состояния карточки по названию.
GetDocStates(DVCardRow, ProcessVariable)	Получает все возможные состояния карточки и сохраняет их в переменную процесса.
GetDocumentByTypeID(String)	Получает справочник по его идентификатору.
GetFunctionLinkedCard(String, Boolean, ProcessVariable)	Получает карточку, связанную с функцией бизнес-процесса.
GetRow(Guid, Guid)	Получает строку секции карточки.
GetVariable(Int32, String, Object)	Получает переменную процесса по её идентификатору и типу.
GetVariableReference(Int32, String)	Возвращает URL (ссылку) для открытия карточки или папки.
ImportCard(String)	Импортирует карточку из XML.
ImportCardFromFile(FSFile)	Импортирует карточку из текстового файла.
ImportCardIntoFolder(String, DVFolder, ProcessVariable)	Импортирует карточку из XML и создаёт ярлык.
ImportCardIntoFolderFromFile(FSFile, DVFolder, ProcessVariable)	Импортирует карточку из текстового файла и создаёт ярлык.

Имя	Описание
`Initialize(Guid, IMessageChannel, ISubscriptionChannel)`	Инициализирует новый экземпляр шлюза.
`IsVariableExists(DVVariableType, String)`	Проверяет существование переменной шлюза
`SendMessageToProcess(Guid, Guid, Int32, String, Guid)`	Отправляет сообщение функции процесса.
`SendMessageToProcessFunctions(Guid, String, Int32, String, Guid)`	Отправляет сообщение функции процесса.

Поля

Имя	Описание
`GateID`	Предоставляет идентификатор шлюза.
`GateName`	Предоставляет название шлюза.

DVGate.BaseURL — свойство

Возвращает URL к корню WEB-сервера Docsvision.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public string BaseURL { get; }
```

Значение свойства

Тип: `System.String`

Базовый URL

DVGate.FoldersCard — свойство

Возвращает карточку папок.

- **Пространство имён:** `DocsVision.Workflow.Gates`

- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public FolderCard FoldersCard { get; }
```

Значение свойства

Тип: FolderCard

Карточка папок

DVGate.MessageChannel — свойство

Возвращает канал сообщений шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public MessageChannel MessageChannel { get; }
```

Значение свойства

Тип: MessageChannel

Канал сообщений

DVGate.RefPartners — свойство

Возвращает справочник контрагентов.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public CardData RefPartners { get; }
```

Значение свойства

Тип: CardData

Справочник контрагентов

DVGate.RefStaff — свойство

Возвращает справочник сотрудников.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public CardData RefStaff { get; }
```

Значение свойства

Тип: CardData

Справочник сотрудников

DVGate.RefUniversal — свойство

Возвращает универсальный справочников (Делопроизводство).

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public CardData RefUniversal { get; }
```

Значение свойства

Тип: CardData

Универсальный справочник

DVGate.AddVariable — метод (Int32, String)

Добавляет новую переменную шлюза

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public IGateVariable AddVariable(int Type, string VariableInfo)
```

Параметры

Type

Тип: System.Int32

Тип добавляемой переменной: DVVariableType.DOCUMENT, DVVariableType.FILe или DVVariableType.PRINCIPAL

VariableInfo

Тип: System.String

Зависит от типа добавляемой переменной:

- DVVariableType.DOCUMENT — должен быть указан идентификатор типа карточки, которая будет создана при добавлении переменной шлюза.
- DVVariableType.PRINCIPAL — должна быть указана фамилия сотрудника из Справочника сотрудников, сущность которого будет загружена в переменную шлюза
- DVVariableType.FILe — должен быть указан путь к файлу, на основе которого будет создан версионный файл.

Возвращаемое значение

Тип: IGateVariable

Реальный возвращаемый тип зависит от типа добавляемой переменной:

- DVVariableType.DOCUMENT — возвращает DVCard
- DVVariableType.PRINCIPAL — возвращает DVPrincipal
- DVVariableType.FILe — возвращает DVFileCard

DVGate.CopyCard — метод (DVCard, Boolean)

Создаёт копию переданной карточки.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVCard CopyCard(DVCard Card, bool ClearTemplateFlag)
```

Параметры

Card

Тип: DVCard

Копируемая карточка

ClearTemplateFlag

Тип: System.Boolean

Если копируемая карточка является шаблоном, то при установленном флаге ClearTemplateFlag, признак шаблона из копии будет убран.

Возвращаемое значение

Тип: DVCard

Копия карточки

DVGate.CopyCards — метод (Array, Boolean, ProcessVariable)

Создаёт копии нескольких карточек.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public void CopyCards(Array cards, bool ClearTemplateFlag, ProcessVariable result);
```

Параметры

cards

Тип: System.Array

Массив элементов типа `DVCard` — копируемых карточек

ClearTemplateFlag

Тип: `System.Boolean`

Если копируемая карточка является шаблоном, то при установленном флаге `ClearTemplateFlag`, признак шаблона из копии будет убран.

result

Тип: `ProcessVariable`

Переменная процесса, в которую будут сохранены копии карточек. Копии сохраняются в `result.Values`.

DVGate.CopyVariable — метод (Int32, String, String)

Создаёт копию переменной шлюза.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public virtual string CopyVariable(int Type, string ID, string Location)
```

Параметры

Type

Тип: `System.Int32`

Тип копируемой переменной: `DVVariableType.DOCUMENT`, `DVVariableType.FILe`, `DVVariableType.FOLDEr` или `DVVariableType.SHORTCUt`

ID

Тип: `System.String`

Идентификатор копируемой переменной

Location

Тип: `System.String`

Путь для копирования переменной

Возвращаемое значение

Тип: System.String

Идентификатор созданной копии

DVGate.CreateCard — метод (String)

Создаёт новую карточку указанного типа.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVCard CreateCard(string CardTypeID)
```

Параметры

CardTypeID

Тип: System.String

Идентификатор типа создаваемой карточки

Возвращаемое значение

Тип: DVCard

Созданная карточка

DVGate.CreateCardInFolder — метод (String, DVFolder)

Создаёт новую карточку указанного типа и ярлык на неё.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVCard CreateCardInFolder(string CardTypeID, DVFolder Folder)
```

Параметры

CardTypeID

Тип: `System.String`

Идентификатор типа создаваемой карточки

Folder

Тип: `DVFolder`

Папка для размещения ярлыка

Возвращаемое значение

Тип: `DVCard`

Созданная карточка

DVGate.CreateDVFileFromExAttachment — метод (ExAttachment)

Создаёт карточку версионного файла на основе вложения письма переданного шлюзом к Exchange.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public DVFileCard CreateDVFileFromExAttachment(ExAttachment Attachment)
```

Параметры

Attachment

Тип: `ExAttachment`

Вложение письма

Возвращаемое значение

Тип: `DVFileCard`

Карточка файла с версиями

DVGate.CreateDVFileFromExAttachmentInFolder — метод (ExAttachment, DVFolder, ProcessVariable)

Создаёт карточку версионного файла на основе вложения письма переданного шлюзом к `Exchange`.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public DVFileCard CreateDVFileFromExAttachmentInFolder(ExAttachment Attachment, DVFolder Folder, ProcessVariable OutShortcut);
```

Параметры

Attachment

Тип: `ExAttachment`

Вложение письма

Folder

Тип: `DVFolder`

Папка, в которую будет добавлен ярлык на карточку

OutShortcut

Тип: `ProcessVariable`

Переменная бизнес-процесса, в значение которой будет записан созданный ярлык. Если не требуется — передать `null`.

Возвращаемое значение

Тип: `DVFileCard`

Карточка файла с версиями

DVGate.CreateDVFileFromFSFile — метод (FSFile)

Создаёт карточку версионного файла на основе файла, полученного из шлюза к файловой системе.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVFileCard CreateDVFileFromFSFile(FSFile File)
```

Параметры

File

Тип: FSFile

Сущность файла, полученного через шлюз к файловой системе

Возвращаемое значение

Тип: DVFileCard

Карточка файла с версиями

DVGate.CreateDVFileFromFSFileInFolder — метод (FSFile, DVFolder, ProcessVariable)

Создаёт карточку версионного файла на основе файла, полученного из шлюза к файловой системе, и ярлык на карточку.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVFileCard CreateDVFileFromFSFileInFolder(FSFile File, DVFolder Folder,
ProcessVariable OutShortcut)
```

Параметры

File

Тип: FSFile

Сущность файла, полученного через шлюз к файловой системе

Folder

Тип: `DVFolder`

Папка для размещения ярлыка

OutShortcut

Тип: `ProcessVariable`

Переменная бизнес-процесса, в значение которой будет записан созданный ярлык. Если не требуется — передать `null`.

Возвращаемое значение

Тип: `DVFileCard`

Карточка файла с версиями

DVGate.CreateFolder — метод (DVFolder, String, DVCardRow)

Создаёт новую папку указанного типа.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public DVFolder CreateFolder(DVFolder ParentFolder, string Name, DVCardRow FolderTypeRow)
```

Параметры

ParentFolder

Тип: `DVFolder`

Родительская папка. Если `null`, то папка будет создана в корневой папке.

Name

Тип: `System.String`

Название создаваемой папки

FolderTypeRow

Тип: `DVCardRow`

Тип создаваемой папки. Передается в виде строки *Справочника типов папок*.

Возвращаемое значение

Тип: `DVFolder`

Созданная папка

DVGate.DeleteCard — метод (DVCard, Boolean)

Удаляет карточку.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public void DeleteCard(DVCard Card, bool FinalDeletion)
```

Параметры

Card

Тип: `DVCard`

Удаляемая карточка

FinalDeletion

Тип: `System.Boolean`

Признак окончательного удаления карточки: `true` — удалить окончательно.

DVGate.DeleteFolder — метод (DVFolder, Boolean)

Удаляет папку.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public void DeleteFolder(DVFolder Folder, bool FinalDeletion)
```

Параметры

Folder

Тип: DVFolder

Удаляемая папка

FinalDeletion

Тип: System.Boolean

Признак окончательного удаления карточки: true — удалить окончательно.

DVGate.DeleteShortcut — метод (DVShortcut, Boolean)

Удаляет ярлык на карточку.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public void DeleteShortcut(DVShortcut Shortcut, bool FinalDeletion)
```

Параметры

Shortcut

Тип: DVShortcut

Ссылка

FinalDeletion

Тип: System.Boolean

Признак окончательного удаления карточки: true — удалить окончательно.

DVGate.DeleteVariable — метод (Int32, String)

Удаляет переменную шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public void DeleteVariable(int Type, string ID)
```

Параметры

Type

Тип: System.Int32

Тип переменной шлюза: DVVariableType.DOCUMENT, DVVariableType.FILe, DVVariableType.DATAROW, DVVariableType.FOLDER, DVVariableType.SHORTCUT, DVVariableType.PRINCIPAL, DVVariableType.ROLe, DVVariableType.GROUP, DVVariableType.DEPARTMENT

ID

Тип: System.String

Идентификатор удаляемой переменной. В зависимости от типа удаляемой переменной, указываются: идентификатор карточки, файла, папки или ярлыка; идентификатор строки, соответствующей сотруднику, группе или роли.

DVGate.GetCard — метод (Guid, Boolean)

Получает карточку.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVCard GetCard(Guid cardId, bool refresh)
```

Параметры

cardId

Тип: System.Guid

Идентификатор карточки

refresh

Тип: System.Boolean

Флаг обновления данных кэша: `true` — обновлять кэш для загрузки актуальных данных

Возвращаемое значение

Тип: `DVCard`

Карточка

DVGate.GetCardHardLink — метод (String)

Получает сильный ярлык на карточку.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public DVShortcut GetCardHardLink(string CardID)
```

Параметры

CardID

Тип: `System.String`

Идентификатор карточки, для которой возвращается ярлык

Возвращаемое значение

Тип: `DVShortcut`

Ярлык

DVGate.GetCardSecurable — метод (Guid)

Возвращает описатель прав для карточки.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public CardSecurableImpl GetCardSecurable(Guid cardID)
```

Параметры

cardID

Тип: System.Guid

Идентификатор карточки

Возвращаемое значение

Тип: CardSecurableImpl

Описатель прав

DVGate.GetCardTypeByID — метод (Guid)

Получает тип карточки по его идентификатору.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVCardType GetCardTypeByID(string TypeID)
```

Параметры

TypeID

Тип: System.Guid

Идентификатор типа карточки

Возвращаемое значение

Тип: DVCardType

Тип карточки

DVGate.GetCurrentProcess — метод (ProcessVariable)

Получает текущий бизнес-процесс.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public void GetCurrentProcess(ProcessVariable Variable)
```

Параметры

Variable

Тип: ProcessVariable

Переменная текущего бизнес-процесса, из которой будет получен процесс. Бизнес-процесс (карточка) записывается в переданную переменную в Variable.Value.

DVGate.GetDocStateByName — метод (DVCardRow, String)

Получает строку состояния карточки по названию.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVCardRow GetDocStateByName(DVCardRow TypeRow, string StateName)
```

Параметры

TypeRow

Тип: DVCardRow

Строка вида карточки из *справочника видов*

StateName

Тип: System.String

Название (по умолчанию) состояния

Возвращаемое значение

Тип: DVCardRow

Строка состояния из *конструктора состояний*

DVGate.GetDocStates — метод (DVCardRow, ProcessVariable)

Получает все возможные состояния карточки и сохраняет их в переменную процесса.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public void GetDocStates(DVCardRow TypeRow, ProcessVariable StateRows)
```

Параметры

TypeRow

Тип: DVCardRow

Строка вида карточки из *справочника видов*

StateRows

Тип: ProcessVariable

Переменная процесса, в которую будут сохранены строки возможных состояний из *конструктора состояний*

DVGate.GetDocumentByTypeID — метод (String)

Получает справочник по его идентификатору.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVCard GetDocumentByTypeID(string TypeID)
```

Параметры

TypeID

Тип: System.String

Идентификатор справочника

Возвращаемое значение

Тип: `DVCard`

Справочник

DVGate.GetFunctionLinkedCard — метод (String, Boolean, ProcessVariable)

Получает карточку, связанную с функцией бизнес-процесса.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public void GetFunctionLinkedCard(string FunctionName, bool HardLinked, ProcessVariable
Card)
```

Параметры

FunctionName

Тип: `System.String`

Название функции бизнес-процесса, из которой будет получена карточка

HardLinked

Тип: `System.Boolean`

Тип ссылки, по которой получается карточка: `true` — по сильной ссылке, `false` — по слабой

Card

Тип: `ProcessVariable`

Переменная процесса, в которую будет сохранена полученная карточка. Карточка сохраняется в `Card.Value`.

Заметки

Метод `GetFunctionLinkedCard` позволяет получить (**если это предусмотрено**)

идентификатор карточки, из которой был запущен бизнес-процесс. В частности, данный метод может быть использован для получения карточки задания из бизнес-процесса отправки данного задания на исполнение.

DVGate.GetRow — метод (Guid, Guid)

Получает строку секции карточки.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVCardRow GetRow(Guid rowID, Guid sectionID)
```

Параметры

rowID

Тип: System.Guid

Идентификатор получаемой строки

sectionID

Тип: System.Guid

Идентификатор секции получаемой строки

Возвращаемое значение

Тип: DVCardRow

Строка секции

DVGate.GetVariable — метод (Int32, String, Object)

Получает переменную процесса по её идентификатору и типу.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public IGateVariable GetVariable(int TypeID, string SubTypeID, object ID)
```

TypeID

Тип: System.Int32

Тип получаемой переменной из типов DVVariableType, кроме DVVariableType.GROUPROLE

SubTypeID

Тип: System.String

Идентификатор подтипа

ID

Тип: System.Object

Идентификатор переменной

Возвращаемое значение

Тип: IGateVariable

Переменная процесса

DVGate.GetVariableReference — метод (Int32, String)

Возвращает URL (ссылку) для открытия карточки или папки.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public virtual string GetVariableReference(int Type, string ID)
```

Type

Тип: System.Int32

Тип переменной: DVVariableType.DOCUMENT, DVVariableType.FILe или

DVVariableType.FOLDER

ID

Тип: System.String

Идентификатор карточки или папки

Возвращаемое значение

Тип: System.String

Сформированная ссылка

Заметки

Данный метод предоставляется ссылку для открытия карточки или папки из браузера.

DVGate.ImportCard — метод (String)

Импортирует карточку из XML.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVCard ImportCard(string cardXml)
```

Параметры

cardXml

Тип: System.String

XML-документ с данными импортируемой карточки

Возвращаемое значение

Тип: DVCard

Импортированная карточка

DVGate.ImportCardFromFile — метод (FSFile)

Импортирует карточку из текстового файла.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVCard ImportCardFromFile(FSFile file)
```

file

Тип: FSFile

Файл с XML-данными импортируемой карточки

Возвращаемое значение

Тип: DVCard

Импортированная карточка

DVGate.ImportCardIntoFolder — метод (String, DVFolder, ProcessVariable)

Импортирует карточку из XML и создаёт ярлык.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public DVCard ImportCardIntoFolder(string cardXml, DVFolder folder, ProcessVariable
outShortcut)
```

cardXml

Тип: System.String

XML-документ с данными импортируемой карточки

folder

Тип: `DVFolder`

Папка для создания ярлыка

outShortcut

Тип: `ProcessVariable`

Переменная процесса, в которую будет сохранен ярлык. Ярлык (тип DVShortcut) сохраняется в `outShortcut.Value`.

Возвращаемое значение

Тип: `DVCard`

Импортированная карточка

DVGate.ImportCardIntoFolderFromFile — метод (FSFile, DVFolder, ProcessVariable)

Импортирует карточку из текстового файла и создаёт ярлык.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public DVCard ImportCardIntoFolderFromFile(FSFile file, DVFolder folder, ProcessVariable outShortcut)
```

Параметры

file

Тип: `FSFile`

Файл с XML-данными импортируемой карточки

folder

Тип: `DVFolder`

Папка для создания ярлыка

outShortcut

Тип: `ProcessVariable`

Переменная процесса, в которую будет сохранен ярлык. Ярлык (тип DVShortcut) сохраняется в `outShortcut.Value`.

Возвращаемое значение

Тип: `DVCard`

Импортированная карточка

DVGate.IsVariableExists — метод (DVVariableType, String)

Проверяет существование переменной шлюза.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public bool IsVariableExists(DVVariableType Type, string ID)
```

Параметры

Type

Тип: `DVVariableType`

Тип переменной

ID

Тип: `System.String`

Идентификатор переменной

Возвращаемое значение

Тип: `System.Boolean`

`true` — переменная существует, иначе — `false`

Заметки

Для проверки существования строки, идентификатор следует передавать в виде: `CardId,SectionId,RowId`.

DVGate.SendMessageToProcess — метод (Guid, Guid, Int32, String, Guid)

Отправляет сообщение функции процесса.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public void SendMessageToProcess(Guid ProcessID, Guid FunctionID, int Type, string Data,
Guid SourceID)
```

Параметры

ProcessID

Тип: System.Guid

Идентификатор процесса

FunctionID

Тип: System.Guid

Идентификатор функции процесса, которой отправляется сообщение. Если не указан, то сообщение будет разослано всем активным функциям

Type

Тип: System.Int32

Тип сообщения

Data

Тип: System.String

Отправляемые данные

SourceID

Тип: System.Guid

Идентификатор источника сообщения

DVGate.SendMessageToProcessFunctions — метод (Guid, String, Int32, String, Guid)

Отправляет сообщение функции процесса.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public void SendMessageToProcessFunctions(Guid ProcessID, string FunctionName, int Type,
string Data, Guid SourceID)
```

Параметры

ProcessID

Тип: System.Guid

Идентификатор процесса

FunctionName

Тип: System.String

Название функции процесса, которой отправляется сообщение

Type

Тип: System.Int32

Тип сообщения

Data

Тип: System.String

Отправляемые данные

SourceID

Тип: System.Guid

Идентификатор источника сообщения

DVGate.GateID — поле

Предоставляет идентификатор шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public static readonly Guid GateID
```

DVGate.GateName — поле

Предоставляет название шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public const string GateName = "DocsVision Gate"
```

ExGate — класс

Содержит методы и свойства шлюза к почте.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public class ExGate : IGate, IGateMessageSource
```

Конструкторы

Имя	Описание
ExGate()	Инициализирует новый экземпляр класса ExGate.

Свойства

Имя	Описание
Data	Задаёт или возвращает конфигурацию шлюза.
ExecutionMode	Возвращает поддерживаемый режим исполнения процесса.
ID	Возвращает уникальный идентификатор шлюза.
ServerAddress	Возвращает e-mail адрес сервера.
SubscriptionChannel	Возвращает канал подписки на события шлюза.

Методы

Имя	Описание
AddVariable(Int32, String)	Создаёт указанного типа переменную шлюза.

Имя	Описание
AddVariable(Int64, String)	Создаёт указанного типа переменную шлюза

Поля

Имя	Описание
GateId	Предоставляет идентификатор шлюза.
GateName	Предоставляет название шлюза.

ExGate.ServerAddress — свойство

Возвращает e-mail адрес сервера.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public string ServerAddress { get; }
```

Значение свойства

Тип: System.String

E-mail адрес сервера, указанный в настройках шлюза

ExGate.AddVariable — метод (Int32, String)

Создаёт указанного типа переменную шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public IGateVariable AddVariable(int Type, string VariableInfo)
```

Type

Тип: `System.Int32`

Тип переменной:

- `0` — сообщение
- `1` — получатель сообщения
- `2` — вложение

VariableInfo

Тип: `System.String`

Не используется. Может быть передан `null`

Возвращаемое значение

Тип: `IGateVariable`

Переменная шлюза.

Заметки

Действительный тип возвращаемого объекта зависит от значения `Type`

FSGate — класс

Содержит методы и свойства шлюза к файловой системе.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public class FSGate : IGate, IGateMessageSource
```

Конструкторы

Имя	Описание
FSGate()	Инициализирует новый экземпляр класса FSGate.

Свойства

Имя	Описание
Data	Задаёт или возвращает конфигурацию шлюза.
ExecutionMode	Возвращает поддерживаемый режим исполнения процесса.
ID	Возвращает уникальный идентификатор шлюза.
SubscriptionChannel	Возвращает канал подписки на события шлюза.

Методы

Имя	Описание
AddVariable(Int32, String)	Создаёт новый файл или папку в указанном каталоге.

Имя	Описание
AddVariable(Int64, String)	Создаёт новый файл или папку в указанном каталоге.
CompareVariables(Int32, String, String)	Сравнивает пути к двум файлам или папкам.
CompareVariables(Int64, String, String)	Сравнивает пути к двум файлам или папкам.
CopyVariable(Int32, String, String)	Копирование не поддерживается.
CopyVariable(Int64, String, String)	Копирование не поддерживается.
CreateFile(String)	Создаёт в файловой системе новый файл.
CreateFolder(String)	Создаёт в файловой системе новую папку.
DeleteVariable(Int32, String)	Удаляет указанный файл или папку.
DeleteVariable(Int64, String)	Удаляет указанный файл или папку.
GetVariable(Int32, String)	Возвращает сущность указанного файла или папки.
GetVariable(Int64, String)	Возвращает сущность указанного файла или папки.
GetVariable(Int32, String, String)	Возвращает сущность указанного файла или папки.
GetVariableInternalReference(Int32, String)	Возвращает ссылку на переменную шлюза во внутреннем формате.
GetVariableReference(Int32, String)	Возвращает внешнюю ссылку на заданную переменную шлюза.
Initialize(Guid, IMessageChannel, ISubscriptionChannel)	Инициализирует механизм подписки на события шлюза.

Поля

Имя	Описание
GateID	Предоставляет идентификатор шлюза.

Имя	Описание
GateName	Предоставляет название шлюза.

Примечание

При работе с файловой системой из бизнес-процесса (особенно) в кластере СУБП необходимо учитывать, что конкретная функция может обрабатываться на любом из серверов СУБП, работающих в кластере, т.е. список файлов локального диска для одной функции может отличаться от списка файлов, полученного другой функцией.

Примеры

Ниже приведён пример простого скрипта бизнес-процесса, который получает из шлюза список файлов определённой папки, содержащей новые приказы, и на их основе создаёт документы.

```
public void Execute (ProcessInfo process, PassState passInfo, ObjectContext context) ①
{
 try
 {
  FSGate fsGate = (FSGate)process.Gates[FSGate.GateID]; ②

  IDocumentService documentService = context.GetService<IDocumentService>(); ③

  string newOrderFolder = (string)process.Variables.GetVariableByName("newOrderFolder"
).Value;
  string processedOrderFolder = (string)process.Variables.GetVariableByName
("processedOrderFolder").Value; ④

  FSFileFolder folder = fsGate.GetVariable(0, newOrderFolder) as FSFileFolder; ⑤

  string[] orders = folder.GetFiles("*.docx"); ⑥

  for(int i =0; i<orders.Length; i++)
  {
   FSFile currentFile = (FSFile)fsGate.GetVariable(1, orders[i]); ⑦

   if(currentFile.CreationTime < DateTime.Today) ⑧
   {
    try{

     Document document = documentService.CreateDocument(currentFile.ID);
     context.SaveObject(document); ⑨
```

```
      currentFile.Move(System.IO.Path.Combine(processedOrderFolder, currentFile.Name));
      process.LogMessage ("Создан документ по файлу: " + currentFile.Name); ⑩
     } catch
     {
      throw;
     }
    }
   }
  } catch (Exception ex)
  {
   process.LogMessage("Ошибка выполнения скрипта:" + ex.Message); ⑪
  }
  return;
 }
```

① Стандартная точка входа функции БП, дополненная параметром `context`.

② Получаем шлюз к файловой системе.

③ Получаем сервис для работы с документами для создания документов.

④ Получаем из настроек БП папки: с новыми приказами и для обработанных БП приказов.

⑤ Получаем объект-папку из шлюза.

⑥ Получаем список файлов (`.docx`) из папки.

⑦ Получаем объект-файл.

⑧ Обрабатываем только приказы, выпущенные вчера (или ранее).

⑨ Создаем документ.

⑩ Убираем обработанный файл.

⑪ Записываем в журнал ошибки исполнения.

FSGate.AddVariable — метод (Int32, String)

Создаёт новый файл или папку в указанном каталоге.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public IGateVariable AddVariable(int Type, string VariableInfo)
```

Параметры

Type

Тип: System.Int32

Тип переменной: 0 — папка; 1 — файл

VariableInfo

Тип: System.String

Путь в файловой системе для создания папки или файла

Возвращаемое значение

Тип: IGateVariable

В зависимости от типа создаваемой переменной вернет объект типа: FSFileFolder — для папки; FSFile — для файла.

Заметки

Данный метод создаёт в файловой системе файл или папку (в зависимости от значения параметра Type) по пути, переданному в VariableInfo. Если файл или папка существует, будет возвращена ошибка FSGateException#. Также для создания файла или папки можно использовать соответственно методы CreateFile и CreateFolder.

FSGate.CompareVariables — метод (Int32, String, String)

Сравнивает пути к двум файлам или папкам.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public int CompareVariables(int Type, string ID1, string ID2)
```

Параметры

Type

Тип: `System.Int32`

Тип переменной: `0` — папка; `1` — файл. Не влияет на результат

ID1

Тип: `System.String`

Путь к первому файлу или папке

ID2

Тип: `System.String`

Путь ко второму файлу или папке

Возвращаемое значение

Тип: `System.Int32`

`0` — если пути равны, иначе — `1`

Заметки

Метод `CompareVariables` выполняет простое сравнение двух строковых переменных.

FSGate.CreateFile — метод (String)

Создаёт в файловой системе новый файл.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public FSFile CreateFile(string filePath)
```

Параметры

filePath

Тип: `System.String`

Путь к создаваемому файлу

Возвращаемое значение

Тип: FSFile

Сущность созданного файла

FSGate.CreateFolder — метод (String)

Создаёт в файловой системе новую папку.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public FSFileFolder CreateFolder(string folderPath)
```

Параметры

folderPath

Тип: System.String

Путь к создаваемой папке

Возвращаемое значение

Тип: FSFileFolder

Сущность созданной папки

FSGate.DeleteVariable — метод (Int32, String)

Удаляет указанный файл или папку.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public void DeleteVariable(int Type, string ID)
```

Параметры

Type

Тип: System.Int32

Тип переменной: 0 — папка; 1 — файл

ID

Тип: System.String

Путь к удаляемому файлу или папке

Заметки

Если указанные файл или папка отсутствую, будет возвращена ошибка FSGateException.

FSGate.GetVariable — метод (Int32, String)

Возвращает сущность указанного файла или папки.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public IGateVariable GetVariable(int Type, string ID)
```

Параметры

Type

Тип: System.Int32

Тип переменной: 0 — папка; 1 — файл

VariableInfo

Тип: System.String

Путь к запрашиваемому файлу или папке

Возвращаемое значение

Тип: IGateVariable

В зависимости от типа переменной будет возвращен объект типа: `FSFileFolder` — для папки; `FSFile` — для файла.

FSGate.GateID — поле

Предоставляет идентификатор шлюза.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public static readonly Guid GateID
```

FSGate.GateName — поле

Предоставляет название шлюза.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public const string GateName = "FileSystem Gate"
```

FSSubscriptionInfo — класс

Предоставляет значения для фильтра при подписке на события шлюза файловой системы.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public class FSSubscriptionInfo
```

Свойства

Имя	Описание
`Path`	Путь к папке или файлу.
`Filter`	Шаблон имён искомых файлов.
`IncludeSubFolder`	Просматривать дочерние папки.

MailFilter — класс

Предоставляет значения для фильтра почтовых сообщений.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public class MailFilter
```

Свойства

Имя	Описание
`DateFrom`	Начальная дата для диапазона даты отправки.
`DateTo`	Конечная дата для диапазона даты отправки.
`Subject`	Тема сообщения.
`Body`	Текст сообщения.
`Sender`	Отправитель.
`Recipients`	Получатель.
`UnreadFlag`	Признак того, что сообщение не прочитано.
`Importance`	Важность.
`Size`	Минимальный размер письма (байт).

MessageChannel — класс

Предоставляет методы для работы с сообщениями шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class MessageChannel : IMessageChannel
```

Конструкторы

Имя	Описание
MessageChannel(UserSession)	Инициализирует новый экземпляр класса MessageChannel.

Свойства

Имя	Описание
Session	Возвращает пользовательскую сессию канала сообщений.

Методы

Имя	Описание
RemoveMessages(Guid, DateTime)	Удаляет все сообщения указанного процесса, отправленные до определённого времени.
RemoveMessages(Guid, Guid, DateTime)	Удаляет все сообщения указанной функции процесса, отправленные до определённого времени.
SendMessage(Int32, Guid)	Отправляет сообщение указанному процессу.
SendMessage(Int32, Guid, Guid)	Отправляет сообщение указанной функции процесса.

Имя	Описание
`SendMessage(Int32, Guid, Guid, String)`	Отправляет сообщение указанной функции процесса.
`SendMessage(Int32, Guid, Guid, String, Guid, MessageSourceType)`	Отправляет сообщение указанной функции процесса.
`SendMessage(Guid, Int32, String, Guid)`	Отправляет сообщение указанной функции процесса.

ICollectionalProperty — интерфейс

Данный интерфейс описывает способ взаимодействия с коллекционным свойством стандартной переменной шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
public interface ICollectionalProperty : IEnumerator, IEnumerable
```

Свойства

Имя	Описание
Count	Возвращает количество элементов коллекции.
Item	Возвращает элемент коллекции с заданным идентификатором.

Методы

Имя	Описание
Add(Object)	Добавляет существующий элемент в коллекцию.
AddNew	Добавляет новый элемент в коллекцию.
AddNew(Object)	Добавляет новый элемент, с указанным идентификатором, в коллекцию.
AddRange(IEnumerator)	Добавляет набор элементов в коллекцию.
Clear	Удаляет все элементы коллекции.
GetByIndex(Int32)	Возвращает элемент с заданным индексом.
Remove(Object)	Удаляет элемент коллекции с заданным идентификатором.

Имя	Описание
`RemoveAt(Int32)`	Удаляет элемент коллекции с заданным индексом.

IGate — интерфейс

Данный интерфейс реализуется серверными компонентами шлюзов. При помощи методов, представленных в данном интерфейсе, сервис СУБП работает с переменными шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
public interface IGate
```

Свойства

Имя	Описание
Data	Задаёт или возвращает конфигурацию шлюза.
ExecutionMode	Возвращает поддерживаемый режим исполнения процесса.
ID	Возвращает уникальный идентификатор шлюза.

Методы

Имя	Описание
AddVariable(Int32, String)	Добавляет переменную с указанным типом и дополнительными данными.
CompareVariables(Int32, String, String)	Сравнивает значения двух однотипных переменных шлюза.
CopyVariable(Int32, String, String)	Создаёт копию указанной переменной (объекта).
DeleteVariable(Int32, String)	Удаляет переменную шлюза.
GetVariable(Int32, String)	Получает переменную по её типу и строке-идентификатору.

Имя	Описание
GetVariable(Int64, String)	Получает переменную по её типу и строке-идентификатору.
GetVariable(Int32, String, Object)	Получает переменную по её типу и строке-идентификатору.
GetVariableInternalReference(Int32, String)	Возвращает ссылку на переменную шлюза во внутреннем формате.
GetVariableReference(Int32, String)	Возвращает внешнюю ссылку на заданную переменную шлюза.

Примечание

В методах под переменной подразумевается переменная шлюза, но не переменная процесса — ProcessVariable.

Примеры

См. примеры в описании конкретного шлюза.

IGate.Data — свойство

Задаёт или возвращает конфигурацию шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
string Data { get; set; }
```

Значение свойства

Тип: System.String

Строка, содержащая конфигурацию

Заметки

Строка конфигурации может содержать данные, которые будут переданы в серверный компонент шлюза и клиентского. Эти данные могут быть

использованы, к примеру, для установки значений внутренних переменных (адреса серверов, различные ограничения и пр.).

IGate.ExecutionMode — свойство

Возвращает поддерживаемый режим исполнения процесса.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
ExecutionModeEnum ExecutionMode { get; }
```

Значение свойства

Тип: ExecutionModeEnum

Режим исполнения

IGate.ID — свойство

Возвращает уникальный идентификатор шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
Guid ID { get; }
```

Значение свойства

Тип: System.Guid

Идентификатор шлюза

IGate.AddVariable — метод (Int32, String)

Добавляет переменную с указанным типом и дополнительными данными.

- **Пространство имён:** DocsVision.Workflow.Gates

- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
IGateVariable AddVariable(int Type, string VariableInfo)
```

Параметры

Type

Тип: System.Int32

Тип переменной

VariableInfo

Тип: System.String

Дополнительная информация для создания переменной

Возвращаемое значение

Тип: IGateVariable

Созданная переменная

IGate.CompareVariables — метод (Int32, String, String)

Сравнивает значения двух однотипных переменных шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
int CompareVariables(int Type, string ID1, string ID2)
```

Type

Тип: System.Int32

Тип переменной

ID1

Тип: `System.String`

Идентификатор первой сравниваемой переменной

ID2

Тип: `System.String`

Идентификатор второй сравниваемой переменной

Возвращаемое значение

Тип: `System.Int32`

Результат сравнения

Заметки

Возвращаемое методом `CompareVariables` значение зависит от реализации шлюза. В стандартных шлюзах Docsvision: `0` — если объекты идентичны; `1` — если сравниваемые объекты различаются.

IGate.CopyVariable — метод (Int32, String, String)

Создаёт копию указанной переменной (объекта).

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Interfaces.dll`

Синтаксис

```
string CopyVariable(int Type, string ID, string Location);
```

Параметры

Type

Тип: `System.Int32`

Тип переменной

ID

Тип: `System.String`

Идентификатор копируемой переменной

Location

Тип: System.String

Путь для копирования переменной

Возвращаемое значение

Тип: System.String

Идентификатор созданной копии

IGate.DeleteVariable — метод (Int32, String)

Удаляет переменную шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
void DeleteVariable(int Type, string ID);
```

Параметры

Type

Тип: System.Int32

Тип переменной

ID

Тип: System.String

Идентификатор переменной

IGate.GetVariable — метод (Int32, String, Object)

Получает переменную по её типу и строке-идентификатору.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
IGateVariable GetVariable(int TypeID, string SubTypeID, object ID)
```

Параметры

TypeID

Тип: System.Int32

Тип переменной

SubTypeID

Тип: System.String

Идентификатор подтипа переменной

ID

Тип: System.Object

Строка-идентификатор

Возвращаемое значение

Тип: IGateVariable

Переменная

IGate.GetVariableInternalReference — метод (Int32, String)

Возвращает ссылку на переменную шлюза во внутреннем формате.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
object GetVariableInternalReference(int Type, string ID);
```

Параметры

Type

Тип: System.Int32

Тип переменной

ID

Тип: `System.String`

Идентификатор переменной

Возвращаемое значение

Тип: `System.Object`

Ссылка

Заметки

Идентификатор объекта, переданный в `ID`, не обязательно содержит действительный идентификатор объекта в целевой системе. К примеру, в шлюзе к Docsvision при работе со строкой секции в `ID` содержатся идентификаторы карточки, секции и собственно строки. Метод `GetVariableInternalReference` в таком случае позволяет получить корректный идентификатор строки.

IGate.GetVariableReference — метод (Int32, String)

Возвращает внешнюю ссылку на заданную переменную шлюза.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Interfaces.dll`

Синтаксис

```
string GetVariableReference(int Type, string ID);
```

Параметры

Type

Тип: `System.Int32`

Тип переменной

ID

Тип: `System.String`

Идентификатор переменной

Возвращаемое значение

Тип: `System.String`

Ссылка на переменную шлюза

Заметки

К примеру, в шлюзе к Docsvision данный метод может вернуть ссылку в виде URL для открытия карточки, файла или папки.

IGateMessageSource — интерфейс

Определяет метод инициализации механизма подписки на события шлюза, а также свойство, через которое может быть получен канал подписки на события.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
public interface IGateMessageSource
```

Свойства

Имя	Описание
SubscriptionChannel	Возвращает канал подписки на события шлюза.

Методы

Имя	Описание
Initialize(Guid, IMessageChannel, ISubscriptionChannel)	Инициализирует механизм подписки на события шлюза.

IGateMessageSource.SubscriptionChannel — свойство

Возвращает канал подписки на события шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
ISubscriptionChannel SubscriptionChannel { get; }
```

Значение свойства

Тип: ISubscriptionChannel

Канал подписки на события

IGateMessageSource.Initialize — метод (Guid, IMessageChannel, ISubscriptionChannel)

Инициализирует механизм подписки на события шлюза.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
void Initialize(Guid instanceId, IMessageChannel messageChannel, ISubscriptionChannel
subscriptionChannel)
```

Параметры

instanceId

Тип: System.Guid

Идентификатор экземпляра шлюза

messageChannel

Тип: IMessageChannel

Канал отправки сообщений

subscriptionChannel

Тип: ISubscriptionChannel

Канал подписок на события

IGateVariable — интерфейс

Данный интерфейс описывает свойства стандартной переменной шлюза, единые для всех типов переменных. Все переменные, уникальные для конкретного шлюза, должны реализовывать этот интерфейс.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Interfaces.dll`

Синтаксис

```
public interface IGateVariable
```

Свойства

Имя	Описание
`DisplayValue`	Возвращает отображаемое значение.
`GateId`	Возвращает уникальный идентификатор шлюза.
`Id`	Возвращает идентификатор переменной.
`Name`	Возвращает название переменной.
`SubTypeId`	Возвращает подтип переменной.
`Type`	Возвращает тип переменной в шлюзе.

Методы

Имя	Описание
`Convert(VarTypeEnum, IGate, Int32)`	Приводит значение переменной к заданному типу.
`LogicalOperation(String, IGateVariable)`	Выполняет логическую операцию над переменными.

IGateVariableCustomProperties — интерфейс

Данный интерфейс описывает методы для работы с пользовательскими свойствами объекта шлюза, поддерживающего обработку подобных свойств.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Interfaces.dll`

Синтаксис

```
public interface IGateVariableCustomProperties
```

Методы

Имя	Описание
`GetCustomProperty(String)`	Получает значение переменной с заданным названием.
`SetCustomProperty(String, Object)`	Задаёт значение переменной с указанным названием.

ILicensableObject — интерфейс

Определяет поддержку шлюзом лицензирования, предоставляет тип лицензии.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Interfaces.dll`

Синтаксис

```
public interface ILicensableObject
```

Свойства

Имя	Описание
`Type`	Возвращает тип лицензии.

Методы

Имя	Описание
`CheckLicense(Object)`	Осуществляет проверку лицензии на шлюз. Проверка применима для типов лицензии: `LicenseTypeEnum.FileLicense` и `LicenseTypeEnum.StringLicense`. В метод будет передан объект `GateInfo.LicenseInfo`, содержащий лицензию полученную из справочника *шлюзов*.
`GetRequestedFeature`	Получает идентификатор опции лицензии. Применимо для типа лицензии: `LicenseTypeEnum.DocsVisionLicenseFeature`.

ILockableObject — интерфейс

Данный интерфейс описывает методы и свойства объекта шлюза, поддерживающего блокировку доступа к данным объекта.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Interfaces.dll`

Синтаксис

```
public interface ILockableObject
```

Свойства

Имя	Описание
`IsLocked`	Возвращает признак того, что объект заблокирован.

Методы

Имя	Описание
`PlaceLock`	Выполняет блокировку объекта.
`PlaceAutoLock`	Выполняет блокировку объекта и получает объект блокировки. Блокировка автоматически снимется при уничтожении объекта `ObjectLock`.
`RemoveLock`	Снимает блокировку с объекта.

ISubscriptionChannel — интерфейс

Данный интерфейс представляет источник сообщений и добавляет методы подписки на события от шлюза Workflow.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
public interface ISubscriptionChannel
```

Методы

Имя	Описание
Subscribe(Int32, Guid, Guid)	Выполняет подписку на событие шлюза Workflow.
Subscribe(Int32, Guid, Guid, String)	Выполняет подписку на событие шлюза Workflow.
Subscribe(Int32, Guid, Guid, Guid)	Выполняет подписку на событие шлюза Workflow.
Unsubscribe(Guid, Guid)	Отменяет подписку на указанное событие.
GetSubscriptions(Guid, Guid)	Возвращает список подписок для указанной функции и бизнес-процесса.
GetSubscriptions	Возвращает список всех подписок.

События

Имя	Описание
Subscribed	Позволяет обработать подписку на события Workflow.
Unsubscribed	Позволяет обработать отмену подписки на события Workflow.

ISubscriptionChannel.Subscribe — метод (Int32, Guid, Guid)

Выполняет подписку на событие шлюза Workflow.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
void Subscribe(int messageType, Guid processId, Guid functionId)
```

Параметры

messageType

Тип: System.Int32

Тип события

processId

Тип: System.Guid

Идентификатор бизнес-процесса, из которого выполняется подписка

functionId

Тип: System.Guid

Идентификатор функции бизнес-процесса, которая выполняет подписку

Заметки

Набор доступных типов событий индивидуален для каждого шлюза. Например, для шлюза к Docsvision, разрешены подписки на следующие типы событий:

- Изменение определённой карточки;
- Изменение любой карточки определённого типа;
- Разблокировка объекта;
- Блокировка объекта.

DocsVisionSubscriptionType — перечисление

Типы подписок на события, поддерживаемые шлюзом к Docsvision.

- **Пространство имён:** `DocsVision.Workflow.Gates`
- **Сборка:** `DocsVision.Workflow.Gates.dll`

Синтаксис

```
public enum DocsVisionSubscriptionType
```

Члены

Имя члена	Описание
`CardChange`	Изменение карточки.
`AnyCardChange`	Изменение любой карточки определённого типа.
`ObjectUnlock`	Разблокирование объекта.
`ObjectLock`	Блокировка объекта.

DVVariableType — перечисление

Типы переменных шлюза к Docsvision.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public enum DVVariableType
```

Члены

Имя члена	Описание
DOCUMENt	Карточка.
DATASECTIOn	Секция карточки.
DATAROw	Строка карточки.
DEPARTMENt	Подразделение.
PRINCIPAl	Сотрудник.
GROUp	Группа.
ROLe	Роль.
GROUPROLe	Групповая Роль.
FOLDEr	Папка.
FILe	Файл.
SHORTCUt	Ярлык.
DOCUMENT_TYPe	Тип карточки.
ENUm	Перечисление (поле типа enum).
PROCESs	Бизнес-процесс.
VARIABLe	Переменная бизнес-процесса.
VARIABLE_VALUe	Значение переменной бизнес-процесса.

FSSubscriptionType — перечисление

Типы подписок на события, поддерживаемые шлюзом к файловой системе.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public enum FSSubscriptionType
```

Члены

Имя члена	Описание
NewFileInFolder	Создание нового файла в папке.
FileChange	Изменение файла.

MailSubscriptionType — перечисление

Типы подписок на события, поддерживаемые шлюзом к почте.

- **Пространство имён:** DocsVision.Workflow.Gates
- **Сборка:** DocsVision.Workflow.Gates.dll

Синтаксис

```
public enum MailSubscriptionType
```

Члены

Имя члена	Описание
NewMessage	Появление нового сообщения.
MessageWithSubstringInSubject	Появление нового сообщение с определённым текстом в названии письма.

DocsVision.Workflow.Functions — пространство имён

В пространстве имён DocsVision.Workflow.Functions сосредоточены объекты, отвечающие за предоставление доступ к базовым функции Workflow, функциям взаимодействия с объектами Docsvision, файловой системой и с Microsoft Exchange.

Интерфейсы

Интерфейс	Описание
IFunction	Описывает свойства и методы серверного компонента функции.

Перечисления

Перечисление	Описание
`ExecResultEnum`	Определяет результаты выполнения *функции* и *сценария* в бизнес-процессе Workflow.

IFunction — интерфейс

Описывает свойства и методы серверного компонента функции.

- **Пространство имён:** `DocsVision.Workflow.Functions`
- **Сборка:** `DocsVision.Workflow.Runtime.dll`

Синтаксис

```
public interface IFunction
```

Свойства

Имя	Описание
`Data`	Задаёт или возвращает настройки функции. Позволяет передать функции настройки, заданные в клиентской части.
`FunctionData`	Задаёт или возвращает данные функции в процессе.
`AutoRemoveSubscriptions`	Задаёт или возвращает требование автоматического удаления подписки после обработки функции, действовавшие на момент начала работы функции
`AutoRemoveMessages`	Задаёт или возвращает требование автоматического удаления сообщения после обработки функции, полученные на момент начала работы функции.

Методы

Имя	Описание
`AllowExecute(ProcessInfo, PassState)`	Определяет возможность исполнения функции.
`Execute(ProcessInfo, PassState)`	Запускает функцию на исполнение.

Имя	Описание
`GetNextExecuteTime(Int32)`	Получает время следующего выполнения функции.
`GetExecuteDisallowReason(ProcessInfo, PassState)`	Возвращает причину запрета исполнения функции.
`GetName`	Получает название функции.

ExecResultEnum — перечисление

Определяет результаты выполнения *функции* и *сценария* в бизнес-процессе Workflow.

- **Пространство имён:** DocsVision.Workflow.Functions
- **Сборка:** DocsVision.Workflow.Runtime.dll

Синтаксис

```
public enum ExecResultEnum
```

Члены

Имя члена	Описание
KeepActive	Функция должна оставаться активной.
Done	Функция успешно выполнилась.
Stop	Функция успешно выполнилась, поток выполнения должен быть прерван.
Error	Произошла ошибка во время выполнения функции.
WaitForAnyFunctionFinished	Функцию нужно активировать после завершения какой-либо иной функции.
WaitForMessage	Функцию нужно активировать после получения сообщения.

DocsVision.Workflow.Objects — пространство имён

Пространство имён DocsVision.Workflow.Objects содержит описания структур данных карточек Workflow (карточка процесса, справочники)

Классы

Класс	Описание
Constraint	Предоставляет данные по предшествующему состоянию процесса.
ConstraintWrapper	Предоставляет идентификатор предшествующей функции для типа Constraint.
DocType	Предоставляет параметры инициирующего документа.
DocTypeWrapper	Предоставляет параметры инициирующего документа.
DictGate	Добавляет дополнительную информацию к основной информации по шлюзу, определённому в справочнике шлюзов Workflow.
DictGateWrapper	Представляет основную информацию по шлюзу, определённому в справочнике шлюзов Workflow.
DictFunction	Добавляет параметры преднастроенной функции к основной информации по функции Workflow, определённой в справочнике функций.
DictFunctionWrapper	Представляет основную информацию по функции Workflow, определённой в справочнике функций.
Dictionary	Представляет доступ к справочникам Workflow.
EnumValue	Предоставляет значение переменной перечислимого типа.
EnumValueWrapper	Предоставляет значение переменной перечислимого типа.
Gate	Представляет параметры шлюза в бизнес-процессе.

Класс	Описание
GateWrapper	Представляет параметры шлюза в бизнес-процессе.
Library	Представляет объектную модель библиотеки карточек Workflow.
Link	Описывает связь между двумя функциями в процессе.
LinkWrapper	Описывает связь между двумя функциями в процессе.
LogMessage	Представляет запись в журнале бизнес-процесса.
LogMessageWrapper	Представляет запись в журнале бизнес-процесса.
LogValue	Представляет запись в журнале изменения значения переменной.
LogValueWrapper	Представляет запись в журнале изменения значения переменной.
PassState	Предоставляет данные о текущем проходе функции.
PassStateWrapper	Предоставляет дополнительные данные о состоянии функции при проходе функции.
Process	Предоставляет доступ к данным бизнес-процесса.
ProcessWrapper	Предоставляет основную информацию по бизнес-процессу.
ProcFunction	Предоставляет параметры функции в бизнес-процессе.
Variable	Предоставляет данные переменной бизнес-процесса.
VariableWrapper	Предоставляет данные переменной бизнес-процесса.

Перечисления

Перечисление	Описание
ActionTypeEnum	Определяет тип записи в журнале процесса.
ExecutionModeEnum	Определяет режим исполнения процесса.
FuncStateEnum	Определяет состояние функции в текущем проходе.
LinkTypeEnum	Определяет связи между двумя функциями в бизнес-процессе.
ProcessStateEnum	Определяет состояние бизнес-процесса.
TemplateStateEnum	Определяет состояние шаблона бизнес-процесса.
VarTypeEnum	Определяет тип переменной бизнес-процесса.

Constraint — класс

Предоставляет идентификатор предшествующей функции.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class Constraint : ConstraintWrapper
```

Свойства

Имя	Описание
Id	Предоставляет данные по предшествующему состоянию процесса.

ConstraintWrapper — класс

Предоставляет идентификатор предшествующей функции для типа Constraint.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class ConstraintWrapper : IDisposable
```

Конструкторы

Имя	Описание
ConstraintWrapper(RowData)	Инициализирует новый экземпляр класса ConstraintWrapper с помощью указанного значения.

Свойства

Имя	Описание
PrevFunction	Задаёт или возвращает идентификатор предыдущей функции в процессе.

Методы

Имя	Описание
Dispose	Очищает ресурсы.

DocType — класс

Предоставляет параметры инициирующего документа.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class DocType : DocTypeWrapper
```

Заметки

См. описание базового класса `DocTypeWrapper`

DocTypeWrapper — класс

Предоставляет параметры инициирующего документа.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class DocTypeWrapper : IDisposable
```

Конструкторы

Имя	Описание
`DocTypeWrapper(RowData)`	Инициализирует новый экземпляр класса DocTypeWrapper с помощью указанного значения.

Свойства

Имя	Описание
`TypeId`	Задаёт или возвращает идентификатор типа инициирующей карточки.

Методы

Имя	Описание
`Dispose`	Очищает ресурсы.

DictGate — класс

Добавляет дополнительную информацию к основной информации по шлюзу, определённому в справочнике шлюзов Workflow.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class DictGate : DictGateWrapper
```

Свойства

Имя	Описание
Definition	Задаёт или возвращает описание шлюза.
LicenseInfo	Задаёт или возвращает информацию о лицензии.

DictGateWrapper — класс

Представляет основную информацию по шлюзу, определённому в справочнике шлюзов Workflow.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class DictGateWrapper : IDisposable
```

Конструкторы

Имя	Описание
DictGateWrapper(RowData)	Инициализирует новый экземпляр класса DictGateWrapper с помощью указанного значения.

Свойства

Имя	Описание
AddGateInNewProcess	Задаёт или возвращает признак того, что шлюз должен быть автоматически добавлен в новые процессы.
Assembly	Задаёт или возвращает название сборки, содержащей класс шлюза.
Class	Задаёт или возвращает название класса шлюза.
DefaultSettings	Задаёт или возвращает настройки по умолчанию.
Id	Задаёт или возвращает идентификатор шлюза.
Name	Задаёт или возвращает название.

Имя	Описание
`UIClass`	Задаёт или возвращает название класса, реализующего пользовательский интерфейс шлюза.

Методы

Имя	Описание
`Dispose`	При переопределении в производном классе, освобождает ресурсы.

DictFunction — класс

Добавляет параметры преднастроенной функции к основной информации по функции Workflow, определённой в справочнике функций.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class DictFunction : DictFunctionWrapper
```

Свойства

Имя	Описание
PredefinedFunctions	Возвращает параметры преднастроенной функции.

DictFunctionWrapper — класс

Представляет основную информацию по функции Workflow, определённой в справочнике функций.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class DictFunctionWrapper : IDisposable
```

Конструкторы

Имя	Описание
Public DictFunctionWrapper(RowData)	Инициализирует новый экземпляр класса DictFunctionWrapper с помощью указанного значения.

Свойства

Имя	Описание
Assembly	Задаёт или возвращает название сборки, в которой определён класс функции Workflow.
Class	Задаёт или возвращает название класса, реализующего функцию Workflow.
Id	Задаёт или возвращает идентификатор функции.
IsStart	Задаёт или возвращает признак того, что функция является стартовой.
IsStop	Задаёт или возвращает признак того, что функция является конечной.

Имя	Описание
`Name`	Задаёт или возвращает название функции Workflow.
`Obsolete`	Задаёт или возвращает признак того, что функция является устаревшей.
`PoolingInterval`	Задаёт или возвращает временной интервал опроса функции.
`UIClass`	Задаёт или возвращает название класса, реализующего интерфейс пользователя функции Workflow.

Методы

Имя	Описание
`Dispose`	При переопределении в производном классе, освобождает ресурсы.

Dictionary — класс

Представляет доступ к справочникам Workflow.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class Dictionary
```

Свойства

Имя	Описание
`Assemblies`	Возвращает зарегистрированные сборки.
`Functions`	Возвращает зарегистрированные функции.
`Gates`	Возвращает зарегистрированные шлюзы.
`GateSettings`	Возвращает настройки шлюзов.

EnumValue — класс

Предоставляет значение переменной перечислимого типа.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class EnumValue : EnumValueWrapper
```

Заметки

См. описание базового класса `EnumValueWrapper`

EnumValueWrapper — класс

Предоставляет значение переменной перечислимого типа.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class EnumValueWrapper : IDisposable
```

Конструкторы

Имя	Описание
EnumValueWrapper(RowData)	Инициализирует новый экземпляр класса EnumValueWrapper с помощью указанного значения.

Свойства

Имя	Описание
NumValue	Задаёт или возвращает номер переменной в коллекции.
Value	Задаёт или возвращает значение переменной.

Методы

Имя	Описание
Dispose	Очищает ресурсы.

Gate — класс

Представляет параметры шлюза в бизнес-процессе.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class Gate : GateWrapper
```

Заметки

См. описание базового класса `GateWrapper`.

GateWrapper — класс

Представляет параметры шлюза в бизнес-процессе.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class GateWrapper : IDisposable
```

Конструкторы

Имя	Описание
`GateWrapper(RowData)`	Инициализирует новый экземпляр класса GateWrapper с помощью указанного значения.

Свойства

Имя	Описание
`Caption`	Задаёт или возвращает название шлюза в процессе.
`Data`	Задаёт или возвращает персональные данные шлюза.
`Description`	Задаёт или возвращает описание.
`Id`	Задаёт или возвращает идентификатор.
`TypeId`	Задаёт или возвращает идентификатор шлюза в справочнике шлюзов.

Методы

Имя	Описание
`Dispose`	Очищает ресурсы.

Library — класс

Представляет объектную модель библиотеки карточек Workflow.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class Library
```

Конструкторы

Имя	Описание
`Library(UserSession)`	Инициализирует новый экземпляр класса Library с помощью указанного значения.
`Library(UserSession, Int32, Int32, Boolean)`	Инициализирует новый экземпляр класса Library с помощью указанного значения.

Свойства

Имя	Описание
`Dictionary`	Возвращает справочник СУБП.
`ExecutionMode`	Возвращает ссылку на справочники Workflow.
`FolderCard`	Возвращает ссылку карточку папок Docsvision.
`GatePool`	Возвращает пул шлюзов Workflow.
`Session`	Возвращает текущую сессию.
`SessionPool`	Возвращает пул сессий.

Методы

Имя	Описание
`CreateProcess(Boolean)`	Создаёт экземпляр бизнес-процесса.
`CreateProcess(Process)`	Создаёт экземпляр бизнес-процесса на основе имеющегося шаблона.
`CreateProcess(Boolean, Boolean)`	Создаёт экземпляр бизнес-процесса и инициализирует все шлюзы.
`CreateProcess(Boolean, Boolean)`	Создаёт экземпляр бизнес-процесса и добавляет в процесс все шлюзы.
`CreateProcess(Boolean, Boolean, Guid)`	Создаёт экземпляр бизнес-процесса и добавляет в процесс все шлюзы. В бизнес-процесс будет добавлена функция (в формате "Старт-Функция-Стоп") с указанным идентификатором.
`DeleteProcess(Guid)`	Удаляет бизнес-процесс с заданным идентификатором.
`GetProcess(Guid)`	Получает бизнес-процесс с указанным идентификатором.
`GetProcess(String, Guid)`	Получает бизнес-процесс с указанным идентификатором. При запросе будет использована сессия, выбранная из пула сессий по переданной строке соединения.
`GetProcesses(Boolean, Boolean)`	Возвращает коллекцию бизнес-процессов, используя переданные параметры в качестве фильтра.
`HandleProcessFinishing(Guid, Int32)`	Обрабатывает завершённой бизнес-процесс, который в зависимости от настроек может быть перемещен в архив, либо удалён.
`Shutdown`	Выполняет очистку пулов сессии и шлюзов, освободив занимаемые ресурсы.

Примеры

Ниже приведён пример использования объекта типа `Library` для запуска бизнес-процесса, полученного из имеющегося шаблона

```
①

Library library = new Library(userSession); ②

Process template = library.GetProcess(new Guid("00000000-0000-0000-0000-000000000000"));
③

Process process = library.CreateProcess(template); ④

process.Variables[new Guid("00000000-0000-0000-0000-000000000001")].Value = "Новое
значение"; ⑤

process.Start(userSession.Properties["AccountName"].Value.ToString(), library.Dictionary,
ExecutionModeEnum.Automatic, true); ⑥
```

① Инициализация контекста объектов.

② Инициализация объекта для работы с библиотекой карточкой СУБП.

③ Получение существующего шаблона бизнес-процесса.

④ Создание нового экземпляра бизнес-процесса.

⑤ Присвоение значения переменной бизнес-процесса.

⑥ Запуск экземпляра бизнес-процесса.

Идентификатор (`00000000-0000-0000-0000-000000000001`) переменной бизнес-процесса, может быть получен из шаблона БП при помощи утилиты Docsvision Explorer (поле "ID" секции "Переменные" карточки БП, используемой в качестве шаблона).

Link — класс

Описывает связь между двумя функциями в процессе.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class Link : LinkWrapper
```

Заметки

См. описание базового класса `LinkWrapper`.

LinkWrapper — класс

Описывает связь между двумя функциями в процессе.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class LinkWrapper : IDisposable
```

Конструкторы

Имя	Описание
`LinkWrapper(RowData)`	Инициализирует новый экземпляр класса `LinkWrapper` с помощью указанного значения.

Свойства

Имя	Описание
`Caption`	Задаёт или возвращает название связи.
`Destination`	Задаёт или возвращает идентификатор конечной функции.
`Disabled`	Задаёт или возвращает признак того, что связь является не активной.
`FunctionId`	Задаёт или возвращает идентификатор связанной функции.
`Id`	Задаёт или возвращает идентификатор связи.
`LinkType`	Задаёт или возвращает тип срабатывания связи.
`Source`	Задаёт или возвращает идентификатор начальной функции.

Методы

Имя	Описание
`Dispose`	Очищает ресурсы.

LogMessage — класс

Представляет запись в журнале бизнес-процесса.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class LogMessage : LogMessageWrapper
```

Свойства

Имя	Описание
Id	Возвращает идентификатор сообщения.
InUpdate	Возвращает признак включения для объекта режима отложенных изменения.

Методы

Имя	Описание
BeginUpdate	Включает режим отложенных изменений.
CancelUpdate	Отменяет несохранённые изменения и выключает режим отложенных изменений.
EndUpdate	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
Refresh	Выполняет обновление данных.
UpdateNow	Отправляет накопленные изменения на сервер.

LogMessageWrapper — класс

Представляет запись в журнале бизнес-процесса.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class LogMessageWrapper : IDisposable
```

Конструкторы

Имя	Описание
LogMessageWrapper(RowData)	Инициализирует новый экземпляр класса LogMessageWrapper с помощью указанного значения.

Свойства

Имя	Описание
Action	Задаёт или возвращает описание производимого действия.
ActionType	Задаёт или возвращает тип производимого действия.
ChangeState	Задаёт или возвращает смена состояния функции (для событий от функций).
FunctionName	Задаёт или возвращает имя функции (для событий от функций).
InputParameters	Задаёт или возвращает входные данные функции (для событий от функций).
Message	Задаёт или возвращает текст сообщение.

Имя	Описание
`MessageDate`	Задаёт или возвращает дату и время события.
`OutputParameters`	Задаёт или возвращает выходные данные функции (для событий от функций).
`Priority`	Задаёт или возвращает приоритет события (от `1` до `100`). Приоритет выставляется инициатором события по собственному усмотрению.

Методы

Имя	Описание
`Dispose`	Очищает ресурсы.

LogValue — класс

Представляет запись в журнале изменения значения переменной.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class LogValue : LogValueWrapper
```

Свойства

Имя	Описание
`Id`	Возвращает идентификатор записи.

LogValueWrapper — класс

Представляет запись в журнале изменения значения переменной.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class LogValueWrapper : IDisposable
```

Конструкторы

Имя	Описание
LogValueWrapper(RowData)	Инициализирует новый экземпляр класса LogValueWrapper с помощью указанного значения.

Свойства

Имя	Описание
Author	Задаёт или возвращает автора изменения.
ChangeDate	Задаёт или возвращает дату и время изменения.
Description	Задаёт или возвращает описание изменения.
Value	Задаёт или возвращает новое значение.

Методы

Имя	Описание
Dispose	Очищает ресурсы.

PassState — класс

Предоставляет данные о текущем проходе функции.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class PassState : PassStateWrapper
```

Свойства

Имя	Описание
Constraints	Возвращает коллекцию предшествующих ограничений.
Data	Задаёт или возвращает данные прохода.
ExecuteTime	Задаёт или возвращает время, за которое должен быть обработан проход функции.
FunctionId	Возвращает идентификатор функции.
HasErrors	Задаёт или возвращает признак того, что возникла ошибка.
HasWarnings	Задаёт или возвращает признак того, что имеются предупреждения.
Id	Возвращает идентификатор прохода.
InUpdate	Возвращает признак включения для объекта режима отложенных изменения.
StartTime	Задаёт или возвращает дату и время запуска функции.
State	Задаёт или возвращает состояние процесса.

Имя	Описание
TimeoutCount	Задаёт или возвращает счетчик таймаутов.
UseSparedData	Задаёт или возвращает признак того, что используются разделенные данные.

Методы

Имя	Описание
BeginUpdate	Включает режим отложенных изменений.
CancelUpdate	Отменяет несохранённые изменения и выключает режим отложенных изменений.
EndUpdate	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
GetLocalizedPassState(Int32)	Возвращает локализованное название текущего состояния функции.
Refresh	Запускает принудительное обновление данных карточки с сервера.
UpdateNow	Отправляет накопленные изменения на сервер.

PassStateWrapper — класс

Предоставляет дополнительные данные о состоянии функции при проходе функции.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class PassStateWrapper : IDisposable
```

Конструкторы

Имя	Описание
PassStateWrapper(RowData)	Инициализирует новый экземпляр класса PassStateWrapper с помощью указанного значения.

Свойства

Имя	Описание
Pass	Задаёт или возвращает номер прохода функции.
TimeoutCount	Задаёт или возвращает счетчик количества задержек исполнения.
UseSparedData	Задаёт или возвращает признак того, что используются разделенные данные.

Методы

Имя	Описание
Dispose	Очищает ресурсы.

Process — класс

Предоставляет доступ к данным бизнес-процесса.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public class Process : ProcessWrapper
```

Конструкторы

Имя	Описание
Process(CardData)	Инициализирует новый экземпляр класса Process с помощью указанного значения.

Свойства

Имя	Описание
DateCreated	Возвращает дата создания.
DateModified	Возвращает дату изменения.
Digest	Задаёт или возвращает дайджест.
DocTypes	Возвращает коллекцию типов инициирующих документов.
Functions	Возвращает коллекцию функций выполняемых в рамках бизнес-процесса (БП).
Gates	Возвращает коллекцию шлюзов.
Hash	Возвращает хэш. Фактически возвращает значение строки "Дополнительная информация".
Id	Возвращает идентификатор карточки БП.

Имя	Описание
`InitialDocument`	Задаёт или возвращает идентификатор карточки, которая вызвала создание данного экземпляра БП.
`InUpdate`	Возвращает признак режима отложенных изменения.
`IsTemplate`	Задаёт или возвращает признак того, что карточка является шаблоном.
`Links`	Возвращает коллекцию связей функций БП.
`LockStatus`	Возвращает статус блокировки карточки.
`Name`	Задаёт или возвращает название БП.
`Priority`	Задаёт или возвращает приоритет БП.
`SimpleMode`	Задаёт или возвращает признак того, что включён простой режим отображения панели инструментов СУБП.
`State`	Задаёт или возвращает состояние экземпляра процесса.
`Topic`	Задаёт или возвращает тему карточки.
`TopicId`	Возвращает идентификатор темы карточки.
`TopicIndex`	Возвращает индекс карточки внутри темы обработки.
`Variables`	Возвращает коллекцию переменных экземпляра БД.

Методы

Имя	Описание
`AddFunction(DictFunction, Int32, Int32, String, String, String, String, String)`	Добавляет функцию с бизнес-процессу.

Имя	Описание
`AddLink(Guid, Guid)`	Создаёт связь между двумя функциями.
`AddLogMessage(ActionTypeEnum, String, Int32, String, String, String, String, String)`	Создаёт запись в журнале текущего БП.
`AddToTopicChain(String)`	Добавляет БП к теме обработки карточки с заданным идентификатором.
`BeginUpdate`	Включает режим отложенных изменений.
`CancelUpdate`	Отменяет несохранённые изменения и выключает режим отложенных изменений.
`ClearLogMessages`	Удаляет записи журнала БП.
`ComputeExecutionMode(Dictionary)`	Определяет режим/платформу исполнения БП и задаёт `base.ExecutionMode`. При любых конфликтах определения платформы будет возвращено исключение.
`EndUpdate`	Отправляет накопленные изменения на сервер и выключает режим отложенных изменений.
`ForceUnlock`	Принудительно снимает блокировку с БП.
`GetFunctionByPassState(PassState)`	Возвращает функцию БП соответствующую заданному состоянию исполнения.
`GetMessages`	Возвращает очередь сообщений процесса.
`GetMessages(Guid)`	Возвращает очередь сообщений процесса от функции с заданным идентификатором.

Имя	Описание
`GetProcessDescription(ProcessStateEnum)`	Возвращает сформированное описание для указанного состояния БП.
`GetString(String)`	Возвращает значение указанного ресурса.
`GetString(String, Int32)`	Возвращает значение ресурса для указанного кода локализации.
`MarkCardForDeletion(String)`	Требует удаление указанной карточки при отмене изменений, если задействован режим отложенных изменений.
`Pause(String)`	Приостанавливает выполнение БП.
`PlaceLock`	Позволяет установить постоянную блокировку.
`Prepare`	Устанавливает статус подготовки БП к выполнению в false.
`Refresh`	Выполняет принудительное обновление данных карточки с сервера.
`RemoveLock`	Снимает блокировку БП в свой сессии.
`RemoveMessage(Guid)`	Удаляет сообщений процесса с указанным идентификатором.
`Resume(String, Dictionary)`	Продолжает выполнение приостановленного бизнес-процесса.
`SetNextMilestone(ProcFunction)`	Устанавливает следующую веху БП.
`Start(String, Dictionary)`	Запускает бизнес-процесс.
`Stop(String)`	Останавливает исполнение БП.
`UpdateNow`	Отправляет накопленные изменения на сервер.

Заметки

Некоторые из свойств данного класса относятся к интерфейсу СУПБ. Например, свойство `SimpleMode` — определяет режим отображения панели инструментов,

уменьшая или увеличивая количество доступных элементов конструктора.

ProcessWrapper — класс

Предоставляет основную информацию по бизнес-процессу.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class ProcessWrapper : IDisposable
```

Конструкторы

Имя	Описание
`ProcessWrapper(RowData)`	Инициализирует новый экземпляр класса ProcessWrapper с помощью указанного значения.

Свойства

Имя	Описание
`AfterFinishBehavior`	Задаёт или возвращает действие, которое должно быть совершено с процессом после завершения.
`AuthorCreated`	Задаёт или возвращает автора шаблона процесса.
`AuthorModified`	Задаёт или возвращает автора изменения.
`BuildNumber`	Задаёт или возвращает номер сборки Docsvision.
`Calendar`	Задаёт или возвращает идентификатор бизнес-календаря.
`CurrentPriority`	Задаёт или возвращает текущий приоритет.

Имя	Описание
DateBegin	Задаёт или возвращает дату начала работы бизнес-процесса.
DateBeginMsecs	Задаёт или возвращает число миллисекунд
DateEnd	Задаёт или возвращает дату окончания работы бизнес-процесса.
Description	Задаёт или возвращает описание процесса.
EncryptScripts	Задаёт или возвращает признак того, что скрипт шифруется.
ExecutionMode	Задаёт или возвращает режим исполнения бизнес-процесса.
Folder	Задаёт или возвращает папку экземпляров процесса.
FunctionsCount	Задаёт или возвращает число функций, выполняемых за единицу времени.
HasLayout	Задаёт или возвращает признак того, что имеется нарисованный граф функций.
Info	Задаёт или возвращает дополнительную информацию.
InstanceAuthor	Задаёт или возвращает автора экземпляра бизнес-процесса.
InstanceName	Задаёт или возвращает название экземпляра.
LastRunDate	Задаёт или возвращает дата и время последней обработки процесса.
LocaleId	Задаёт или возвращает идентификатор языка.
LoggingLevel	Задаёт или возвращает уровень журналирования.

Имя	Описание
`Name`	Задаёт или возвращает название бизнес-процесса.
`NextRunDate`	Задаёт или возвращает дата и время следующей обработки процесса.
`ParentProcess`	Задаёт или возвращает родительский процесс.
`Prepared`	Задаёт или возвращает признак того, что процесс подготовлен для выполнения.
`Priority`	Задаёт или возвращает исходный приоритет процесса.
`ReadyToRun`	Задаёт или возвращает того, что процесс готов к выполнению.
`Singleton`	Задаёт или возвращает признак того, что может быть запущен единственный экземпляр процесса.
`State`	Задаёт или возвращает состояние процесса.
`SubProcess`	Задаёт или возвращает признак того, что процесс является подпроцессом.
`TemplateProcess`	Задаёт или возвращает ссылку на шаблон процесса.
`TemplateState`	Задаёт или возвращает статус шаблона процесса.
`Version`	Задаёт или возвращает версию бизнес-процесса

Методы

Имя	Описание
`Dispose`	Освобождает ресурсы.

ProcFunction — класс

Предоставляет параметры функции в бизнес-процессе.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class ProcFunction
```

Свойства

Имя	Описание
`Caption`	Задаёт или возвращает название функции в бизнес-процессе.
`CardId`	Задаёт или возвращает идентификатор связанной карточки.
`Data`	Задаёт или возвращает персональные данные функции.
`Description`	Задаёт или возвращает описание.
`ErrCodeVarId`	Задаёт или возвращает идентификатор переменной, используемой для хранения кода ошибки.
`ErrDescriptionVarId`	Задаёт или возвращает идентификатор переменной, используемой для хранения описания ошибки.
`ExecutionCounter`	Задаёт или возвращает число отрабатываний функции.
`Height`	Задаёт или возвращает высоту отображаемого элемента функции.
`Id`	Задаёт или возвращает идентификатор.
`IsLinkedFunction`	Задаёт или возвращает признак того, что функция предназначена для связи.

Имя	Описание
IsMilestone	Задаёт или возвращает признак того, что функция является вехой.
MilestoneDateVarId	Задаёт или возвращает идентификатор переменной даты вехи.
MilestoneDelayType	Задаёт или возвращает тип задержки.
MilestoneDelayVarId	Задаёт или возвращает идентификатор переменной задержки вехи.
MilestoneNextDate	Задаёт или возвращает дату вехи.
MilestoneType	Задаёт или возвращает тип вехи.
ParentFunctionId	Возвращает идентификатор родительской функции.
PoolingInterval	Задаёт или возвращает интервал опроса функции.
ReuseStep	Задаёт или возвращает признак того, что разрешено повторное использование данных прохода.
States	Возвращает состояние функции в проходе.
TypeId	Задаёт или возвращает идентификатор функции в справочнике функций.
UseSparedData	Задаёт или возвращает признак того, что используются разделенные данные.
WeakCardId	Задаёт или возвращает слабую ссылку на карточку.
Width	Задаёт или возвращает ширину отображаемого элемента функции.
XPos	Задаёт или возвращает координату по оси X для отображаемого элемента функции.

Имя	Описание
`YPos`	Задаёт или возвращает координату по оси Y для отображаемого элемента функции.

Методы

Имя	Описание
`Refresh`	Обновляет данные функции.

Variable — класс

Предоставляет данные переменной бизнес-процесса.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class Variable : VariableWrapper
```

Свойства

Имя	Описание
`DisplayValue`	Задаёт или возвращает отображаемое значение.
`EnumValues`	Возвращает коллекцию значений перечисления (для переменных перечислимого типа).
`LogValues`	Возвращает журнал значений переменной (для переменных с историей изменения значений).
`Value`	Задаёт или возвращает значение переменной.
`Values`	Возвращает коллекцию значений переменной.

VariableWrapper — класс

Предоставляет данные переменной бизнес-процесса.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public class VariableWrapper : IDisposable
```

Конструкторы

Имя	Описание
`VariableWrapper(RowData)`	Инициализирует новый экземпляр класса `VariableWrapper` с помощью указанного значения.

Свойства

Имя	Описание
`CreateCopy`	Задаёт или возвращает признак того, что необходимо создавать копию объекта в шлюзе при создании экземпляра процесса.
`Description`	Задаёт или возвращает описание переменной.
`GateId`	Задаёт или возвращает идентификатор шлюза, которому принадлежит переменная (для переменной типа, зарегистрированного в шлюзе).
`HiddenInParentProcess`	Задаёт или возвращает признак того, что переменная скрыта в родительском процессе.
`Id`	Задаёт или возвращает идентификатор переменной.

Имя	Описание
IsAdded	Задаёт или возвращает признак того, что переменная была добавлена во время выполнения процесса.
IsAdditive	Задаёт или возвращает признак переменной с сохранением истории изменения значений.
IsDefault	Задаёт или возвращает признак наличия значения "по умолчанию".
IsMultipleValued	Задаёт или возвращает признак переменной-коллекции.
IsRequired	Задаёт или возвращает признак обязательного заполнения значения переменной.
Name	Задаёт или возвращает название переменной.
SubTypeId	Задаёт или возвращает идентификатор подтипа переменной в шлюзе.
TypeId	Задаёт или возвращает идентификатор типа переменной в шлюзе (для переменной типа, зарегистрированного в шлюзе).
UseSparedValue	Задаёт или возвращает признак того, что переменная использует дополнительное значение.
VarType	Задаёт или возвращает тип переменной.

Методы

Имя	Описание
Dispose	Очищает ресурсы.

ActionTypeEnum — перечисление

Определяет тип записи в журнале процесса.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public enum ActionTypeEnum
```

Члены

Имя члена	Описание
`Error`	Ошибка.
`Warning`	Предупреждение.
`Information`	Информационное сообщение.

ExecutionModeEnum — перечисление

Определяет режим исполнения процесса.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Interfaces.dll

Синтаксис

```
public enum ExecutionModeEnum
```

Члены

Имя члена	Описание
Automatic	Определить автоматически
X86	32-разрядный
X64	64-разрядный
Any	Любой

FuncStateEnum — перечисление

Определяет состояние функции в текущем проходе.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public enum FuncStateEnum
```

Члены

Имя члена	Описание
NonActive	Не активна.
Wait	Готова к исполнению.
Active	Выполняется.
Done	Проход завершен.
Finished	Процесс завершен.
Error	Завршено с ошибкой.
WaitForAnyFunctionFinished	Ожидание завершения любой другой функции.
WaitForLinkedCardChange	Ожидание изменений в связанной карточке. *Не используется и не обрабатывается.*
WaitForWeakLinkedCardChange	Ожидание изменений в слабо-связанной карточке. *Не используется и не обрабатывается.*
WaitForMessage	Ожидание поступления сообщения.

LinkTypeEnum — перечисление

Определяет связи между двумя функциями в бизнес-процессе.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public enum LinkTypeEnum
```

Члены

Имя члена	Описание
`LinkSuccess`	Выполняется только при успешном завершении функции.
`LinkFail`	Выполняется только при неудачном завершении функции.
`LinkCompletion`	Выполняется всегда.
`LinkFunction`	Выполняется при завершении ассоциированной со связью функцией.

ProcessStateEnum — перечисление

Определяет состояние бизнес-процесса.

- **Пространство имён:** `DocsVision.Workflow.Objects`
- **Сборка:** `DocsVision.Workflow.Objects.dll`

Синтаксис

```
public enum ProcessStateEnum
```

Члены

Имя члена	Описание
`NonActive`	Остановлен.
`Active`	Активен.
`Paused`	Приостановлен.
`Failed`	Приостановлен из-за ошибки.
`Finished`	Завершён успешно.

TemplateStateEnum — перечисление

Определяет состояние шаблона бизнес-процесса.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public enum TemplateStateEnum
```

Члены

Имя члена	Описание
Design	В разработке.
InUse	Используется.
Test	Тестирование.
ReadyToStart	Готов к созданию экземпляров.

VarTypeEnum — перечисление

Определяет тип переменной бизнес-процесса.

- **Пространство имён:** DocsVision.Workflow.Objects
- **Сборка:** DocsVision.Workflow.Objects.dll

Синтаксис

```
public enum VarTypeEnum
```

Члены

Имя члена	Описание
Generic	Переменная типа, зарегистрированного в шлюзе.
Integer	Целое.
Float	Дробное.
String	Строка.
Boolean	Логическое значение.
DateTime	Дата и время.
Enum	Перечисление.

DocsVision.Workflow.ObjectsUI — пространство имён

Пространство имён DocsVision.Workflow.ObjectsUI содержит описания элементов пользовательского интерфейса, взаимодействующих с реализуемыми объектами Workflow.

Интерфейсы

Интерфейс	Описание
IObjectLicenseInfo	Определяет поддержку объектом функции лицензирования.

Интерфейс	Описание
`IObjectLocalize`	Дополнительный интерфейс для объектов, поддерживающих локализацию.
`IObjectUI`	Интерфейс, реализующий свойства клиентских объектов СУБП (функций, шлюзов).

Перечисления

Перечисление	Описание
`LicenseTypeEnum`	Определяет возможные типы лицензий для шлюза Workflow.

IObjectLicenseInfo — интерфейс

Определяет поддержку объектом функции лицензирования.

- **Пространство имён:** DocsVision.Workflow.ObjectsUI
- **Сборка:** DocsVision.Workflow.Settings.dll

Синтаксис

```
public interface IObjectLicenseInfo
```

Свойства

Имя	Описание
Data	Задаёт или возвращает данные лицензии.
Type	Возвращает тип лицензии.

IObjectLocalize — интерфейс

Дополнительный интерфейс для объектов, поддерживающих локализацию.

- **Пространство имён:** `DocsVision.Workflow.ObjectsUI`
- **Сборка:** `DocsVision.Workflow.Settings.dll`

Синтаксис

```
public interface IObjectLocalize
```

Методы

Имя	Описание
`GetLocaleId`	Возвращает идентификатор языка.
`SetLocaleID(Int32)`	Задаёт идентификатор языка, используемого при локализации интерфейса.

IObjectUI — интерфейс

Интерфейс, реализующий свойства клиентских объектов СУБП (функций, шлюзов).

- **Пространство имён:** `DocsVision.Workflow.ObjectsUI`
- **Сборка:** `DocsVision.Workflow.Settings.dll`

Синтаксис

```
[Guid("6883CB9D-7FED-427F-993C-E8580DB0D6B6"), TypeLibType(4288)]
[ComImport]
internal interface IObjectUI
```

Свойства

Имя	Описание
`Id`	Возвращает идентификатор объекта.
`TypeName`	Возвращает название типа объекта.
`Icon`	Возвращает иконку.
`Data`	Задаёт или возвращает данные объекта (функции, шлюза).

LicenseTypeEnum — перечисление

Определяет возможные типы лицензий для шлюза Workflow.

- **Пространство имён:** DocsVision.Workflow.ObjectsUI
- **Сборка:** DocsVision.Workflow.Settings.dll

Синтаксис

```
public enum LicenseTypeEnum
```

Члены

Имя члена	Описание
NO_LICENSE_REQUIREd	Лицензия не требуется.
DOCSVISION_FEATURe	Лицензия определена в лицензии на платформу Docsvision.
STRING_LICENSe	Лицензия задана в виде строки.
FILE_LICENSe	Лицензия задана в виде лицензионного файла.

DocsVision.Workflow.Runtime — пространство имён

Пространство имён DocsVision.Workflow.Runtime содержит различные типы, используемые при запуске и обработке бизнес-процессов.

Классы

Класс	Описание
ProcessInfo	Объект типа ProcessInfo выступает в качестве входного параметра функции сценария. С его помощью сценарий может оперировать данными процесса.

Класс	Описание
`ProcessVariable`	Тип `ProcessVariable` используется при определении переменной бизнес-процесса.

ProcessInfo — класс

Объект типа `ProcessInfo` выступает в качестве входного параметра функции *сценария*. С его помощью *сценарий* может оперировать данными процесса.

- **Пространство имён:** `DocsVision.Workflow.Runtime`
- **Сборка:** `DocsVision.Workflow.Runtime.dll`

Синтаксис

```
public class ProcessInfo
```

Конструкторы

Имя	Описание
`ProcessInfo(Process, Library)`	Инициализирует новый экземпляр типа `ProcessInfo` с помощью указанного значения.
`ProcessInfo(Process, Library, UserSession)`	Инициализирует новый экземпляр типа `ProcessInfo` с помощью указанного значения.

Свойства

Имя	Описание
`ActivePassStates`	Возвращает список с информацией об активных проходах функций.
`ExecutionStartDate`	Задаёт или возвращает дату и время начала обработки процесса.
`FunctionsExecuted`	Задаёт или возвращает число обработанных функций.
`Gates`	Возвращает шлюзы процесса.
`Id`	Возвращает идентификатор процесса.
`InLinks`	Возвращает таблицу со связями, входящими в функцию.
`Library`	Возвращает объектную модель библиотеки Workflow.

Имя	Описание
`LocaleId`	Возвращает идентификатор текущей локализации.
`MessageChannel`	Возвращает канал сообщений.
`OutLinks`	Возвращает таблицу со связями, исходящими из функции.
`Prepared`	Задаёт или возвращает признак того, что процесс подготовлен для выполнения.
`ProcessData`	Возвращает данные процесса.
`Session`	Возвращает активную сессию.
`Variables`	Возвращает коллекцию переменных процесса.
`Version`	Возвращает версию процесса.

Методы

Имя	Описание
`AddLogMessage(ActionTypeEnum, String, Int32, String, String, String, String)`	Создаёт записи в журнале работы Workflow.
`BuildLinkTables`	Создаёт таблицы исходящих и входящих связей для функций процесса.
`GetActiveItems`	Возвращает список функций процесса готовых к выполнению.
`GetDefaultGate(Guid)`	Возвращает шлюз с указанным идентификатором.
`GetGateByName(String)`	Возвращает шлюз с заданным названием.
`GetHasActiveFunctions`	Возвращает признак того, что в процессе есть активные функции.
`GetProcessState`	Возвращает текущее состояние процесса.
`GetValue(String, Guid, Int32, String)`	Возвращает из шлюза переменную с заданным идентификатором типа и названием/идентификатором.

Имя	Описание
`GetVariableByName(String)`	Возвращает первую переменную с заданным именем.
`Prepare`	Выполняет подготовку процесса к запуску — осуществляет загрузку необходимых переменные из шлюза.
`PrepareExecution`	Сбрасывает состояния функций процесса, переводит начальную функцию (установлен флаг `IsStart`) в состояние `FuncStateEnum.Wait`.
`Refresh`	Перегружает данные из карточки процесса в процесс.
`ReleaseGates`	Освобождает все занятые объекты всех шлюзов.
`ReportError(String, Exception)`	Добавляет в журнал сообщение об ошибке.
`ReportError2(String, Exception, Object[])`	Добавляет в журнал сообщение об ошибке с передачей в шаблон сообщения дополнительных данных.
`ReportInfo(String)`	Добавляет в журнал информационное сообщение.
`ReportInfo2(String, Object[])`	Добавляет в журнал информационное сообщение с передачей в шаблон сообщения дополнительных данных.
`ReportMessage(ActionTypeEnum, Int32, String, String)`	Добавляет в журнал сообщение заданного типа.
`ReportMessage2(ActionTypeEnum, Int32, String, String, Object[])`	Добавляет в журнал сообщение заданного типа с передачей в шаблон сообщения дополнительных данных.
`ReportWarning(String, String)`	Добавляет в журнал предупреждение.
`ReportWarning2(String, String, Object[])`	Добавляет в журнал предупреждение с передачей в шаблон сообщения дополнительных данных.
`SetProcessInfo(ProcessStateEnum, DateTime, Boolean)`	Принудительно задаёт состояние процесса, время последнего запуска и признак подготовки к выполнению.

Имя	Описание
`UpdateFunctionStatistics(Guid, Double)`	Обновляет время последнего запуска указанной функции.

ProcessVariable — класс

Тип `ProcessVariable` используется при определении переменной бизнес-процесса.

- **Пространство имён:** `DocsVision.Workflow.Runtime`
- **Сборка:** `DocsVision.Workflow.Runtime.dll`

Синтаксис

```
public class ProcessVariable
```

Конструкторы

Имя	Описание
`ProcessVariable(Variable, ProcessInfo)`	Инициализирует новый экземпляр типа ProcessVariable с помощью указанного значения.

Свойства

Имя	Описание
`AutoRefreshDisplayValue`	Задаёт или возвращает признак того, что отображаемое значение переменной будет обновляться автоматически при изменении её значения.
`CreateCopy`	Возвращает признак того, что при подготовке процесса к выполнению будет создана копия переменной.
`DisplayValue`	Возвращает отображаемое значение объекта, хранящегося в переменной.
`GateId`	Возвращает идентификатор шлюза, которому принадлежит переменная. Для простых типов этот идентификатор является пустым `GUID`.

Имя	Описание
`Id`	Возвращает идентификатор переменной.
`IsMultipleValued`	Возвращает флаг, показывающий, что переменная хранит несколько значений — отмечена флагом "Коллекция значений".
`IsNull`	Возвращает признак того, что переменная является пустой (null) — значение не задано.
`IsRequired`	Возвращает признак того, что параметр является обязательным.
`KeepHistory`	Задаёт или возвращает признак того, что для переменной будет вестись журнал изменений значений
`LogValues`	Возвращает историю значений переменной.
`Name`	Возвращает название переменной.
`ProcessInfo`	Возвращает информацию о процессе.
`SubTypeId`	Возвращает подтип переменной в шлюзе.
`Type`	Возвращает тип переменной.
`TypeId`	Возвращает идентификатор типа переменной в шлюзе. Для простых типов переменных не заполняется.
`Value`	Возвращает значение переменной. Для простых типов переменных содержит значение соответствующего типа (`string`, `long` и т.д.). Для шлюзовых переменных содержит экземпляр переменной шлюза, реализующий стандартный интерфейс `IGateVariable`.

Имя	Описание
`Values`	Возвращает коллекцию значений для переменной, отмеченной флагом "Коллекция значений".
`VarEnumValues`	Возвращает коллекцию строчных значений (для переменных типа "Перечисление").

Методы

Имя	Описание
`Refresh`	Выполняет принудительное обновление данных процесса.
`RefreshDisplayValue`	Выполняет обновление отображаемого значения переменной.
`SetValue(Object, String, String)`	Задаёт значение переменной.

Руководство разработчика Web-клиента

Руководство по разработке

Данный раздел посвящён разработке программных компонентов для решений на базе Web-клиента.

В разделе приводится информация по разработке:

- Клиентских скриптов,
- Серверных и клиентских расширений,
- Расширений программы Конструктор Web-разметок,
- Элементов управления,
- Другую информацию, связанную с разработкой Решений,
- Дополнительных компонентов Web-клиента,
- Комментарии к исходному коду примеров дополнительных компонентов,
- Описания членов объектной модели Web-клиента, задействованных при разработке компонентов,
- Справочные сведения, касающиеся разработки на базе Web-клиента.

Руководство предназначено для специалистов, обладающих знаниями в области веб-разработки, знакомых с разработкой компонентов на ReactJS, обладающих опытом разработки на платформе Docsvision 6.

Обратите внимание. При обращении к API, методы следует вызывать последовательно, один за другим. Параллельный вызов методов API не гарантирует их работу.

Перевод решений на .NET 8.0

Web-клиент версии dv6 работает на .NET 8.0 с возможностью установки на ОС Astra Linux, поэтому при переходе с версии 5.5.17 и ниже необходимо также перевести на .NET 8.0 собственные решения. Подробнее см. "Перевод решений на .NET 8.0".

Изменение файла проекта

Конвертация существующего проекта может быть выполнена:

- При помощи автоматических инструментов (см. документацию Microsoft).
- При помощи создания нового проекта, нацеленного на .NET 8. При создании проекта предпочтительно использовать шаблоны VisualStudio из примеров разработки расширений для версии 6.1.

В проектах нового типа нет необходимости добавлять файлы с исходным кодом в `.csproj`-файл проекта. По умолчанию автоматически все файлы в папке проекта считаются частью проекта. Это существенно упрощает процесс преобразования проекта.

Важной особенностью нового проекта является атрибут `Sdk` на корневом элементе `Project` файла `.csproj`, это xml-файл, который можно открыть в блокноте. Для web-расширений рекомендуется значение `Microsoft.NET.Sdk.Web`, для расширений конструктора разметок `Microsoft.NET.Sdk`.

Значение тега `TargetFramework` для web-расширений должно быть `net8.0`, для расширений конструктора разметок — `net8.0-windows`.

Для проектов расширений конструктора разметок также могут быть полезны следующие элементы:

```
<UseWindowsForms>true</UseWindowsForms>
<UseWPF>true</UseWPF>
<ImportWindowsDesktopTargets>true</ImportWindowsDesktopTargets>
```

Особенности сборки

Для сборки C# кода для .NET 8 требуется VisualStudio 2022 или старше.

При использовании сторонних зависимостей требуется обновить их до версий, поддерживающих .NET 8. В случае нарушений совместимости при сборке будет выдаваться предупреждение `CA1416`. Данные предупреждения следует все устранять.

При разработке расширений для программы Конструктор Web-разметок и другого кода, рассчитанного на работу в ОС Windows следует использовать атрибут `SupportedOSPlatform`. Его можно назначить на отдельный класс, либо на всю сборку:

```
[assembly: SupportedOSPlatform("windows")]
```

Без этого атрибута также может возникать предупреждение CA1416, при обращении к API, рассчитанному на работу в ОС Windows.

Логика привязки параметров в запросах

- Метод контроллера либо принимает несколько параметров простых типов через query-параметры, либо один параметр, значение которого читается из тела запроса.
- Рекомендуется явно указывать, откуда читается параметр с помощью атрибутов [FromBody] и [FromQuery]. Если ранее использовался атрибут FromUri, то нужно заменить на FromQuery.
- Рекомендуется явно помечать методы атрибутами [HttpGet], [HttpPost], для указания метода вызова

Например, метод может быть объявлен так:

```
public class MathController : ControllerBase
{
    [HttpGet]
    public double GetSquare([FromQuery] double value) {
        return
    }
}
```

Обращение к методу будет, соответственно, выглядеть так:

```
let result = await requestManager.get(`/Math/GetSquare?value=2`, null);
```

Метод может быть объявлен так:

```
class MinValueParams {
    public List<double> Values { get; set; };
}

public class MathController : ControllerBase
{
    [HttpPost]
    public double MinValue([FromBody] MinValueParams  arg) {
        return arg.Values.Min();
    }
}
```

Если не указать [FromBody] для сложного параметра, его значение

не привяжется.

Обращение к такому методу будет выглядеть следующим образом:

```
let arg = { values: [ 1, 5, 3, 2] };
let result = await requestManager.post(`/Math/MinValue`, JSON.stringify(arg));
```

При передаче параметра в теле запроса нельзя указывать имя параметра, т.к. он может быть только один.

Следующий код не будет работать:

```
let data = { arg: { values: [ 1, 5, 3, 2] } }; ①
let result = await requestManager.post(`/Math/MinValue`, JSON.stringify(data)); ②
```

① Неверный код.

② Код не будет работать.

Внедрение зависимостей

В Web-клиенте 18 (6.1) вместо Autofac используется стандартный механизм внедрения зависимостей .NET. В связи с этим в классе серверного расширения нужно произвести следующие изменения:

Вместо метода `InitializeContainer` *определить метод:*

```
public override void InitializeServiceCollection(IServiceCollection services)
{
①
}
```

① Регистрация сервисов.

Регистрация сервисов изменяется следующим образом:

```
  containerBuilder.RegisterType<YourServiceClass>().As<YourServiceInterface
>().SingleInstance();
  ->
  services.AddSingleton<YourServiceInterface, YourServiceClass>();

  containerBuilder.RegisterOrderedType<YourBindingConverterType, IBindingConverter>();
  ->
  services.AddSingleton<IBindingConverter, YourBindingConverterType>();

  containerBuilder.RegisterOrderedType<YourBindingResolverType, IBindingResolver>();
```

```
  ->
  services.AddSingleton<IBindingResolver, YourBindingResolverType>();

  containerBuilder.RegisterOrderedType<YourControlResolverType, IControlResolver>();
  ->
  services.AddSingleton<IControlResolver, YourControlResolverType>();

  containerBuilder.RegisterOrderedType<YourPropertyResolverType, IPropertyResolver>();
  ->
  services.AddSingleton<IPropertyResolver, YourPropertyResolverType>();

  containerBuilder.RegisterType<YourCardLifeCycle>().Keyed<ICardLifeCycle>(CardTypeID
).SingleInstance();
  ->
  services.AddTransient<ICardLifeCycle, YourCardLifeCycle>();

  containerBuilder.RegisterType<YourRowLifeCycle>().Keyed<IRowLifeCycle>(SectionID
).SingleInstance();
  ->
  services.AddTransient<IRowLifeCycle, YourRowLifeCycle>();
```

Прочие изменения

- В конструкторе класса расширения при вызове базовой реализации теперь не нужно передавать `serviceProvider`:

  Было

  ```
  public ServerExtension(IServiceProvider serviceProvider)
    : base(serviceProvider)
  {
  }
  ```

  Стало

  ```
  public ServerExtension(IServiceProvider serviceProvider)
    : base()
  {
  }
  ```

- Классы контроллеров рекомендуется наследовать от `Microsoft.AspNetCore.Mvc.ControllerBase`. При переводе методы `requestManager get` и `post` в некоторых случаях могут изменить возвращаемое значение. В частности, ранее они могли возвращать строку, а после перевода — объект — результат `JSON.parse` этой строки. С примером можно ознакомиться ниже.

Было

```
const response = JSON.parse(await requestManager.get(url));
```

Стало

```
const response = await requestManager.get(url)
```

- Также в отдельных случаях может измениться регистр первой буквы в названии свойств ответа сервера. Например, следующим образом:

  Было

  ```
  const name = response.IssueName;
  ```

  Стало

  ```
  const name = response.issueName;
  ```

- Изменился способ загрузки и скачивания файлов. Пример реализации соответствующего метода на .NET6 можно найти в примере `ExternalWebService`, в файле `DocumentsController.cs`, методах `UploadFile` и `DownloadFile`.
- Чтение настроек из конфигурационного файла Web-сервиса изменилось. Прежние сервисы оставлены для обратной совместимости, однако рекомендуется использовать стандартный механизм работы с настройками .NET 8. В частности, использовать внедрение интерфейса `IOptions` через конструктор для получения настройки и `AddOptions` + `BindConfiguration` при регистрации сервисов для регистрации настройки. Пример работы с настройками можно найти в примере `KonturIntegration` (регистрация см. `KonturServerExtension.cs/InitializeServiceCollection`, чтение см. `KonturRequestService.cs`).
- Библиотека `iTextSharp`, которая использовалась в примере `Watermark`, поддерживает .NET6 только в новой версии, под названием `itext7`. В этой новой версии полностью изменился API.

С чего начать

Исходные коды примеров серверных и клиентских расширений, расширений программы Конструктор Web-разметок, а также элементов управления содержатся в репозитории WebClient-Samples на ресурсе GitHub.

Загрузите к себе репозиторий, чтобы начать работать с примерами.

Пример рассчитан на версию Web-клиента dv6 или выше.

Подготовка

Для сборки примеров в Visual Studio[1] (серверные расширения и расширения программы Конструктор Web-разметок) необходимы актуальные версии компонентов Web-клиента и системы Docsvision, которые можно скопировать в папку Assemblies из каталога установки Web-клиента и системы Docsvision.

Для проверки зависимостей и быстрой сборки сразу всех примеров можно воспользоваться установочным пакетом Docsvision Web-client SDK. При установке файлы примеров будут собраны и помещены в соответствующие каталоги. При необходимости (например, если Web-клиент уже установлен) вы можете изменить путь сборки через переменную окружения в node и проектах Visual Studio[2]. Если в примере требуется загрузка разметки или библиотеки карточек, это необходимо произвести вручную.

После установки SDK в `C:\Program Files (x86)\Docsvision\WebClient\SDK\USAGE.md` указаны инструкции по разворачиванию примеров.

1. Подготовьте среду разработки.

   На компьютере разработчика должны быть установлены:

   - Microsoft Visual Studio 2022.
   - NodeJS v22.14.0 и выше или выше.
   - TypeScript 3.8.

   Если указанные компоненты не установлены, при установке SDK будет предложено их автоматически загрузить и установить.

2. Загрузите на компьютер репозиторий WebClient-Samples.
3. Установите шаблоны проектов из полученного репозитория:
   - *TemplateDesignerExtension* — шаблон проекта для разработки расширения программы Конструктор Web-разметок.
   - *TemplateServerExtension* — шаблон проекта для разработки серверного расширения.

Чтобы установить шаблоны:

1. Скопируйте папки `Docsvision Web-layout designer extension template` и `Docsvision Web-client server extension template` из каталога репозитория `ProjectTemplates\VSTemplates\` в каталог шаблонов проектов Visual Studio[3] (`C:\Users\\%username%\Documents\Visual Studio %version%\Templates\ProjectTemplates\`).
2. Запустите Visual Studio[4]. Будут добавлены новые типы проектов: *TemplateDesignerExtension* и *TemplateServerExtension*.

Что дальше

1. Определите цель разработки.

   Web-клиент имеет несколько точек расширения функциональных возможностей.

   Программист может разработать:

   - Серверное расширение, добавляющее методы WebAPI, для вызова с клиентского уровня.

     Серверные методы могут быть использованы для взаимодействия с сервером Docsvision: создание карточек, запуск БП и т. д.

   - Новый тип элементов управления для создания разметок.

     Данный тип расширения включает два компонента: *расширение программы Конструктор Web-разметок* (обеспечивает возможность настройки разметки с использованием ЭУ) и *клиентское расширение* (обеспечивает функционирование ЭУ на клиентской стороне).

   - Клиентский скрипт, содержащий обработчики событий разметки.

     Данные скрипты могут добавлять в Web-клиент необходимую бизнес-логику или обеспечивать другие функции клиентского уровня.

   - Клиентское расширение — общее название компонентов, обеспечивающих функциональность клиентского уровня.
2. Ознакомьтесь с инструкцией по разработке требуемого типа компонента Web-клиента в данном руководстве.
3. Ознакомьтесь с предоставляемыми примерами.

Примеры

Папка репозитория `WebClient-Samples` *содержит каталоги:*

- `Assemblies` — папка с подключаемыми сборками Web-клиента, платформы Docsvision и др..
- `ClientScripts` — папка с исходными кодами клиентских расширений, предоставляющих обработчики событий. Исходные коды приведены на TypeScript (подкаталог TypeScript).
- `ControlProperties` — папка с исходным кодом расширения программы Конструктор Web-разметок, добавляющего новый тип свойств элемента управления.
- `Controls` — папка с исходными кодами элементов управления.
- `Others` — папка с прочими примерами.
- `Obsolete` — устаревшие примеры, работа которых не поддерживается в текущей версии Web-клиента.
- `ProjectTemplates` — папка с шаблонами проектов серверного расширения и расширения программы Конструктор Web-разметок.
- `ServerExtensions` — папка с исходными кодами серверных расширений.
- `Install` и другие папки — служебные директории.

Примеры сопровождаются файлами `README.md`, в которых приведено краткое описание примера и инструкция по его сборке и проверке.

Часть исходных кодов представляет собой проекты Visual Studio[5] (решение `Samples`), а часть — отдельные файлы клиентских расширений (TypeScript, CSS).

Список примеров

Примеры клиентских расширений

Все клиентские скрипты также включают в себя проекта расширения Web-клиента.

Примеры клиентских расширений с обработчиками событий (скриптов):

- `ClientScripts\TypeScript\ChangeLogoAndBackground` — демонстрирует пример реализации скрипта, изменяющего заголовок страницы и её фоновое изображение для папок и дашборда.
- `ClientScripts\TypeScript\ControlsRelation` — пример демонстрирует способ

организации взаимодействия элементов разметки, через обработчики событий (в примере видимость одного элемента управления зависит от значения другого).

- `ClientScripts\TypeScript\CopyValueToControl` — пример метода, в котором выполняется копирование значения одного элемента управления в другой.
- `ClientScripts\TypeScript\ExtendServerQueryData` — индикатор высокого приоритета для просроченных заданий.
- `ClientScripts\TypeScript\FileListHighlightRow` — добавляет красную обводку при скачивании файла.
- `ClientScripts\TypeScript\GridColumnBackgroundRows` — новая колонка таблицы грида — *Фон строки*.
- `ClientScripts\TypeScript\GridContextMenuItem` — новый пункт контекстного меню Открыть карточку в модальном окне.
- `ClientScripts\TypeScript\GridHoverPanelGrouping` — группировка по столбцу.
- `ClientScripts\TypeScript\GridToolbarButton` — кнопка в гриде отображает общее количество карточек, а также количество непрочитанных.
- `ClientScripts\TypeScript\LoginPage` — настройка страницы авторизации.
- `ClientScripts\TypeScript\ShowRequiredFields` — отображение незаполненных обязательных полей.
- `ClientScripts\TypeScript\ValidateOnStateChanging` — пример реализации обработчика изменения состояния карточки, осуществляющего проверку значения текстового поля, с отменой изменения состояния, если поле не заполнено или количество знаков в строке больше `100`.
- `ClientScripts\AdvancedTypeScript` — примеры клиентских расширений, в которых задействован React API:
  - `ClientScripts\AdvancedTypeScript\AdaptiveMenuBar` — пример реализации панели меню с собственным набором элементов.
  - `ClientScripts\AdvancedTypeScript\MessageBox` — пример вызова стандартных диалоговых окон Web-клиента.
  - `ClientScripts\AdvancedTypeScript\ModalDialog` — пример реализации диалогового окна с собственным содержимым.
  - `ClientScripts\AdvancedTypeScript\ModalSidebar` — пример вызова стандартной боковой панели Web-клиента.

Примеры серверных расширений

- `ServerExtensions\CSPSignatureVisualization` — пример изменения визуализации штампа электронной подписи: вставки собственных изображений штампов электронной подписи в файл, приложенный к карточке Docsvision.
- `ServerExtensions\CreateCard` — пример реализации функции создания исходящих документов, представленной в виде клиентской и серверной частей:
  - `ServerExtensions\CreateCard\CreateCardServerExtension` — серверное расширение, добавляющее метод WebAPI, создающий карточку исходящего документа.
  - `ServerExtensions\CreateCard\CreateCardWebExtension` — клиентское расширение, добавляющее клиентский сервис создания документа (вызывает метод расширения `CreateCardServerExtension`) и пример обработчика событий, использующего клиентский сервис.
- `ServerExtensions\CreateCardDialog` — серверное расширение, добавляющее в Web-клиент возможность создания карточек типа *Этап согласования*:
  - `ServerExtensions\CreateCardDialog\CreateCardDialogServerExtension` — серверное расширение, добавляющее фабрику карточек, позволяющую создавать карточки Этап согласования.
  - `ServerExtensions\CreateCardDialog\CreateCardDialogWebExtension` — клиентское расширение, регистрирующее новый тип карточек на стороне клиента.
- `ServerExtensions\CustomConditionTypes` — отображение разметки в соответствии с заданными условиями.
  - `ServerExtensions\CustomConditionTypes\CustomConditionTypesDesignerExtension` — проект-расширение для программы Конструктор Web-разметок. Содержит класс `CustomConditionTypesProvider`, в котором реализованы методы для отображения новых условий выбора разметки по группе, пользователю, роли
  - `ServerExtensions\CustomConditionTypes\CustomConditionTypesLayoutExtension` — проект-расширение для Web-клиента, которое позволяет определить — выполняется условие выбора разметки или нет для текущего контекста.
  - `ServerExtensions\CustomConditionTypes\CustomConditionTypesInterfaces` — проект-расширение для программы Конструктор Web-разметок и серверного расширения, содержащий все константные значения для этих проектов.
- `ServerExtensions\CustomLibrary` — пример демонстрирует способ добавление нового типа карточек. Пример включает несколько компонентов:

- ServerExtensions\CustomLibrary\CardDefs — схема новой библиотеки карточек с карточкой справочника.
- ServerExtensions\CustomLibrary\CustomLibrary.ObjectModel — классы объектной модели библиотеки карточек.
- ServerExtensions\CustomLibrary\CustomLibraryServerExtension — серверное расширение, добавляющее в Web-клиент поддержку нового типа карточек и WebApi функцию получения данных карточек нового типа.
- ServerExtensions\CustomLibrary\CustomLibraryWebExtension — клиентское расширение, добавляющее клиентский сервис для вызова функции серверного расширения CustomLibraryServerExtension и пример обработчика событий, использующего клиентский сервис.

- ServerExtensions\DataGridControlExtension — плагин загружает информацию о файлах для документа.
  - ServerExtensions\DataGridControlExtension\DataGridControlServerExtension — проект-расширение для Web-клиента. Содержит плагин Files (FilesDataGridControlPlugin), реализующий загрузку информации о файлах документа.
- ServerExtensions\ExcelExport — экспорт в Excel.
  - ServerExtensions\ExcelExport\ExcelExportServerExtension — проект-расширение для Web-клиента, позволяющего вмешать в процесс экспорта в Excel.
- ServerExtensions\ExtendedCardInfo — пример демонстрирует возможность получения данных карточки, которые по умолчанию не загружаются и не передаются клиенту. Пример представлен в виде клиентской и серверной частей:
  - ServerExtensions\ExtendedCardInfo\ExtendedCardInfoServerExtension — серверное расширение, добавляющее WebApi функцию загрузки из Docsvision данных карточки.
  - ServerExtensions\ExtendedCardInfo\ExtendedCardInfoWebExtension — клиентское расширение, добавляющее клиентский сервис для вызова функции серверного расширения ExtendedCardInfoServerExtension и пример обработчика событий, использующего клиентский сервис.
- ServerExtensions\LicenseCheck — демонстрация способа проверки лицензии Docsvision на наличие дополнительной опции. Пример представлен в виде клиентской и серверной частей:

    - ServerExtensions\LicenseCheck\LicenseCheckServerExtension — серверное расширение, реализующее функцию проверки лицензии.
    - ServerExtensions\LicenseCheck\LicenseCheckWebExtension — клиентское расширение, добавляющее клиентский сервис для вызова функции проверки лицензии из серверного расширения LicenseCheckServerExtension и пример обработчика событий, использующего клиентский сервис.
- ServerExtensions\ShiftTasksEndDate — пример реализации функции изменения данных связанных карточек. Пример представлен в виде клиентской и серверной частей:
    - ServerExtensions\ShiftTasksEndDate\ShiftTasksEndDateServerExtension — серверное расширение, в котором реализована функция изменения времени исполнения в заданиях, связанных с документом.
    - ServerExtensions\ShiftTasksEndDate\ShiftTasksEndDateWebExtension — клиентское расширение, добавляющее клиентский сервис для вызова функции изменения связанных карточек и пример обработчика событий, использующего клиентский сервис.
- ServerExtensions\TableControl — пример демонстрирует способ получения данных из справочника контрагентов и отображения их в таблице. Пример представлен в виде клиентской и серверной частей:
    - ServerExtensions\TableControl\TableControlServerExtension — серверное расширение с функцией WebApi, предоставляющей данные контрагентов;
    - ServerExtensions\TableControl\TableControlWebExtension — клиентское расширение, реализующее несколько функций: добавляет клиентский сервис для вызова функции серверного расширения TableControlServerExtension. Предоставляет обработчик открытия карточки, использующий клиентский сервис для получения данных контрагентов. Обеспечивает загрузку данных контрагентов в таблицу.

Примеры элементов управления

- Controls\AcquaintancePanel — элемент управления Панель отправки на ознакомление, предназначен для запуска БП отправки документа на ознакомление. Пример включает несколько компонентов:
    - AcquaintancePanel\AcquaintancePanelDesignerExtension — текстовый описатель элемента управления, а также расширение программы Конструктор Web-разметок с новым типом свойств и локализованными ресурсами.

    - AcquaintancePanel\AcquaintancePanelServerExtension — серверное расширение, предоставляющее функцию запуска бизнес-процесса отправки документа на ознакомление.
    - AcquaintancePanel\AcquaintancePanelWebExtension — клиентское расширение с клиентским компонентом элемента управления и сервисом вызова функции запуска БП из расширения AcquaintancePanelServerExtension.
- Controls\CheckBox — элемент управления "Флаг", предназначен для установки и отображения значения булева типа. Пример включает два компонента:
    - CheckBox\CheckBoxDesignerExtension — расширение программы Конструктор Web-разметок с бинарным описателем элемента управления, новым свойством DefaultValue и редактором BooleanMetadataEditor значения свойства DataField, который ограничивает список доступных для выбора полей карточки.
    - CheckBox\CheckBoxWebExtension — клиентское расширение с клиентским компонентом элемента управления.
- Controls\DocumentSignBatchOperation — пример реализации элемента управления Групповая операция / Подписание.
    - Controls\DocumentSignBatchOperation\DocumentSignBatchOperationWebExtension — проект-расширение серверной части Web-клиента. Содержит ЭУ групповой операции Подписание документа.
    - Controls\DocumentSignBatchOperation\DocumentSignBatchOperationDesignerExtension — содержит решение для программы Конструктор Web-разметок с локализованными ресурсами.
- Controls\DownloadFilesGroupOperation — пример реализации элемента управления группой операции — загрузки файлов выбранных карточек Документ на компьютер. Пример включает несколько компонентов:
    - DownloadFilesGroupOperation\DownloadFilesGroupOperationDesignerExtension — расширение программы Конструктор Web-разметок с бинарным описателем элемента управления, новыми свойствами downloadDocumentFileMode и BatchOperationRestrictionFoldersPropertyDescription, и источником данных DownloadDocumentFileModeSource для свойства downloadDocumentFileMode.
    - DownloadFilesGroupOperation\DownloadFilesGroupOperationServerExtension — серверное расширение с функцией, возвращающей файлы карточки.
    - DownloadFilesGroupOperation\DownloadFilesGroupOperationWebExtension — клиентское расширение с клиентским компонентом элемента управления и

сервисом вызова функции получения файлов карточки из расширения `DownloadFilesGroupOperationServerExtension`.

- `Controls\ExchangeRates` — элемент управления `ExchangeRates`, предназначен для отображения курса валют, получаемого с внешнего ресурса. Пример включает два компонента:
  - `ExchangeRates\ExchangeRatesDesignerExtension` — текстовый описатель элемента управления, а также расширение программы Конструктор Web-разметок с локализованными ресурсами.
  - `ExchangeRates\ExchangeRatesWebExtension` — клиентское расширение с клиентским компонентом элемента управления.
- `Controls\HyperComments` — элемент управления `HyperComments`, предназначен для отображения виджета комментариев внешнего веб-приложения `HyperComments`. Пример включает два компонента:

  Обратите внимание, пример устарел. Для его использования требуется лицензия.

  - `HyperComments\HyperCommentsDesignerExtension` — расширение программы Конструктор Web-разметок с бинарным описателем элемента управления, новым типом свойств и локализованными ресурсами.
  - `HyperComments\HyperCommentsWebExtension` — клиентское расширение с клиентским компонентом элемента управления.
- `Controls\Image` — элемент управления `Image`, предназначен для отображения галереи изображений. В примере демонстрируется возможность работы со значением элемента управления сложного типа и локализацией. Пример включает несколько компонентов:
  - `Image\ImageDesignerExtension` — расширение программы Конструктор Web-разметок с бинарным описателем элемента управления, новыми типами свойств и новым редактором `SliderEditor`.
  - `Image\ImageServerExtension` — серверное расширение с конвертером значения `SliderConverter`, который применяется при загрузке значения элемента управления в клиент.
  - `Image\ImageWebExtension` — клиентское расширение с клиентским компонентом элемента управления.
- `Controls\Link` — элемент управления `Link`, предназначен для отображения

ссылки. Пример включает два компонента:

  - `Link\LinkDesignerExtension` — текстовый описатель элемента управления, а также расширение программы Конструктор Web-разметок с локализованными ресурсами.
  - `Link\LinkWebExtension` — клиентское расширение с клиентским компонентом элемента управления.

- `Controls\RefCases` — элемент управления `RefCases`, предназначен для выбора и отображения дела из Справочника номенклатуры дел.

  Обратите внимание, пример устарел. Модуль *Делопроизводство 5* устарел и не поддерживается. Функциональность

  - `RefCases\RefCasesDesignerExtension` — текстовым описателем элемента управления, а также расширение программы Конструктор Web-разметок с дополнительными редакторами;
  - `RefCases\RefCasesServerExtension` — серверное расширение с конвертером значения RefCasesConverter, который применяется при загрузке значения элемента управления в клиент, и сервисами для получения данных из *Справочника номенклатуры дел*.
  - `RefCases\RefCasesWebExtension` — клиентское расширение с реализацией элемента управления.

- `Controls\TextBox` — элемент управления `TextBox`, предназначен для текстового поля с меткой. Пример включает два компонента:

  - `TextBox\TextBoxDesignerExtension` — расширение программы Конструктор Web-разметок с бинарным описателем элемента управления, новым типом свойств и локализованными ресурсами.
  - `TextBox\TextBoxWebExtension` — клиентское расширение с клиентским компонентом элемента управления.

- `ControlProperties\Url` — расширение программы Конструктор Web-разметок с описателем нового свойства **`Url`**.

Прочие примеры

- `Others\CustomStageService` — демонстрационный вариант специального сервиса логики этапа согласования.
- `Others\ExternalWebService` — пример отдельного веб-сервиса, предоставляющего

методы для взаимодействия с платформой Docsvision через WebApi:

- Получение информации о карточке документа по её идентификатору.
- Создание карточки документа по переданной модели.
- Обновление данных карточки документа.
- Удаление карточки документа по её идентификатору.
- Изменение состояния карточки документа.
- Прикрепление файла к карточке документа.
- Получение файла по его идентификатору.
- Получение результата выполнения расширенного отчёта.

- `Others\KonturIntegration` — пример реализации сервиса интеграции с системой Контур.Фокус, который открывает возможность:

  a. При создании Контрагента заполнить реквизиты, полученные из Контур.Фокус на основе указанного ИНН.

  b. В карточке Документа получить из Контур.Фокус отчёт о Контрагенте и приложить его в карточку Документа.

  Содержит следующие компоненты:

  - `KonturServerExtension` — папка с серверным расширением Web-клиента, в котором реализовано проксирование запросов к Контур.Фокус (для обхода ограничений CORS в браузере).
  - `KonturWebExtension` — папка с клиентским расширением, в котором реализовано открытие отчета о контрагенте и загрузка информации.

- `Others\SignalForUsers` — пример использования сервиса отправки оповещений пользователям Web-клиента:
  - `Others\SignalForUsers\SignalServerExtension` — серверное расширение, предоставляющее функцию рассылки оповещений.
  - `Others\SignalForUsers\SignalWebExtension` — клиентское расширение с клиентским сервисом и обработчиком, вызывающим функцию рассылки оповещений.

- `Others\Watermark` — пример расширения программы _DVWebTool_s, добавляющего водяной знак в PDF файлы документа:
  - `Others\Watermark\WatermarkServerExtension` — серверное расширение, предоставляющее методы для получения и добавления файлов в документ.

- Others\Watermark\WatermarkWebExtension — клиентское расширение, вызывающее функцию добавления водяного знака.
- Others\Watermark\WatermarkWebToolExtension — расширение программы _DVWebTool_s, реализующее функцию добавления водяного знака в PDF-файлы на компьютере пользователя.

Шаблоны проектов серверных расширений

Папка ProjectTemplates содержит шаблоны проектов для создания серверных расширений Web-клиента, программы Конструктор Web-разметок, а также клиентских расширений.

- ProjectTemplates\TemplateDesignerExtension — шаблон проекта расширения программы Конструктор Web-разметок.
- ProjectTemplates\TemplateServerExtension — шаблон проекта серверного расширения Web-клиента.
- ProjectTemplates\TemplateWebExtension — шаблон проекта web-расширения.
- ProjectTemplates\TemplateXmlDesignerExtension — XML-шаблон нового ЭУ.

Репозиторий "WebClient-Samples"

Репозиторий "WebClient-Samples" — репозиторий компании "ДоксВижн" на ресурсе GitHub.

Адрес репозитория: https://github.com/Docsvision/WebClient-Samples/.

Репозиторий "WebClient-Samples" содержит исходные коды примеров и шаблоны проектов для Visual Studio[6] для текущей и предыдущих версий Web-клиента.

Как работать со справочником по JS API

Справочник по JavaScript API модуля Web-клиент — ресурс, содержащий описание классов, функций и других членов предоставляемого Web-клиентом клиентского API, используемого при разработке клиентских расширений и скриптов.

Справочник по JS API размещен по адресу: https://docsvision.com/docs/webclient/версия-Web-клиента/JsDocApi/, например, для Web-клиента 6.1 адрес будет следующим: https://docsvision.com/docs/webclient/6.1/JsDocApi/.

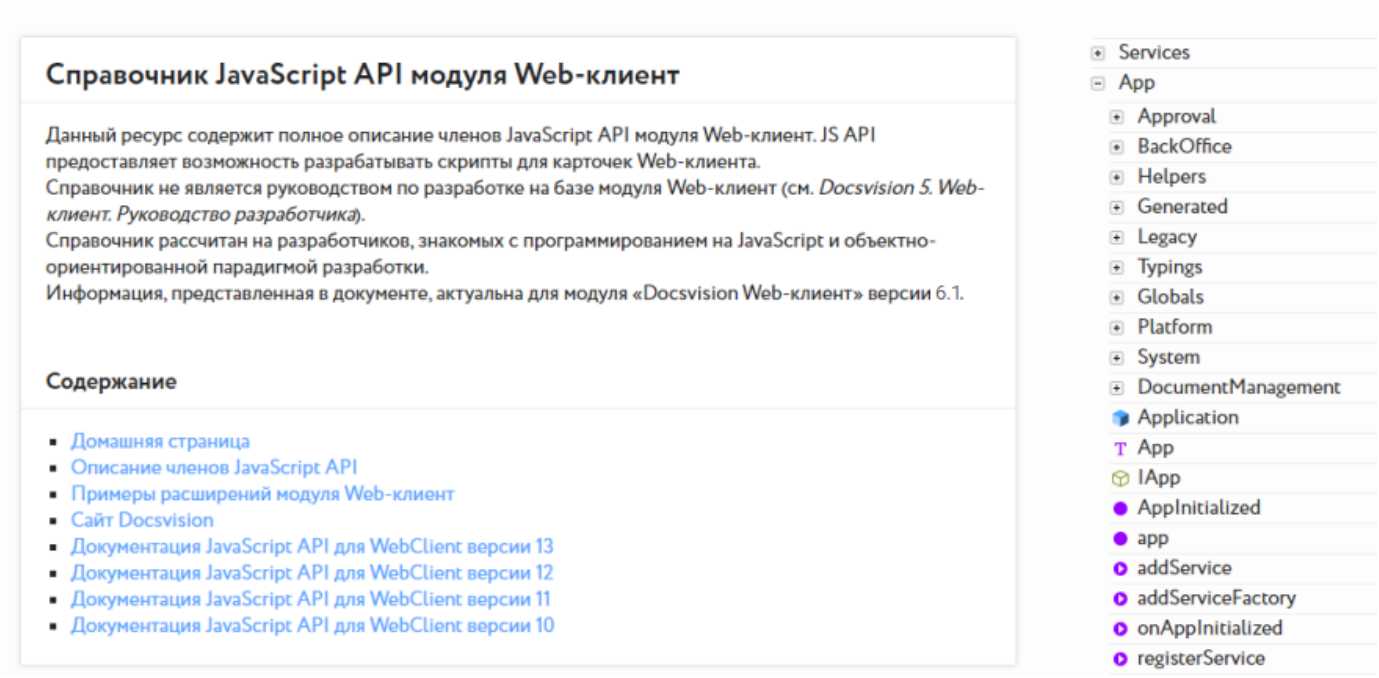


Рисунок 103. Главная страница справочника по JS API

Окно справочника содержит:

- Строку поиска,
- Флаги `Показывать protected-члены` (включает показ защищенных членов JS API) и `Показывать унаследованные члены` (включает показ унаследованных членов JS API),
- Меню для перехода к описанию элементов JS API,
- Легенду с описанием условных обозначений.

Как работать с описаниями сущностей

Чтобы получить описание для определённого класса, метода и т.п., нужно воспользоваться поиском или перейти к описанию из правого меню. К примеру, чтобы получить описание для класса элемента управления `Сотрудник`, нужно выполнить поиск по слову `Employee` (название класса элемента управления `Сотрудник`), или открыть из правого блока элемент **App › BackOffice › Controls › Employee › Employee ›** .

Значок слева от названия ссылки указывает, что данный раздел посвящен описанию класса. Условные обозначения всех типов сущностей приведены в легенде.

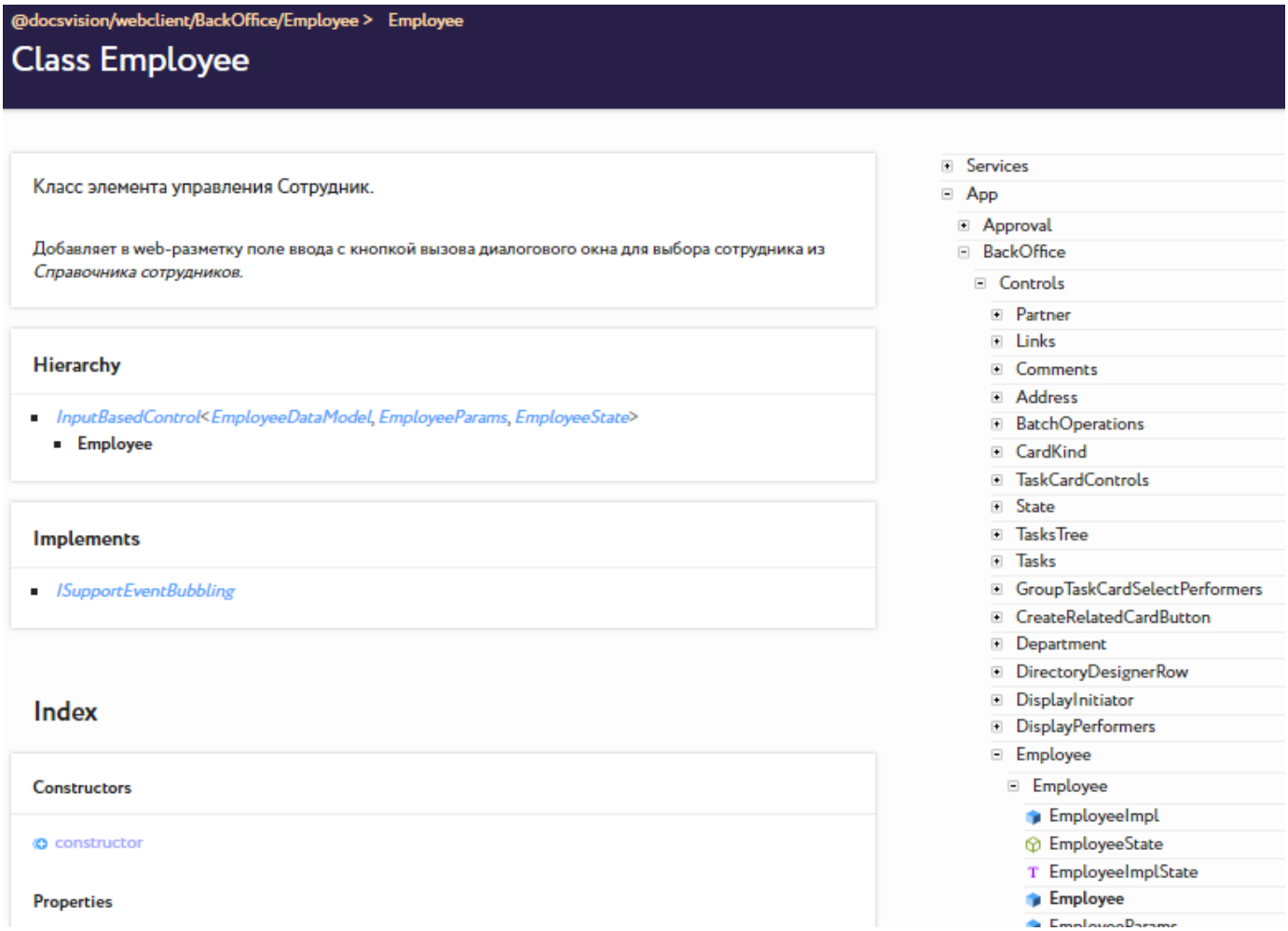


Рисунок 104. Страница с описанием класса Employee

На странице с описанием класса приведены:

- Общая информация о классе,
- Родительский класс,
- Конструкторы,
- Методы,
- Свойства и др.

В зависимости от типа описываемого члена JS API список может изменяться.

Некоторые члены JS API могут быть отмечены предупреждением:

Это внутренний нестабильный API, который может измениться в следующих версиях Web-клиент.

Использовать такой класс, интерфейс или метод следует с осторожностью, т.к. его сигнатура не стабильна и может измениться в следующих версиях Web-клиента.

Над названием сущности указан модуль Web-клиента (на изображении — `@docsvision/webclient/BackOffice/Employee`), содержащий данную сущность.

Чтобы перейти к описанию метода, полей и пр., кликните по ссылке с названием данного члена API.

Описание поля класса содержит:

- Название поля,
- Тип поля,
- Примечание.

Описание метода содержит:

- Название метода,
- Сигнатуру, примечание,
- Список параметров метода с типами,
- Тип возвращаемого значения.

Получение описания класса элемента управления

1. Определите название класса элемента управления.

Название класса ЭУ обычно совпадает с названием его типа, которое указано в свойстве `Название ЭУ` в программе Конструктор Web-разметок.

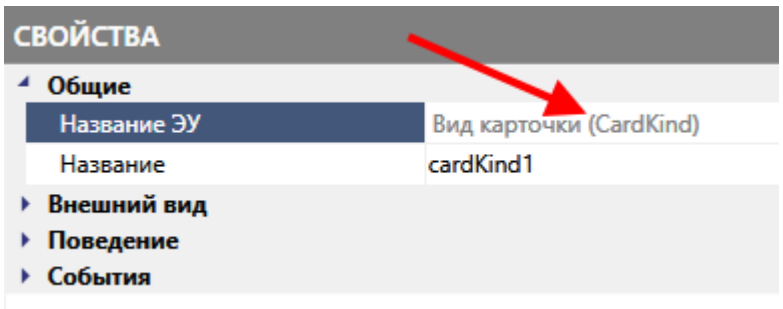


Рисунок 105. Тип элемента управления в программе Конструктор Web-разметок

2. Перейдите к описанию класса в справочнике:
    - Воспользуйтесь прямой ссылкой из описания элемента управления,

которое приведено в данной документации.

- Воспользуйтесь поиском по справочнику или правым меню.

Публичные свойства и события элемента управления программно связаны с его классом и содержатся в свойстве **params** класса.

Чтобы получить описание для класса публичных свойств ЭУ:

a. Найдите на странице секцию Properties.

b. Найдите описание свойства params.

c. Щелкните по ссылке с типом свойства. Будет открыто описание типа свойства params.

   Описание типа свойства params аналогично описанию других типов.

Чтобы получить описание для класса значения ЭУ:

a. Перейдите к описанию класса публичных свойств ЭУ.

b. Найдите на странице секцию **Properties**.

c. Найдите описание свойства **value**.

d. Щелкните по ссылке с типом свойства. Будет открыто описание типа свойства **value**.

   Описание типа свойства **value** аналогично описанию других типов.

Чтобы получить описание для класса события ЭУ:

a. Перейдите к описанию класса публичных свойств ЭУ.

b. Найдите на странице секцию **Properties**.

c. Найдите описание требуемого события — на уровне объектной модели ЭУ его событие является свойством ЭУ.

d. Щелкните по ссылке с типом события. Будет открыто описание класса.

К примеру, тип события **click** элемента управления `Кнопка`, которое возникает при нажатии, — `BasicApiEvent`. Тип аргумента, передаваемого с событием, указан в угловых скобках.

Событие типа `BasicApiEvent` (реализует интерфейс **IBasicEvent**) является непрерываемым, т. е. остановить обработку данного события другими обработчиками нельзя. Если тип события `CancelableApiEvent` (реализует интерфейс `ICancelableEvent`), то данное событие является

прерываемым (подробнее в разделе *Прерывание выполнения операции*).

Как работать с описанием веб-API

Справочник с описанием веб-методов, предоставляемых сервером Docsvision и его расширениями, доступен по адресу http://WebC-server.domain.com:5004/swagger/). Справочник также включает REST Web-API, доступный на стороне клиента Web-клиента, который может вызывать функции Docsvision API (работа с дискреционной безопасностью, карточками, файлами, метаданными и т.д.). API предназначается для работы из клиентских скриптов.

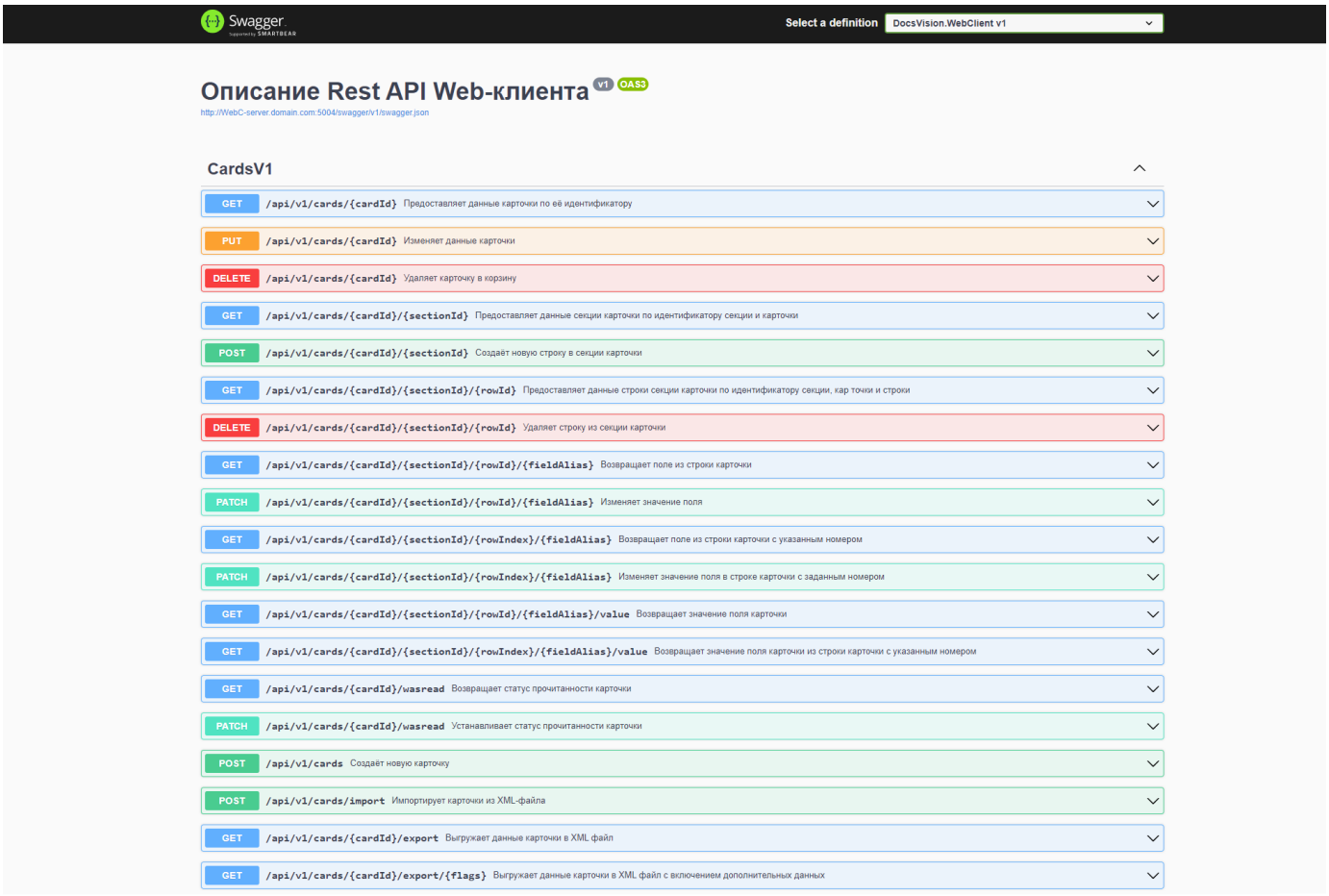


Рисунок 106. Страница описания веб-API

Описание методов сгруппировано по контролёрам, предоставляющим методы.

Чтобы перейти к описанию метода:

1. Нажмите на название контролёра, затем по названию метода. Будет раскрыт блок описания.

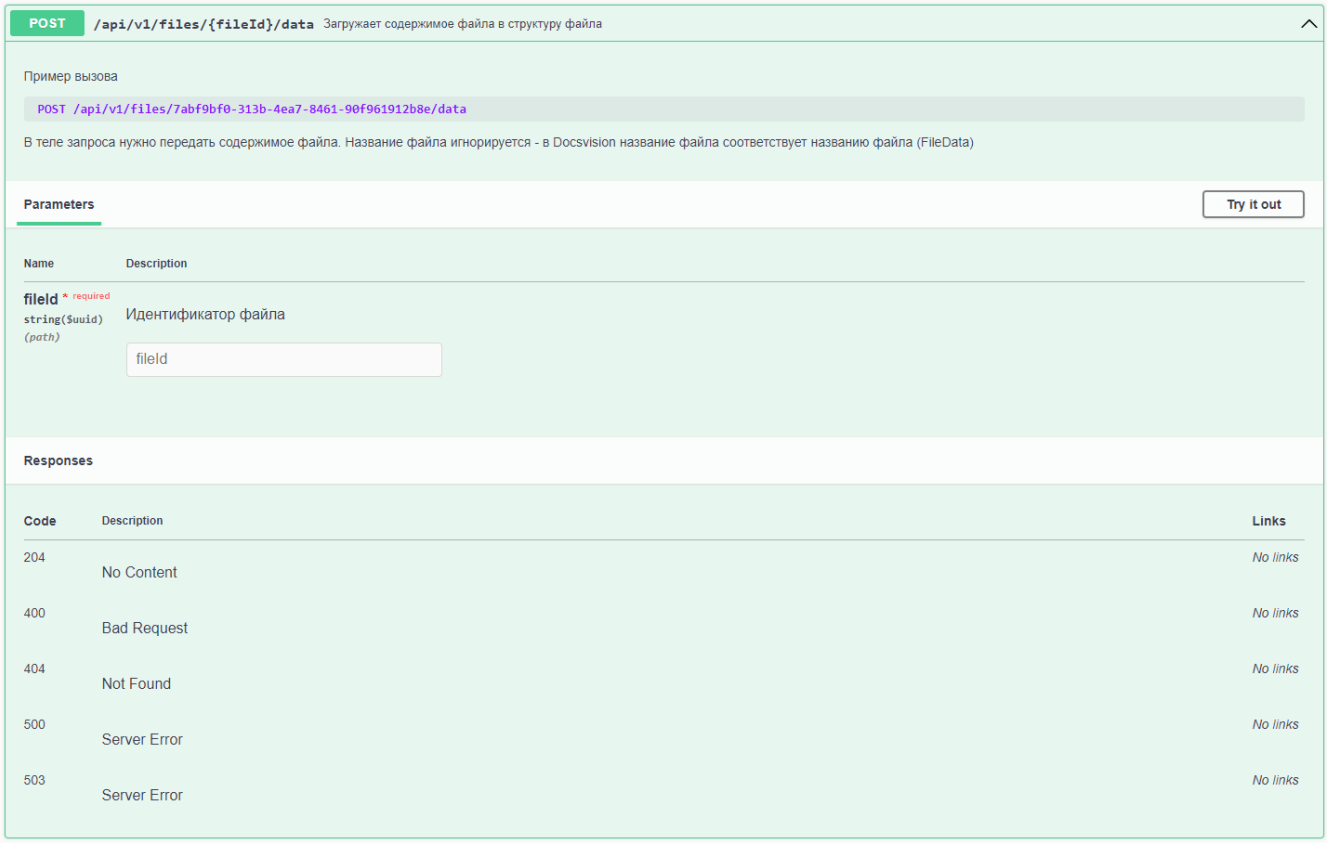


Рисунок 107. Описание метода "BarcodeSearchButton/{id}"

Описание метода включает описание модели возвращаемых данных и перечень параметров метода с примером из модели.

2. Если требуется выполнить метод, введите параметры метода в разделе *Parameters* и нажмите кнопку **Try it out!**. Ниже будет отображён результат выполнения запроса.

Обратите внимание: методы выполняются в текущей базе данных Web-клиента с правами текущего пользователя веб-браузера.

Как работать с описанием REST Web-API

В данной версии модуля с описанием REST API можно ознакомиться в справочнике веб-API, см. подробнее "Как работать с описанием веб-API".

Использование "общих справочников" в разработке

Функциональная возможность "общие справочники" включает режим использования данных из справочников только в режиме чтения и призвана сократить потребление памяти и повысить производительность любого приложения, реализованного с применением клиентского API Docsvision. Функциональность доступна для Web-клиента, начиная с версии 17, данная

статья описывает взаимодействие с режимом при разработке собственных решений.

Если при разработке кода решения для работы со справочниками используется сборка `DocsVision.Platform.ObjectModel.dll` (или `DocsVision.BackOffice.ObjectModel.dll`) решение будет автоматически работать и с общими справочниками. Стандартная логика Docsvision реализует автоматическое переключение режима работы со справочниками в зависимости от того читаются данные или изменяются.

Функциональность доступна только при использовании `DocsVision.Platform.ObjectManager.Rest.dll` в качестве API для работы с системой.

Если код решения взаимодействует со справочниками через `DocsVision.Platform.ObjectManager.dll` разработчику доступны два варианта:

- При работе на стороне Web-клиента для доступа к общим справочникам необходимо использовать методы класса `CardManager` — `GetDictionaryAsReadonly` и `GetDictionaryDataAsReadonly`. Эти методы передают доступные только для чтения `CardData` запрошенных справочников.
- Если код работает вне Web-клиента, необходимо до использования методов `GetDictionaryAsReadonly` и `GetDictionaryDataAsReadonly` создать сессию, через которую будет осуществляться чтение данных справочников и установить эту сессию как общую путём вызова `SessionManager.SetCommonSession`. Для создания сессии следует использовать УЗ, обладающую правами чтения всех данных справочников, например, состоящую в группе **DocsVision Administrators**.

Основные особенности использования функциональности общих справочников

1. При использовании функциональности общих справочников нельзя использовать операции, которые их модифицируют. Данные справочников в режиме чтения нельзя передавать как аргументы в методы, где они могут быть изменены, даже если изменения требуются крайне редко. Несоблюдение этого правила приведёт к ошибкам, источник которых сложно определить.
2. Вызов `RowData.Refresh` у строчек справочников в режиме чтения приведёт к ошибке для данных, полученных таким образом.
3. Для сохранения совместимости не рекомендуется возвращать справочники в режиме чтения в старых публичных методах, где возвращались справочники не в режиме чтения.

4. Рекомендуется использовать интерфейс `IReadOnly<T>` при работе со справочниками в режиме чтения через классы `ObjectManager`, см. ниже.
5. При работе со справочниками через API `ObjectModel` дополнительных действий не требуется.
6. Справочники в режиме чтения не повышают производительность в однопользовательских приложениях как, например, Конструктор Web-разметок или Панель управления Web-клиентом.

Использование интерфейса IReadOnly<T>

При вызове методов с параметрами, которые имеют типы справочников или строчек справочников из `ObjectManager` (`CardData`, `RowData`, `FolderCard`, `Folder` и т.д.) во время компиляции отсутствует проверка того, что в качестве аргументов были переданы справочники или строки в режиме чтения. В результате разработчик может случайно изменить вызываемый метод, добавив в него код модификации для передаваемых в качестве аргументов справочников, не зная, что в этот метод могут быть переданы справочники в режиме чтения. Если аргументы попадут в метод по цепочке вызовов, отследить такие операции, определить, в каком режиме они используются, будет сложно.

Рекомендуется использовать в объявлении аргументов интерфейс-обертку `IReadOnly<T>`, чтобы не забывать, что метод работает со справочниками в режиме чтения.

Интерфейс служит напоминанием о том, что обертываемый объект (справочник или строка) может быть в режиме чтения. Обычный справочник или строка справочника могут быть приведены к `IReadOnly<T>` с помощью метода расширения `AsReadOnly()`, что является безопасным, т.к. с обычным справочником можно делать то же, что со справочником в режиме чтения плюс модификация и обновление.

Интерфейс `IReadOnly<T>` содержит метод `Apply`, который используется, например, следующим образом:

```
IReadOnly<FolderCard> folderCard = ⋯
IReadOnly<Folder> folder = folderCard.Apply((FolderCard f)=>f.GetFolder(folderId));
```

Если типы не проставлены явно:

```
Var folder = folderCard.Apply( f => f.GetFolder(folderId));
```

Метод-расширение `Unwrap()` возвращает обертываемое значение. Его использование является небезопасным, но необходимым. Использовать `Unwrap` желательно как можно позже, в противном случае использование маркерного интерфейса теряет своё значение. Метод реализован как расширение, чтобы не вызывать ошибку при вызове, когда объект типа `IReadOnly<T> null`.

Расширение возможностей клиентской части Web-клиента

Клиентское расширение — исполняемая в веб-браузере пользователя программа, реализующая дополнительную функциональность клиентского уровня.

Клиентское расширение может добавлять:

- Локализованные ресурсы, которые можно использовать при формировании локализованного пользовательского интерфейса.
- Элементы управления (клиентскую часть).
- Обработчики событий, которые могут быть вызваны из разметки,
- Сервисы (публичные методы).

Создание и публикация клиентского расширения

Клиентское расширение разрабатывается на языке TypeScript с последующей компиляцией в JavaScript.

Чтобы создать клиентское расширение:

1. Скопируйте к себе в рабочую папку каталог `ProjectTemplates\TemplateWebExtension\` из репозитория WebClient-Samples.
2. Настройте проект:
3. Переименуйте папку `TemplateWebExtension` — укажите название расширения (например, "MyWebExtension").
4. В файле `MyWebExtension/copy-path.js` измените значения:
   - `SITE_ROOT` — укажите относительный путь к папке, в которую будут выгружены файлы, полученные при сборке.
   - `EXTENSION_NAME` — укажите название расширения (в данном примере — "MyWebExtension").
5. Откройте папку `MyWebExtension` в командной строке и выполните команду: `npm`

install.

6. Создайте в папке MyWebExtension/src файл ts/tsx с кодом расширения.

Примеры клиентских расширений приведены в следующих пунктах раздела, а также в репозитории "ДоксВижн".

7. Измените содержимое файла src/Index.ts. В файле должна выполняться регистрация расширения с учетом реализованных в нем функций:

```
extensionManager.registerExtension({
    name: "My web Extension",
    version: "1.0",

    getLocalizations: ... , ①
    controls: ... , ②
    globalEventHandlers: ... , ③
    globalServices: ... , ④
    layoutServices: ··· ⑤

})
```

① Расширение добавляет локализованные ресурсы.

② Расширение добавляет элементы управления.

③ Расширение добавляет обработчики событий.

④ Расширение добавляет публичные сервисы.

⑤ Расширение добавляет публичные сервисы, имеющие особое поведение в рамках каждой конкретной разметки.

Описание изменений, которые необходимо выполнить в файле Index.ts, приведено в описании примеров расширений, добавляющих обработчики событий, локализованные ресурсы и клиентские компоненты элементов управления.

1. Откройте папку MyWebExtension в командной строке и выполните команду: npm run build.

Данная команда выполнит компиляцию проекта. В папку, указанную в параметре SITE_ROOT, будут выгружены файлы: extension.js, extension.js.map, extension.css и extension.css.map. Файлы .map

используются для отладки кода в браузере.

Npm может отслеживать изменения в исходных файлах, и при этом самостоятельно выполнять сборку. Выполните команду, чтобы включить режим отслеживания: `npm run watch`.

2. Скопируйте папку `%SITE_ROOT%\Content/Modules/MyWebExtension` в `../../../lib/docsvision/webclient/Content/Modules`.
3. После завершения отладки клиентского расширения может быть собрана "релизная версия". Для этого, находясь в командной строке в папке MyWebExtension, выполните команду: `npm run build:prod`.

Действие данной команды аналогично действию команды `npm run build`, за исключением того, что к полученным файлам будет применена процедура минификации (минимизации). Полученные файлы также необходимо скопировать в папку `../../../lib/docsvision/webclient/Content/Modules`.

Информация о загруженных клиентских расширениях отображается в Web-клиенте в окне "О программе".

Разработка расширения с обработчиками событий

Данный тип расширений предназначен для создания функций обработки событий разметки. Такие функции включают, например, функцию открытия карточки, функцию нажатия кнопки, функцию выбора значения из списка.

Одно клиентское расширение может содержать несколько функций-обработчиков.

1. Создайте проект клиентского расширения **без публикации**.
2. Добавьте код функции-обработчика в существующий файл `src/EventHandlers.tsx` или в новый файл TypeScript (`.ts`, `.tsx`) в папке `src`.

Следующий код демонстрирует пример функции-обработчика.

```
import { Employee } from "@docsvision/webclient/BackOffice/Employee";
```

```
import { IEventArgs } from "@docsvision/webclient/System/IEventArgs";
import { MessageBox } from "@docsvision/webclient/Helpers/MessageBox/MessageBox";

export function someHandler(sender: Employee, e: IEventArgs) { ①

 MessageBox.ShowInfo(sender.params.value.displayName); ②

}
```

① Пример обработчика.

② Отображает сообщение с именем сотрудника.

В первых строчках кода выполняется импорт внешних сущностей, типы которых используются в коде, из JS-модулей Web-клиента. При использовании Visual Studio Code[7] редактор автоматически подключит необходимые модули. При использовании другого редактора — обратитесь к справочнику по JS API, в котором указаны названия JS-модулей.

Функция `someHandler` — реализация обработчика

- В определении функции должно быть указано ключевое слово `export`.
- Тип аргумента `sender` должен соответствовать типу элемента управления, событие которого обрабатывается. В примере, `sender` с типом `Employee` обрабатывается событие элемента управления `Сотрудник`.
- Тип аргумента `e` зависит от обрабатываемого события:
  - `IEventArgs` — если событие вызывается после выполнения действия, т.е. не может его прервать.
  - `ICancelableEventArgs` — если событие вызывается при выполнении действия, т.е. может его прервать. Работа с прерываемыми действиями рассмотрена в пункте Прерывание выполнения операции.

3. Если на шаге 2 был создан новый tsx файл, добавьте в файле `src/Index.ts` строку импорта его сущностей.

```
import * as Handlers from "./FileWithHandlers"; ①
import { extensionManager } from "@docsvision/webclient/System/ExtensionManager";
```

```
extensionManager.registerExtension({
 name: "My web Extension",
 version: "1.0",
 globalEventHandlers: [Handlers] ②
})
```

① Импорт сущностей из файла FileWithHandlers (расширение файла не указывается).

② В массив globalEventHandlers нужно передать ссылку на импорт сущностей из файла FileWithHandlers.

4. Соберите проект клиентского расширения командой `npm run build`.
5. Скопируйте полученный файл `%BuildDir%/extension.js` на сервер Web-клиента в папку `../../../lib/docsvision/webclient/Content/Modules/%Каталог Решения%`.

Обратитесь к разделу Создание и публикация клиентского расширения, чтобы получить больше информации, связанной со сборкой проекта.

6. Настройте обработчик в программе Конструктор Web-разметок.
7. Откройте в программе Конструктор Web-разметок разметку, событие которой обрабатывается.
8. Выберите элемент управления, для которого настраивается обработчик события.
9. Укажите название функции-обработчика, реализованной в клиентском расширении, в значении события.

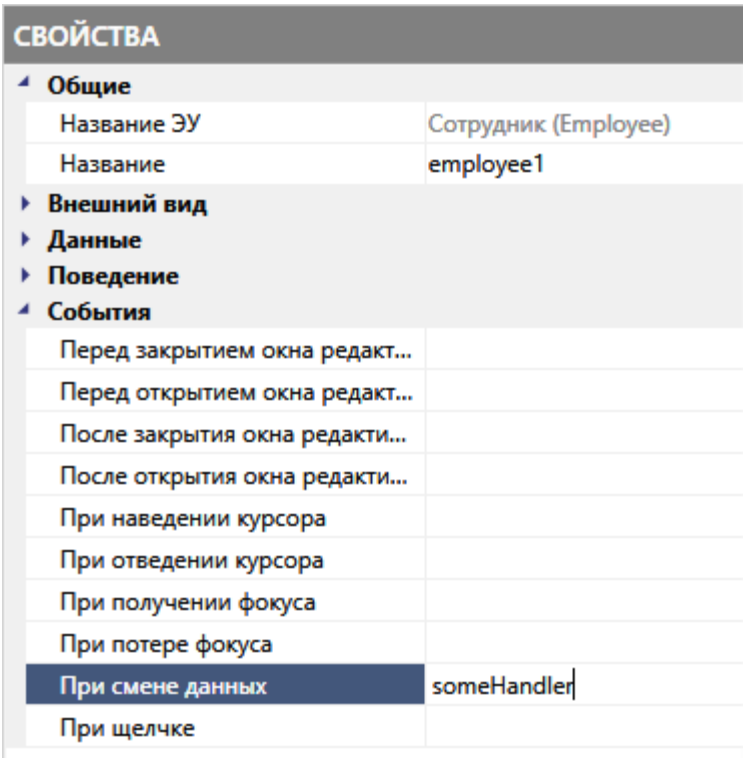


Рисунок 108. Настройка обработчика для события "При смене данных" в программе Конструктор Web-разметок

10. Сохраните настройки разметки.

> Инструкция по работе в программе Конструктор Web-разметок приведена в документации по работе с программной /dv6/webclient/6.1/layouts/info-install/[Конструктор Web-разметок].

11. Откройте локацию с настроенной разметкой (например, карточку) и выполните действие, вызывающее реализованный обработчик.
12. После завершения отладки скрипта, соберите и опубликуйте на сервере Web-клиента "релизную версию" клиентского расширения. "Релизная версия" собирается командой `npm run build:prod`.

Получить все обработчики событий

При необходимости, например для поиска и исправления искать ошибок, можно получить имена всех скриптов, назначенных на разметку. Для этого выполните в консоли браузера следующий скрипт:

```
(function getHandlers(sender) {
    let controlsVisitor = new WebClient.ControlsRecursiveVisitor();
    controlsVisitor.visitDeep(sender.layout, (control) => {
```

```
        const paramsKey = Object.keys(control.params);
        const eventsKey = paramsKey.filter(key => control.params[key] instanceof
WebClient.BasicEvent);
        const handlers = eventsKey.reduce((handlers, key) =>
            control.props[key] && !handlers.includes(control.props[key])
                ? `${control.props[key]}${handlers.length ? ', ' : ''}${handlers}`
                : handlers
            , "");
        if (handlers !== "") {
            console.info(`${control.props.name}: ${handlers}`);
        }
    });
})(layoutManager.cardLayout);
```

Получение доступа к разметке

Прежде чем работать с элементами управления разметки, необходимо в скрипте получить доступ к данной разметке. Это можно сделать несколькими способами.

Получение разметки, из которой вызван обработчик события

```
export function someHandler(sender: Employee, e: IEventArgs) {
    let currentLayout = sender.layout;
  }
```

Получение разметки по имени локации

```
let someLayout = layoutManager.getLayout("DV.MainMenu");
```

В метод getLayout нужно передать название локации с разметкой:

- *DV.Dashboard* — главная странице (страница "дашборд").
- *DV.MainMenu* — основное меню (с деревом папок).
- *DV.GridFilter* — панель фильтра папки.
- *DV.SearchParameters* — панель поисковых параметров.
- *DV.UserProfile* — диалоговое окно профиля пользователя.

Локация, разметка которой получается, должна быть на текущей странице Web-клиента.

Получение разметки открытой карточки:

Для быстрого доступа к разметке из консоли браузера можно использовать `layoutManger.cardLayout`. Однако использование в расширениях этого свойства не рекомендуется, следует использовать `sender.layout`.

```
let cardLayout = layoutManager.cardLayout;
```

В примерах кода используются функции объекта `layoutManager` (менеджер разметок). Для импорта объекта добавьте в скрипт следующий код:

```
import { layoutManager } from "@docsvision/webclient/System/LayoutManager";
```

Работа с элементами управления

Элементы управления являются частью разметки. Ссылка на коллекцию элементов управления разметки содержится в её поле `controls`:

```
let layoutControls = someLayout.controls; ①
```

① someLayout — разметка.

Доступ к элементу управления осуществляется по названию, которое ему присвоено в программе Конструктор Web-разметок:

```
let registrar = layoutManager.cardLayout.controls.get<Employee>("registrar"); ①
```

① Получаем элемент управления Сотрудник (Employee) с названием "registrar" из разметки карточки (layoutManager.cardLayout).

Доступ к свойствам и значению элемента управления осуществляется через его поле `params`:

```
let registrarName = registrar.params.value.displayName; ①

registrar.params.visibility = false; ②
```

① Получаем значение элемента управления registrar (params.value), из которого получаем отображаемое имя сотрудника (displayName).

② Изменяем значения свойства Видимость на "false" (элемент управления будет скрыт).

Изменение значения элемента управления осуществляется через свойство

params.value.

Способ изменения зависит от типа значения:

Для простых типов (string, bool, int и пр.) значение изменяется напрямую:

```
let onControl = layoutManager.cardLayout.controls.get<CheckBox>("onControl"); ①

onControl.params.value = true; ②
```

① Получаем из разметки карточки элемент управления Флаг (CheckBox) с названием "onControl".

② Изменяем значения элемента управления на true (флаг будет установлен).

Для элемента управления Дата/время значение должно быть предварительно сформировано:

```
let endDate = layoutManager.cardLayout.controls.get<DateTimePicker>("endDate"); ①

endDate.value = new Date("2019-03-01T12:00:00"); ②
```

① Получаем из разметки карточки элемент управления Дата/время (DateTimePicker) с названием "endDate".

② Изменяем значения элемента управления на дату 01.03.2019 и время 12:00:00.

Для элементов управления, работающих с моделями объектов, значение должно быть предварительно сформировано:

```
export async function someHandler(sender: CustomButton, e: IEventArgs) { ①

    let departmentControllerService = sender.layout.getService($DepartmentController); ②

    let departmentModel = await departmentControllerService.getStaffDepartment("67E331F0-
EF31-4E2C-A1F6-D527606EC8CA"); ③

    let department = sender.layout.controls.get<Department>("department"); ④

    department.params.value = departmentModel; ⑤
}
```

① Функция является асинхронной.

② Получаем сервис, загружающий модель объекта для элемента управления Подразделение и Подразделение контрагента.

③ Получаем модель объекта для подразделения с идентификатором "67E331F0-EF31-4E2C-A1F6-D527606EC8CA".

④ Получаем из разметки карточки элемент управления Подразделение (Department) с названием "department".

⑤ Сохраняем в значение элемента управления полученную ранее модель. В элементе управления будет выбрано подразделение с идентификатором "67E331F0-EF31-4E2C-A1F6-D527606EC8CA".

Список сервисов генерации моделей объектов приведён в пункте Сервисы генерации моделей объектов.

Управление блокировкой элемента управления от изменений осуществляется с помощью параметра disabled:

```
let control = layoutManager.cardLayout.controls.superControl; ①

control.params.disabled = true ②
```

① Получение элемента управления superControl.

② Блокировка элемента управления.
Параметр *disabled* работает совместно с параметром *editOperation*:

- если *disabled* равен `false` и *editOperation* разрешён — редактирование ЭУ разрешено;
- иначе — редактирование ЭУ запрещено.

Для отрисовки заблокированного элемента управления вызывается метод `BaseControlImpl.renderDisabledControl`, который может быть переопределён, если требуется собственная реализация отрисовки заблокированного элемента управления.

Получение номера в нумераторе

Следующий код демонстрирует пример получения номера в нумераторе перед изменением состояния карточки. Предполагается, что функция будет указана в качестве обработчика для события изменения состояния карточки.

```
export async function onStateChanging(sender: Layout, e: CancelableEventArgs
<ICardStateChangingEventArgs>) {
①
    let regNumber = sender.controls.get<Numerator>("regNumber");
```

```
    if (!regNumber.hasValue()) {
        e.wait(); ②
        try {
            await regNumber.generateNewNumber() ③
            console.log("Присвоен регистрационный номер");
            e.accept(); ④
        } catch (err) {
            MessageBox.ShowError("Ошибка при получении регистрационного номера: " + err);
            e.cancel();
        }
    }
}
```

① Получаем элемент Нумератор из разметки.

② Приостанавливаем изменения состояние карточки.

③ Используем метод generateNewNumber для получения нового номера. При этом будет выполнено сохранение карточки. Метод generateNewNumber нужно вызывать асинхронно.

④ Продолжаем обработку изменения состояния.

Получение информации из ЭУ "Записи справочника сотрудников"

Для выбора записей из *Справочника сотрудников* предназначен элемент управления "/dv6/webclient/6.1/layouts/ctrl/directories/staffDirectoryItems/[Записи справочника сотрудников]". В зависимости от настроек пользователь может выбирать сотрудников, организации, подразделения, группы, роли и должности, а также поисковые слова.

Каждый из этих типов имеет свою модель данных:

- Сотрудники: `EmployeeDataModel`.
- Организации и подразделения: `DepartmentModel`.
- Группы, роли, должности, поисковые слова: `StaffDataModel`

Поля, которые содержит каждая модель, можно посмотреть в справочнике по JavaScript API.

Чтобы корректно в коде обрабатывать полученные значения из ЭУ `Записи справочника сотрудников` нужно явно указывать тип. Если в области выбора элемента указан только один из типов, в коде достаточно прописать этот тип:

```
sender.layout.controls.get<StaffDirectoryItems<GenModels.EmployeeDataModel>>(название
ЭУ);
```

Пример для ЭУ с множественным выбором:

```
sender.layout.controls.get<StaffDirectoryItems<GenModels.EmployeeDataModel[]>>(название
ЭУ);
```

Если в области выбора указано несколько типов, необходимо написать логику, определяющую тип элемента. Пример такой функции приведён ниже:

```
function showName(sender: Layout) {
    const control = sender.layout.controls.get<StaffDirectoryItemsSingle>(Название ЭУ);

    if(control.value.dataType === GenModels.DirectoryDataType.Employee) {

        const value = control.value as GenModels.EmployeeDataModel;

        console.log(value.firstName + " " + value.lastName); ①

    } else if (control.value.dataType === GenModels.DirectoryDataType.Department

        || control.value.dataType === GenModels.DirectoryDataType.Organization) {

            const value = control.value as GenModels.DepartmentModel;

            console.log(value.fullName); ②
    } else {

        const value = control.value as GenModels.StaffDataModel;

        console.log(value.name); ③
    }
}
```

① Логика работы с `value` типа `EmployeeDataModel`.

② Логика работы с `value` типа `DepartmentModel`.

③ Логика работы с `value` типа `StaffDataModel`.

Если установлен множественный выбор, код аналогичный. При получении элемента управления необходимо указать тип значения `StaffDirectoryItemsMultiple` вместо `StaffDirectoryItemsSingle`, далее циклом пройти

все значения и в зависимости от типа совершить определенные действия с каждым из значений.

Проверка данных

Данный раздел содержит описание примера проверки действительности данных. Выполнить проверку, что заполнено определенное текстовое поле и количество символов не превышает 100.

Проект-расширение клиентской части Web-клиента. Содержит клиентский скрипт с функцией `validateTextBoxControl`, реализующей валидацию данных ЭУ.

Ссылка на пример на GitHub: ControlValidation.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше

Сборка и установка

1. Откройте консоль в папке `ControlValidationWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
2. Скопируйте каталог `SamplesOutput\Content\Modules\ControlValidationWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
3. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Импортируйте решение из файла `ControlValidationLayout.xml`.
3. Сделайте решение *ControlValidationLayout* активным для карточки типа Документ вида *ДокументУД/Исходящий*.
4. На разметке расположен ЭУ `Текст` и кнопки сохранения/отмены.
5. Перезапустите **dvwebclient**
6. Откройте документ ДокументУД\Исходящий и нажать кнопку редактировать. При этом должна открыться разметка с ЭУ `Текст` и кнопками сохранения.
7. Если в ЭУ `Текст` нет текста или длина текста превышает 100 символов, то при нажатии кнопки **Сохранить** возникает сообщение об ошибке и изменения не сохраняются.

Данный пример демонстрирует только доступ к элементу разметки. При реализации расширений рекомендуется использовать экспорт разметок в виде Решения.

Копирование значения ЭУ

Данный раздел содержит описание примера взаимодействия ЭУ: копирование значения. При завершении ввода значения в один ЭУ, это же значение копируется в другой. Проект-расширение клиентской части Web-клиента. Содержит клиентский скрипт с функцией `copyValueToControl`, реализующей копирование данных одного ЭУ в другой.

Ссылка на пример на GitHub: CopyValueToControl.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше

Сборка и установка

1. Откройте консоль в папке CopyValueToControlWebExtension и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`
2. Скопируйте каталог SamplesOutput\Content\Modules\CopyValueToControlWebExtension в каталог "Путь к сайту Web-клиента\Content\Modules"
3. Перезапустите **dvwebclient**

Проверка примера

1. Запустите конструктор разметок.
2. Импортируйте решение из файла `CopyValueToControlLayout.xml`.

   Либо создайте собственное решение:

   a. Создайте для расширения разметку.

   b. Добавьте в разметку `Блок`, в него два ЭУ `Текст`.

   c. Задайте для первого ЭУ `Текст` режим редактирования по месту, на событие **После смены данных** добавьте вызов функции `copyValueToControl:CopyValueToControlExt`.
3. Сделайте решение `CopyValueToControlLayout` или собственное активным для карточки типа *Документ* вида *ДокументУД/Исходящий*.

4. На разметке расположено два ЭУ `Текст`.
5. Перезапустите **dvwebclient**.
6. Откройте документ *ДокументУД\Исходящий* и нажать кнопку **Редактировать**. При этом должна открыться разметка с двумя ЭУ `Текст`.
7. Если в первый из них добавить текст, текст появится во втором.

Данный пример демонстрирует только доступ к элементу разметки. При реализации расширений рекомендуется использовать экспорт разметок в виде Решения.

Получение информации о карточке, разметке

JS API предоставляет возможность получать в скрипте информацию об открытой карточке и разметке, с использованием которой она открыта.

Следующий код демонстрирует пример получения информации о карточке и разметке:

```
export function showInformation(sender: Layout, e: IEventArgs) {
    let cardInfo = sender.getService($CardInfo); ①
    let cardId = sender.getService($CardId);
    console.log("Идентификатор карточки: " + cardId);
    console.log("Тип карточки: " + cardInfo.typeId);
    console.log("Вид карточки: " + cardInfo.kindId);

    let layoutInfo = sender.getService($LayoutInfo); ②
    console.log("Название разметки: " + layoutInfo.name);
    console.log("Тип разметки: " + layoutInfo.type);
    console.log("Тип пользовательского устройства: " + layoutInfo.deviceType);
}
```

① Информация о карточке из сервиса `$CardInfo`.

② Информация о разметке из сервиса `$LayoutInfo`.

Получение данных карточки на стороне клиента

Web-клиент предоставляет REST API для получения любых данных, см. подробнее "Как работать с описанием веб-API".

Отображение сообщений, запросов, предупреждений

Сервис отображения окна сообщений

Показать на экране простое информационное сообщение можно с помощью сервиса `$MessageWindow`.

Основные методы этого сервиса: `showInfo`, `showWarning`, `showError`. Они различаются только цветом оформления и отображают диалог следующего вида:

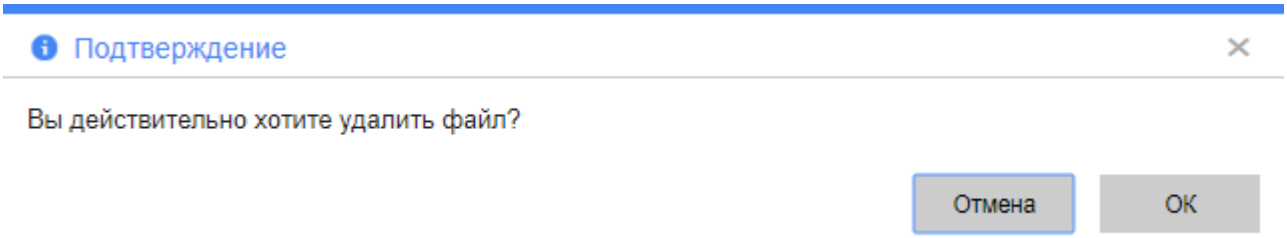


Рисунок 109. Диалоговое окно подтверждения

В качестве параметра для этих методов можно передать как простую строку, так и React-разметку для отображения произвольного содержимого.

```
export async function onButtonClick(sender: LayoutControl, args: IEventArgs) {
     const messageWindow = sender.getService($MessageWindow);
     await messageWindow.showConfirmation("Вы действительно хотите удалить файл?");
     deleteFile();
}
```

Работа с боковой панелью Web-клиента

Боковая панель Web-клиента — отображаемая по запросу правая панель, на которую могут быть добавлены требуемые элементы управления. Данный компонент реализован в классе `ModalSidebar` JS-модуля `@docsvision/webclient/Helpers/ModalSidebar/ModalSidebar`.

Следующий код демонстрирует пример использования боковой панели для отображения текущего времени.

```
export function showSidebar(sender: CustomButton, e: IEventArgs) { ①
    let host: ModalHost = new ModalHost("time-dialog", () => {
        return (
            <ModalBackdrop visible={true} onClick={() => host.unmount()} > ②
                {/* Элемент Боковая панель */}
                <ModalSidebar isOpen={true}>

                  {/* Кнопка */} ③
                  <ModalSidebarCloseButton onClick={() => host.unmount()} />
```

```
                    {/* Шапка */} ④
                    <ModalSidebarHeader>Заголовок панели</ModalSidebarHeader>

                    {/* Содержимое */} ⑤
                    <div>Текущее время: {(new Date()).toTimeString()}</div>
                </ModalSidebar>
            </ModalBackdrop>
        );
    });

    host.mount(); ⑥

    setInterval(() => host.forceUpdate(), 1000); ⑦
}
```

① Создаем контейнер для размещения модальных окон, которым является боковая панель.

Первый параметр конструктора `ModalHost` — класс, который будет задан элементу-контейнеру в DOM. Второй параметр — функция, выполняющая отрисовку модального окна.

② Элемент `ModalBackdrop` создаёт темный фон при отображении боковой панели.

При клике по элементу `ModalBackdrop` боковая панель будет закрыта.

③ Кнопка для закрытия боковой панели.

④ Заголовок боковой панели.

⑤ Содержимое боковой панели. В данном случае на боковой панели отображается текущее время.

⑥ Вызов метода, добавляющего проинициализированный контейнер с боковой панелью в DOM.

⑦ Периодически вызываемый метод перерисовки контейнера с боковой панелью. Метод обновляет отображаемое значение текущего времени.

Обработка события изменения состояния карточки

При изменении состояния карточки из пользовательского интерфейса Web-клиента вызывается событие **Перед изменением состояния карточки** разметки карточки (корневой элемент разметки), которое может быть обработано в клиентском скрипте.

Ссылка на пример на GitHub: ChangeStateByScript.

Данное событие является прерываемым — может быть использовано для отмены изменения состояния карточки.

Ниже приведён код обработчика события изменения состояния карточки (Документ, Задание):

```
async function changeState(sender: Layout, e: CancelableEventArgs
<ICardStateChangingEventArgs>) {
    e.wait();

    if (await myCheckPossibility()){ ①
        e.accept(); ②
    } else {
        sender.getService($MessageWindow).showWarning("Операция " + e.data.operationId  +
" запрещена"); ③
        e.cancel(); ④
    }
}
```

① Функция `myCheckPossibility` — пользовательский код, содержащий собственную логику.

② Подтверждаем выполнение операции. Обязательно, если вызвали `e.wait`.

③ Показываем сообщение о запрете операции с указанием идентификатора операции.

④ Для отмены операции, при необходимости, вызываем `e.cancel`.

Для карточки *Задание* после изменения состояния карточки будет вызвано событие **После изменения состояния карточки** элемента управления Кнопки операций задания.

Изменение внешнего вида и заголовка

Данный раздел содержит описание примера изменения внешнего вида и заголовка Web-клиента.

Ссылка на пример на GitHub: ChangeLogoAndBackground.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше

Сборка и установка

1. Откройте консоль в папке `ChangeLogoAndBackgroundWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
2. Скопируйте каталог `SamplesOutput\Content\Modules\ChangeLogoAndBackgroundWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
3. Перезапустите **dvwebclient**.

Проверка примера

1. Откройте стартовую страницу или любую папку.

Данный пример демонстрирует только изменение внешнего вида и заголовка системы с помощью скрипта.

Сохранение изменений карточки

Код сохраняет все изменения карточки:

```
export async function saveCard(sender: CustomButton, e: IEventArgs) {
    await MessageBox.ShowConfirmation("Сохранить карточку?");

    layoutManager.cardLayout.saveCard(); ①
}
```

① Получаем разметку карточки (cardLayout) и вызываем её метод saveCard для сохранения.

Чтобы сохранить изменения определённого элемента управления, используйте его метод save:

```
export async function saveDescription(sender: CustomButton, e: IEventArgs) {
    let cardDescription = sender.layout.controls.get<TextArea>("Description"); ①

    await cardDescription.save(); ②
}
```

① Получаем элемент управления Description, данные которого сохраняются.

② Вызываем метод save для сохранения изменений.

Если за сохранением следует другая операция, изменяющая карточку, сохранение карточки может привести к ошибке `Содержимое карточки было изменено другим пользователем. Пожалуйста, обновите карточку и выполните операцию снова`.

> Например, данная ошибка возникнет при сохранении карточки, если сохранение вызывается в обработчике изменения состояния карточки.

Чтобы обойти данную ошибку, приостановите обработку основного события до сохранения карточки, а сохранение сделайте асинхронным. Следующий код демонстрируется данный способ:

```
export async function onChangeState(sender: Layout, e: CancelableEventArgs
<ICardStateChangingEventArgs>) {
        let description = sender.controls.get<TextArea>("Description");

        description.value = description.value + "\nЗарегистрировано"; ①

        e.wait(); ②
        await sender.layout.saveCard(); ③
        e.accept(); ④
}
```

① Изменяем значение поля карточки.

② Приостанавливаем обработку основного события.

③ Асинхронное сохранение.

④ Продолжаем обработку.

Обработка события сохранения карточки

При сохранении карточки последовательно вызываются события **Подготовка к сохранению карточки**, **Перед сохранением карточки** и **При сохранении карточки** корневого элемента разметки карточки. Данные события могут быть обработаны в пользовательской функции.

Событие **Перед сохранением карточки** является прерываемым, т. е. предоставляет возможность отменять сохранение изменений карточки в зависимости от условий:

```
function onSaveCard(sender: Layout, e: CancelableEventArgs<ICardSavingEventArgs>) {
    if (CheckPossibility()){ ①
        e.accept(); ②
        return;
    }
```

```
    MessageBox.ShowWarning("Сохранение отклонено");
    e.cancel(); ③
}
```

① Функция CheckPossibility проверяет выполнение неких требований

② Подтверждение сохранения. Необязательно.

③ Отмена сохранения, если условия не выполняются.

Если при сохранении карточки нужно изменить значение элемента управления разметки (и данное изменение должно быть включено в сохранение), изменение следует применять до сохранения в обработчике события **Подготовка к сохранению карточки**.

Обработка события элемента Кнопки сохранения

Другой способ обработать событие попытки сохранения карточки — подписаться на событие **Перед щелчком** элемента управления Кнопки сохранения. В аргументе события будет передан тип нажатой кнопки, который может использоваться при реализации логики обработчика.

```
export function onClicking(sender: SavingButtons, e: ICancelableEventArgs
<ISavingButtonClickEventArgs>) {
    if (e.data.button == SavingButton.Save){ ①
    }else{ ②
    }
}
```

① Обработка события, когда нажата кнопка Сохранить/ОК.

② Обработка события, когда нажата кнопка Отмена.

Прерывание выполнения операции

Функция, настроенная для обработки события, возникающего при выполнении операции, позволяет прервать её выполнение, если аргумент события имеет интерфейс ICancelableEventArgs. Данный интерфейс определяет два метода: cancel — прерывает выполнение операции; accept — продолжает выполнение операции.

Следующий код демонстрирует пример обработчика события сохранения карточки, прерывающего сохранение, если в названии карточки (cardName) содержится слово "договор".

```
export function onCardSaving(sender: Layout, e: ICancelableEventArgs<ISaveControlData>) {
①

    let cardName = sender.controls.get<TextBox>("cardName"); ②

    if (cardName.params.value.includes("договор")) { ③
        MessageBox.ShowError("Для оформления договоров используйте вид документа
'Договор'");
        e.cancel(); ④
    } else {
        e.accept(); ⑤
    }
}
```

① Метод `onCardSaving` настроен в качестве обработчика события, вызываемого перед сохранением карточки

Список параметров метода соответствует аргументам события `cardSaving` класса `LayoutParams`

② Получение элемента управления с названием карточки.

③ Если в названии используется слово "договор", сохранении карточки будет прервано.

④ Метод `cancel` прерывает операцию сохранения.

⑤ Метод `accept` продолжает операцию сохранения.

Обычно вызов метода `accept` не является обязательным. Однако `accept` или `cancel` обязателен в случае, если вызван `wait`.

Другой пример демонстрирует использование метода `wait`, который позволяет отложить решение до момента выполнения асинхронной операции (например, запроса к серверу или показа диалога подтверждения).

```
export async function onCardSaving(sender: Layout, e: ICancelableEventArgs
<ISaveControlData>) {

    e.wait(); ①
    try {
        await MessageBox.ShowConfirmation("Вы действительно хотите сохранить карточку?");
②

        e.accept(); ③
    } catch {
```

```
        e.cancel(); ④
    }
}
```

① Отменяем автоматический вызов `e.accept()`.

② Показ диалога с ожиданием решения пользователя.

③ Разрешаем выполнение операции.

④ Отменяем событие.

Чтобы определить, является ли событие прерываемым, обратитесь к его описанию в справочнике по JS API. События, возникающие при начале выполнения операции, реализуют типы `CancelableApiEvent` и `BasicApiEvent`.

Получение сервиса клиентского уровня

Сервисы представляют основной API взаимодействия с системой в части функциональности, не связанной с конкретными элементами управления. Они позволяют получить информацию о текущем контексте выполнения, выполнять запросы к серверу, показывать модальные окна, взаимодействовать с разметкой и использовать множество других функций системы.

Доступ к сервисам предоставляет функция `getService` разметки (тип `Layout`). Функция `getService` принимает тип сервиса, который нужно предоставить. Список сервисов клиентского уровня приведён в справочнике по JS API в разделе *Services*.

В Web-клиент версии 17 и выше функция `getService` есть у всех ЭУ. Рекомендуется получать сервис в ближайшем элементе управления.

Следующий код демонстрирует пример получения списка всех операций редактирования, зарегистрированных для карточки, с помощью сервиса `$EditOperationStore`.

```
export function onMyButtonClick(sender: CustomButton, e: IEventArgs) { ①
    let editOperationStore = sender.getService($EditOperationStore); ②

    let editOperations = editOperationStore.getAll(); ③
}
```

① Метод onMyButtonClick обрабатывает событие нажатия кнопки.

② Получаем разметку layout из элемента, вызвавшего событие. С помощью метода getService получаем сервис $EditOperationStore.

③ Вызываем метод getAll сервиса для получения списка всех операций редактирования.

Переопределение сервиса отрисовки элемента "Ход согласования"

Элемент управления **Ход согласования** /dv6/webclient/6.1/layouts/ctrl/approval/agreementHistory/[представляет собой] таблицу, в которой отображаются события заданий согласования (выполнение, делегирование и др.).

Программист может переопределить функцию отрисовки всей таблицы и отдельных элементов: заголовков и ячеек определённых столбцов. Для этого нужно передать в поле `stageRenderer` класса элемента управления `Ход согласования` (тип `AgreementHistory`) сервис с интерфейсом `IAgreementHistoryStageRenderer`:

```
interface IAgreementHistoryStageRenderer {
    renderStageTable?(props: IApprovalCycleInfoProps): React.ReactNode;
    renderStageHeaderCells?(): React.ReactNode;
    renderApproverCell?(stageItemProps: IApprovalStageItemRowProps): React.ReactNode;
    renderDecisionCell?(stageItemProps: IApprovalStageItemRowProps): React.ReactNode;
    renderDateCell?(stageItemProps: IApprovalStageItemRowProps): React.ReactNode;
    renderCommentCell?(stageItemProps: IApprovalStageItemRowProps): React.ReactNode;
    renderCorrectionsCell?(stageItemProps: IApprovalStageItemRowProps): React.ReactNode;
}
```

Интерфейс `IAgreementHistoryStageRenderer` определяет функции, обеспечивающие отрисовку определённых элементов таблицы `Ход согласования`:

- `renderStageTable` — отрисовка всей таблицы.
- `renderStageHeaderCells` — отрисовка заголовков таблицы.
- `renderApproverCell` — отрисовка ячеек столбца *Согласующий*.
- `renderDecisionCell` — отрисовка ячеек столбца *Решение*.
- `renderDateCell` — отрисовка ячеек столбца *Дата*.
- `renderCommentCell` — отрисовка ячеек столбца *Комментарий*.
- `renderCorrectionsCell` — отрисовка ячеек столбца *Правки*.

При реализации объекта `IAgreementHistoryStageRenderer` могут быть

переопределены все или только требуемые функции интерфейса.

Ниже приведён пример реализации компонента, который переопределяет функцию отрисовки ячеек столбца `Комментарий`: ставит `-`, если комментария не было, и текст комментария — если был. Компонент реализуется в виде стандартного клиентского расширения.

Компонент включает две составляющие:

- Сервис, который будет передан в `stageRenderer`.
- Метод, который будет вызываться при открытии вкладки `Ход согласования` карточки *ДокументУД/Исходящий*, и передавать сервис в `stageRenderer`.

Ниже приведено содержимое файла `EventHandlers.tsx` (стандартное название основного файла клиентского расширения, создаваемого из шаблона).

```
import { IAgreementHistoryStageRenderer } from
"@docsvision/webclient/Approval/$AgreementHistoryStageRenderer";
import { serviceName } from "@docsvision/webclient/System/ServiceUtils";
import { IApprovalStageItemRowProps } from
"@docsvision/webclient/Approval/IApprovalStageItemRowProps";
import { IEventArgs } from "@{Docsvision/webclient/System/IEventArgs";
import { AgreementHistory } from "@docsvision/webclient/Approval/AgreementHistory";
import { Tab } from "@docsvision/webclient/Platform/Tab";
import { $RequestManager } from "@docsvision/webclient/System/$RequestManager";

import * as React from 'react' ①

export function loadRender(sender: Tab, e: IEventArgs) { ②

    if(sender.params.activeTabPage.key == "agreementHistoryTabPage"){ ③

        let agreementHistory = sender.layout.controls.get<AgreementHistory>(
"agreementHistory"); ④

        agreementHistory.params.stageRenderer = sender.layout.getService
($AgreementHistoryRenderService); ⑤
    }
}

export class AgreementHistoryRenderService implements IAgreementHistoryStageRenderer { ⑥
    constructor(private services: $RequestManager) {
    }

    renderCommentCell(stageItemProps: IApprovalStageItemRowProps): React.ReactNode{ ⑦
```

```
            if (!stageItemProps.stageItem.hasComment){
                return (
                <td key="comment"> - </td> ⑧
                )
            }

            return (
            <td key="comment">{stageItemProps.stageItem.comment}</td> ⑨
            )
    }
}

export type $AgreementHistoryRenderService = { renderService:
AgreementHistoryRenderService };
export const $AgreementHistoryRenderService = serviceName((s:
$AgreementHistoryRenderService) => s.renderService); ⑩
```

① Импорт зависимостей.

② Определяется функция, которая будет вызываться при открытии страниц элемента "Вкладки".

③ Следующий код применяется только для страницы с названием `agreementHistoryTabPage`.

④ Получение элемента управления `Ход согласования`.

⑤ Передача сервиса отрисовки.

⑥ Реализация сервиса отрисовки, наследуемого от `IAgreementHistoryStageRenderer`.

⑦ Переопределяется метод отрисовки ячеек столбца "Комментарий".

⑧ Если комментария нет, пишется `-`.

⑨ Если комментарий есть, пишется его текст.

⑩ Экспорт сервиса AgreementHistoryRenderService.

Инструкция по разработке и регистрации клиентских сервисов приведена в пункте Разработка расширения с клиентским сервисом.

В файле `Index.ts` нужно добавить регистрацию сервиса `$AgreementHistoryRenderService`.

```
import * as EventHandlers from "./EventHandlers";
import { $AgreementHistoryRenderService, AgreementHistoryRenderService } from
"./EventHandlers";
import { Service } from "@docsvision/webclient/System/Service";
```

```
import { $RequestManager } from "@docsvision/webclient/System/$RequestManager";
import { extensionManager } from "@docsvision/webclient/System/ExtensionManager";

extensionManager.registerExtension({
    name: "Template web extension",
    version: "6.1",
    globalEventHandlers: [ EventHandlers ],
    layoutServices: [
        Service.fromFactory($AgreementHistoryRenderService, (services: $RequestManager)
=> new AgreementHistoryRenderService(services))
    ]
})
```

После загрузки клиентского расширения на сервер Web-клиента нужно указать обработчик для переключения страниц элемента `Вкладки` карточки *Документ*:

1. Откройте программу Конструктор Web-разметок и перейдите к разметке просмотра карточки *ДокументУД/Исходящий*.
2. Перейдите к свойствам элемента `contentTab`.
3. В событии **После переключения активной вкладки** укажите обработчик loadRender.
4. Сохраните изменения и перезапустите **dvwebclient** (для загрузки скриптов).

Чтобы проверить пример, откройте исходящий документ с согласованием и перейдите на страницу `Ход согласования`. В столбце *Комментарий* у событий согласования с комментарием будет указан комментарий, у событий без комментария — знак `-`.

Чтобы получить другие примеры, включая пример получения списка файлов для отображения столбца *Правки*, обратитесь к исходному коду элемента управления `Ход согласования` в каталоге установки Web-клиента: `../../../lib/docsvision/webclient/Content/App/Approval/Controls/AgreementHistory`.

Разработка расширения с локализованными ресурсами

При необходимости использовать локализованные ресурсы в расширении можно их добавить при регистрации расширения в `Index.ts`.

Чтобы создать расширение, добавляющее новые локализованные ресурсы, выполните приведенную далее инструкцию.

1. Создайте проект клиентского расширения.
2. Создайте в папке `src` файл TypeScript (`.ts`, `.tsx`) со следующим содержимым:

```
import { ILocalizationsMap } from "@docsvision/webclient/System/ILocalizationsMap"; ①

export function getLocalizedResources(): ILocalizationsMap {
    let cultureMap: ILocalizationsMap = {};
    cultureMap["en"] = {
        "Resource_One": "Sample",
        "Resource_Two": "Sample 2"
    };
       cultureMap["ru"] = {
        "Resource_One": "Пример",
        "Resource_Two": "Пример 2"
    };
    return cultureMap;
}
```

① Импортируем необходимые сущности из модуля.

В данном коде выполняется добавление двух ресурсов `Resource_One` и `Resource_Two` для языков русский (`ru`) и английский (`en`).

Функция `getLocalizedResources` (можно использовать другое название) возвращает массив добавляемых локализованных ресурсов, в котором каждый элемент соответствует определённой локали. Например, следующий участок кода отвечает за добавление ресурсов для локали `ru` (русский):

```
cultureMap["ru"] = {
    "Resource_One": "Пример",
    "Resource_Two": "Пример 2"
};
```

3. Измените содержимое файла `Index.ts`.

```
import { getLocalizedResources } from "./LocalizedResource"; ①
import { extensionManager } from "@docsvision/webclient/System/ExtensionManager";
import { ILocalizationsMap } from "@docsvision/webclient/System/ILocalizationsMap";

extensionManager.registerExtension({ ②
```

```
    name: "Client extension with localization",
    version: "1.0",
    getLocalizations(): ILocalizationsMap {
        return getLocalizedResources();
        }
})
```

① Добавьте строку импорта функции getLocalizedResources из добавленного в 2 файла (в примере — LocalizedResource).

② Добавьте в registerExtension функцию getLocalizations, которая должна вызывать импортированную в 1 функцию.

Пример использования ресурсов

```
export function onButtonClick(sender: LayoutControl, args: IEventArgs) {
     const messageWindow - sender.getService($MessageWindow);
     messageWindow.showInfo(resoures.Resource_One);
}
```

Разработка расширения с клиентским сервисом

Данный раздел посвящён созданию новых сервисов клиентского уровня. Сервисы предоставляют публичные методы, которые могут быть вызваны (с помощью функции получения сервиса) из клиентских расширений (скриптов, компонентов элементов управления). Программист может реализовать сервисы различного назначения.

Рекомендуется использовать сервис в одном клиентском расширении. Использование разработанного сервиса в другом расширении сопряжено со сложностями настройки сборки.

Создание расширения

Чтобы создать расширение, добавляющее новые сервисы, выполните приведенную далее инструкцию.

1. Создайте проект клиентского расширения.
2. Создайте в папке `src` файл TypeScript (`.ts`, `.tsx`) со следующим содержимым:

```
import { $RequestManager } from "@docsvision/webclient/System/$RequestManager";
import { urlStore } from "@docsvision/webclient/System/UrlStore";
import { serviceName } from "@docsvision/webclient/System/ServiceUtils";
```

```
export class CardService {
    constructor(private services: $RequestManager) {
    }
    exists(cardId: string): Promise<boolean> { ①
        let url = urlStore.urlResolver.resolveApiUrl("Exists", "CardService"); ②
        url = url + "?cardId=" + cardId;
        return this.services.requestManager.get<boolean>(url);
    }
}

export type $CardService = { cardService: CardService };
export const $CardService = serviceName((s: $CardService) => s.cardService); ③
```

① Функция `exists` проверяет существование карточки.

② Проверка выполняется с помощью серверной функции `Exists` контроллера `CardService`.

③ Объявление сервиса `$CardService`.

> В данном коде реализован сервис `CardService`, содержащий единственную функцию `exists`, возвращающую признак существования карточки в системе Docsvision.
>
> При реализации собственного сервиса, используйте его объявление, по аналогии с приведенным примером, заменив `CardService` на название собственного сервиса.

3. Измените содержимое файла `Index.ts`.

```
import { $CardService, CardService } from "./WebService"; ①
import { extensionManager } from "@docsvision/webclient/System/ExtensionManager";
import { Service } from "@docsvision/webclient/System/Service";
import { $RequestManager } from "@docsvision/webclient/System/$RequestManager";

extensionManager.registerExtension({
    name: "Client extension with service",
    version: "1.0",
    layoutServices: [ ②
        Service.fromFactory($CardService, (services: $RequestManager) => new
CardService(services)) ③
        ]
})
```

① Добавьте строку импорта тип/константы $CardService и класса CardService из добавленного в 2 файла (в примере — WebService).

② Добавьте в registerExtension поле layoutServices, в значении которого должна быть специальная конструкция.

③ Service.fromFactory(**<Тип сервиса>**, (services: $RequestManager) ⇒ new **<Класс реализации сервиса>**(services)).

4. Соберите проект клиентского расширения командой `npm run build`.
5. Скопируйте полученный файл `%BuildDir%/extension.js` на сервер Web-клиента в папку `../../../lib/docsvision/webclient/Content\/%Каталог Решения%`.

Обратитесь к разделу Создание и публикация клиентского расширения, чтобы получить больше информации, связанной со сборкой проекта.

Получить разработанный сервис в клиентском расширении можно с помощью метода `getService`. Дополнительная информация по использованию функции getService приведена в пункте Получение сервиса клиентского уровня.

```
let cardService = layout.getService($CardService); ①
```

① Получение сервиса $CardService.

Проверка работы расширения

Чтобы проверить работу сервиса CardService:

1. Добавьте в данное расширение обработчик событий, вызывающий функцию `$CardService.exists`.

В следующем коде в метод exists сервиса `$CardService` передается идентификатор карточки, введённый в строку `cardId`. Результат проверки отображается на экране. Реализованный метод `onMyButtonClick` должен быть назначен обработчиком события нажатия **Кнопки**.

```
export async function onMyButtonClick(sender: CustomButton, e: IEventArgs) {

let cardId = sender.layout.controls.get<TextBox>("cardId").value;
```

```
let result = await sender.layout.getService($CardService).exists(cardId);

if (result)
MessageBox.ShowInfo('Карточка существует');
else
MessageBox.ShowInfo('Карточка не существует');
}
```

2. Разработайте серверное расширение, предоставляющее функцию `Exists`. Инструкция по разработке такого расширения приведена в разделе Разработка расширения с методами WebApi.
3. Опубликуйте клиентское и серверное расширения на сервере Web-клиента.
4. Перезапустите сервер Web-клиента.
5. Настройте разметку карточки, указав для любой **Кнопки** обработчик `onMyButtonClick`. Добавьте в разметку *Строку* с названием *cardId*.
6. Откройте карточку с настроенной разметкой. Введите идентификатор существующей карточки в *Строку* и нажмите **Кнопку**. На экране отобразится сообщение: `Карточка существует`.

Настройка страницы авторизации

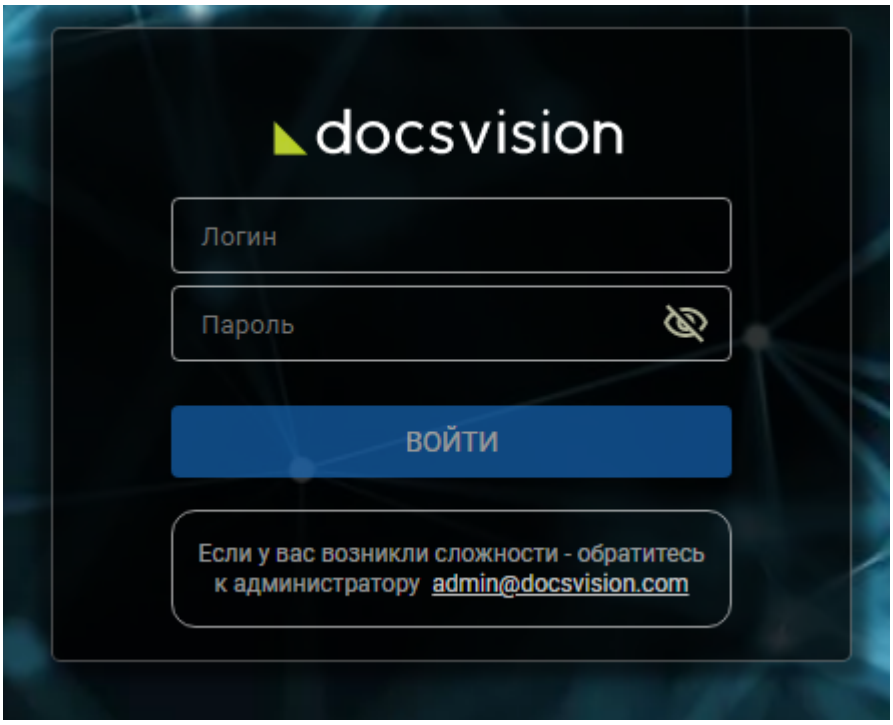


Рисунок 110. Страница входа с указанным адресом электронной почты администратора

Данный раздел описывает расширение, которое содержит пример внедрения

своего кода на страницу входа Web-клиента, а именно:

1. Выполнение кода при определённых событиях — тип `LoginEvents` в модуле `@docsvision/webclient/Account`.
2. Подмена текста на странице — объект `defaultLoginResources` в модуле `@docsvision/webclient/Account`.
3. Замена компонентов страницы — объект `LoginView` в модуле `@docsvision/webclient/Account/LoginContainer`.
4. Добавление стилей на страницу.

Данный раздел документации описывает настройку внешнего вида страницы авторизации. Инструкцию по добавлению расширений авторизации см. в /dv6/engineer/dv6/authorization-extensions/[документации по настройке Docsvision].

Чтобы изменить вид страницы входа, разработайте собственные скрипты По инструкции ниже и поместите их в папку `../../../lib/docsvision/webclient/Content/Account/Scripts`.

Ссылка на пример на GitHub: LoginPage.

Дополнительная информация

В примере используется UMD система модулей.

в результате сборки получается файл с расширением `.js`, который нужно поместить в папку `DocsVision/WebClient/Content/Account/Scripts`, код будет выполнен перед загрузкой приложения и выполнением `ReactDOM.render()`.

Чтобы добавить стили, нужно создать файл или файлы с расширением `.css` и положить в папку `DocsVision/WebClient/Content/Account/Styles`, стили загружаются после основных стилей страницы.

Входной точкой для внедрения кода на страницу является объект `loginExtensionManager` доступный глобально через объект `windows`.

Код расширения будет выполнен перед загрузкой приложения и выполнением `ReactDOM.render()`, но после загрузки библиотек `React` и `ReactDOM`.

Методы `loginExtensionManager`, а также возможные ресурсы, компоненты и события для замены обозначены в виде типов и интерфейсов в модуле

`@docsvision/webclient/Account`.

Процесс настройки

Чтобы настроить вид страницы авторизации, необходимо:

1. Скопируйте проект на компьютер для разработки.
2. Откройте консоль в папке `ClientScripts/TypeScript/LoginPage` и выполните команду `npm install`, затем выполнить `npm run build:prod`.
3. Скопируйте каталог `SamplesOutput/Content/Account/Scripts` в каталог `../../../lib/docsvision/webclient/Content/Account/Scripts`.
4. Перезапустите **dvwebclient**.

Пример кода расширения указан ниже и в файле `Index.ts` в репозитории примеров Web-клиента.

```
import React from 'react';
import { windowWithLoginExtensionManager, ILoginPlugin, ILoginExtensionManager,
LoginEvents } from '@docsvision/webclient/Account';
import { LoginView } from '@docsvision/webclient/Account/LoginContainer';
import { LoginLogic } from '@docsvision/webclient/Account/LoginLogic';
import "index.scss"

const loginExtensionManager: ILoginExtensionManager = (window as
windowWithLoginExtensionManager).loginExtensionManager;
loginExtensionManager.setResources({ Login_AdminMail: 'admin@docsvision.com' }); ①
if (loginExtensionManager.currentLocale === 'ru') {
    loginExtensionManager.setResources({ Login_Greeting: 'Приветствуем,' });
} else {
    loginExtensionManager.setResources({ Login_Greeting: 'Hello,' });
}

const plugin: ILoginPlugin = {
    order: 1,
    eventName: "LogicEventConstructor",
    fn: (data) => {
②
    },
    name: 'myPlugin',
};
const plugin2: ILoginPlugin = {
    order: 2,
    eventName: "EventContainerUseEffect",
    fn: (data) => {
③
```

```
    },
    name: 'myPlugin2',
};

④
loginExtensionManager.addPlugin(plugin);
loginExtensionManager.addPlugins([plugin2]);

⑤
loginExtensionManager.onSetupView = function (view: LoginView): LoginView {
⑥
    view.logo = React.memo((logic: LoginLogic): React.ReactElement => {
        return <img src=
'https://www.google.com/images/branding/googlelogo/2x/googlelogo_color_92x30dp.png'
className="my-custom-logo" />;
    });

⑦
    view.authBoxBefore = (logic: LoginLogic): React.ReactElement => null;
    view.authBoxAfter = (logic: LoginLogic): React.ReactElement => null;
    view.formBefore = (logic: LoginLogic): React.ReactElement => null;
    view.formAfter = (logic: LoginLogic): React.ReactElement => <a href="#" className=
"my-custom-element" >Забыли пароль?</a>;
    view.providersBefore = (logic: LoginLogic): React.ReactElement => null;
    view.providersAfter = (logic: LoginLogic): React.ReactElement => null;
    return view;
};
```

① Замените текст.

② Добавьте код выполнения некоторого действия.

③ Добавьте код выполнения ещё одного действия.

④ Добавьте события.

⑤ Измените компоненты представления.

⑥ Замените логотип.

⑦ Точки расширения.

Проверка примера

1. Перейти на страницу логина в браузере http://company.com:5004/account/login.
2. В результате на странице входа будут изменены логотип и приветствие, а также добавлена почта администратора.

Расширение серверных вызовов

Пример серверного расширения для взаимодействия с панелью настроек таблицы. Пример содержит клиентский скрипт позволяющий добавить элемент в список контекстного меню с возможностью открыть карточку в модальном окне или скрыть существующий элемент.

Ссылка на пример на GitHub: ExtendServerQueryData.

```
extensionManager.registerExtension({ ①
    name: "ExtendServerQueryData",
    version: "6.1",
    globalEventHandlers: [ExtendServerQueryData]
});
```

① Регистрация расширения позволяет корректно установить все обработчики событий, сервисы и прочие сущности web-приложения.

На следующем шаге необходимо клонировать и заменить собственный сервис контроллера. Ниже приведён код обработчика события, в котором показано, как это сделать:

```
export function extendServerQueryData(sender: Layout) {
    const services = cloneServices(sender.params.services);
    replaceService(services, { tasksTreeController: new TasksTreeControllerCustom(sender
.params.services)});
    sender.params.services = services;
}
```

Затем на основе метода базовой реализации стандартного класса сформировать новые данные:

```
export class TasksTreeControllerCustom extends GenControllers.TasksTreeController {
    constructor(services) {
        super(services);
    }
    async getTasksTable(request: GenModels.TasksTableLoadOptions, options?:
RequestOptions) {
        var data = await super.getTasksTable(request, options);
        if (data && data.rows) {
            data.rows.forEach(row => {
                if (row.indicators.includes(GenModels.TasksIndicators.Overdue)) {
                    row.indicators.push(GenModels.TasksIndicators.HighPriority)
                }
```

```
            })
        };
        return data;
    }
}
```

Проверка примера

1. Запустите конструктор разметок.
2. Импортируйте решение из файла `ExtendServerQueryData.xml`.
3. Сделайте решение `ExtendServerQueryData` активным для карточки типа *Документ* вида *ДокументУД/Исходящий*.
4. В примере используется стандартная разметка просмотра исходящего документа. На событие **`Перед открытием карточки`** ЭУ `Layout` разметки назначается обработчик `extendServerQueryData`.
5. Перезапустите **dvwebclient**.
6. Откройте исходящий документ, создайте подчиненное задание с просроченной датой исполнения.
7. Убедитесь, что в таблице исполнения просроченные задания автоматически помечаются индикатором высокого приоритета.

Управление диалогом подписания через API

Управлять диалогом подписания можно при помощи специального API.

*Свойства интерфейса `ISignatureDialogProps` в событии **`Перед открытием диалога подписи`** (`beforeSignatureDialogShown`) позволяют:*

- Скрывать выбор метки.
- Скрывать выбор сертификата.
- Отключать простую подпись.
- Скрывать кнопку настройки способов подписания.
- Скрывать сам диалог: диалог открывается, но оказывается невидим на экране для пользователя.
- Добавлять содержимое в нижней и верхней части диалога.
- Обрабатывать доступные способы подписания: фильтровать, модифицировать, добавлять свои.
- Управлять отрисовкой кнопок: добавлять свои, заменять стандартные.

Управление диалогом через интерфейс `IDigitalSignatureDialogManagement` *события* **`После открытия диалога подписи`** *(*`afterSignatureDialogShown`*) позволяет получать информацию о загруженных метках и способах подписания:*

- Программно выбирать метку и способ подписания
- Изменять видимость диалога для пользователя
- Динамически менять свойства диалога (см. п. 3)
- Программно подтверждать или отменять операцию подписания
- Также событие `beforeSignatureDialogShown` позволяет подменять логику формирования подписи и прикрепления к документу. Это можно использовать для подключения сторонних провайдеров подписания.

Входная точка — сервис `$DigitalSignature`. Для настройки диалога следует использовать события `beforeSignatureDialogShown` и `afterSignatureDialogShown` указанного сервиса. В первом событии можно настроить свойства диалога, во втором получить доступ к управлению диалогом. В событиях доступна информация о карточке, из которой осуществляется подпись: вид, тип, состояние и другая информация.

Следующий пример кода демонстрирует автоматический выбор сертификата для документов вида *Исходящий* и подписание без показа диалога:

```
export function initExtension(app) {
        app.digitalSignature.beforeSignatureDialogShown.watch(args => { ①
            if (args.cardInfo.kindId == '8e40f327-9517-4a43-998d-bf2bd619588d') {
                 args.dialogProps.showHidden = true;
            }
        });
        app.digitalSignature.afterSignatureDialogShown.watch(async (args) => { ②
            let methods = await args.management.getSignatureMethods();
            let outgoingMethod = methods.find(x => x?.certificateInfo?.friendlyName
.includes('для исходящего'));
            if (outgoingMethod) {
                args.management.selectSignatureMethod(outgoingMethod);
                args.management.accept();
            } else {
                const props = args.management.getDialogProps();
                props.footer = "Не удалось найти сертификат для исходящих документов.";
                args.management.update(props);
                args.management.setDialogVisible(true);
            }
        });
```

```
}
```

① По умолчанию скрываем диалог для документов вида *Исходящий*.

② Автоматически выбираем сертификат и нажимаем подписать.

Можно вставить в метод `init` при вызове `registerExtension`, либо просто скопировать в консоль браузера и вызвать `initExtension(WebClient.app)`.

Пример интеграции со сторонним облаком:

```
export function initExtension(app) {
    app.digitalSignature.beforeSignatureDialogShown.watch(args => {
       args.options.onCreateSignature = (options) => { return signByCustomCloud(options);
}
    });
}
```

Свойства диалога объекта `dialogProps` в событии `beforeSignatureDialogShown`:

```
    hideLabelSelection?: boolean; ①
    hideSignatureMethodsSetup?: boolean; ②
    hideSimpleSign?: boolean; ③
    hideSignatureMethodsSelect?: boolean; ④

    showHidden?: boolean; ⑤
    footer?: React.ReactNode; ⑥
    header?: React.ReactNode; ⑦

    onSign?: (options: IDigitalSignOptions) => Promise<void>; ⑧
    onCancel?: () => void; ⑨
    onPrepareSignatureMethods?: (methods: GenModels.SignatureMethod[]) => GenModels
.SignatureMethod[]; ⑩
    onRenderButtons?: (props: ISignatureDialogButtonsProps, defaultButtons: JSX.
Element[]) => JSX.Element[]; ⑪
```

① Не отображать выбор метки.

② Не отображать кнопку настройки вариантов подписания.

③ Скрыть метод простой подписи.

④ Не отображать раскрывающийся список выбора способа подписания. В таком случае будет выбран первый вариант по умолчанию.

⑤ Чтобы не отображать диалог на экране, используйте API `@link IDigitalSignatureDialogManagement`.

⑥ Сообщение, показываемое в нижней части диалога

⑦ Сообщение, показываемое в верхней части диалога

⑧ Вызывается при подписи.

⑨ Вызывается при отмене подписи.

⑩ Вызывается для обработки отображаемого списка доступных способов подписания.

⑪ Вызывается при отрисовке кнопок диалога.

Свойства объекта `management` *в событии* `afterSignatureDialogShown`*:*

```
    close(reason?: unknown): Promise<void>; ①
    accept?(data?: IDigitalSignOptions): Promise<void>; ②
    getSignatureMethods(): Promise<GenModels.SignatureMethod[]>; ③
    getLabels(): Promise<ISignatureLabel[]>; ④
    selectSignatureMethod(method: GenModels.SignatureMethod): void; ⑤
    selectLabel(label: ISignatureLabel): void; ⑥
    getDialogVisible(): boolean; ⑦
    setDialogVisible(visible: boolean): void; ⑧
    getDialogProps(): ISignatureDialogProps; ⑨
    update(props?: ISignatureDialogProps): void; ⑩
```

① Закрыть диалог без сохранения данных.

② Применить изменения и закрыть диалог. Параметр не обязателен.

③ Возвращает список доступных сертификатов и других способов подписания.

④ Возвращает список доступных меток подписи.

⑤ Выбирает в раскрывающемся списке указанный способ подписания.

⑥ Выбирает в раскрывающемся списке указанную метку подписи.

⑦ Возвращает значение свойства видимости диалога для пользователя.

⑧ Устанавливает видимость диалога для пользователя.

⑨ Возвращает текущие свойства диалога подписания.

⑩ Обновляет диалог с указанными свойствами. См. `@link getDialogProps`.

Примеры тестирования

Скрытие раскрывающегося списка для выбора метки в диалоге подписания:

```
WebClient.app.digitalSignature.beforeSignatureDialogShown.watch(args => {

    args.dialogProps.hideLabelSelection = true;
```

```
});
```

Отображение сообщения в нижней части диалога подписания:

```
WebClient.app.digitalSignature.afterSignatureDialogShown.watch(async (args) => {

    const props = args.management.getDialogProps();

    props.footer = "Добрый день!";

    args.management.update(props);

});
```

Автоматический выбор метки для карточек определенного вида:

```
WebClient.app.digitalSignature.afterSignatureDialogShown.watch(async (args) => {

    let labels = await args.management.getLabels();

    if (args.cardInfo.kindId == '8e40f327-9517-4a43-998d-bf2bd619588d') {

        args.management.selectLabel(labels[1]);

    }

});
```

Реализация собственной логики формирования подписи:

```
WebClient.app.digitalSignature.beforeSignatureDialogShown.watch(args => {

       args.options.onCreateSignature = (options) => { return alert('custom sign create
'); }

});
```

Работа со строкой данных элемента управления "Список файлов" (fileList)

Пример взаимодействия со строками элемента управления Список файлов (fileList). Проект представляет собой клиентское расширение. Расширение добавляет к строке элемента управления красную обводку при скачивании файла.

Основной код расширения содержит скрипт `fileListHighlightRow` в качестве обработчика события **После загрузки версии файла** элемента управления `fileList` в конструкторе разметок.

EventHandler.ts

```
export async function fileListHighlightRow(sender: FileListControl, event:
IFileVersionDownloadingArgs) {
    const fileItem = event.fileItem;
    const customClass = "file-list-row_red-border";

    if (!event.fileItem?.cssClasses?.includes(customClass)) {
        sender.toggleRowClass(fileItem, customClass);
    }
}
```

Данный скрипт представляет пример получения объекта `fileItem`, представляющего информацию о файле из аргументов события. Выполняется исследование его свойства, после чего вызывается метод `toggleRowClass` ЭУ `FileListControl` для переключения стилей файла.

Index.ts

```
extensionManager.registerExtension({ ①
    name: "FileListHighlightRow",
    version: "1.0",
    globalEventHandlers: [ EventHandler ]
})
```

① Регистрация расширения позволяет корректно установить все обработчики событий, сервисы и прочие сущности web-приложения.

Проверка примера

1. Откройте папку с элементом управления и добавленным обработчиком события.
2. Скачайте версию файла.

Настройка страницы входа

Данный раздел содержит описание примера кастомизации страницы входа в Web-клиент.

В этом примере демонстрируется возможность внедрения своего кода на

страницу входа в Docsvision Web-клиент, а именно:

- Выполнение кода при определённых событиях(тип `LoginEvents` в модуле `@docsvision/webclient/Account`).
- Подмена текста на страницы(объект `defaultLoginResources` в модуле `@docsvision/webclient/Account`).
- Замена компонентов страницы(объект `LoginView` в модуле `@docsvision/webclient/Account/LoginContainer`).
- Добавление стилей на страницу.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше

Сборка и установка

1. Откройте консоль в папке `ClientScripts\TypeScript\LoginPage` и выполните команду `npm install`, затем `npm run build:prod`.
2. Скопируйте каталог `SamplesOutput\Content\Account\Scripts` в каталог `Путь к сайту Web-клиента\Content\Account\Scripts`.
3. Перезапустите **dvwebclient**.

Дополнительная информация

- Рекомендуется осуществлять сборку в виде umd модуля.
- Для корректной работы расширения нужно создать файл или файлы с расширением `.js` и положить в папку `Путь к сайту Web-клиента\Content\Account\Scripts`, код будет выполнен перед загрузкой приложения и выполнением `ReactDOM.render()`.
- Чтобы добавить стили, нужно создать файл или файлы с расширением `.css` и положить в папку `Путь к сайту Web-клиента\Content\Account\Styles`, стили загружаются после основных стилей страницы.
- Входной точкой для внедрения кода на страницу является объект `loginExtensionManager` доступный глобально через объект `windows`.
- Код расширения будет выполнен перед загрузкой приложения и выполнением `ReactDOM.render()`, но после загрузки библиотек React и ReactDOM.
- Методы `loginExtensionManager`, а также возможные ресурсы, компоненты и

события для замены обозначены в виде типов и интерфейсов в модуле `@docsvision/webclient/Account`.

Блок сообщений

Данный раздел содержит описание примера последовательного вывода нескольких информационных сообщений. Проект-расширение клиентской части Web-клиента. Содержит клиентский скрипт с функцией `showMessages` реализующей последовательный вывод `MessageBox`.

Ссылка на пример на GitHub: MessageBox.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше

Сборка и установка

1. Откройте консоль в папке `MessageBoxWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
2. Скопируйте каталог `SamplesOutput\Content\Modules\MessageBoxWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
3. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок
2. Импортируйте разметку из файла `MessageBoxLayout.xml` без условий использования. При этом будет добавлена разметка в тип карточки документ, вид *ДокументУД\Исходящий*
3. Задайте для разметки условия использования, поменяв также порядок разметок в условиях использования, чтобы разметка стала первой разметкой для редактирования
4. Перезапустите **dvwebclient**.
5. Откройте новый документ *ДокументУД\Исходящий* и нажмите кнопку **Редактировать**. При этом должна открыться разметка с кнопкой вызова сообщений.
6. При нажатии на кнопку `ShowMessages` последовательно появляются диалоговые окна: информационное сообщение с текстом `Это информационное сообщение`,

предупреждающее сообщение с текстом `Это предупреждающее сообщение`, сообщение об ошибке с текстом `Это сообщение об ошибке`, диалог подтверждения с текстом `Выберите кнопку`. В случае нажатия той или иной кнопки выводится сообщение.

Модальное окно

Данный раздел содержит описание примера работы с модальным окном. Проект-расширение клиентской части Web-клиента. Содержит клиентский скрипт с функцией `showModalDialog` реализующей отображение `ModalDialog`. `ModalDialogComponent` — React компонент, отвечающий за представление модельного окна и его событий.

Ссылка на пример на GitHub: ModalDialog.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше

Сборка и установка

1. Откройте консоль в папке `ModalDialogWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
2. Скопируйте каталог `SamplesOutput\Content\Modules\ModalDialogWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
3. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок
2. Импортируйте разметку из файла `ModalDialogLayout.xml` без условий использования. При этом будет добавлена разметка в тип карточки документ, вид *ДокументУД\Исходящий*
3. Задайте для разметки условия использования, поменяв также порядок разметок в условиях использования, чтобы разметка стала первой разметкой для редактирования
4. Перезапустите **dvwebclient**.
5. Откройте новый документ *ДокументУД\Исходящий* и нажмите кнопку **Редактировать**. При этом должна открыться разметка с кнопкой вызова модального окна и текстовым полем.

6. При нажатии на кнопку **ShowModalDialog** отображается модальное окно, в котором присутствует текстовое поле для ввода. После ввода сообщения и нажатия на кнопку **OK**, значение из этого поля копируется в текстовое поле на разметке и сохраняется в качестве имени документа.

Боковая панель

Данный раздел содержит описание примера работы с боковой панелью. Проект-расширение клиентской части Web-клиента. Содержит клиентский скрипт с функцией `showModalSidebar` реализующей отображение `ModalSidebar`. `ModalSidebarComponent` — React компонент, отвечающий за представление боковой панели и её событий.

Ссылка на пример на GitHub: ModalSidebar.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше

Сборка и установка

1. Откройте консоль в папке `ModalSidebarWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
2. Скопируйте каталог `SamplesOutput\Content\Modules\ModalSidebarWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
3. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок
2. Импортируйте разметку из файла `ModalSidebarLayout.xml` без условий использования. При этом будет добавлена разметка в тип карточки документ, вид *ДокументУД\Исходящий*.
3. Задайте для разметки условия использования, поменяв также порядок разметок в условиях использования, чтобы разметка стала первой разметкой для редактирования.
4. Перезапустите **dvwebclient**.
5. Откройте новый документ *ДокументУД\Исходящий* и нажмите кнопку **Редактировать**. При этом должна открыться разметка с кнопкой вызова боковой панели и текстовым полем.

6. При нажатии на кнопку **ShowModalSidebar** отображается боковая панель, в котором присутствует текстовое поле для ввода. После ввода сообщения и нажатия на кнопку **ОК**, значение из этого поля копируется в элемент управления `textBox1` разметки.

Отображение незаполненных обязательных полей

Данный раздел содержит описание примера отображения незаполненных обязательных полей. Проект-расширение клиентской части Web-клиент. Расширение отображает окно со списком незаполненных обязательных полей при создании или сохранении карточки.

Ссылка на пример на GitHub: ShowRequiredFields.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше

Сборка и установка

1. Откройте консоль в папке `ShowRequiredFields` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
2. Скопируйте каталог `SamplesOutput\Content\Modules\ShowRequiredFields` в каталог `Путь к сайту Web-клиента\Content\Modules`.
3. Перезапустите **dvwebclient**.
4. Добавьте скрипт `showRequiredFields` в качестве обработчика события **`Перед сохранением карточки`** `root` для разметок создания и/или редактирования в конструкторе разметок.

Проверка примера

1. Создайте карточку с обязательными полями и добавленным обработчиком события.
2. Не заполняя обязательные поля, нажмите на кнопку создать.

Проверка данных ЭУ

Данный раздел содержит описание примера проверки, изменения и отмены изменения состояния карточки. Подписаться на событие изменения состояния, провести проверку какого-нибудь поля и случае, если проверка не прошла, выдать предупреждающее сообщение и состояние не менять.

Ссылка на пример на GitHub: ValidateOnStateChanging.

Проект-расширение клиентской части Web-клиент. Содержит клиентский скрипт с функцией-обработчиком события `validateOnStateChanging`, проверку данных ЭУ при смене состояния.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше

Сборка и установка

1. Откройте консоль в папке `ValidateOnStateChangingWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
2. Скопируйте каталог `SamplesOutput\Content\Modules\ValidateOnStateChangingWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
3. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Импортируйте решение из файла `ValidateOnStateChangingLayout.xml`.
3. Сделайте решение `ValidateOnStateChangingLayout` активным для карточки типа Документ вида *ДокументУД/Исходящий*.
4. На разметке расположен ЭУ `Текст` и кнопки операций (Start approving и др.) по изменению состояния карточки.
5. Перезапустите **dvwebclient**.
6. Откройте новый документ *ДокументУД\Исходящий* и нажать кнопку редактировать. При этом должна открыться разметка с ЭУ `Текст` и кнопками изменения состояния. Часть кнопок будет неактивна, это зависит от настроек ролевой модели.
7. Если в ЭУ `Текст` нет текста или длина текста превышает 100 символов, то при нажатии на кнопку изменения состояния возникает сообщение об ошибке и состояние не изменяется.
8. В случае, если ЭУ `Текст` проходит валидацию, то состояние карточки меняется. Например при выполнении операции `Start approving` с состояния `Drafting` на состояние `Is approving`.

9. При этом кнопки перехода исчезают, т.к. в состоянии `Is approving` операции `Start approving` и др. не доступны.

Данный пример демонстрирует только доступ к элементу разметки. При реализации расширений рекомендуется использовать экспорт разметок в виде Решения.

Расширения нового грида

Руководство по разработке решений Web-клиента

Расширения нового грида

Описание примеров разработки расширений нового грида.

- Расширить контекстное меню
- Применить группировку кнопкой в ячейке
- Отобразить количество непрочитанных карточек
- Больше примеров категории в навигационном меню слева…

Расширить контекстное меню

Грид позволяет расширить функциональность контекстного меню при помощи расширений. Данный раздел описывает пример добавления нового пункта контекстного меню. Новый пункт контекстного меню позволяет открыть карточку в модальном окне или скрыть существующий элемент.

Ссылка на пример на GitHub: GridContextMenuItem.

Начальный код расширения должен импортировать прямо или косвенно все остальные строки, чтобы затем объединить код в один пакет.

```
extensionManager.registerExtension({ ①
    name: "GridContextMenuItem",
    version: "6.1",
    initialize() {
        app.folderPluginProvider.addFactory(new GridContextMenuItemPluginFactory());
    },
})
```

① Регистрация расширения позволяет корректно установить все обработчики событий, сервисы и прочие сущности web-приложения.

Добавить новый пункт контекстного меню

Расширение добавляет пункт контекстного меню, который открывает карточку в модальном окне.

```
export const AddContextMenuItemProvider: TablePlugins.ServiceProvider<$ContextMenuOptions
& $FolderGrid & $LayoutCardController & $DeviceType & $LayoutManager> = {
    name: "AddContextMenuItemProvider",
    description: "Добавляет пункт контекстного меню, который открывает карточку в
модальном окне",
    feature: GridContextMenuItemFeature,
    composition: TableCompositionNames.Root,
    addServices: (composition) => {
        const { contextMenuOptions, folderGrid } = composition.services;

        if (contextMenuOptions) {

            const isAdded = contextMenuOptions.isAddedProvider
(CONTEXT_MENU_ITEM_CUSTOM_PROVIDER_ID, TableCompositionNames.TableCell); ①

            if (!isAdded) {
                const provider = {
                    id: CONTEXT_MENU_ITEM_CUSTOM_PROVIDER_ID, ②
                    composition: TableCompositionNames.TableCell, ③
                    getItemsToAdd: () => ([ ④
                        {
                            id: "CONTEXT_MENU_CUSTOM_ITEM", ⑤
                            name: "Предпросмотр карточки", ⑥
                            action: (context: TableCompositions.Cell) => showCard(
context, composition.services), ⑦
                            blockId: folderGrid?.getContextMenuBlockNames().
linkOperation, ⑧
                            order: 0, ⑨
                        }
                    ])
                } as IDataItemProvider;

                contextMenuOptions.addItemProvider(provider);
            }
        }
    }
};
```

① Поскольку метод `addServices` вызывается при каждой прорисовке, проверим не был ли добавлен ранее наш провайдер.

② Уникальный ID провайдера, содержащего элементы списка контекстного меню.

③ Композиция, где будет вызвано контекстное меню.

④ Метод, который возвращает массив элементов контекстного меню.

⑤ Уникальный ID элемента контекстного меню.

В данном примере с помощью ID осуществляется доступ к скрытию элемента контекстного меню.

⑥ Отображаемое название элемента контекстного меню.

⑦ Метод, который будет вызван после клика на элемент контекстного меню.

⑧ Указание на расположение элемента в блоке элементов контекстного меню.

⑨ Порядковый номер расположения элемента в блоке элементов контекстного меню.

Скрыть пункт контекстного меню

Следующий пример скрывает пункт "Открыть" в контекстном меню грида.

```
export const HideContextMenuItemProvider: TablePlugins.ServiceProvider
<$ContextMenuOptions> = {
    name: "HideContextMenuItemProvider",
    description: "Скрывает пункт «Открыть» в контекстном меню",
    feature: GridContextMenuItemFeature,
    composition: TableCompositionNames.Root,
    addServices: (composition) => {
        const { contextMenuOptions } = composition.services;

        if (contextMenuOptions) {

            const isAdded = contextMenuOptions.isAddedFilter
(CONTEXT_MENU_ITEM_CUSTOM_FILTER_ID, TableCompositionNames.TableCell); ①

            if (!isAdded) {
                const filter = {
                    id: CONTEXT_MENU_ITEM_CUSTOM_FILTER_ID, ②
                    composition: TableCompositionNames.TableCell, ③
                    getItemsToHide: () => [CONTEXT_MENU_FOLDER_OPEN_ID] ④
                } as IDataItemFilter;
```

```
                contextMenuOptions.addItemFilter(filter);
            }
        }
    }
};
```

① Поскольку метод addServices вызывается при каждой прорисовке, проверим не был ли добавлен ранее наш фильтр.

② Уникальный ID фильтра.

③ Композиция, где будет вызвано контекстное меню.

④ Метод, который возвращает массив ID элементов контекстного меню.

Отображение модального окна

Данный пример демонстрирует создание и отображение модального окна.

```
export async function showCard(context: TableCompositions.Cell, services:
$ContextMenuOptions & $FolderGrid & $LayoutCardController & $DeviceType & $LayoutManager)
{
    const modalHost = new CommonModalDialogHost("grid-context-menu__card-layout",
services); ①

②

    const dialogProps = modalHost.dialogProps as CommonModalDialogProps;

    dialogProps.header = "Предпросмотр"; ③

    dialogProps.maxWidth = "80vw"; ④

    dialogProps.maxHeight = "80vh"; ⑤

    dialogProps.maximizeButtonEnabled = true;
    dialogProps.onMaximize = () => {
        dialogProps.maximized = !dialogProps.maximized;
        modalHost.forceUpdate(); ⑥
    }

    modalHost.onCancelCallback = () => services.layoutManager.destroy
(GRID_CONTEXT_MENU_SHOW_CARD_LOCATION_NAME); ⑦

    const renderLoading = (state: LoadingState) => <LoadingViewWithText state={state}
className="padding-20 margin-30" />; ⑧

    const loadLayout = async (): Promise<GenModels.LayoutViewModel> => { ⑨
```

```
        const model = await services.layoutCardController.view(context.data.row.entityId,
PlatformModeConditionTypes.VIEW);
        return model;
    }

    const dialogServices = addServices(services, modalHost.service, true); ⑩

    modalHost.renderCallback = () => (
        <LocationContainerControl
            async={true}
            renderLoadingState={renderLoading}
            customLayoutLoader={loadLayout} ⑪
            locationName={GRID_CONTEXT_MENU_SHOW_CARD_LOCATION_NAME}
            services={dialogServices}
            name={GRID_CONTEXT_MENU_SHOW_CARD_LOCATION_NAME} />
    );

    await modalHost.showDialog(); ⑫
}
```

① Модальное окно.

② Конфигурируем модальное окно.

③ Заголовок модального окна.

④ Максимальная ширина.

⑤ Максимальная высота.

⑥ Изменение отображения модального окна (свернуть / развернуть).

⑦ Обработчик закрытия модального окна.

⑧ Отображение текста загрузки разметки.

⑨ Получение разметки карточки в режиме просмотра.

⑩ Сервисы модального окна.

⑪ Тело модального окна.

⑫ Показываем модальное окно.

Зарегистрируйте плагин в `GridContextMenuItemPluginFactory`:

```
export class GridContextMenuItemPluginFactory implements IFolderPluginFactory {
    id: string = "GridContextMenuItemPluginFactory";

    getTablePlugins(): ITablePlugins[] {
        return [
```

```
            GridContextMenuItemPlugins
        ]
    }
}
```

Проверка примера

1. Откройте папку с таблицей и вызовите контекстное меню у произвольной карточки.
2. Кликните на пункт "Предпросмотр карточки" в контекстном меню.
3. Убедитесь, что открылось модальное окно "Предпросмотр", в котором отображается разметка просмотра выбранной карточки.

Применить группировку кнопкой в ячейке

Данный раздел описывает расширение, добавляющее элемент в панель инструментов грида. Элемент будет отображаться в ячейках таблицы при наведении на них курсора мыши. Нажатие на элемент будет выполнять группировку по столбцу.

Ссылка на пример на GitHub: GridHoverPanelGrouping.

```
extensionManager.registerExtension({
    name: "HoverPanelGroping",
    version: "6.1",
    initialize() {
        app.setService($HoverPanel, new CustomHoverPanelService()); ①
    },
})
```

① Заменяем реализацию стандартного сервиса `$HoverPanel` на `CustomHoverPanelService`.

Следующим шагом добавляем в ячейку таблицы элемент, который будет выполнять функцию группировки:

```
export const HoverPanelGroupingByColumn: HeaderHoverPanelPlugin<$Grouping & $SidePanel> =
{
    name: "HoverPanelGroupingByColumn",
    description: `
        Добавляет элемент в панель при наведении, который находится в ячейке шапки
таблицы.
        При клике на элемент, происходит группировка по столбцу.
```

```
    `,
    feature: CustomHoverPanelFeature,
    composition: HeaderHoverPanelName,
    placement: PluginPlacement.AfterContent,
    component: (props) => {
①
        const { data: { column }, services: { grouping, sidePanel } } = props.
composition;



②
        const groupingPath = useStore(grouping.$groupingPath);

        const groupByColumn = () => {
③
            if (grouping.canAddColumnGrouping(column.id) && !sidePanel.$isOpen.
getState()) {
                sidePanel.toggle();
            }

            grouping.addColumnGrouping(column.id); ④
        };

        const isGrouped = Boolean(groupingPath.find(path => path.columnId == column.id));
⑤

        if (grouping.$groupingAvailable.getState() && (isGrouped || !grouping
.canAddColumnGrouping(column.id))) {
            return <></>; ⑥
        }

        return (
            <Icon
                iconClass="dv-ico context-menu-group hover-panel-custom-grouping-by-
column"
                onClick={groupByColumn}
                dv-tooltip="Сгруппировать по столбцу" /> ⑦
        );
    }
};
```

① Деструктуризируем объект `props`, получая все необходимые данные для реализации группировки по столбцу

② С помощью хука useStore получаем данные из хранилища `$groupingPath`.

Хук предоставляет возможность перерисовывать компонент при изменении

состояния хранилища `$groupingPath`.

Так как мы следим за состоянием хранилища и перерисовываем компонент, мы можем учитывать следующую логику:

- Если столбец уже сгруппирован, то не показывать иконку
- Если столбец не сгруппирован, то показывать иконку

③ Проверяем доступна ли колонка для группировки. Если колонка доступна и боковая панель закрыта, открываем боковую панель с группировкой.

④ Группируем по колонке.

⑤ Проверяем сценарий, что группировка по колонке уже применена.

⑥ Если группировка доступна и уже применена или недоступна для этой колонки, то ничего не отрисовываем.

⑦ Если группировка доступна, рисуем иконку. Группировка будет применена при нажатии на иконку.

```
export class CustomHoverPanelService implements IHoverPanelService { ①


    getPlugins(defaultPlugins: HeaderHoverPanelPlugin<any>[]): HeaderHoverPanelPlugin<
any>[] { ②

③

        defaultPlugins.push(HoverPanelGroupingByColumn); ④

        return defaultPlugins;
    }
}
```

① Создаём сервис алогичный `IHoverPanelService`..

② При инициализации плагинов для `HoverPanel`, будет вызван этот метод.

Внутри него мы можем убрать, добавить интересующие нас плагины для панели при наведении и вернуть новый массив плагинов.

③ К примеру:

- `defaultPlugins = defaultPlugins.filter(plugin => plugin.name !== target)`
- `defaultPlugins.push(ownPlugin)`

④ Добавляем плагин для группировки по колонке.

Проверка примера

1. Откройте папку с таблицей.
2. Наведите курсор на ячейку в столбце таблицы, которая доступна для группировки.
3. После нажатия на иконку "Сгруппировать по столбцу", должна открыться боковая панель, если она была закрыта, и добавиться колонка в группируемых элементах.

Отобразить количество непрочитанных карточек

Расширение в данном примере предназначено для добавления кнопки на панель инструментов грида. Расширение получает список карточек и по нажатию на кнопку отображает общее количество карточек, а также количество непрочитанных.

Ссылка на пример на GitHub: GridToolbarButton.

```
extensionManager.registerExtension({ ①
    name: "GridToolbarButton",
    version: "6.1",
    initialize() {
        app.folderPluginProvider.addFactory(new ToolbarButtonPluginFactory());
    },
})
```

① Регистрация расширения позволяет корректно установить все обработчики событий, сервисы и прочие сущности web-приложения.

Следующим шагом необходимо получить количество карточек:

```
export function ToolbarButtonMessageContent(props: IToolbarButtonMessageContent) {
    const { data, services } = props;

    useStore(services.tableData.$data); ①

    const loadedRows = data.rows.filter(x => getRowLoaded(x).loaded).length; ②
    const unreadRows = data.rows.filter(x => getUnreadRow(x).isUnread).length; ③

    return (
        <>
            <LabeledText label={"На текущий момент загружено карточек"} text={loadedRows}
/>
            <LabeledText label={"Из них не прочитано"} text={unreadRows} />
        </>
```

```
    )
}
```

① Обновляем компонент при изменении хранилища `$data` из `services.tableData`.

② Количество загруженных карточек.

③ Количество непрочитанных карточек.

Затем нужно получить плагины грида:

```
export class ToolbarButtonPluginFactory implements IFolderPluginFactory {
    id: string = "ToolbarButtonPluginFactory";

    getTablePlugins(): ITablePlugins[] { ①
        return [ ToolbarButtonPlugins ];
    }
}
```

① Возвращаем плагины грида.

Добавляем кнопку, отображающую информацию в модальном окне при нажатии:

```
export const ToolbarButtonButtonPlugin: TablePlugins.Toolbar.Component<$MessageWindow &
$TableData> = {
    name: "ToolbarButtonButtonPlugin",
    description: "Отображает кнопку в панели инструментов таблицы",
    feature: ToolbarButtonFeature,
    composition: TableCompositionNames.Toolbar,
    order: CompositionPluginOrder.Priority,
    component: (props) => {
        const { data, services } = props.composition;

        const showInfo = () => {
            services.messageWindow.showInfo(() => <ToolbarButtonMessageContent data={
data} services={services} />); ①
        };

        return <IconButton iconClassName="dv-ico ico-modal-info" onClick={showInfo} />;
②
    }
};
```

① Вызывается модальное окно с информацией.

② Кнопка, при нажатии на которую открывается модальное окно с информацией.

Проверка примера

1. Откройте папку с таблицей и нажмите на информационную кнопку в панели настроек таблицы.
2. Убедитесь, что открылось модальное окно с информацией. В окне должно отображаться количество загруженных карточек на данный момент и количество непрочитанных из них.

Изменить цвет строки по кнопке

В данном примере создаётся новая колонка таблицы грида. В новую колонку добавляется элемент, нажатие на который открывает диалог выбора цвета. При изменении цвета в диалоге, будет изменяться фон строки, в которой находится ячейка.

```
extensionManager.registerExtension({ ①
    name: "BackgroundRows",
    version: "6.1",
    initialize() {
        app.folderPluginProvider.addFactory(new BackgroundRowsPluginFactory());
    },
})
```

① Регистрация расширения позволяет корректно установить все обработчики событий, сервисы и прочие сущности web-приложения.

Создание кнопки, открывающей диалог выбора цвета:

```
export interface IBackgroundRowsButton extends PropsWithChildren<ICompositionPluginProps
<"TableCell", ICellData, $BackgroundRows>> {
}

export function BackgroundRowsButton (props: IBackgroundRowsButton) { ①
    const { services: { backgroundRows }, data: { row } } = props.composition;

    const [ color, setColor ] = useState( backgroundRows.$rows.getState()[row.entityId]
?? "#ffffff" ); ②

    const changeColor = (ev: ChangeEvent<HTMLInputElement>) => {
        backgroundRows.setBackground({ id: row.entityId, color: ev.target.value }); ③
        setColor(ev.target.value); ④
    };

    const click = (ev: React.MouseEvent) => {
        ev.stopPropagation(); ⑤
```

```
    };

    if (!getRowLoaded(row).loaded) { ⑥
        return <></>
    }

    return (
        <input
            className="background-rows__button"
            type="color"
            value={color}
            onClick={click}
            onChange={changeColor} />
    )
};
```

① Предоставляет возможность отобразить палитру цветов

② Исходный цвет: белый.

③ Обновляем хранилище.

④ Обновляем локальное состояние `input` с `type=color`.

⑤ Прекращает дальнейшую передачу текущего события.

⑥ Если строка ещё не загружена, то ничего не показываем.

Создаём сервис, который будет хранить и обновлять цвет строки:

```
export class BackgroundRowsService implements IBackgroundRowsService {
    $rows: Store<IBackgroundRows>;
    setBackground: Event<IBackgroundRow>;

    constructor() {
        const domain = createDomain("BackgroundRows"); ①

        this.setBackground = domain.event("setBackground");
        this.$rows = domain.store({}, { name: "$rows" })
            .on(this.setBackground, (store, background) => ({...store, [background.id]:
background.color }));
    }
}
```

① Инициализируем сущности необходимые для хранения и обновления хранилища.

Добавляем интерфейс сервиса:

```
export interface IBackgroundRows {
    [id: string]: string;
}

export interface IBackgroundRow {
    id: string;
    color: string;
}

export interface IBackgroundRowsService { ①
    readonly $rows: Store<IBackgroundRows>;
    setBackground: Event<IBackgroundRow>;
}
```

① Интерфейс сервиса.

Инициализируем сущности, необходимые для хранения и обновления хранилища:

```
export class BackgroundRowsService implements IBackgroundRowsService {
    $rows: Store<IBackgroundRows>;
    setBackground: Event<IBackgroundRow>;

    constructor() {
        const domain = createDomain("BackgroundRows");

        this.setBackground = domain.event("setBackground");
        this.$rows = domain.store({}, { name: "$rows" })
            .on(this.setBackground, (store, background) => ({...store, [background.id]:
background.color }));
    }
}
```

Подключаем плагины грида:

```
export class BackgroundRowsPluginFactory implements IFolderPluginFactory {
    id: string = "BackgroundRowsPluginFactory";

    getDataLoadingPlugins(): IFolderDataLoadingPlugin[] {
        return [
            new BackgroundRowsResponseResolver()
        ]
    }

    getTablePlugins(): ITablePlugins[] {
        return [
```

```
            BackgroundRowsPlugins
        ];
    }
}
```

Добавляем новую колонку для элемента переключения цвета:

```
export class BackgroundRowsResponseResolver implements IFolderDataLoadingPlugin {
    id: string = "BackgroundRowsResponseResolver";
    description: string = "Добавляет новую колонку.";
    order: PluginOrder = PluginOrder.Normal;

    async resolveResponse(data: ITableData, response: GenModels.GridViewModelEx):
Promise<void | ResponseResolveResult> { ①
        if (data.columns.length && !data.columns.find(x => x.id ==
BackgroundRowsColumnId)) { ②

            const backgroundRowsColumn = {
                id: BackgroundRowsColumnId, ③
                name: "Фон строки", ④
            } as IColumn;

            data.columns.push(backgroundRowsColumn);
        }
    }
}
```

① Вызывается после каждого ответа загрузки данных таблицы.

② Сделаем проверку на существование столбцов в таблице, а также проверим не был ли добавлен столбец, который мы хотим создать.

③ Уникальный идентификатор столбца.

④ Отображаемое название столбца.

Настраиваем элемент в ячейке:

```
export const BackgroundRowsColumnId = "backgroundRows";
export const BackgroundRowsFeature = "BackgroundRowsFeature";

export const BackgroundRowsServiceProvider: TablePlugins.ServiceProvider<$BackgroundRows>
= {
    name: "BackgroundRowsServiceProvider",
    description: "Добавляет сервис $BackgroundRows.",
    feature: BackgroundRowsFeature,
    composition: TableCompositionNames.Root,
    addServices: (composition) => {
```

```
        if (!composition.services.backgroundRows) { ①
            composition.services.backgroundRows = new BackgroundRowsService();
        }
    }
};

export const BackgroundRowsCellButtonPlugin: TablePlugins.Cell.Component<$BackgroundRows>
= {
    name: "BackgroundRowsCellButtonPlugin",
    description: "Отображает кнопку изменения фона строки.",
    feature: BackgroundRowsFeature,
    composition: TableCompositionNames.TableCell,
    shouldRender: (composition) => composition.data.column.id == BackgroundRowsColumnId,
②
    component: BackgroundRowsButton ③
};

export const BackgroundRowsMountEffect: TablePlugins.Row.MountEffect<$BackgroundRows> = {
    name: "BackgroundRowsMountEffect",
    description: "Обновляет композицию при изменении $rows",
    feature: BackgroundRowsFeature,
    composition: TableCompositionNames.TableRow,
    compositionDidMount: (composition) => { ④
        const { backgroundRows } = composition.services;

        const optimalUpdate = throttle({ source: backgroundRows.$rows, timeout: 100 });
⑤

        const update = optimalUpdate.watch(() => composition.update()); ⑥
        return () => update.unsubscribe(); ⑦
    }
};

export const BackgroundRowDecorator: TablePlugins.Row.Decorator<$BackgroundRows> = {
    name: "BackgroundRowDecorator",
    description: "Изменяет фон строки.",
    feature: BackgroundRowsFeature,
    composition: TableCompositionNames.TableRow,
    jsxDecorator: (node, composition) => { ⑧
        const { services, data } = composition;
        const backgroundRows = services.backgroundRows.$rows.getState(); ⑨

        return decorate(node, { style: { background: backgroundRows[data.row.entityId]
}}); ⑩
    }
};

export const BackgroundRowsPlugins: ITablePlugins = {
```

```
    serviceProviders: [ BackgroundRowsServiceProvider ],
    row: {
        mountEffects: [ BackgroundRowsMountEffect ],
        containerDecorators: [ BackgroundRowDecorator ]
    },
    cell: {
        content: [ BackgroundRowsCellButtonPlugin ]
    }
};
```

① Поскольку `addServices` вызывается при каждой отрисовке, сделаем проверку на существование сервиса, который хотим добавить.

② Компонент будет отображён, если условие правдиво.

③ Компонент, который будет отображён в ячейке добавленной колонки.

④ Вызывается при монтировании компоненты.

⑤ Поскольку хранилище `backgroundRow.$rows` может меняться достаточно быстро (зависит от скорости изменения цвета в палитре), необходимо установить ограничение обновления композиции.

В данном случае, `optimalUpdate` будет срабатывать не чаще, чем раз в `100` мс для оптимизации браузера.

⑥ Когда сработает событие, композиция таблицы будет обновлена.

⑦ Отписка от события, когда компонент будет размонтирован.

⑧ Метод, с помощью которого появляется возможность декорирования `ReactNode`.

⑨ Получаем хранилище, которое содержит информацию о цвете фона строк.

⑩ В данном методе вторым аргументом является объект свойств `ReactNode`, как в `React.createElement`.
В данном случае, если хранилище содержит информацию о фоне строки, она будет применена, в противном случае стиль не применится.

Проверка примера

1. Откройте папку с таблицей. В колонке "Фон строки" должен отображаться созданный элемент.
2. Нажмите на любой элемент в колонке и измените цвет.

Взаимодействие колонки таблицы со строками

Данный раздел содержит описание примера взаимодействия новой колонки

таблицы с её строками. Проект-расширение клиентской части Web-клиента. Содержит клиентский скрипт позволяющий добавить новую колонку в таблицу, отрисовать элемент в её ячейках, при нажатии на который открывается палитра. Изменяя палитру, будет изменяться фон строки, в которой находится ячейка.

Ссылка на пример на GitHub: GridColumnBackgroundRows.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше

Сборка и установка

1. Откройте консоль в папке `GridColumnBackgroundRowsWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
2. Скопируйте каталог `SamplesOutput\Content\Modules\GridColumnBackgroundRowsWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
3. Перезапустите **dvwebclient**.

Проверка примера

1. Откройте папку с таблицей, где в колонке *Фон строки*, нажмите на любое поле ввода и изменить цвет.

Данный пример демонстрирует только взаимодействие новой колонки, её ячеек и фон строки, в которой они находятся, с помощью скрипта. Сохранение выбранного цвета строки не реализовано, поэтому при обновлении страницы цвет сбросится. При необходимости сохранение можно добавить в классе `BackgroundRowsService`.

Расширение возможностей серверной части Web-клиента

Серверное расширение — загружаемый в процесс сервера Web-клиента компонент, реализующий дополнительную функциональность серверного уровня.

Серверное расширение может добавлять:

- Методы WebApi.

- Поддержку новых типов карточек.
- Преобразователи значений элементов управления (binding-конвертер).
- Расширение или замена стандартных сервисов, с дополнительной логикой.
- Поддержка новых условий выбора разметок.
- Методы жизненного цикла карточек и строк.
- И другие точки расширения стандартной функциональности.

Создание и публикация серверного расширения

Серверное расширение представляет собой библиотеку DLL, разработанную на платформе .NET.

Ссылка на пример на GitHub: TemplateServerExtension.

Перечень необходимых инструментов

Пример рассчитан на версию Web-клиента dv6 или выше. * Microsoft Visual Studio 2022. * Docsvision WebClient SDK (запрашивается через портал ТП).

Описание файлов проекта

- В корне репозитория расположено решение для Visual Studio (`Samples.sln`) и ключ для подписания сборок примеров (`StrongNameKey.snk`). Рекомендуется использовать Visual Studio 2022.
- `Assemblies` — сборки, необходимые для использования примеров
- `ServerExtension.cs` — входная точка расширение, в которой регистрируются сервисы и прочие сущности.
- `Sign.snk` — файл подписи сборки. Для установки сборки должны быть подписаны. Рекомендуется сгенерировать новый файл подписи в настройках проекта.
- `Resource.resx`, `Resource.ru.resx` — локализации, используемые в расширении.
- `Feature1` — папка, содержащая реализацию некоторой функциональности.
- `Feature1/Feature1Controller.cs` — класс, регистрирующий конечные точки Web API.
- `Feature1/IFeature1Service.cs` и `Feature1/Feature1Service.cs` — интерфейс и реализация сервиса, реализующего логику обработки запроса.
- `Feature1/Models` — классы моделей, используемых в сервисе и контроллере.

Сборка и установка

Чтобы создать серверное расширение:

1. Убедитесь, что установлен Docsvision WebClient SDK.

   В этом случае в переменной окружения `DocsvisionWebClientSDK` содержится путь к его папке, например, `C:\Program Files (x86)\Docsvision\WebClient\SDK`.

2. Установите шаблон серверного расширения и создайте на его основе новый проект с названием `LearnDocvisionServerExtension`.

3. Обновите в проекте ссылки на зависимости.

   > После крупного обновления версии Docsvision, необходимо повторно скомпилировать расширение с новыми версиями компонентов. В остальных случаях такой необходимости нет.
   >
   > Это необходимо, чтобы версии подключаемых компонентов соответствовали версиям компонентов, используемых сервером Web-клиента, для которого разрабатывается расширение.

4. Реализуйте в проекте функции серверного расширения.

   > Примеры кода серверных расширений приведены в следующих пунктах раздела, а также в репозитории "ДоксВижн".

5. Настройте ядро серверного расширения.

   > Ядром серверного расширения называется файл `ServerExtension.cs` в проекте расширения. Ядро является производным от `WebClientExtension` классом, содержащим ссылки на реализуемые функции расширения.

```
public class ServerExtension : WebClientExtension
{
    public ServerExtension(IServiceProvider serviceProvider)
        : base()
    {
    }

    public override string ExtensionName ①
    {
```

```
        get { return Assembly.GetAssembly(typeof(ServerExtension)).GetName().
Name; }
    }

    public override Version ExtensionVersion ②
    {
        get { return new Version(FileVersionInfo.GetVersionInfo(Assembly
.GetExecutingAssembly().Location).FileVersion); }
    }

    public override void InitializeServiceCollection(IServiceCollection services)
③
    {
④
        services.AddSingleton<IFeature1Service, Feature1Service>();
    }

    protected override List<ResourceManager> GetLayoutExtensionResourceManagers()
    {
        return new List<ResourceManager>
        {

        };
    }
⑤

}
```

① Название расширения.

② Версия расширения.

③ Метод регистрации добавляемых расширением сервисов, преобразователей, конвертеров и т.п. в IoC контейнере.

④ Примеры регистрации.

⑤ Далее идут функции, возвращающие реализованные функции серверного расширения.

Описание изменений, которые необходимо выполнить в ядре расширения, приведено в описании примеров расширений, добавляющих обработчики событий и локализованные ресурсы.

В серверном расширении Web-клиента используется IoC контейнер, реализованный с использованием библиотеки Autofac. Описание данной библиотеки и её возможностей см. на сайте autofac.org.

6. Скомпилируйте проект.

   В результате будет получен файл DLL с ядром серверного расширения и реализуемыми функциями, а также файлы DLL с локализованными ресурсами.

7. Скопируйте `safeprojectnamesafeprojectname.dll` и `safeprojectnamesafeprojectname.dll.pdb` из папки **bin** › **WebClient** › **Bin** в **Каталог-установки-Web-клиента** › **Site** › **bin**.

   Расширения можно устанавливать в папки **Site** › **Extensions** и **Site** › **bin**. В ходе разработки расширения можно удобнее устанавливать в папку **Site** › **bin**, т.к. это избавляет от необходимости перезапускать **dvwebclient** при установке сборок.

   В ресурсной поставке рекомендуется устанавливать сборки в папку **Site** › **Extensions**, чтобы проще было видеть, какие расширения установлены в Web-клиенте. Однако этот способ требует перезапуска **dvwebclient** при установке или обновлении сборки.

8. Перезапустите **dvwebclient**.

Проверка работы

1. Откройте Web-клиент.
2. Откройте диалог "О программе" и убедитесь, что расширение присутствует в списке подключенных расширений.
3. Откройте произвольный документ и в консоли браузера выполните:

   ```
   $ await
   layoutManager.cardLayout.getService("requestManager").post("api/Feature1/Action1",
   JSON.stringify({ documentId: layoutManager.cardLayout.getService("cardId") }));
   ```

4. Обновите документ, убедиться, что в названии добавился знак.

Разработка расширения с методами WebApi

Данный раздел содержит пример добавления новых web-методов, реализующих функции взаимодействия с сервером Docsvision (создание и изменение карточек, изменение данных, запуск бизнес-процессов), а также выполнения других задач, для которых необходимы возможности сервера Web-клиента.

Чтобы создать расширение, добавляющее новые методы WebApi, выполните приведенную далее инструкцию.

1. Создайте проект серверного расширения.
2. Добавьте в проект класс WebApi контроллера (производный от `System.Web.Http.ApiController`).
3. Реализуйте в классе контроллера веб-методы.

Следующий код демонстрирует пример реализации контроллера с методом, проверяющим существование карточки в Docsvision.

```
using DocsVision.BackOffice.ObjectModel;
using DocsVision.Platform.WebClient.Helpers;
using System;
using System.Web.Http;

namespace WebApiServerExtension
{
    public class CardServiceController : ApiController
    {
        private readonly ICurrentObjectContextProvider
currentObjectContextProvider; ①

        public CardServiceController(ICurrentObjectContextProvider
currentObjectContextProvider) ②
        {
            this.currentObjectContextProvider = currentObjectContextProvider;
        }

        [HttpGet]
        public bool Exists(Guid cardId) ③
        {
            var objectContext = currentObjectContextProvider
.GetOrCreateCurrentSessionContext().ObjectContext; ④

            try
            {
```

```
                var card = objectContext.GetObject<BaseCard>(cardId); ⑤

                return (card != null); ⑥
            }
            catch
            {
                return false;
            }
        }
    }
}
```

① В конструкторе контроллера получаем ссылку на сервис-провайдер,

② Из сервис-провайдера можно получать различные сервисы объектной модели, а также сервис для получения контекста объектов.

③ Метод, проверяющий существует ли карточка с идентификатором cardId. Если карточка существует, метод возвращает true, иначе — false.

④ Получаем пользовательский контекст объектов.

Всегда нужно получать контекст объектов заново. Дополнительная информация в пункте Получение контекста объектов в серверном расширении.

⑤ Получаем карточку с указанным идентификатором.

⑥ Возвращаем true, если при получении карточки из контекста объектов не было ошибки и полученный объект не null.

4. Скомпилируйте проект и скопируйте полученную сборку на сервер Web-клиента в папку `../../../lib/docsvision/webclient/Extensions/%Каталог Решения%`. Перезапустите **dvwebclient**.

Для проверки работоспособности примера, откройте в веб-браузере страницу `http://%WebCinstallDir%/DocsvisionWebClient/api/CardService/Exists?cardId=<CardID>`. Вместо *CardID* укажите идентификатор любой существующей карточки. Метод вернет значение **true** (если карточки не существует — **false**).

Пример клиентского расширения, использующего метод `Exists`, приведён

в пункте Разработка расширения с клиентским сервисом.

Получение контекста объектов в серверном расширении

Программист может использовать в серверном расширении контекст объектов Docsvision для доступа к данным системы.

Разработчику доступен пользовательский контекст объектов. Пользовательский контекст создаётся для каждого пользователя при его входе в Web-клиент. Права на работу с объектами соответствуют правам данного пользователя в системе Docsvision.

Получить контекст объектов можно используя функции сервиса `ICurrentObjectContextProvider`:

- `GetOrCreateCurrentSessionContext` — возвращает контекст объектов текущего пользователя.

Не используйте метод `GetOrCreateApplicationPoolSessionContext`, это приведет к блокировкам потоков под нагрузкой.

Сервис `ICurrentObjectContextProvider` может быть получен следующим способом:

```
private readonly ICurrentObjectContextProvider _currentObjectContextProvider;

public DocumentRegistrationController(ICurrentObjectContextProvider
currentObjectContextProvider)
{
    _currentObjectContextProvider = currentObjectContextProvider;
}
```

Web-клиент периодически закрывает неиспользуемые пользовательские сессии, из-за чего контекст объектов, сохранённый в статический объект класса (серверного расширения или контроллера), спустя время станет недействительным. **Всегда получайте контекст объектов заново** (с помощью функции `GetOrCreateCurrentSessionContext`).

Примеры работы с контекстом объектов рассмотрены в Руководстве разработчика. Следующий код демонстрирует пример получения объектной модели карточки *Документ*.

```
Document document = objectContext.GetObject<Document>(Guid.Parse("9E11083F-D0EC-4F9F-8981-285498280BD9")); ①
```

① В метод GetObject передаются тип объекта (Document) и идентификатор объекта (ID карточки).

Добавление сервисов и поддержки новых типов карточек в контекст объектов

По умолчанию в контекст объектов, предоставляемый в серверном расширении, загружаются сервисы и преобразователи данных (см. документ Руководство разработчика Docsvision), позволяющие работать с карточками библиотек *Базовые объекты* и *Конструктор согласований*, системными карточками и карточками приложения *Управление документами*: `SystemCardsMapperFactory`, `BackOfficeMapperFactory`, `ApprovalDesignerMapperFactory`, `SystemCardsServiceFactory`, `BackOfficeServiceFactory`, `DocumentsManagementServiceFactory` и `ApprovalDesignerServiceFactory`.

Если требуется поддержать собственные типы карточек в объектной модели Docsvision, программист должен добавить в контекст объектов преобразователи данных для этих типов, и сервисы для работы с данными карточками. Инструкция по разработке фабрик сервисов и фабрик преобразователей данных приведена в Руководстве разработчика Docsvision.

Выполните указанные ниже действия для добавления сервисов и преобразователей данных в контекст объектов. Все действия выполняются в проекте серверного расширения (нового или существующего).

1. Подключите к проекту сборки, в которых реализованы классы разработанных фабрик сервисов и преобразователей данных.
2. Откройте файл ядра расширения.
3. Переопределите метод `OnObjectContextCreate`, вызываемый при создании контекста объектов, добавив в него код регистрации фабрик сервисов и преобразователей данных.

   Следующий код демонстрирует пример добавления в контекст объектов фабрики сервисов `CustomLibraryServiceFactory` и фабрики преобразователей данных `CustomLibraryMapperFactory`.

```
public override void OnObjectContextCreate(ObjectContext objectContext)
 {
    base.OnObjectContextCreate(objectContext);
```

```
①

    IObjectMapperFactoryRegistry mapperFactoryRegistry = objectContext.GetService
<IObjectMapperFactoryRegistry>(); ②

    mapperFactoryRegistry.RegisterFactory(typeof(CustomLibraryMapperFactory)); ③

    IServiceFactoryRegistry serviceFactoryRegistry = objectContext.GetService
<IServiceFactoryRegistry>(); ④

    serviceFactoryRegistry.RegisterFactory(typeof(CustomLibraryServiceFactory)); ⑤
}
```

① Далее следует стандартный код добавления фабрик в контекст объектов (см. Руководство разработчика Docsvision).

② Получаем из существующего контекста объектов сервис регистрации фабрик преобразователей данных.

③ Регистрируем фабрику преобразователей данных, реализованную в классе CustomLibraryMapperFactory.

④ Получаем из существующего контекста объектов сервис регистрации фабрик сервисов.

⑤ Регистрируем фабрику сервисов, реализованную в классе `CustomLibraryServiceFactory`.
Добавленный в контекст объектов сервис можно получить стандартным образом:

```
ISomeService someService = objectContext.GetService<ISomeService>();
```

+ . Опубликуйте серверное расширение.

+ NOTE: Сборки, содержащие классы фабрик сервисов и преобразователей данных, нужно скопировать в каталог со сборкой данного серверного расширения.

Методы управления жизненным циклом объектов Web-клиента

В данном пункте рассмотрены механизмы API Web-клиента, которые позволяют влиять на функции создания, изменения и удаления карточек и строк секций из разметок, а также сервис `ILifeCycleService`, предоставляющий доступ к данным механизмам.

Реализация методов для работы с карточками из разметок

Программисту предоставляется возможность добавить новую функциональность в механизмы создания и работы с карточками из разметок. Данная возможность реализуется с помощью специального класса с интерфейсом `ICardLifeCycleEx`. Интерфейс `ICardLifeCycleEx` предоставляет методы, которые вызываются на различных этапах работы с карточкой:

```
    public interface ICardLifeCycleEx
    {
        Guid CardTypeId { get; } ①
        Guid Create(SessionContext sessionContext, CardCreateLifeCycleOptions options);
②
        bool Validate(SessionContext sessionContext, CardValidateLifeCycleOptions
options, out List<ValidationResult> validationResults); ③
        void OnSave(SessionContext sessionContext, CardSaveLifeCycleOptions options); ④
        bool CanDelete(SessionContext sessionContext, CardDeleteLifeCycleOptions options,
out string message); ⑤
        void OnDelete(SessionContext sessionContext, CardDeleteLifeCycleOptions options);
⑥
        string GetDigest(SessionContext sessionContext, CardDigestLifeCycleOptions
options); ⑦
    }
```

① Идентификатор типа карточек.

② `Create` — Вызывается при получении разметки создания карточки.

③ `Validate` — Вызывается перед сохранением изменений карточки в БД с возможностью прервать сохранение.

④ `OnSave` — Вызывается при сохранении изменений карточки. Позволяет добавить дополнительную логику в сохранение карточки, но без возможности прервать сохранение.

⑤ `CanDelete` — Вызывается перед удалением карточки. Позволяет прервать удаление.

⑥ `OnDelete` — Вызывается при удалении карточки. Позволяет добавить дополнительную логику в удаление карточки, но без возможности прервать удаление.

⑦ `GetDigest` — Вызывается при получении дайджеста карточки при её сохранении. Позволяет добавить собственную логику формирования дайджеста. Метод `GetDigest` используется, только когда способ формирования дайджеста не настроен в Docsvision и карточка сохраняется из

пользовательского интерфейса (при нажатии кнопки Сохранить, сохранении по месту или скриптом).

Указанные методы используются **исключительно** при выполнении стандартных операций со строками из разметок или с использованием JS WebAPI разметок. В других случаях методы интерфейса `ICardLifeCycleEx` не вызываются и не влияют на работу сервера Web-клиента с карточками.

Следующий код демонстрирует пример реализации класса, в котором определены собственные функции создания и сохранения карточки.

```
using System;
using DocsVision.BackOffice.ObjectModel.Services;
using DocsVision.BackOffice.ObjectModel;
using DocsVision.Platform.WebClient;
using DocsVision.WebClientLibrary.ObjectModel.Services.EntityLifeCycle;
using DocsVision.WebClientLibrary.ObjectModel.Services.EntityLifeCycle.Options;
using DocsVision.BackOffice.CardLib.CardDefs;


internal class DocumentCardLifeCycle : BaseCardLifeCycleEx
{
    public override Guid CardTypeId => CardDocument.ID;

    public override Guid Create(SessionContext sessionContext, CardCreateLifeCycleOptions
options)
    {
        if (!options.CardKindId.HasValue)
            throw new ArgumentNullException(nameof(options.CardKindId));

        var objectContext = sessionContext.ObjectContext;
        var kindsCardKind = objectContext.GetObject<KindsCardKind>(options.CardKindId);
        var documentService = objectContext.GetService<IDocumentService>();
        var document = documentService.CreateDocument(null, kindsCardKind);
        document.Description = document.SystemInfo.CardKind.Name;
        document.MainInfo.Name = "Наименование документа";

        objectContext.AcceptChanges();
        return document.GetObjectId();
    }


    public override void OnSave(SessionContext sessionContext, CardSaveLifeCycleOptions
options)
    {
        base.OnSave(sessionContext, options);
```

```
        var objectContext = sessionContext.ObjectContext;
        var document = objectContext.GetObject<Document>(options.CardId);
        document.MainInfo.Name = "Принудительно установленное наименование документа";

        objectContext.AcceptChanges();
    }
}
```

В приведенном коде класс `DocumentCardLifeCycle` наследуется от базовой реализации интерфейса `ICardLifeCycleEx` — класса `BaseCardLifeCycle` с абстрактными методами. Помимо класса `BaseCardLifeCycle` с базовой реализацией, API включает реализующий интерфейс `ICardLifeCycleEx` класс `DefaultCardLifeCycle` с реализацией по умолчанию методов `Create` и `CreateFromTemplate`.

Метод `Create` в приведенном выше коде создаёт новую карточку *Документ* с предустановленным названием (`document.MainInfo.Name`). Метод `CreateFromTemplate` создаёт карточку *Документа* по шаблону. Метод `Save` изменяет значение поля `document.MainInfo.Name` при сохранении карточки.

Тип с интерфейсом `ICardLifeCycleEx` нужно добавить в IoC-контейнер в ядре расширения:

```
public override void InitializeServiceCollection(IServiceCollection services)
{
    services.AddTransient<ICardLifeCycleEx, DocumentCardLifeCycle>();
}
```

Серверное расширение должно быть опубликовано стандартным образом.

Реализация методов для работы со строками секций из разметок

Программисту предоставляется возможность добавить новую функциональность в механизмы создания и работы со строками секциями из разметок. Данная возможность реализуется с помощью специального класса с интерфейсом `IRowLifeCycleEx`. Интерфейс `IRowLifeCycleEx` предоставляет методы, которые вызываются на различных этапах работы со строкой секции:

```
public interface IRowLifeCycleEx
{
    Guid SectionId { get; } ①
    Guid Create(SessionContext sessionContext, RowCreateLifeCycleOptions options); ②
    bool Validate(SessionContext sessionContext, RowValidateLifeCycleOptions options, out
```

```
List<ValidationResult> validationResults); ③
    void OnSave(SessionContext sessionContext, RowSaveLifeCycleOptions options); ④
    bool CanDelete(SessionContext sessionContext, RowDeleteLifeCycleOptions options, out
string message); ⑤
    void OnDelete(SessionContext sessionContext, RowDeleteLifeCycleOptions options); ⑥
}
```

① Идентификатор секции, для строк которой предоставляется реализация интерфейса IRowLifeCycleEx.

② `Create` — Вызывается при получении разметки создания новой строки секции.

③ `Validate` — Вызывается перед сохранением изменений строки в БД, с возможностью прервать сохранение.

④ `OnSave` — Вызывается при сохранении изменений строки — позволяет добавить дополнительную логику в сохранение строки, но без возможности прервать сохранение.

⑤ `CanDelete` — Вызывается перед удалением строки секции — позволяет прервать удаление.

⑥ `OnDelete` — Вызывается при удалении строки секции — позволяет добавить дополнительную логику в удаление строки, но без возможности прервать удаление.

Указанные методы используются **исключительно** при выполнении стандартных операций со строками из разметок или с использованием JS WebAPI разметок. В других случаях методы интерфейса `IRowLifeCycleEx` не вызываются и не влияют на работу сервера Web-клиента со строками секций.

Следующий код демонстрирует пример реализации класса, в котором определена собственная функция проверки данных строки секции перед сохранением изменений (`Validate`).

```
using System.Collections.Generic;
using DocsVision.BackOffice.CardLib.CardDefs;
using DocsVision.Platform.WebClient;
using DocsVision.WebClientLibrary.ObjectModel.Services.EntityLifeCycle;
using DocsVision.WebClientLibrary.ObjectModel.Services.EntityLifeCycle.Options;

public class RefPartnersCompaniesRowLifeCycle : DefaultRowLifeCycleEx ①
{
    public RefPartnersCompaniesRowLifeCycle()
        : base(RefPartners.Companies.ID)
    {
```

```
    }

    public override bool Validate(SessionContext sessionContext,
RowValidateLifeCycleOptions options, out List<ValidationResult> validationResults)
    {
        var isValid = base.Validate(sessionContext, options, out validationResults);
        var rowData = sessionContext.AdvancedCardManager.GetRowData(options.CardId,
RefPartners.Companies.ID, options.RowId);
        var companyName = rowData.GetString(RefPartners.Companies.Name);
        if (string.IsNullOrEmpty(companyName))
        {
            isValid = false;
            var validationResult = new ValidationResult
            {
                Message = "Наименование компании не заполнено"
            };
            validationResults.Add(validationResult);
        }
        return isValid;
    }
}
```

① `RefPartnersCompaniesRowLifeCycle` наследуется от стандартной реализации интерфейса `IRowLifeCycleEx` — класса `DefaultRowLifeCycle`, в котором уже реализованы необходимые методы. Помимо класса `DefaultRowLifeCycle` с реализацией по умолчанию API включает реализующий интерфейс `IRowLifeCycleEx` базовый класс `BaseRowLifeCycle` с абстрактными методами.

Метод `Validate` проверяет строку `rowId` на заполненность поля `RefPartners.Companies.Name`, и, если данное поле пустое, возвращает ошибку `validationResult`.

Полученный тип с интерфейсом `IRowLifeCycleEx` нужно добавить в IoC-контейнер в ядре расширения:

```
public override void InitializeServiceCollection(IServiceCollection services)
{
    services.AddTransient<IRowLifeCycleEx, RefPartnersCompaniesRowLifeCycle>();
}
```

Серверное расширение должно быть опубликовано стандартным образом.

Работа с сервисом жизненного цикла объектов Web-клиента

Рассмотренные в пунктах Реализация методов для работы с карточками из разметок и Реализация методов для работы со строками секций из разметок компоненты, которые позволяют внести изменения в механизмы создания, изменения и удаления карточек и строк секций из разметок, доступны программисту через специальный сервис `ILifeCycleServiceEx`. Данный сервис определяет методы:

- `GetCardLifeCycle` — Возвращает реализацию интерфейса `ICardLifeCycleEx` для указанного типа карточек.
- `GetRowLifeCycle` — Возвращает реализацию интерфейса `IRowLifeCycleEx` для указанной секции.

Следующий код демонстрирует способ получения сервиса `ICardLifeCycleEx` и его использования в веб-методе для создания карточки по шаблону:

```
public class CardServiceController : ApiController
{
    private readonly ILifeCycleServiceEx lifeCycleService;
    private readonly ICurrentObjectContextProvider currentObjectContextProvider;

    public CardServiceController(ILifeCycleServiceEx lifeCycleService,
ICurrentObjectContextProvider currentObjectContextProvider) ①
    {
        this.lifeCycleService = lifeCycleService;
        this.currentObjectContextProvider = currentObjectContextProvider;
    }

    [HttpGet]
    public Guid CreateByTemplate(Guid templateId)
    {
        var sessionContext = currentObjectContextProvider
.GetOrCreateCurrentSessionContext();
        var documentCardLifeCycle = lifeCycleService.GetCardLifeCycle(CardDocument.ID);
②
        var newDocumentId = Guid.Empty;
        if (templateId != Guid.Empty)
        {
            newDocumentId = documentCardLifeCycle.Create(sessionContext, new
CardCreateLifeCycleOptions()
            {
                TemplateId = templateId ③
            });
        }
```

```
            return newDocumentId;
        }
    }
```

① Принимаем требуемый сервис в конструкторе.

② Получаем интерфейс для работы с карточкой типа `CardDocument`.

③ Используем метод создания карточки по шаблону.

Изменение штампа электронной подписи

Пример демонстрирует вставку собственных изображений штампов электронной подписи в файл, приложенный к карточке Docsvision.

Ссылка на пример на GitHub: CSPSignatureVisualization.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- Microsoft Visual Studio 2022.

Создайте описание расширения для Web-клиента, которое задано в текущей сборке

```
    public class CSPSignatureVisualizationExtension : WebClientExtension
    {
        public CSPSignatureVisualizationExtension() ①
        {
        }

        #region Свойства

        public override string ExtensionName ②
            => Assembly.GetAssembly(typeof(CSPSignatureVisualizationExtension)).
GetName().Name;

        public override Version ExtensionVersion ③
            => new Version(FileVersionInfo.GetVersionInfo(Assembly.
GetExecutingAssembly().Location).FileVersion);

        #endregion Свойства

        #region Обработчики событий

        public override void InitializeServiceCollection(IServiceCollection services) ④
        {
```

```
            services.AddSingleton<IImageGenerator, CSPSignatureImageGenerator>(); ⑤

            services.AddSingleton<IImagePositionSelector,
CSPSignatureImagePositionSelector>(); ⑥
        }

        #endregion Обработчики событий
    }
```

① Создаём новый экземпляр класса `CSPSignatureVisualizationExtension`.

Параметры:

- `serviceProvider` — сервис-провайдер.

② Получаем название расширения.

③ Получаем версию расширения.

④ Регистрируем типы в IoC контейнере.

Параметры:

- `containerBuilder` — контейнер.

⑤ Регистрируем тип `CSPSignatureImageGenerator` как реализацию интерфейса `IImageGenerator` в `containerBuilder`.

⑥ Регистрируем тип `CSPSignatureImagePositionSelector` как реализацию интерфейса `IImagePositionSelector` в `containerBuilder`.

Настройка штампа ЭП

Описание генератора изображений штампов электронной подписи выглядит следующим образом:

```
    public class CSPSignatureImageGenerator : IImageGenerator ①
    {
        private const double ViewBoxHeight = 88; ②

        private const double ViewBoxWidth = 314; ③

        private const double ContentTop = 64; ④

        private const double FontHeight = 14; ⑤

        private const int MaxStringsLength = 38; ⑥
⑦
```

① Высота изображения штампа электронной подписи.

② Ширина изображения штампа электронной подписи.

③ Левая верхняя координата информационного содержимого штампа электронной подписи.

④ Высота используемого шрифта в пикселях (Используется десятый размер шрифта).

⑤ Максимальная длина используемого в изображении штампа электронной подписи текста. При стандартном интервале между символами. Используется для разбиения текста на отдельные строки.

⑥ Полный код приведён в файле **CSPSignatureVisualization › CSPSignatureVisualizationServerExtension › DataVisualization › CSPSignatureImageGenerator.cs** в репозитории примера на GitHub.

Описание класса, вставляющего в PDF-файл изображения штампов электронной подписи:

```
namespace CSPSignatureVisualizationServerExtension.DataVisualization
{
    public class CSPSignatureImagePositionSelector : IImagePositionSelector
    {
        private static readonly double ImageMargin = ImageHelper.PixelsToUnits(5); ①

        private static readonly int PagePadding = 30; ②

        private const int PageNumber = 0; ③

        #region Реализация интерфейса IImagePositionSelector

        public GeneratedFileInfo Generate(SessionContext sessionContext,
ImageGeneratorType generatorType, List<byte[]> images, Stream fileStream) ④
        {
            ArgumentNullException.ThrowIfNull(images);
            ArgumentNullException.ThrowIfNull(fileStream);

            using var pdfDocument = PdfReader.Open(fileStream);
            var page = pdfDocument.Pages[PageNumber];

            var pageWidth = page.Width.Value;
            var pageHeight = page.Height.Value;

            double xIndent = PagePadding; ⑤
            double yIndent = PagePadding;
```

```
            double maxImageHeight = 0; ⑥

            foreach (var imageHelper in images.Select(x => new ImageHelper(x)))
            {
                if (yIndent + imageHelper.Height > pageHeight) ⑦
                    break;

                if (xIndent + imageHelper.Width > pageWidth) ⑧
                {
                    yIndent += maxImageHeight + ImageMargin;
                    xIndent = PagePadding;
                    maxImageHeight = imageHelper.Height;
                }
                else if (imageHelper.Height > maxImageHeight) ⑨
                {
                    maxImageHeight = imageHelper.Height;
                }

                InsertImage(pdfDocument, page, imageHelper, xIndent, yIndent); ⑩

                xIndent += imageHelper.Width + ImageMargin; ⑪
            }

            var outputStream = new MemoryStream(); ⑫
            pdfDocument.Save(outputStream);
            pdfDocument.Close();
            outputStream.Seek(0, SeekOrigin.Begin);

            return new GeneratedFileInfo { Stream = outputStream };
        }
⑬
```

① Расстояние между изображениями штампов электронной подписи.

② Координаты первого изображения штампа электронной подписи. 30 миллиметров.

③ Номер страницы, на которую требуется вставить изображение штампа.

④ Вставляет изображения штампов электронной подписи на первую страницу указанного PDF-файла.

Параметры:

- `sessionContext` — контекст сессии.
- `generatorType` — тип генератора изображений штампов электронной подписи.

- `images` — массив изображений штампов электронной подписи.
- `fileStream` — поток, содержащий PDF-файл, в который необходимо вставить указанные изображения.

  Возвращаемое значение: модель файла со вставленными изображениями.

⑤ Задаём начальные координаты первого изображения штампов электронной подписи.

⑥ Максимальная высота изображения штампа в текущей строке (нужна для выравнивания изображений в строке).

⑦ Если верхняя левая координата текущего изображения больше высоты страницы PDF-документа, прекращаем вставку изображений.

⑧ Если ширина строки вставленных изображений превышает ширину страницы, переходим на следующую строку.

⑨ В противном случае вычисляем максимальную высоту изображения штампа в текущей строке.

⑩ Вставляем текущее изображение по указанным координатам.

⑪ Вычисляем горизонтальную координату следующего изображения.

⑫ Копируем в поток сформированный PDF-файл со вставленными изображениями штампов электронной подписи.

⑬ Полный код приведён в файле **CSPSignatureVisualization › CSPSignatureVisualizationServerExtension › DataVisualization › CSPSignatureImagePositionSelector.cs** в репозитории примера на GitHub.

Пример взаимодействия с сервисом конвертации

Этот пример серверного расширения демонстрирует взаимодействие с сервисом конвертации.

Ссылка на пример на GitHub: ConversionSample.

Необходимое ПО

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Для работы расширения требуется наличие в лицензии опции *Docsvision Сервис конвертации файлов*.

Сборка

1. Откройте `/Samples.sln`
2. Соберите проект **ServerExtensions** › **ConversionSample** › **ConversionSampleServerExtension**.
3. Откройте консоль в папке **ServerExtensions** › **ConversionSample** › **ConversionSampleWebExtension**, выполните `npm install`, `npm update` и `npm run build:prod`.
4. Скопируйте каталог `SamplesOutput\Content\Modules\ConversionSampleWebExtension` в каталог `Путь-к-установленному-Web-клиенту\Content\Modules`.
5. Скопируйте каталог `SamplesOutput\Extensions\ConversionSampleServerExtension` в каталог `Путь к установленному Web-клиенту\Extensions`.
6. Перезапустите **dvappserver**.
7. Перезапустите **dvwebclient**.

Проверка примера

1. Откройте Конструктор Web-разметок.
2. В разметке просмотра документа на событие **`Перед стартом согласования`** элемента управления `agreementManagement` назначьте обработчик `attachPdfa`.
3. Создайте карточку документа, приложите к ней `.docx`-файл и отправьте на согласование.
4. К документу будет добавлен файл, конвертированный в PDF/A.

Изменение экспорта в Excel

Пример серверного расширения, позволяющего вмешаться в процесс экспорта представления в Excel.

Ссылка на пример на GitHub: ExcelExport.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- Microsoft Visual Studio 2022.

Настраиваем собственные параметры экспорта:

```
namespace ExcelExportServerExtension.ExcelExport ①
{
    public class ExcelExportExampleService : ExcelExportService
    {
        private readonly Color oddRowColor = Color.LightBlue; ②

        protected override void ApplyWorkbookSettings(XLWorkbook workbook) ③
        {
            foreach (var worksheet in workbook.Worksheets)
            {
                for (var i = 1; i < worksheet.RangeUsed().RowCount(); i += 2) ④
                {
                    var row = worksheet.RangeUsed().Row(i);
                    row.Style.Fill.BackgroundColor = XLColor.FromColor(oddRowColor);
                }
            }
        }

        protected override string GetGridRowParamValueAsString(object paramValue, bool
noNullValues = false) ⑤
        {
            if (paramValue is DateTime dateTime) ⑥
            {
                return dateTime.ToLongDateString();
            }

            return base.GetGridRowParamValueAsString(paramValue);
        }

        protected override IEnumerable<GridColumn> GetGridColumns(GridViewModel
viewModel) =>
            viewModel.Columns.Select(gridColumn => CreateGridColumnModel(gridColumn,
viewModel))
                .OrderBy(column => column.UserColumn?.Order ?? 0)
                .Select(column => column.GridColumn); ⑦

        private static GridColumnModel CreateGridColumnModel(GridColumn gridColumn,
GridViewModel viewModel) => new GridColumnModel
        {
            GridColumn = gridColumn,
            UserColumn = viewModel.GridUserSettings
                .GetColumnsForCurrentPresentation()
                .FirstOrDefault(x => x.Name == gridColumn.Name)
        };
    }
```

```
}
```

① Описание класса, реализующего свой экспорт в Excel.

② Изменяем цвет нечётных строк.

③ Модифицирует документ Excel.

④ Меняем фоновый цвет нечётных строк в таблице.

⑤ Получаем строковое значение ячейки таблицы.

⑥ Для ячейки типа `DateTime` устанавливаем длинный формат.

⑦ Получаем все колонки из грида и сортируем их по параметру `Order`.

В методе `InializeContainer` серверного расширения зарегистрируйте сервис.

```
        public override void InitializeContainer(ContainerBuilder containerBuilder) ①
        {
            containerBuilder.RegisterType<ExcelExportExampleService>()
                .As<IExcelExportService>()
                .SingleInstance(); ②
        }
```

① Регистрируем типы в IoC контейнере.

② Регистрируем тип `ExcelExportExampleService` как реализацию интерфейса `IExcelExportService` в `containerBuilder`.

Проверка примера

1. Откройте папку в Web-клиенте Docsvision.
2. Нажмите кнопку **Экспортировать в Excel**.
3. Загрузится документ в формате `.xlsx`, содержащий выгрузку представления со всеми колонками и выделенными цветом нечетными строками.

Создание плагина для наполнения контрола табличных данных

ЭУ /dv6/webclient/6.1/layouts/ctrl/table/dataGridControl/[Контрол табличных данных] отображает данные в виде таблицы, наполняемой при помощи плагина. На данной странице описан способ создания плагина.

Ссылка на пример на GitHub: DataGridControlExtension.

`Название плагина` указывается в конструкторе разметок, в свойствах элемента управления. При открытии разметки карточки в Web-клиенте из настроек ЭУ

будет считано `Название плагина`. Затем отправляется запрос на сервер, который обнаруживает нужную реализацию интерфейса и возвращает модель для отображения таблицы.

Чтобы реализовать собственный плагин:

1. Выполните наследование от интерфейса `IDataGridControlPlugin (DocsVision.Layout.WebClient.Services)`.

   Наименование плагина:

   ```
   string Name { get; }
   ```

   Метод формирования модели таблицы:

   ```
   TableModel GetTableData(SessionContext sessionContext, List<ParamModel> parameters);
   ```

2. Затем зарегистрируйте тип в IoC контейнере:

   ```
   public override void InitializeContainer(ContainerBuilder containerBuilder) {

   containerBuilder.RegisterType<TestTablePlugin>().As<IDataGridControlPlugin>().SingleInstance();

   }
   ```

В случае, если это разметка карточки, то в массив параметров будет добавлен её идентификатор (`CurrentCardId`).

Создание карточки при помощи скрипта

Данный раздел содержит описание примера создания карточки скриптом с заполнением атрибутов.

Ссылка на пример на GitHub: CreateCard.

Пример рассчитан на версию Web-клиента dv6 или выше.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Сборка и установка

1. Откройте /Samples.sln.
2. Соберите проект ServerExtensions > CreateCard > CreateCardServerExtension.
3. Откройте консоль в папке ServerExtensions > CreateCard > CreateCardWebExtension и выполнить команду npm install, затем npm update и в конце npm run build:prod.
4. Скопируйте каталог SamplesOutput\Content\Modules\CreateCardWebExtension в каталог Путь к сайту Web-клиента\Content\Modules.
5. Скопируйте каталог SamplesOutput\Extensions\CreateCardServerExtension в каталог Путь к сайту Web-клиента\Extensions.
6. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Скопируйте любую разметку просмотра.
3. Выбрать условия использования этой разметки.
4. Откройте разметку и добавить в нее элемент управления Кнопка.
5. На событие onClick задать функцию createOutgoingDocument.
6. Сохраните разметку.
7. Перезапустите **dvwebclient**.
8. Откройте карточку с этой разметкой.
9. Убедитесь, что появился новый элемент управления — кнопка, при нажатии на которую создается исходящий документ.
10. Должен открыться исходящий документ, с заданными атрибутами.

Проект "CreateCardServerExtension"

1. Проект-расширение для Web-клиент. Содержит бизнес-логику и скрипт для создания новой карточки.

```
namespace CreateCardServerExtension
{
    public class LayoutWebClientExtension : WebClientExtension ①
    {
        public LayoutWebClientExtension(IServiceProvider serviceProvider) ②
            : base()
        {
```

```
        }

        public override string ExtensionName ③
        {
            get { return Assembly.GetAssembly(typeof(LayoutWebClientExtension
)).GetName().Name; }
        }

        public override Version ExtensionVersion ④
        {
            get { return new Version(FileVersionInfo.GetVersionInfo(Assembly
.GetExecutingAssembly().Location).FileVersion); }
        }

        #region WebClientExtension Overrides

        public override void InitializeServiceCollection(IServiceCollection services)
⑤
        {
            services.AddSingleton<ISampleDocumentService, SampleDocumentService>();
        }

        #endregion
    }
}
```

① Задаёт описание расширения для Web-клиента, которое задано в текущей сборке.

② Создаёт новый экземпляр `LayoutWebClientExtension`. `serviceProvider` — сервис-провайдер.

③ Получить название расширения.

④ Получить версию расширения.

⑤ Регистрация типов в IoC контейнере.
Параметры:

- `containerBuilder` — сборщик контейнеров.

2. Демонстрирует расширение функционала с помощью добавления новых сервисов, контроллеров.
3. Реализован контроллер `SampleDocumentController` с методом `CreateOutgoingDocument`, который вызывает сервис `ISampleDocumentService`, для создания исходящего документа и заполнения его полей:

- "Исходящий.Дата создания" = текущая дата.
- "Исходящий.Регистратор" = текущий пользователь.
- "Исходящий.Автор" = текущий пользователь.
- "Исходящий.Организация-Контрагет" = Входящий.Организация-Отправитель.
- "Исходящий.Название" = "В ответ на " + <Входящий.Исходящий номер>.
- В ссылки Исходящего и исходного Входящего добавить ссылку друг на друга, тип "ответ" — "в ответ на".

```
namespace CreateCardServerExtension.Controllers
{
    public class SampleDocumentController : Controller
    {
        private readonly ICurrentObjectContextProvider currentObjectContextProvider;
        private readonly ISampleDocumentService sampleDocumentService;

        public SampleDocumentController(ICurrentObjectContextProvider
currentObjectContextProvider, ISampleDocumentService sampleDocumentService) ①
        {
            this.currentObjectContextProvider = currentObjectContextProvider;
            this.sampleDocumentService = sampleDocumentService;
        }

        public ActionResult CreateOutgoingDocument(Guid parentDocId) ②
        {
            var sessionContext = this.currentObjectContextProvider
.GetOrCreateCurrentSessionContext();
            var response = sampleDocumentService.CreateOutgoingDocument(
sessionContext, parentDocId);

            return Content(JsonHelper.SerializeToJson(response), "application/json");
        }
    }
}
```

① Создаёт новый экземпляр, см. `SampleDocumentController`.

② GET: `/LayoutCreateDocumentController/SendToAcquaintance`

Проект "CreateCardWebExtension"

Содержит клиентские скрипты, в которых при нажатии на кнопку с помощью сервиса `requestManager` отправляется запрос на сервер. Пользователю

отображается созданная карточка Исходящего документа в режиме просмотра.

Создание карточки пользовательского типа

Данный раздел содержит описание примера добавления в диалог создания карточки пользовательского типа.

Ссылка на пример на GitHub: CreateCardDialog.

Пример рассчитан на версию Web-клиента dv6 или выше.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Сборка и установка

1. Открыть `/Samples.sln`
2. Собрать проект `ServerExtensions > CreateCardDialog > CreateCardDialogServerExtension`
3. Открыть консоль в папке `ServerExtensions > CreateCardDialog > CreateCardDialogWebExtension` и выполнить команду `npm install`, затем `npm update` и в конце `npm run build:prod`
4. Скопировать каталог `SamplesOutput\Extensions\CreateCardDialogServerExtension` в каталог `Путь к сайту Web-клиента\Extensions`
5. Скопировать файл `SamplesOutput\Extensions\ru\CreateCardDialogServerExtension.resources.dll` в каталог `Путь к сайту Web-клиента\Extensions\ru`
6. Скопировать каталог `SamplesOutput\Content\Modules\CreateCardDialogWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`
7. Перезапустите **dvwebclient**

Проверка примера

1. Запустить конструктор разметок.
2. Выбрать библиотеку `ApprovalDesigner карточки`.
3. Выбрать тип *Этап согласования*.

4. Создать разметку создания и просмотра (Например, добавить элемент управления `Строка` и привязать его к свойству `MainInfo\Name`).
5. Выбрать условия использования этих разметок.
6. В конфигурационном файле Web-клиента `appsettings.json` в секции `Docsvision > Platform > CardTypes` добавить строку:

```
{
    {

        "CardTypeId": "0DB13C90-21B6-49D8-9070-8144DF97552A", ①
        "CssClass": "approval-stage"
    }
}
```

① `0DB13C90-21B6-49D8-9070-8144DF97552A` — идентификатор типа карточки этапа согласования.

7. Разрешить создание карточек вида *Этап согласования* в справочнике видов карточек.
8. Перезапустите **dvwebclient**.
9. Открыть Web-клиент и нажать на кнопку **Создать**.
10. Должен появиться новый тип карточки *Этап*, доступный для создания (фон — фиолетовый).
11. При выборе его, открывается созданная разметка, при этом фон боковой панели и шапки становится фиолетовым.

Проект "CreateCardDialogServerExtension"

Проект реализует серверное расширение, в котором регистрируется фабрика для создания типа карточки *Этап*, отличная от стандартной.

Проект "CreateCardDialogWebExtension"

Содержит клиентские скрипты реализующие веб-расширение, которое регистрирует тип карточки для его корректного отображения в Web-клиенте.

Примечания

В базовом варианте, чтобы подключить к Web-клиенту карточку нового типа, достаточно внести изменения в конфигурационный файл Web-клиента, как описано в инструкции проверки примера, а так же описать стиль класса

approval-stage-color, например как:

```
.approval-stage-color {
  background-color: black;
}
```

Этот стиль применяется к элементам добавляемого типа карточки в диалоге создания карточки.

Если необходимо изменять цвет основных элементов управления Web-клиента, например, основной кнопки, шапки, боковых панелей, следует реализовать стили по примеру, описанному в approval-stage.scss.

Локализованные наименования всех зарегистрированных типов карточек доступны на клиенте и содержатся в объекте resources. Наименование пользовательского типа доступно по ключу CardType_{CardTypeName}_DisplayName, где {CardTypeName} - наименование типа карточки в файле метаданных.

В собственном решении или элементе управления можно использовать цвета, связанные с каким-то типом карточки. Для этого следует воспользоваться системными CSS классами:

- card-type-background-color — стандартный фон элемента типа карточки.
- card-type-background-color-hover — стандартный фон элемента типа карточки при наведении мыши.
- card-type-background-color-light — светлый фон элемента типа карточки.
- card-type-background-color-light-hover — светлый фон элемента типа карточки при наведении мыши.
- card-type-background-color-disabled — заблокированный фон элемента типа карточки, например, когда кнопка недоступна для нажатия.
- card-type-foreground-color — стандартный цвет шрифта типа карточки.

Список может быть расширен, при необходимости обратитесь к документации разработчика.

Если определенная бизнес-логика реализована или необходимо её реализовать, например если требуется задать значение какого-то поля по умолчанию при создании карточки, можно подключить серверное расширение (см. LayoutWebClientExtension.cs) и реализовать в нем интерфейс ICardFactory, в

котором реализуется собственная логика по созданию карточки или созданию карточки по шаблону.

Фрагмент файла "LayoutWebClientExtension.cs"

```
namespace CreateCardDialogServerExtension
{

    public class LayoutWebClientExtension : WebClientExtension ①
    {

        public LayoutWebClientExtension(IServiceProvider serviceProvider) ②
            : base()
        {
        }

        public override string ExtensionName ③
        {
            get { return Assembly.GetAssembly(typeof(LayoutWebClientExtension)).
GetName().Name; }
        }

        public override Version ExtensionVersion ④
        {
            get { return new Version(FileVersionInfo.GetVersionInfo(Assembly
.GetExecutingAssembly().Location).FileVersion); }
        }


        #region WebClientExtension Overrides

        public override void InitializeServiceCollection(IServiceCollection services) ⑤
        {
            services.AddTransient<ICardLifeCycleEx, ApprovalStageCardLifeCycle>();
        }

        protected override List<ResourceManager> GetLayoutExtensionResourceManagers() ⑥
        {
            return new List<ResourceManager>
            {
                { Resources.ResourceManager }
            };
        }

        #endregion
    }
}
```

① Задаёт описание расширения для Web-клиента, которое задано в текущей сборке.

② Создаёт новый экземпляр `LayoutWebClientExtension`.

Параметры:

- `serviceProvider` — сервис-провайдер.

③ Получить название расширения.

④ Получить версию расширения.

⑤ Регистрация типов в IoC-контейнере.

Параметры:

- `containerBuilder` — сборщик контейнеров.

⑥ Получает менеджеры ресурсов для расширения разметки.

Добавление условия выбора разметок

Данный раздел содержит описание примера серверного расширения. Пример демонстрирует добавление условия выбора разметок по группе, пользователю, роли.

Ссылка на пример на GitHub: CustomConditionTypes.

После сохранения изменений в конструкторе разметок, разметка отображается в соответствии с заданными условиями.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Сборка и установка

1. Откройте `/Samples.sln`.
2. Соберите проект `ServerExtensions > CustomConditionTypes > CustomConditionTypesDesignerExtension`.
3. Соберите проект `ServerExtensions > CustomConditionTypes > CustomConditionTypesLayoutExtension`.
4. Соберите проект `ServerExtensions > CustomConditionTypes > CustomConditionTypesInterfaces`.

5. Скопируйте каталог `SamplesOutput\Plugins\CustomConditionTypesDesignerExtension` в каталог `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins`.
6. Скопируйте каталог `SamplesOutput\Plugins\CustomConditionTypesInterfaces` в каталог `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins`.
7. Скопируйте каталог `SamplesOutput\Extensions\CustomConditionTypesInterfaces` в каталог `Путь к сайту Web-клиента\Extensions`.
8. Скопируйте каталог `SamplesOutput\Extensions\CustomConditionTypesLayoutExtension` в каталог `Путь к сайту Web-клиента\Extensions`.
9. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите Конструктор Web-разметок.
2. Выберите или создайте любую разметку в локации **WebFrame › MainMenu**.
3. Откройте вкладку условия использования разметки.
4. Добавьте новое условие выбора или измените существующее.
5. Измените в условии выбора роль, группу, пользователя или все сразу.
6. Запустите Web-клиент.
7. Заданная разметка отображается только если выполняются условия выбора разметки.

Проект "CustomConditionTypesWebLayoutsDesignerExtension"

Проект-расширение для конструктора разметок. Содержит класс `CustomConditionTypesProvider`, в котором реализованы методы для отображения новых условий выбора разметки по группе, пользователю, роли.

```
    [SupportedOSPlatform("windows")]
    internal class CustomConditionTypesProvider : IConditionTypesProvider
    {
        public Guid Id => Constants.ExtensionId;

        public List<Staff> GetStaffTree() ①
        {
            var staffTree = new List<Staff>();

            var sessionContext = this.SessionContextProvider
.GetOrCreateCurrentSessionContext();
```

```
            var staffService = sessionContext.ObjectContext.GetService<IStaffService>();
            var units = staffService.GetUnits(null, true).OrderBy(u => u.Name);

            foreach (var unit in units)
            {
                var nodeStaff = unit.Staff;
                staffTree.Add(nodeStaff);
            }

            return staffTree;
        }

        public List<StaffEmployee> GetUsersTree() ②
        {
            var usersTree = new List<StaffEmployee>();

            var sessionContext = this.SessionContextProvider
.GetOrCreateCurrentSessionContext();
            var staffService = sessionContext.ObjectContext.GetService<IStaffService>();
            var units = staffService.GetUnits(null, true).OrderBy(u => u.Name);
            foreach (var unit in units)
            {
                foreach (var employee in unit.Employees)
                    usersTree.Add(employee);
            }

            return usersTree;
        }

        private readonly IServiceProvider serviceProvider;
        private ICurrentObjectContextProvider sessionContextProvider;

        public CustomConditionTypesProvider(IServiceProvider serviceProvider)
        {
            this.serviceProvider = serviceProvider;
        }

        private ICurrentObjectContextProvider SessionContextProvider
        {
            get
            {
                if (this.sessionContextProvider == null)
                    this.sessionContextProvider = DocsVision.Platform.Tools.LayoutEditor
.ServiceUtil.GetService<ICurrentObjectContextProvider>(this.serviceProvider);
                return this.sessionContextProvider;
            }
        }
```

① Получение списка штатов сотрудников.

② Получение списка сотрудников.

Проект "CustomConditionTypesLayoutExtension"

Проект-расширение для Web-клиента, которое позволяет определить, выполняется условие выбора разметки или нет для текущего контекста.

Фрагмент "CustomConditionTypesLayoutExtension.cs"

```
    public class CustomConditionTypesLayoutExtension : ILayoutExtension
    {
        public CustomConditionTypesLayoutExtension()
        {
            this.LayoutExtensionDescription = new LayoutExtensionDescription
            {
                Id = Constants.ExtensionId,
                Name = "Custom condition types layout extension",
                Version = FileVersionInfo.GetVersionInfo(Assembly.GetExecutingAssembly
().Location).FileVersion,
                ConditionTypes = this.GetConditionTypes(),
                ModeConditionTypes = this.GetLayoutTypes()
            };
        }
```

При внесении изменений необходимо увеличивать номер версии расширения в `AssemblyFileVersion`.

Проект "CustomConditionTypesInterfaces"

Проект-расширение для конструктора разметок и серверного расширения, содержащий все константные значения для этих проектов.

Фрагмент "Constants.cs"

```
    public static class Constants
    {
        public static readonly Guid ExtensionId = new Guid("7BDD6F65-B6F2-414F-B97F-
13E87E972F85");
        public static readonly Guid LocationExtensionId = new Guid("7e794753-46ed-42f3-
85b2-2e172c00d045");

        public static class ConditionTypes
        {
            public static readonly Guid GroupsConditionType = new Guid("AC83CC41-CB31-
4804-B330-A180FA02BA9D");
```

```
            public static readonly Guid UnitConditionType = new Guid("103B4D34-594A-479B-
BE72-D481677C5136");
            public static readonly Guid RolesConditionType = new Guid("986E19CD-6101-
4979-A99B-77CA50C0D781");
        }

        public static class MainMenu
        {
            public const string Id = "DV.MainMenu";
        }
    }
```

Взаимодействие с пользовательским типом карточки

Данный раздел содержит описание примера серверного расширения. Пример демонстрирует взаимодействие с пользовательским типом карточки Docsvision.

Ссылка на пример на GitHub: CustomLibrary.

В библиотеке `Sample custom library` реализован тип карточки `Custom Directory`, который представляет справочник с полем `Counter`. Значение поля получается скриптом и выводится во всплывающем окне.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.
- Docsvision Resource Kit
- Docsvision DVExplorer

Сборка и установка

1. Откройте `/Samples.sln`.
2. Соберите проекты `ServerExtensions > CustomLibrary > CustomLibrary.ObjectModel` и `CustomLibraryServerExtension`.
3. Откройте консоль в папке `ServerExtensions > CustomLibrary > CustomLibraryWebExtension` и выполнить команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
4. Скопируйте каталог `SamplesOutput\Extensions\CustomLibraryServerExtension` в каталог `Путь к сайту Web-клиента\Extensions`.

5. Скопируйте каталог `SamplesOutput\Content\Modules\CustomLibraryWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.

Перед выполнением следующего шага, рекомендуется сделать резервную копию базы данных Docsvision.

6. На сервере Docsvision установите *Docsvision Resource Kit*, установить обновление *DVExplorer*. С помощью утилиты *DVCardManager* следует загрузить в БД Docsvision библиотеку `CustomCardLib` из каталога `SamplesOutput\SamplesCardDefs\CustomLibrary`. Подробное описание процедуры загрузки пользовательской библиотеки карточек содержится в руководстве разработчика Docsvision.
7. С помощью утилиты *DVExplorer* подключитесь к БД, откройте карточку справочника `Custom Directory`, добавьте строку секции `MainInfo` и установите значение `777` в поле `Counter`.

   Сохраните изменения. Подробное описание работы с утилитой *DVExplorer* содержится /dv6/resource-kit/6.1/dvexplorer/util/[в документации комплекта утилит Resource Kit].

8. Перезапустите **dvwebclient** на сервере Docsvision.
9. Перезапустите **dvwebclient** на сервере Docsvision Web-клиент.

Проверка примера

1. Запустить конструктор разметок.
2. Скопировать любую разметку просмотра.
3. Выбрать условия использования этой разметки.
4. В разметке просмотра на событие `OnCardOpened` элемента `root` указать обработчик с названием `getCustomData`.
5. Сохранить разметку.
6. Перезапустите **dvwebclient**.
7. Создайте карточку документа.
8. При открытии разметки просмотра появится всплывающее сообщение: `Custom data: 777`.

Каталог "CardDefs"

Содержит описание библиотеки `Sample custom library`, в которой описан тип

карточки CustomDirectory — справочник. В секции MainInfo справочника есть поле Counter. Подробности содержатся в документации разработчика Docsvision.

Проект "CustomLibrary.ObjectModel"

Проект содержит объектную модель карточки типа CustomDirectory. Подробности содержатся в руководстве разработчика Docsvision.

```
namespace CustomLibrary.ObjectModel
{

    public class CustomDirectory : BaseCard ①
    {

        public static readonly ObjectProperty MainInfoProperty = ObjectProperty.Register
("MainInfo", typeof(ObjectCollection<MainInfo>), typeof(CustomDirectory)); ②

        [DebuggerHidden]
        static CustomDirectory()
        {
        }

        internal protected CustomDirectory() ③
            : base()
        {
        }

        internal protected CustomDirectory(ObjectInitializationData data) ④
            : base(data)
        {
        }

        #region Properties

        public MainInfo MainInfo ⑤
        {
            get
            {
                if (((ObjectCollection<MainInfo>)GetValue(MainInfoProperty)).Count == 0)
                    ((ObjectCollection<MainInfo>)GetValue(MainInfoProperty)).Add(new
MainInfo());

                return ((ObjectCollection<MainInfo>)GetValue(MainInfoProperty)).First();
            }
        }

        #endregion
```

```
    }
}
```

① Представляет собой пользовательский справочник.

② Настройки базы данных.

③ Создаёт новый экземпляр `CustomDirectory`.

④ Создаёт новый экземпляр `CustomDirectory`.

⑤ Главная информация.

Проект "CustomLibraryServerExtension"

Проект-расширение для Web-клиент. Содержит сервис по работе со справочником `CustomDirectory`, методы контроллера `CustomLibrary` для взаимодействия с клиентскими скриптами веб-приложения.

```
namespace CustomLibraryServerExtension.Services
{
    public class CustomLibraryService : ICustomLibraryService ①
    {
        public CustomLibraryService(IServiceProvider provider) ②
        {
        }
        public int GetCustomData(SessionContext sessionContext) ③
        {
            var settingsDirectory = sessionContext.ObjectContext.GetObject
<CustomDirectory>(CustomLibrary.CustomLibrary.CardLib.CardDefs.CustomDirectory.ID);

            int count = settingsDirectory.MainInfo.Counter;

            return count;
        }
    }
}
```

① Представляет собой пример сервиса, использующего кастомную библиотеку

② Создаёт новый экземпляр <see cref="CustomLibraryService"/>

③ Получить кастомные данные

Проект "CustomLibraryWebExtension"

Содержит клиентский скрипт `getCustomData`.

Информация о расширенной карточке

Данный раздел содержит описание примера разработки и подключения собственного серверного расширения для чтения значения полей, загруженных и не загруженных на страницу.

Ссылка на пример на GitHub: ExtendedCardInfo.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Сборка и установка

1. Откройте `/Samples.sln`.
2. Собрать проект `ServerExtensions > ExtendedCardInfo > ExtendedCardInfoServerExtension`.
3. Откройте консоль в папке `ServerExtensions > ExtendedCardInfo > ExtendedCardInfoWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
4. Скопировать каталог `SamplesOutput\Extensions\ExtendedCardInfoServerExtension` в каталог `Путь к сайту Web-клиента\Extensions`.
5. Скопируйте файл `SamplesOutput\Extensions\ru\ExtendedCardInfoServerExtension.resources.dll` в каталог `Путь к сайту Web-клиента\Extensions\ru`.
6. Скопируйте файл `SamplesOutput\Extensions\uk\ExtendedCardInfoServerExtension.resources.dll` в каталог `Путь к сайту Web-клиента\Extensions\uk`.
7. Скопируйте каталог `SamplesOutput\Content\Modules\ExtendedCardInfoWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
8. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Скопируйте любую разметку просмотра.

3. Выберите условия использования этой разметки.
4. Откройте разметку и для ЭУ `RegDate` на событие **`onChanged`** привязать функцию `extendedCardCheckDates`.

Для разных видов карточки название может варьироваться, например, `RegistrationDate`.

5. Сохраните разметку.
6. Перезапустите **dvwebclient**.
7. Откройте карточку с этой разметкой.
8. Убедитесь, что при изменении даты появляется сообщение.

Проект "ExtendedCardInfoServerExtension"

Проект-расширение для Web-клиента. Содержит бизнес-логику для чтения значения полей, загруженных и не загруженных на страницу.

Демонстрирует расширение функционала с помощью добавления новых сервисов, контроллеров. Реализован контроллер `ExtendedCardController` с методом `Get`, который вызывает сервис `IExtendedCardService` для чтения не загружаемых на страницу полей документа:

- `CreateDate`
- `ChangeDate`
- `Description`
- `BarCode`

Проект "ExtendedCardInfoWebExtension"

Проект-расширение клиентской части Web-клиента. Содержит клиентский скрипт функции `extendedCardCheckDates`, которая вызывается на событие **`onChanged`** ЭУ `RegDate`. С помощью `requestManager.get` отправляем запрос на сервер для получения расширенной модели `IExtendedCardModel` и сравниваем поля.

Проверка опции лицензии

Данный раздел содержит описание примера проверки наличия дополнительной опции лицензионного ключа.

Ссылка на пример на GitHub: LicenseCheck.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Сборка и установка

1. Откройте `/Samples.sln`
2. Соберите проект `ServerExtensions > LicenseCheck > LicenseCheckServerExtension`
3. Откройте консоль в папке `ServerExtensions > LicenseCheck > LicenseCheckWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
4. Скопируйте каталог `SamplesOutput\Content\Modules\LicenseCheckWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
5. Скопируйте каталог `SamplesOutput\Extensions\LicenseCheckServerExtension` в каталог `Путь к сайту Web-клиента\Extensions`.
6. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Скопируйте любую разметку просмотра.
3. Выберите условия использования этой разметки.
4. Откройте разметку и добавить в нее элемент управления `Кнопка`.
5. На событие `onClick` задать функцию `checkLicenseFeature`.
6. Сохраните разметку.
7. Перезапустите **dvwebclient**.
8. Откройте карточку с этой разметкой.
9. Убедитесь, что появился новый элемент управления (кнопка).
10. Нажмите на кнопку. Должно появиться сообщение с результатом проверки.

Проект "LicenseCheckServerExtension"

Проект-расширение для Web-клиента. Содержит бизнес-логику и скрипт для проверки наличия дополнительной опции лицензионного ключа. В данном примере это опция `Docsvision Card Builder`. Демонстрирует расширение

функционала с помощью добавления новых сервисов, контроллеров. Реализован контроллер `LicenseCheckController` с методом `CheckFeature`, который вызывает сервис `ILicenseCheckService`.

Проект "LicenseCheckWebExtension"

Содержит скрипт, который при нажатии на кнопку с помощью сервиса `requestManager` отправляется запрос на сервер. Пользователю отображается сообщение о наличии или отсутствии опции.

Автоматическое изменение срока исполнения заданий

Данный раздел содержит описание примера серверного расширения. Пример демонстрирует реализацию автоматических изменений данных связанных карточек.

Ссылка на пример на GitHub: ShiftTasksEndDate.

После сохранения карточки Документа ищутся все его связанные задания, и срок исполнения для отправленных на исполнение заданий переносится на 3 дня вперед.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Сборка и установка

1. Откройте `/Samples.sln`.
2. Соберите проект `ServerExtensions > ShiftTasksEndDate > ShiftTasksEndDateServerExtension`.
3. Откройте консоль в папке `ServerExtensions > ShiftTasksEndDate > ShiftTasksEndDateWebExtension` и выполнить команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
4. Скопируйте каталог `SamplesOutput\Extensions\ShiftTasksEndDateServerExtension` в каталог `Путь к сайту Web-клиента\Extensions`.
5. Скопируйте файл `SamplesOutput\Extensions\ru\ShiftTasksEndDateServerExtension.resources.dll` в каталог “Путь к сайту Web-клиента\Extensions\ru”.

6. Скопируйте файл `SamplesOutput\Extensions\uk\ShiftTasksEndDateServerExtension.resources.dll` в каталог `Путь к сайту Web-клиента\Extensions\uk`.
7. Скопируйте каталог`SamplesOutput\Content\Modules\ShiftTasksEndDateWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
8. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Скопируйте любую разметку редактирования.
3. Выберите условия использования этой разметки.
4. В разметке редактирования на событие `OnCardSaved` элемента `root` прикрепляем обработчик с названием `shiftTasksEndDate`.
5. Сохраните разметку.
6. Перезапустите **dvwebclient**.
7. Создайте карточку документа.
8. Добавьте дочерние задания (не группы заданий), отправить на исполнение.
9. Нажмите на редактирование карточки, изменить что-нибудь, сохранить.
10. Срок исполнения для отправленных на исполнение заданий сдвинулся на 3 дня.

Проект "ShiftTasksEndDateServerExtension"

Проект-расширение для Web-клиента. Содержит бизнес-логику изменения данных связанных карточек. Демонстрирует расширение функционала с помощью добавления новых сервисов, контроллеров. Реализован контроллер `AdvancedDocumentController` с методом `ShiftTasksEndDate`, который вызывает сервис `IShiftTasksEndDateService`.

Проект "ShiftTasksEndDateWebExtension"

Проект-расширение клиентской части Web-клиента. Содержит клиентский скрипт функции `shiftTasksEndDate`, которая вызывается на событие `OnCardSaved`. С помощью `requestManager.post` отправляем запрос на сервер для изменения даты.

Отображение данных из связанных карточек

Данный раздел содержит описание примера серверного расширения. Пример содержит серверное расширение для отображения данных из связанных карточек в табличном контроле. В разметку карточки *Документ УД/Исходящий* добавляется таблица контрагентов, в колонках которой автоматически показываются e-mail и телефон выбранной в строке организации, загружаемые с сервера отдельным запросом.

Ссылка на пример на GitHub: TableControl.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Сборка и установка

1. Откройте `/Samples.sln`.
2. Соберите проект `ServerExtensions > TableControl > TableControlServerExtension`.
3. Откройте консоль в папке `ServerExtensions > TableControl > TableControlWebExtension` и выполнить команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
4. Скопируйте каталог `SamplesOutput\Content\Modules\TableControlWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
5. Скопируйте каталог `SamplesOutput\Extensions\TableControlServerExtension` в каталог `Путь к сайту Web-клиента\Extensions`.
6. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Скопируйте разметку просмотра исходящего документа.
3. Выберите условия использования этой разметки.
4. Откройте разметку и добавить в нее элемент управления `Таблица` в режиме *Редактирование по месту* и 3 столбца. Для таблицы в поле *Источник данных* секцию `Partners Receivers`.

5. В первый столбец поместите элемент управления `Подразделение котрагента` в режиме *Редактирование по месту*, задайте имя `samplePartnersDepartment`. Значение *Поле данных* выберите `Partner company`.
6. Во второй столбец поместить элемент управления `Метка`, задайте имя `sampleDepartmentEmail`.
7. В третий столбец поместить элемент управления `Метка`, задайте имя `sampleDepartmentPhone`.
8. На событие `После открытия карточки` элемента `root` задайте функцию `sampleDocumentViewCardOpened`.
9. Сохранить разметку.
10. Перезапустите **dvwebclient**.
11. Откройте карточку с этой разметкой, добавить строку в таблицу, выбрать контрагента, обновить страницу.
12. Убедиться, что почта и телефон контрагента заполнились.

Проект "TableControlServerExtension"

Проект-расширение для Web-клиента. Содержит бизнес-логику и скрипт для отображения данных из связанных карточек в табличном контроле. Демонстрирует расширение функционала с помощью добавления новых сервисов, контроллеров. Реализован контроллер `SamplePartnersController` с методом `GetPartnersInfo`, который вызывает сервис `ISamplePartnersService`, для получения информации о контрагентах: `Phone`, `Email`, `Name`.

Проект "TableControlWebExtension"

Содержит клиентский скрипт с функцией `sampleDocumentViewCardOpened`, которая вызывается на событие `После открытия карточки`. Функция отправляет запрос на сервер `getPartnersInfo` и заполняет таблицу вернувшимися данными.

Расширение возможностей программы Конструктор Web-разметок

Конструктор Web-разметок предоставляет возможность дополнять собственные функциональные возможности с помощью расширений.

Ссылка на пример на GitHub: Docsvision Web-layout designer extension template.

Альтернативно расширение можно получит

Расширение программы Конструктор Web-разметок может добавлять:

- Типы элементов управления.
- Свойства (настройки) элементов управления.
- Редакторы для свойств элементов управления.
- Функции проверки доступности операций.
- Критерии для условий выбора разметок.
- Локализованные ресурсы.

Разработка расширения программы Конструктор Web-разметок

Процесс разработки расширения программы Конструктор Web-разметок может включать следующие шаги:

1. Создание в Visual Studio[8] проекта типа TemplateDesignerExtension.
2. Обновление ссылок на подключаемые компоненты (при необходимости).

Версии подключаемых компонентов в проекте должны соответствовать версиям компонентов, используемых Конструктором web-разметок, для которого разрабатывается расширение. При обновлении версии Web-клиента все разработанные компоненты должны быть скомпилированы повторно с использованием новых версий библиотек Web-клиента.

3. Разработка ядра расширения программы Конструктор Web-разметок.

   Ядро расширения программы Конструктор Web-разметок представляет собой производный от `WebLayoutsDesignerExtension` класс. В нём переопределяются методы получения описателей типов элементов управления, описателей типов свойств элементов управления, редакторов значений свойств и пр., в зависимости от назначения расширения программы Конструктор Web-разметок.

   При создании проекта из шаблона `TemplateDesignerExtension` ядро расширения программы Конструктор Web-разметок будет содержаться в файле `Extension\DesignerExtension.cs`.

   Примеры реализации отдельных элементов ядра расширения программы Конструктор Web-разметок рассмотрены в следующих разделах.

4. Компиляция проекта.

> В результате будет получен файл DLL — расширение программы Конструктор Web-разметок, а также файлы DLL с локализованными ресурсами.

Публикация расширения программы Конструктор Web-разметок

Последовательность публикации расширения программы Конструктор Web-разметок на сервере Web-клиента включает следующие шаги:

1. Компиляция проекта.

   > В результате будет получен файл DLL с ядром расширения программы Конструктор Web-разметок и реализуемыми функциями, а также файлы DLL с локализованными ресурсами.

2. Создание каталога для Решения на сервере Web-клиента: `Каталог-установки-Конструктора-Web-Разметок/Plugins/%Каталог Решения%`.
3. Копирование сборки расширения программы Конструктор Web-разметок в `%Каталог Решения%`. Файлы DLL с локализованными ресурсами копируются в существующие в каталоге `/ru/` (для русской локализации), `/uk/` (для английской локализации) и т.д.

   > Реализованная в расширении программы Конструктор Web-разметок функциональность будет доступна после запуска программы Конструктор Web-разметок.

4. Перезапуск Конструктор Web-разметок.

Создание и публикация расширения программы Конструктор Web-разметок

Серверное расширение представляет собой библиотеку DLL, разработанную на платформе .NET.

Ссылка на шаблон на GitHub: TemplateDesignerExtension.

Чтобы создать расширение программы Конструктор Web-разметок:

1. Создайте в Visual Studio[9] новый проект с типом TemplateDesignerExtension.
2. Обновите в проекте ссылки на зависимости.

После крупного обновления версии Docsvision, необходимо повторно скомпилировать расширение с новыми версиями компонентов. В остальных случаях такой необходимости нет.

Это необходимо, чтобы версии подключаемых компонентов соответствовали версиям компонентов, используемых сервером Web-клиента, для которого разрабатывается расширение.

3. Реализуйте в проекте функции расширения программы Конструктор Web-разметок.

Примеры кода расширений программы Конструктор Web-разметок приведены в следующих пунктах раздела, а также в репозитории "ДоксВижн".

4. Настройте ядро расширения.

Ядром расширения является класс, производный от `WebLayoutsDesignerExtension`, который содержится в файле `Extension/DesignerExtension.cs`.

```
class TemplateDesignerExtension : WebLayoutsDesignerExtension
{
public TemplateDesignerExtension(IServiceProvider provider)
: base(provider)
{
}
  protected override List<ControlTypeDescription> GetControlTypeDescriptions() ①
  {
      return new List<ControlTypeDescription>
      {
      };
  }

  protected override Dictionary<string, PropertyDescription>
GetPropertyDescriptions() ②
  {
      return new Dictionary<string, PropertyDescription>
      {
      };
  }

  protected override List<ResourceManager> GetResourceManagers() ③
  {
```

```
        return new List<ResourceManager>
        {
          Resources.ResourceManager
        };
    }
}
```

① Метод регистрации новых типов элементов управления.

② Метод регистрации новых свойств элементов управления.

③ Метод регистрации локализованных ресурсов.

Добавление нового элемента управления

Добавить новый тип элементов управления в Конструктор Web-разметок можно с помощью расширения.

Расширение предоставляет возможность настраивать разметку с использованием нового элемента управления, но не обеспечивает его функционирование на клиентской стороне.

Полностью вопрос разработки новых типов элементов управления будет рассмотрен в разделе Разработка элементов управления.

Основным способом описания элемента управления является текстовый или XML. Подробнее о создании текстового описателя элемента управления см. Формирование текстового описателя элемента управления.

Добавить элемент управления

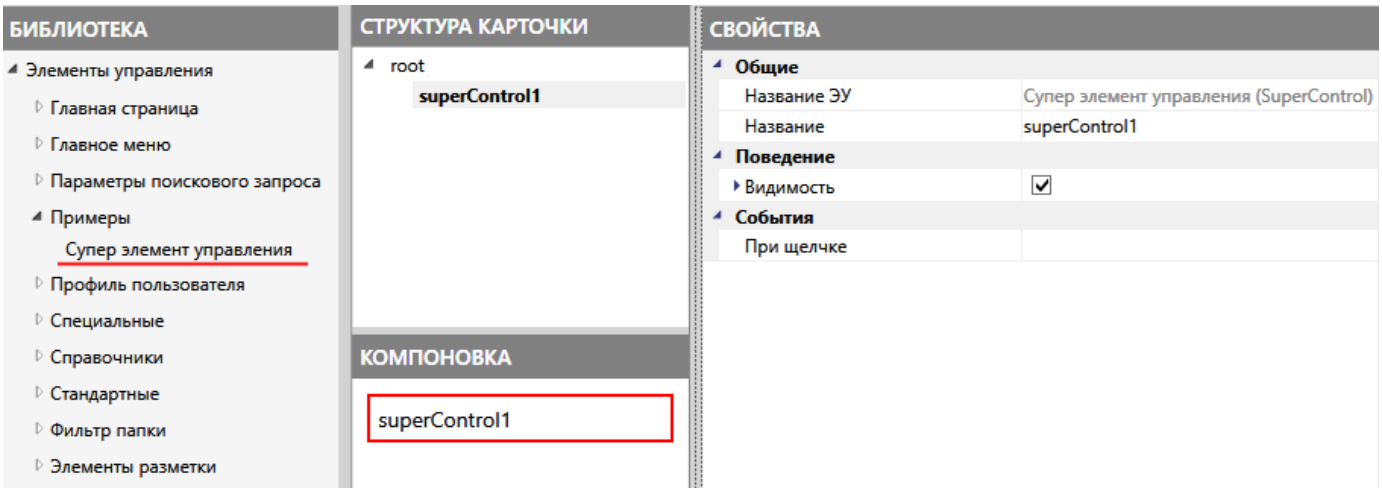


Рисунок 111. Элемент управления из расширения в редакторе разметок

Чтобы создать расширение, добавляющее новый тип элементов управления, выполните приведенную далее инструкцию.

1. Создайте проект расширения программы Конструктор Web-разметок.
2. Добавьте в ядро расширения (файл DesignerExtension.cs) функцию, предоставляющую *описатель элемента управления*.

   Описатель элемента управления — это объект, содержащий информацию об элементе управления: название типа, группа, перечень свойства (настроек). Описатель имеет тип ControlTypeDescription.

```
ControlTypeDescription GetSuperControlTypeDescription ()
{
  var controlTypeDescription = new ControlTypeDescription ("SuperControl") {
      DisplayName = "Супер элемент управления",
      ControlGroupDisplayName = "Примеры",
      PropertyDescriptions = {
          PropertyFactory.GetNameProperty(), ①
          PropertyFactory.GetVisibilityProperty(), ②
          PropertyFactory.GetClickEvent() ③
      }
  };
  return controlTypeDescription;
}
```

   ① Название.

   ② Видимость.

   ③ Событие При щелчке.

   При составлении описания элемента необходимо:

   i. Указать название типа элемента управления. Название передается непосредственно в конструктор класса ControlTypeDescription.

   Описатель элемента управления с типом "SuperControl"

```
var controlTypeDescription = new ControlTypeDescription ("SuperControl"){}
```

   ii. Указать отображаемое название типа элемента управления в DisplayName. Данное название приводится в строке Название ЭУ в блоке настроек элемента управления в программы Конструктор Web-разметок.

```
DisplayName = "Супер элемент управления",
```

iii. Указать группу элемента управления в ControlGroupDisplayName. Группа используется для группировки элементов управления в библиотеке элементов управления программы Конструктор Web-разметок.

```
ControlGroupDisplayName = "Примеры",
```

3. Укажите свойства элемента управления в `PropertyDescriptions`.

> Свойства элемента управления — его параметры, которые определяют внешний вид, связь с данными карточки, функции, вызываемые при работе элемента управления (обработчики событий). К свойствам относятся: видимость, текстовая метка, подсказка, (событие) при получении фокуса и др. Значения свойств (есть исключения) передаются в клиент, где могут быть использованы для формирования пользовательского интерфейса.

4. Переопределите метод `GetControlTypeDescriptions()` базового класса, чтобы он возвращал коллекцию с реализованным в пункте 2 методом:

```
protected override List<ControlTypeDescription> GetControlTypeDescriptions()
{
    return new List<ControlTypeDescription>
    {
        GetSuperControlTypeDescription()
    };
}
```

Добавление свойства в Конструктор Web-разметок

Данный раздел содержит описание примера добавления свойства `Url` в конструктор разметок. Проект-расширение для конструктора разметок. Содержит добавление собственного свойства, используя класс `PropretyDescription` (см. свойство `Url`).

Ссылка на пример на GitHub: ControlProperties/Url.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Сборка и установка

1. Открыть `/Samples.sln`.
2. Собрать проект `ControlProperties > Url > UrlPropertyDesignerExtension`.
3. Скопировать каталог `SamplesOutput\Plugins\UrlPropertyDesignerExtension` в каталог “Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins”.

Проверка примера

Для проверки примера необходима реализация ЭУ, который использует свойство Url (xml-файл или расширение конструктора)

1. Реализуйте и подключите ЭУ, который использует свойство `Url` одним из доступных способов:
   - Через `xml`, см. `Controls\Link\SampleLinkControlDescription.xml`
   - Через расширение, см. `Controls > Image > ImageDesignerExtension`.
2. Запустите конструктор разметок.
3. Скопируйте любую разметку.
4. Выберите условия использования этой разметки.
5. Откройте разметку и добавить в нее новый элемент управления.
6. Убедитесь, что присутствует свойство `Url`.
7. Сохраните разметку.
8. Перезапустите **dvwebclient**.
9. Откройте карточку с этой разметкой.
10. Убедитесь в работоспособности нового элемента управления.

Добавление нового свойства элементов управления

Ссылка на GitHub: TemplateDesignerExtension.

Элементы управления обладают набором свойств, которые определяют их поведение в веб-браузере пользователя: внешний вид, получение данных (для

разметки карточки), обработка событий.

Когда программист формирует список свойств элемента управления нового типа, он может ссылаться только на существующие стандартные свойства (Стандартные свойства и события элементов управления). Если стандартные свойства не позволяют описать все особенности элемента управления, программист может сформировать новое свойство.

Добавить новое свойство элементов управления в Конструктор Web-разметок можно с помощью *Расширения*.

Добавление свойства

Чтобы создать расширение, добавляющее новый тип свойств элементов управления, выполните приведенную далее инструкцию.

1. Убедитесь, что установлен Docsvision WebClient SDK (запрашивается через портал ТП).

   В этом случае в переменной окружения `DocsvisionWebClientSDK` содержится путь к его папке, например, `C:\Program Files (x86)\Docsvision\WebClient\SDK`.

2. Создайте проект расширения программы Конструктор Web-разметок.

3. Добавьте в ядро расширения (файл `DesignerExtension.cs`) функцию, предоставляющую *описатель свойства* элемента управления.

   Описатель свойства элемента управления — это объект с типом `PropertyDescription`, содержащий информацию о свойстве: название свойства, тип, категория, редактор, который должен использоваться для изменения значения, возможность локализации и настройки значения для определённого типа устройств.

   ```
   PropertyDescription GetSuperPropertyDescription()
   {
       var propertyDescription = new PropertyDescription();
       propertyDescription.Name = "SuperProperty";
       propertyDescription.DisplayName = "Супер свойство";
       propertyDescription.Type = typeof(string); ①
       propertyDescription.Category = PropertyCategoryConstants.DataCategory; ②
       return propertyDescription;
   }
   ```

   ① Строковый тип свойства.

② Указывает категорию свойства (`DataCategory` — Данные).

При составлении описания элемента необходимо:

i. Указать название свойства в Name.

```
propertyDescription.Name = "SuperProperty";
```

ii. Указать отображаемое название свойства в DisplayName. С данным названием свойство отображается в списке свойств элемента управления в программы Конструктор Web-разметок.

```
+propertyDescription.DisplayName = "Супер свойство";
```

iii. Указать тип значения свойства в Type.

```
propertyDescription.Type = typeof(string); ①
```

① Строковый тип свойства.

iv. Указать категорию свойства в Category. Категория используется для группировки свойств в настройках элемента управления в программе Конструктор Web-разметок. Категория выбирается из перечня в классе `PropertyCategoryConstants`.

```
propertyDescription.Category = PropertyCategoryConstants.DataCategory; ①
```

① Указывает категорию свойства (`DataCategory` — Данные).

Обратитесь к описанию класса `PropertyDescription`, чтобы получить полный перечень возможностей описателя свойства элементов управления.

4. Переопределите метод `GetPropertyDescriptions` базового класса, чтобы он возвращал коллекцию с парой: ключ — имя. Возвращаемая коллекция позволяет получить описатель данного свойства. Используйте имя, указанное в `propertyDescription.Name` и метод, предоставляющий описатель свойства в качестве значения:

```
protected override Dictionary<string, PropertyDescription> GetPropertyDescriptions()
{
    return new Dictionary<string, PropertyDescription>
    {
        { "SuperProperty", GetSuperPropertyDescription()}
    };
}
```

Проверка свойства

Чтобы проверить пример:

1. Создайте новый тип элемента управления.
2. Добавьте **Супер свойство** в список свойств элемента управления (в `PropertyDescriptions`).

Для получения описателя свойства используйте метод `PropertyFactory.Create`, передав в него имя свойства, указанное при его регистрации в методе `GetPropertyDescriptions`.

```
PropertyDescriptions = {
    PropertyFactory.GetNameProperty(),
    PropertyFactory.GetVisibilityProperty(),
    PropertyFactory.Create("SuperProperty"), ①

    PropertyFactory.GetClickEvent() ②
}
```

① Получаем описатель свойства "SuperProperty".

② Событие "При щелчке".

3. Опубликуйте расширение с элементом управления на сервере Web-клиента.
4. Откройте для настройки любую разметку карточки.
5. Добавьте в разметку элемент управления с новым свойством. В его настройках будет *Супер свойство*.

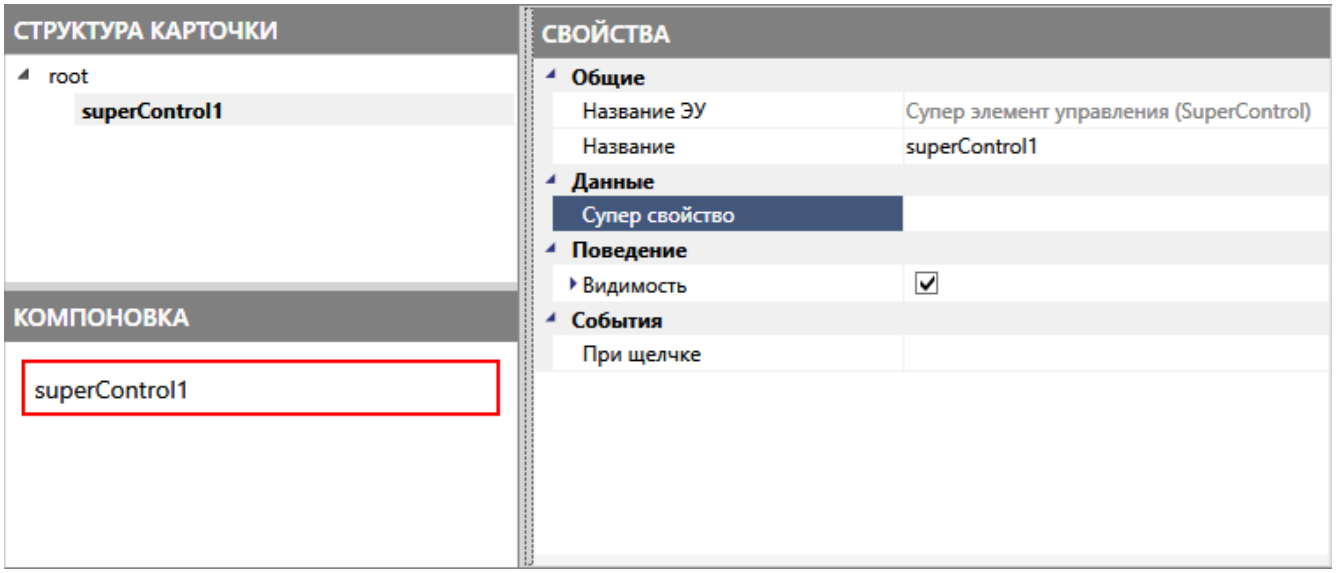


Рисунок 112. Свойство элемента управления из расширения в редакторе разметок

Добавление редактора для значения свойства элемента управления

Редактор предоставляет пользовательский интерфейс для изменения значения свойства в программе Конструктор Web-разметок.

Новый редактор нужно разрабатывать, когда требуется поддержать возможность изменения свойства со сложным типом значения, предоставить новый пользовательский интерфейс для изменения свойства или изменить логику работы со значением свойства.

Редактор представляет собой элемент управления с типом `System.Windows.Controls.UserControl`, в котором реализован интерфейс `Xceed.Wpf.Toolkit.PropertyGrid.Editors.ITypeEditor`. Рекомендуемые размеры элемента управления: `20x200px`. Если для изменения значения свойства требуется форма большего размера, разработайте её в отдельном компоненте с типом `System.Windows.Window`, а в основной компонент `UserControl` добавьте функцию, вызывающую данную форму.

Добавить новый редактор

Добавить новый редактор в Конструктор Web-разметок можно с помощью *Расширения*.

1. Создайте проект расширения программы Конструктор Web-разметок.
2. Добавьте в проект компонент с типом UserControl (WPF) и реализуйте в нем программный интерфейс ITypeEditor.

   Следующий код демонстрирует пример класса редактора.

```
public partial class SuperEditor : UserControl, ITypeEditor
{
    private string startValue;
    private PropertyItem propertyItem; ①

    public SuperEditor()
    {
        InitializeComponent();
    }

    public FrameworkElement ResolveEditor(PropertyItem prop) ②
    {
        propertyItem = prop; ③

        if (prop.Value == null) ④
            prop.Value = string.Empty;

        startValue = prop.Value.ToString(); ⑤

        Placeholder.Text = "{lines ...}"; ⑥

        return this; ⑦
    }

    private void ChangeLines_Click(object sender, RoutedEventArgs e) ⑧
    {
        var dialog = new DialogEditor(startValue);
        dialog.ShowDialog();

        propertyItem.Value = startValue = dialog.Value; ⑨
    }
}
```

① Переменные для хранения текущего значения свойства и его программного компонента (`PropertyItem`).

② Реализация единственного метода интерфейса `ITypeEditor`. Метод должен вернуть элемент управления, который будет отображаться на панели Свойства.

③ Сохраняем ссылку на компонент свойства, для возможности сохранения изменения свойства.

④ Поле `prop.Value` содержит значение элемента управления.

⑤ Значение свойства передается в `prop.Value`.

⑥ Текст, который будет отображаться в качестве значения свойства.

⑦ В качестве элемента управления для отображения на панели свойства используется текущий элемент управления.

⑧ Открытие диалогового окна редактирования значения свойства.

⑨ Новое значение получаем из переменной `Value`. Значение должно быть сохранено в `propertyItem.Value`.
В данном случае редактор представляет собой кнопку **ChangeLines**, при нажатии которой открывается диалоговое окно **DialogEditor** для ввода текстового значения, а также нередактируемый блок текста **Placeholder** для отображения статичного значения `{lines ···}`.

Ниже приведён код графической части редактора.

```
 <UserControl x:Class="TemplateDesignerExtension1.Editors.SuperEditor"
                xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
                xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                xmlns:local="clr-namespace:TemplateDesignerExtension1.Editors"
                mc:Ignorable="d"
                d:DesignHeight="20" d:DesignWidth="200">
        <Grid>
            <TextBlock x:Name="ValueString" HorizontalAlignment="Left" Margin="1"
VerticalAlignment="Stretch" Width="164"/>
            <Button x:Name="ChangeLines" Content="···" HorizontalAlignment="Right"
Margin="1" VerticalAlignment="Top" Width="22" Click="ChangeLines_Click" />;
        </Grid>
</UserControl>
```

Собственная реализация редактора может включать любые элементы управления и использоваться для отображения любых значений (с учетом рекомендуемых значений элемента управления).

Значение свойства можно получить из переменной `propertyItem.Value`. При изменении значения, его нужно сохранить в `propertyItem.Value`.

3. Переопределите метод `GetEditors` базового класса расширения, чтобы он возвращал коллекцию с парой: ключ — имя. Используя данный метод, можно

получить редактор. Значением будет тип редактора:

```
protected override Dictionary<string, Type> GetEditors()
{
    return new Dictionary<string, Type>
    {
        { "SuperEditor", typeof(Editors.SuperEditor) }
    };
}
```

Проверка примера

1. Создайте новое свойство элемента управления в расширении с *Супер редактором*.
2. Укажите *Супер редактор* в качестве редактора свойства в параметре **Editor**.

```
PropertyDescription GetSuperPropertyDescription()
{
    var propertyDescription = new PropertyDescription();
    propertyDescription.Name = "SuperProperty";
    propertyDescription.Type = typeof(string); ①
    propertyDescription.Category = PropertyCategoryConstants.DataCategory;
    propertyDescription.DisplayName = "Супер свойство";
    propertyDescription.Editor = typeof(SuperEditor); ②
    return propertyDescription;
}
```

① Тип свойства должен быть string!

② Указываем тип "Супер редактора".

3. Добавьте свойство с редактором *Супер редактор* в описатель элемента управления. См. пример в пункте Добавление нового свойства элементов управления.
4. Опубликуйте расширение с элементом управления на сервере Web-клиента.
5. Откройте для настройки любую разметку.
6. Добавьте в разметку элемент управления, содержащий свойство с редактором. Для изменения значения свойства будет использован *Супер редактор*.

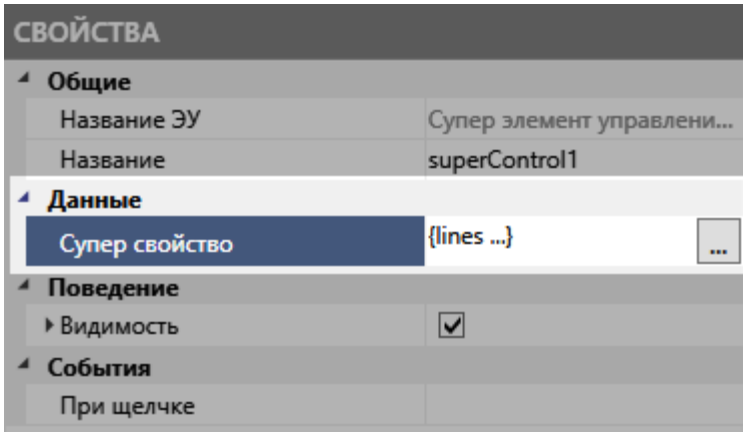


Рисунок 113. Супер свойство с собственным редактором

Изменение значения свойства осуществляется в диалоговом окне, открываемом при нажатии кнопки **...** (три точки).

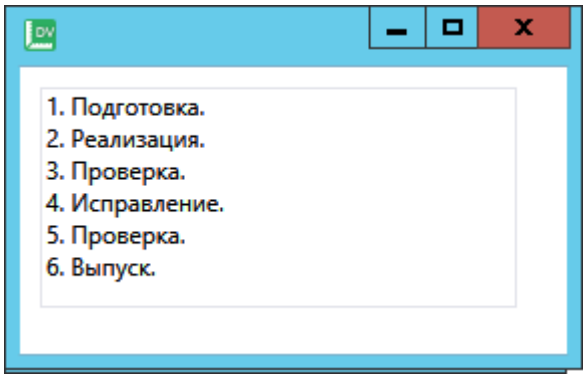


Рисунок 114. Диалоговое окно изменения значения свойства

Редактор для выбора поля карточки

Для выбора поля карточки, с которым связаны данные элемента управления, может использоваться стандартный редактор — `Docsvision.BackOffice.WebLayoutsDesigner.Editors.FieldMetadataEditor`. Данный редактор предоставляет возможность установки фильтра для ограничения доступных для выбора полей.

Следующий пример демонстрирует редактор, предоставляющий возможность выбирать только поля булева типа.

```
class BooleanMetadataEditor : ITypeEditor
{
    public FrameworkElement ResolveEditor(PropertyItem propertyItem)
    {
        var editor = new FieldMetadataEditor();

        editor.FieldFilter = (field) => { return field.FieldType == FieldType.Bool; }; ①
```

```
        return editor.ResolveEditor(propertyItem);
    }
}
```

① Устанавливаем ограничением для отображаемых типов. Текущее обрабатываемое поле передаётся в `field`.

Использование возможностей фильтра `FieldFilter` позволяет установить более точные критерии для отображения списка полей в редакторе.

Например, в следующем коде поля ограничиваются по типу и связанной карточки и секции, с которой должно быть связано поле (для полей ссылающихся на справочники).

```
public FrameworkElement ResolveEditor(PropertyItem propertyItem)
{
    var editor = new FieldMetadataEditor();

    editor.FieldFilter = (field) => { ①
        return (field.FieldType == FieldType.RefId || field.FieldType == FieldType
.UniqueId) &&
            field.LinkedCardTypeId == RefBaseUniversal.ID && ②
            field.LinkedSectionId == RefBaseUniversal.Items.ID; }; ③

    return editor.ResolveEditor(propertyItem);
}
```

① Устанавливаем фильтр.

② Поле должно быть связано с карточкой Конструктор справочников

③ Поле должно быть связано с секцией *Строка справочника*

Добавление локализованных ресурсов

В строковых переменных расширения программы Конструктор Web-разметок (отображаемые названия свойств, диалоги) можно использовать локализованные ресурсы, добавляемые расширением.

Чтобы создать расширение, добавляющее новые локализованные ресурсы, выполните приведенную далее инструкцию.

1. Создайте проект расширения программы Конструктор Web-разметок.
2. Добавьте в проект расширения ресурсные файлы (для английского, русского и др. языков), содержащие требуемые локализованные ресурсы.

Проект расширения, формируемый из шаблона TemplateDesignerExtension, содержит ресурсные файлы, которые можно использовать для формирования локализованных ресурсов.

3. Переопределите метод GetResourceManagers базового класса расширения, чтобы он предоставлял менеджер ресурсов для добавленных ресурсных файлов:

```
protected override List<ResourceManager> GetResourceManagers()
{
    return new List<ResourceManager>
    {
        Resources.ResourceManager
    };
}
```

4. Скомпилируйте проект и скопируйте полученную сборку на сервер Web-клиента в папку `Каталог-установки-Конструктора-Web-Разметок/Plugins/%Каталог Решения%. Ресурсные сборки скопируйте в папки Каталог-установки-Web-клиента/ru/ (для русской локализации), Каталог-установки-Web-клиента/uk/ (для английской локализации) и т.д.

Ограничение выполнения действий с элементом управления

Программист может установить запрет на выполнение определённых действий с элементом управления в программе Конструктор Web-разметок.

Ограничение устанавливается "функцией проверки", которая возвращает набор флагов, разрешающих только определённые действия с элементом управления. Например, *функция проверки* может ограничить использование элемента управления определённым типом карточек.

Добавление функции проверки

Чтобы создать расширение, добавляющее функцию проверки, выполните приведенную далее инструкцию.

1. Создайте проект расширения программы Конструктор Web-разметок.
2. Добавьте в ядро расширения (файл DesignerExtension.cs) функцию проверки следующего вида:

```
public static AllowedOperationsFlag GetPossibleOperation(IServiceProvider
```

```
serviceProvider)
{
    var flags = AllowedOperationsFlag.All; ①

    return flags;
}
```

① По умолчанию доступны все операции
Функция проверки должна вернуть набор флагов AllowedOperationsFlag.

3. Добавьте в код функции проверки алгоритм, добавляющий нужные ограничения.

   Следующий код демонстрирует пример функции проверки, запрещающей добавление элемента управления в разметку карточки, если тип карточки не *Документ*.

```
public static AllowedOperationsFlag GetPossibleOperation(IServiceProvider
serviceProvider)
{
    var flags = AllowedOperationsFlag.All; ①

    var selectedLayoutService = ServiceUtil.GetService<ISelectedLayoutService
>(serviceProvider); ②

    if (selectedLayoutService.SelectedCardTypeId != DocsVision.BackOffice.CardLib
.CardDefs.CardDocument.ID) ③
        flags &= ~AllowedOperationsFlag.Create; ④

    return flags;
}
```

① По умолчанию доступны все операции.

② Получение сервиса для работы с разметкой.

③ Из сервиса получаем идентификатор типа карточек, для которых настраивается разметка.

④ Если тип карточки не документ (CardDocument.ID) из возможных операций удаляется флаг Create. Что фактически приводит к невозможности добавить элемент управления в разметку
В примере информация о контексте использования элемента управления предоставляется сервисом ISelectedLayoutService.

4. Переопределите метод `GetAllowedOperations` базового класса расширения, чтобы он возвращал коллекцию с парой: ключ — имя. Метод позволяет получить функцию проверки. Значение представлено функцией проверки:

```
protected override Dictionary<string, Func<AllowedOperationsFlag>>
GetAllowedOperations()
{
    return new Dictionary<string, Func<AllowedOperationsFlag>>
    {
        {"GetPossibleOperation", () => GetPossibleOperation(this.serviceProvider) } ①
    };
}
```

① Обратите внимание, что в функцию передается сервис-провайдер расширения.

Проверка примера

1. Создайте проект нового элемента управления.
2. В поле GetAllowedOperations (функция проверки) описателя элемента управления укажите функцию GetPossibleOperation.

```
ControlTypeDescription GetSuperControlTypeDescription()
{
    var controlTypeDescription = new ControlTypeDescription("SuperControl")
    {
        DisplayName = "Супер элемент управления",
        ControlGroupDisplayName = "Примеры",
        PropertyDescriptions = {
            PropertyFactory.GetNameProperty (),
            PropertyFactory.GetVisibilityProperty (),
            PropertyFactory.GetClickEvent ()
        },
        GetAllowedOperations = () => GetPossibleOperation(serviceProvider) ①
    };

    return controlTypeDescription;
}
```

① Передаем функцию проверки.
Функция проверки также может быть получена по имени из хранилища функций проверки:

```
GetAllowedOperations = AllowedOperationsStorage.GetAllowedOperations
("GetPossibleOperation")
```

Код получения хранилища функций проверки:

```
IAllowedOperationsStorage allowedOperationsStorage;
protected IAllowedOperationsStorage AllowedOperationsStorage
{
    get
    {
        return this.allowedOperationsStorage ?? (this.allowedOperationsStorage =
ServiceUtil.GetService<IAllowedOperationsStorage>(serviceProvider));
    }
}
```

3. Опубликуйте расширение с элементом управления на сервере Web-клиента.
4. Откройте для настройки разметку карточки *Документ*.

> Элемент управления `Супер элемент управления` будет представлен в библиотеке элементов управления.

5. Откройте любую другую разметку.

> Элемент управления `Супер элемент управления` будет отсутствовать в библиотеке элементов управления.

Разработка элементов управления

Элемент управления в Web-клиенте — дополнительный компонент Модуля, предназначенный для настройки разметки. Элемент управления включает два обязательных компонента:

1. Описатель элемента управления — добавляет в Web-клиент возможность использования данного элемента управления для настройки разметки в программе Конструктор Web-разметок.
2. Клиентский компонент элемента управления — добавляет в Web-клиент пользовательский интерфейс и функции элемента управления: внутреннюю

логику работы, публичные функции.

Формирование и публикация описателя элемента управления

Описатель элемента управления — программный компонент, содержащий общую информацию об элементе управления, и список его свойств (настроек, которые можно изменять в программе Конструктор Web-разметок).

Описатель элемента управления может быть сформирован двумя способами:

1. В виде функции расширения программы Конструктор Web-разметок.
2. В виде файла XML.

Для простоты первый вариант описателя будем называть *бинарным описателем* элемента управления, второй — *текстовым описателем*.

Основное отличие: *текстовый описатель* можно составить с применением уже зарегистрированных редакторов, свойств (пример исключения приведён в пункте Формирование текстового описателя элемента управления) и функций проверки. Для *бинарного описателя* недостающие элементы можно включить в расширение программы Конструктор Web-разметок, в котором реализуется описатель. В остальном выбор способа формирования описателя определяется предпочтениями программиста.

Формирование бинарного описателя элемента управления

Обратитесь к инструкции в пункте Добавление нового элемента управления, чтобы ознакомиться с условиями формирования бинарного описателя элемента управления.

Формирование текстового описателя элемента управления

Ссылка на пример на GitHub: TemplateXmlDesignerExtension.

1. Создайте текстовый файл с расширением `.xml`, кодировкой `UTF-8`, и содержимым:

```
<?xml version="1.0" encoding="utf-8" ?>
<Controls>
  <Control Name="" DisplayName="" ControlGroupDisplayName=""> ①
    <Properties> ②
       <Property Type="" > ③
    </Properties>

    <Events> ④
```

```
      <Event Name="" ResourceKey=""> ⑤
    </Events>
  </Control>
</Controls>
```

① Описатель элемента управления.

② Свойства элемента управления.

③ Здесь перечисляются все свойства элемента управления.

④ События элемента управления.

⑤ Здесь перечисляются все события элемента управления.

2. Укажите основную информацию об элементе управления в атрибутах элемента `Control`.

Name

Название типа элемента управления — английский, без спецсимволов и пробелов.

DisplayName

Отображаемое в программе Конструктор Web-разметок название элемента управления. Если нужно использовать локализованное название, замените атрибут `DisplayName` на `ResourceKey`, и укажите в значении название локализованного ресурса.

Добавить локализованные ресурсы можно с помощью предназначенного для этого расширения программы Конструктор Web-разметок.

ControlGroupDisplayName

Название группы, в которой будет отображаться элемент управления на панели *Библиотека*. Можно указать название существующей группы или новой группы.

Если нужно использовать локализованное название, замените атрибут `ControlGroupDisplayName` на `ControlGroupResourceKey`, и укажите в значении название локализованного ресурса.

AllowedOperations

Название функции проверки. Необязательный атрибут.

3. Перечислите свойства элемента управления в элементе `Properties`.

   Свойства обеспечивают возможность настройки элемента управления в программе Конструктор Web-разметок, и передаются в клиент при загрузке.

4. Добавьте дочерний элемент `Property`.
5. Укажите информацию о свойстве в аргументах элемента `Property`.

   Type

   Название типа свойства, зарегистрированного в программе Конструктор Web-разметок.

   Типы стандартных свойств перечислены в пункте Стандартные свойства и события элементов управления.

   DefaultValue

   Значение по умолчанию (необязательное).

6. Перечислите события элемента управления в элементе `Events`.
7. Добавьте дочерний элемент `Event`.
8. Укажите информацию о событии в аргументах элемента `Event`.

   Name

   Название события. Названия стандартных событий перечислены в пункте Стандартные свойства и события элементов управления.

   DisplayName

   Отображаемое название события. Если нужно использовать локализованное название, замените атрибут `DisplayName` на `ResourceKey`, и укажите в значении название локализованного ресурса.

   DefaultValue

   Обработчик по умолчанию (необязательное).

   1. Сохраните файл `.xml` на сервер Web-клиента в папку `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins`.

9. Перезапустите Конструктор Web-разметок.

Добавление нового типа свойств в текстовом описателе

Новый тип свойств можно добавить с помощью серверного расширения (см. Добавление нового свойства элементов управления) или описать непосредственно в текстовом описателе элемента управления.

Добавление нового типа свойства

Чтобы добавить новый тип свойств:

1. Добавьте дочерний элемент `Property` в секцию `Properties` текстового описателя.
2. Добавьте в элемент `Property` аргументы:

Name

Название нового типа свойств.

DataType

Тип значения свойств. В `DataType` может быть указан только стандартный тип — должно быть указано полное название типа (например, `System.Boolean`).

ResourceKey

Название локализованного ресурса для отображаемого названия.

Category

Категория свойства. Может иметь значения: `GeneralCategory`, `DataCategory`, `BehaviorCategory`, `AppearanceCategory`, `EventsCategory`, `DefaultValueCategory`.

Editor

Имя редактора значения, которое указано при его регистрации в методе GetEditors расширения программы Конструктор Web-разметок.

BindingConverter

Имя конвертера в серверном расширении для свойства `Type="Binding"`.

BindingResolver

Имя резолвера биндинга в серверном расширении для свойства `Type="Binding"`.

ItemsSource

Варианты значений свойства, которые будут отображаться в виде раскрывающегося списка.

В значении атрибута должно быть указано полное имя типа класса, реализующего интерфейс `IItemsSource` (см. ниже).

DefaultValue

Значение по умолчанию.

DeviceDependent

Флаг, указывающий на то, что значение свойства зависит от типа клиентского устройства.

Localized

Флаг, указывающий на то, что свойство является локализуемым.

При объявлении свойств можно указывать одновременно `Type` и другие параметры. Свойство, указанное в `Type`, станет основой для нового, которое может быть дополнительно настроено.

Следующий пример демонстрирует объявление свойства `superBinding`, основанного на стандартном свойстве `Binding`, с переопределённым конвертером (`BindingConverter`):

```
<Property Type="Binding" Name="superBinding" BindingConverter="SuperConverter"
/>
```

Примеры объявления свойств

- Свойство, получающее значения из источника значений (`ItemsSource`):

```
<Property Name="SuperMode" DisplayName="Супер свойство с вариантами"
    Category="AppearanceCategory" DataType="System.Int32" DefaultValue="0"
    ItemsSource="SuperExtension.SuperVariantsSource" />
```

Пример реализации класса `SuperExtension.SuperVariantsSource`, предоставляющего значения для свойства:

```
namespace SuperExtension
{
    public class SuperVariantsSource : IItemsSource
    {
        public ItemCollection GetValues()
        {
            ItemCollection values = new ItemCollection();
            values.Add(0, "Вариант 1"); ①
            values.Add(1, "Вариант 2"); ②
            values.Add(2, Resource.LocalizedVariant3); ③
            return values;
        }
    }
}
```

① При выборе этого варианта, значение свойства будет 0.

② При этом варианте, значение 1.

③ Имя варианта можно объявить в ресурсах с целью локализации.

- Свойство логического типа, зависящее от устройства (DeviceDependent), с категорией Внешний вид (AppearanceCategory) и значением по умолчанию ***false***:

```
<Property Name="ShowPreview" ResourceKey="History_ShowPreview"
    Category="AppearanceCategory" DataType="System.Boolean" DefaultValue="false"
DeviceDependent="true" />
```

- Свойство числового типа с категорией Поведение(BehaviorCategory) и значением по умолчанию 20:

```
<Property Name="RecordsOnPage" Editor="PositiveIntegerEditor"
    Category="BehaviorCategory" DataType="System.Int32" DefaultValue="20"/>
```

- Свойство строкового типа:

```
<Property Name="DurationControlName" Category="BehaviorCategory"
DataType="System.String"/>
```

- Локализованное свойство (Localized) строкового типа со стандартным редактором (Editor) LocalizedPropertyEditor:

```
<Property Name="ButtonText" Editor="LocalizedPropertyEditor" Localized="true"
    Category="AppearanceCategory" DataType="System.String"/>
```

Разработка и публикация клиентского компонента элемента управления

Клиентский компонент элемента управления — программный компонент, обеспечивающий загрузку и функционирование элемента управления на стороне клиента. Клиентский компонент предоставляет графический интерфейс элемента управления.

Клиентский компонент элемента управления является вариантом клиентского расширения.

Чтобы создать клиентский компонент элемента управления, выполните приведенную далее инструкцию.

1. Создайте проект клиентского расширения. Не публикуйте его.
2. Создайте в папке `src` файл TypeScript (`.ts`, `.tsx`) с кодом клиентского компонента.

   Клиентский компонент состоит из трёх блоков кода:

   i. Класс параметров:

```
export class SuperControlParams extends BaseControlParams {
    @r firstProperty?: string;
    @rw secondProperty?: boolean;

    @apiEvent firstEvent?: BasicApiEvent<IEventArgs>;
}
```

   Класс параметров является классом, расширяющим `BaseControlParams`. В данном классе перечисляются публичные свойства и события элемента управления, указанные в его описателе, которые отсутствуют в базовом классе `BaseControlParams`. В данном классе также перечисляются любые другие параметры, значение которых должно быть доступным извне.

   Если свойство, указанное в описателе, не требуется на клиентском уровне, его можно не объявлять в классе параметров.

В приведенном выше коде объявляются два свойства и событие: `firstProperty` (строкового типа), `secondProperty` (булева типа) и `firstEvent` (событие).

При объявлении свойства или события нужно указать декоратор:

- `@r` — если свойство доступно только для чтения.
- `@rw` — если свойство доступно для чтения и записи.
- `@apiEvent` — если свойство является событием.

Значение свойства будут загружено в поле, только если поле класса связано со свойством элемента управления.

Чтобы поле класса могло быть связано со свойством элемента управления:

☑ Имя поля класса должно совпадать с названием свойства/события элемента управления с условием.

Первая буква в названии поля класса должна быть строчной. Например, если в описателе свойство называется `SomeParam`, то поле должно назваться `someParam`.

☑ Тип поля класса должен совпадать с типом свойства элемента управления.

Для событий должен быть установлен тип:

- `BasicApiEvent<IEventArgs>` — для непрерываемых событий.
- `CancelableApiEvent<IEventArgs>` — для прерываемых событий.

Если в описателе элемента управления у свойства указано значение по умолчанию, данное значение должно быть указано в **классе параметров**.

ii. Интерфейс состояния:

```
export interface SuperControlState extends SuperControlParams, BaseControlState {
internalFirstState: boolean;
}
```

Интерфейс состояния является интерфейсом, расширяющим `BaseControlParams`.

В данном интерфейсе перечисляются свойства, которые требуются для описания внутреннего состояния элемента управления (частные переменные).

Если при работе элемента управления хранить внутреннее состояние не требуется, интерфейс состояния не будет содержать полей.

```
export interface SuperControlState extends SuperControlParams,
BaseControlState {
}
```

iii. Класс реализации:

```
export class SuperControlImpl extends BaseControlImpl<SuperControlParams,
SuperControlState> {
    construct() { ①
        super.construct();

        this.state.firstEvent = SimpleEvent.Create(this.state.wrapper);
    }

    renderControl() { ②
        return (
            <div />
        );
    }
}
```

① Инициализация компонента.

② VirtualDOM элемента управления.

Класс реализации является производным от контейнерного класса `BaseControlImpl`, в который передаются `класс параметров` и `интерфейс состояния`. Класс реализации возвращает виртуальный DOM, описывающий пользовательский интерфейс элемента управления.

Класс реализации содержит два обязательных метода:

construct

В данном методе должны быть выполнены следующие действия:

- Вызван конструктор базового класса с передачей ему публичных и внутренних свойств ЭУ:

```
super.construct();
```

- События, объявленные в классе параметров, должны быть проинициализированы, а внутренние обработчики событий связаны с текущим контекстом. Подробности в пункте Работа с событиями в клиентском компоненте.

renderControl

Метод должен вернуть VirtualDOM элемента управления.

Пользовательский интерфейс элемента управления может быть сформирован с использован стандартных элементов React, стандартных элементов управления Web-клиента или разработанных элементов управления (см. дополнительные условия в пункте [Использование вложенных элементов управления](UseNestedControls.md)).

iv. Интерфейсный класс:

```
export class SuperControl extends BaseControl<SuperControlParams,
SuperControlState> {
    protected createParams() {
        return new SuperControlParams(); ①
    }

    protected createImpl() { ②
        return new SuperControlImpl(this.props, this.state);
    }
}
```

① Инициализация параметров.

② Инициализация класса реализации.

Интерфейсный класс является производным от контейнерного класса `BaseControl`, в который передаются *класс параметров* и *интерфейс состояния*.

Интерфейсный класс обеспечивает жизненный цикл элемента управления: связывает элемент управления с его значением, инициализирует класс параметров, а также предоставляет

проинициализированную сущность класса реализации.

Интерфейсный класс должен содержать два обязательных метода:

- `createParams` — метод должен вернуть экземпляр *класса параметров*;
- `createImpl` — метод должен вернуть экземпляр *класса реализации*.

Помимо указанных методов, интерфейсный класс может содержать функции, обрабатывающие загрузку значения свойства элемента управления с сервера Web-клиента.

```
@handler("binding")
private set Binding(binding: IBindingResult<boolean>) {
    this.state.readOnly = editOperationAvailable(this.state.services,
binding) == false;
}
```

В случае простого элемента управления, лишенного сложной внутренней логики, реализация элемента управления может быть помещена в интерфейсный класс.

```
export class SuperSimpleControl extends BaseControl<SuperControlParams,
SuperControlState> {
    protected createParams() {
        return new SuperControlParams();
    }

    protected createImpl() { ①
        return new ControlImpl(this.props, this.state, this.renderControl.bind(this));
    }

    renderControl() { ②
        return (
            <div />
        );
    }
}
```

① Метод `createImpl` в данном случае возвращает экземпляр стандартного типа `ControlImpl`, в которые передана функция, возвращающая VirtualDOM элемента управления

② Метод может иметь любое название, кроме "render".

3. Измените содержимое файла `Index.ts`.

```
import { extensionManager } from "@docsvision/webclient/System/ExtensionManager";

import { SuperControl } from "./Controls/SuperControl"; ①

extensionManager.registerExtension({ ②
    name: "Client extension with controls",
    version: "1.0",
    controls: [{ controlTypeName: "SuperControl", constructor: SuperControl }] ③ ④
})
```

① Добавьте строку импорта элемента управления.

② Добавьте в registerExtension поле controls, в значении которого укажите описание элемента управления.

③ `controlTypeName` — название типа элемента управления, которое указано в описателе элемента управления

④ `constructor` — интерфейсный класс элемента управления.

4. Соберите проект клиентского расширения командой `npm run build`.
5. Скопируйте полученный файл `%BuildDir%\extension.js` на сервер Web-клиента в папку `../../../lib/docsvision/webclient/Content/Modules/%Каталог Решения%`.
6. После завершения отладки, соберите и опубликуйте на сервере Web-клиента "релизную версию" клиентского расширения. "Релизная версия" собирается командой `npm run build:prod`.

Объявление событий элемента управления в клиентском компоненте

Чтобы корректно объявить обрабатываемое событие в клиентском компоненте элемента управления, нужно выполнить два условия:

☑ Генерируемые элементом управления события должны быть проинициализированы.

☑ Событие должно быть сгенерировано.

Инициализация событий

Для примера рассмотрим инициализацию события `onShowed`, поддерживаемого элементом управления:

```
export class SuperControlParams extends BaseControlParams { ①
    @apiEvent onShowed?: BasicApiEvent<IEventArgs>;
}
```

① Класс параметров с объявлением события onShowed.

Для инициализации события onShowed, нужно добавить в конструктор класса реализации следующий код:

```
this.state.onShowed = SimpleEvent.Create<IEventArgs>(this.state.wrapper);
```

Событие onShowed является непрерываемым (имеет тип BasicApiEvent). Для прерываемого события, инициализация будет выглядеть следующим образом:

```
this.state.onShowing = CancelableEvent.Create<IEventArgs>(this.state.wrapper);
```

Таким образом конструктор класса реализации может выглядеть так:

```
construct() {
    super.construct();

    this.state.onShowed = SimpleEvent.Create<IEventArgs>(this.state.wrapper);
    this.state.onShowing = CancelableEvent.Create<IEventArgs>(this.state.wrapper);
}
```

Генерация события

```
SimpleEvent.cast(this.state.onShowed).trigger({} as IEventArgs);

await CancelableEvent.cast(this.state.onShowing).trigger({} as IEventArgs).promise;
```

Связать обработчики с контекстом

Связывание обработчика с контекстом выполняется в конструктор класса реализации, что демонстрируется в следующем коде, в котором выполняется связывание с контекстом обработчика события onClick.

```
construct() {
    super.construct();
```

```
    this.clickHandler = this.clickHandler.bind(this); ①
}
```

① Метод `clickHandler` является внутренним обработчиком события `onClick`.

Работа элемента управления с данными карточки

Чтобы элемент управления мог работать с данными карточки, его необходимо определённым образом подготовить:

1. Добавьте в описатель элемента управления свойства: `DataSource`, `DataField` и `Binding`.

   > Первые два свойства предоставляют возможность выбрать в настройках элемента управления секцию (`DataSource`) и поле карточки (`DataField`) с данными. Свойство `биндинг` (`Binding`) — не отображаемое в настройках свойство, с которым значение элемента управления передается в его клиентский компонент.
   >
   > Элемент управления также может получать данные из карточки, связанной с текущей. Для возможности выбора типа источника (текущая карточки или определённый дополнительный источник) в описатель также нужно добавить свойство `ExtendedDataSource`.

   Текстовый описатель:

   ```
   ...
       <Properties>
          ...
          <Property Type="DataSource" />
          <Property Type="DataField" />
          <Property Type="Binding" />
          <Property Type="ExtendedDataSource" /> ①
        </Properties>
   ```

   ① Не обязательно.

   Бинарный описатель:

   ```
   ...
       PropertyDescriptions =
       {
           ...
           PropertyFactory.GetDataSourceProperty(),
   ```

```
        PropertyFactory.GetDataFieldProperty(),
        PropertyFactory.GetBindingProperty(),
        PropertyFactory.GetExtendedDataSourceProperty(),
    }
```

2. Добавить в клиентский компонент элемента управления в класс параметров поле для хранения значения элемента управления. Тип поля должен соответствовать типу значения элемента управления.

```
export class SomeControlParams extends BaseControlParams {
    @rw value?: string;
}
```

3. Добавить в клиентский компонент элемента управления в интерфейс состояния поле для хранения биндинга. Тип поля должен быть `IBindingResult<string>`. Тип `string` нужно заменить на тип значения элемента управления.

```
export interface SomeControlState extends SomeControlParams, BaseControlState {
    binding: IBindingResult<string>; ①
}
```

① Если элемент управления предназначен только для чтения значения (изменять не требуется), свойство `binding` не требуется.

4. Добавить в клиентский компонент элемента управления в интерфейсный класс set-метод, загружающий значение элемента управления. Метод должен быть помечен декоратором `@handler("binding")`.

```
@handler("binding")
private set binding(binding: IBindingResult<string>) {
    this.state.value =  binding && binding.value;
    this.state.binding = binding;
}
```

5. Добавить в клиентский компонент элемента управления в интерфейсный класс метод, возвращающий биндинги.

"Биндинг" формируется методом `getBindingResult`, в который нужно

передать:

- Текущий "биндинг" (из свойства класса, в которое "биндинг" был сохранен при получении с сервера.
- Текущее значение элемента управления.
- Поле, в котором содержится ассоциированное с ЭУ название элемента карточки.

Переданное значение будет записано в историю изменений карточки: `Выполнилась операция "Редактирование". Значение поля {Значение аргумента}` было изменено с ⋯ на ⋯.

```
protected getBindings() {
    return [getBindingResult(this.state.binding, this.params.value, () => at
(SomeControlParams).controlTitle)];
}
```

Метод `getBindings` должен вернуть `IBindingResult`, для получения которого вызывается метод `getBindingResult`, принимающий:

- Биндинг, полученный при загрузке элемента управления.
- Новое значение элемента управления (в примере, оно хранится в поле `this.params.value`).
- Поле, в котором содержится ассоциированное с элементом управления название элемента карточки. Данное значение используется при формировании истории изменений карточки: `Выполнилась операция "Редактирование". Значение поля {Значение аргумента}` было изменено с ⋯ на ⋯.

6. В класс реализации добавить функцию, сохраняющую значение элемента управления в поле `value`.

```
changedHandler = (event) => {
    this.state.value = event.target.value;
    this.forceUpdate();
}
```

В составе биндинга может быть передана операция редактирования, настраиваемая в программе Конструктор Web-разметок. Операцию

редактирования можно использовать, чтобы ограничить возможность изменять значение элемента управления или его скрытия.

Чтобы использовать операцию редактирования, необходимо:

☑ Добавить в описатель элемента управления свойство EditOperation:

```
<Properties>
①
        <Property Type="DataSource" />
        <Property Type="DataField" />
        <Property Type="Binding" />

        <Property Type="EditOperation" /> ②
</Properties>
```

① Свойства, перечисленные ранее, также должны быть указаны.

② Добавляем операцию редактирования.

☑ Добавить код проверки операции редактирования в функцию загрузки значения элемента управления (binding):

```
@handler("binding")
protected set binding(binding: IBindingResult<boolean>) {
    this.state.canEdit = !binding || editOperations.available(binding.
editOperation);
}
```

Проверка прав пользователя на изменение значения

Возможность изменения значения ЭУ (или выполнения других операций) проверяется по доступности пользователю операции редактирования.

Чтобы добавить возможность настраивать для ЭУ операцию редактирования, нужно добавить в описатель элемента управления свойство `EditOperation`.

```
<?xml version="1.0" encoding="utf-8" ?>
<Controls>
    <Control Name="Somecontrol" DisplayName="Somecontrol">
        <Properties>
            <Property Type="Name" />

            <Property Type="DataSource" /> ①
            <Property Type="DataField" />
```

```
            <Property Type="Binding" />
            <Property Type="EditOperation" /> ②
        </Properties>
    </Control>
</Controls>
```

① Свойства, используемые для связывания ЭУ с данными карточки.

② Свойство `Операция редактирования`.

Свойство `Операция редактирования` передается в представительную часть в биндинге, поэтому свойство `Binding` также должно быть в описателе элемента управления.

Чтобы добавить проверку операции редактирования в представительную часть ЭУ, нужно:

1. Добавить в "интерфейс состояния" поле, в которое будет сохраняться результат проверки.
2. В "интерфейсном классе" в методе, отвечающем за загрузку биндинга, организовать проверку операции редактирования:

```
@handler("binding")
protected set binding(binding: IBindingResult<boolean>) {
    this.state.canEdit = !binding || this.props.services.editOperations.available
(binding.editOperation) ①
}
```

① Сохраняем результат проверки доступности операции в поле `canEdit`.

В приведенном коде проверка доступности операции осуществляется с помощью клиентского сервиса `$EditOperationStore`. Подробности в пункте Получение сервисов в клиентском компоненте.

Конвертация значения элемента управления

Программист может внедрить в механизм получения значения элемента управления функцию конвертации, которая требуемым образом подготовит данные для использования в клиентском компоненте элемента управления.

Чтобы внедрить функцию конвертации, нужно:

1. Указать в описателе свойства `Binding` название функции конвертации:

   ☑ в текстовом описателе — в атрибуте `BindingConverter`:

```
<Property Type="Binding" BindingConverter="SuperConverter" />
```

☑ в бинарном описателе — в методе `GetBindingLoader`:

```
var bindingProperty = PropertyFactory.GetBindingProperty();
bindingProperty.GetBindingLoader = (() => new SimpleBindingLoader
{
    ConverterType = "SuperConverter"
});
```

2. Добавить класс конвертер в серверное расширение.

   Класс конвертера является производным от `BaseBindingConverter`, в котором переопределяется метод `ConvertForDisplay`. Данный метод должен привести значение элемента управления к требуемому виду.

   Следующий код демонстрирует пример класса конвертера, в котором значение элемента управления преобразуется с использованием функции `ConvertForDisplay`.

```
internal class SuperConverter : BaseBindingConverter
{
    public SuperConverter(IServiceProvider serviceProvider) ①
        : base(serviceProvider, "SuperConverter")
    {
    }

    public override BindingResult ConvertForDisplay(ControlContext controlContext,
Binding binding, BindingResult bindingResult) ②
    {
        var data = SuperMethod(bindingResult.Value); ③

        return bindingResult.Clone(data); ④
    }
}
```

   ① В конструктор базового класса должно быть передано название конвертера, которое указано данных свойства.

   ② Метод `ConvertForDisplay` должен возвращать преобразованные данные свойства ЭУ для использования в клиентском расширении ЭУ.

   ③ Здесь метод `SuperMethod` требуемым образом преобразует текущее

переданное значение ЭУ — `bindingResult.Value`.

④ Метод `Clone` создаёт копию `bindingResult`, в поле `Value` которой записывается преобразованное значение из `data`.
Конвертер вызывается автоматически при загрузке значения элемента управления, при условии совпадения названия конвертера в описателе и в расширении.

Обратите внимание

При разработке конвертера следует помнить следующую особенность: в конвертер всегда поступают данные текущей карточки. Если элемент управления связан с данными связанной карточки (не текущей), то получить её данные можно с помощью методов:

- `LayoutContextHelper.TryGetCardDataSource` — сохраняет в `cardId` идентификатор карточки, с данными которой связан элемент управления.
- `LayoutContextHelper.TryGetDataSource` — сохраняет идентификаторы метаданных карточки, с которыми связан элемент управления: в `cardId` — идентификатор карточки, в `sectionId` — идентификатор секции, в `rowId` — идентификатор строки.

```
public override BindingResult ConvertForDisplay(ControlContext controlContext,
Binding binding, BindingResult bindingResult)
{
   CardKindModel model = null;
if (LayoutContextHelper.TryGetCardDataSource(controlContext, binding, out var
cardId))
   {
         model = ExSuperMethod(controlContext.LayoutContext.SessionContext,
cardId);
   }
   return bindingResult.Clone(model);
}
```

3. Конвертер нужно добавить в IoC-контейнер в ядре серверного расширения:

```
public override void InitializeContainer(ContainerBuilder containerBuilder) {
    containerBuilder.RegisterOrderedType<SuperConverter, IBindingConverter>();
```

```
}
```

Расширение должно быть опубликовано на сервере Web-клиента.

Пример. Элемент управления "SuperControl"

В данном примере будут разработаны компоненты элемента управления `SuperControl`: текстовый описатель элемента управления и клиентский компонент.

Элемент управления `SuperControl` является ссылкой, адрес которой указывается в программе Конструктор Web-разметок. Для элемента управления доступно свойство **Видимость** и событие **При щелчке**.

1. Формируем текстовый описатель элемента управления.
   i. Создаем новый файл `SuperControl.xml` со следующим содержимым:

```
<?xml version="1.0" encoding="utf-8" ?>
<Controls>
    <Control Name="SuperControl" DisplayName="Супер элемент управления">
        <Properties>
            <Property Type="Name" /> ①
            <Property Type="Visibility" /> ②

            <Property Name="Url" Category="BehaviorCategory" DataType=
"System.String" /> ③
        </Properties>
        <Events>
            <Event Name="Click" ResourceKey="ControlTypes_ClickEventProperty"/>  ④
        </Events>
    </Control>
</Controls>
```

   ① Название.

   ② Видимость.

   ③ Свойство строкового типа для указания адреса ссылки.

   ④ Событие **При щелчке**.

   ii. Файл `SuperControl.xml` копируем в папку `Каталог-установки-Конструктора-Web-Разметок/Plugins/%Каталог-Решения%`.

2. Формируем клиентское расширение с клиентским компонентом элемента

управления.

i. Создаем новое клиентское расширение.

ii. Добавляем в папку расширения файл `src/Controls/SuperControl.tsx` со следующим содержимым:

```
import { BaseControlParams, BaseControlState, BaseControl } from
"@docsvision/webclient/System/BaseControl";
import { r } from "@docsvision/webclient/System/Readonly";
import { apiEvent } from "@docsvision/webclient/System/Event";
import { BasicApiEvent } from "@docsvision/webclient/System/ApiEvent";
import { IEventArgs } from "@docsvision/webclient/System/IEventArgs";
import { BaseControlImpl } from "@docsvision/webclient/System/BaseControlImpl";
import { SimpleEvent } from "@docsvision/webclient/System/SimpleEvent";
import React from "react"; ①

export class SuperControlParams extends BaseControlParams { ②
    @r url?: string; ③
    @apiEvent onClick?: BasicApiEvent<IEventArgs>;
}

export interface SuperControlState extends SuperControlParams, BaseControlState {
④
}

export class SuperControlImpl extends BaseControlImpl<SuperControlParams,
SuperControlState> { ⑤
    construct() {
        super.construct();

        this.state.onClick = SimpleEvent.Create<IEventArgs>(this.state.wrapper); ⑥

        this.clickHandler = this.clickHandler.bind(this); ⑦
    }

    clickHandler() { ⑧
        SimpleEvent.cast(this.state.onClick).trigger(); ⑨
        window.open(this.props.url, "_blank");
    }

    renderControl() { ⑩

        if (this.props.visibility) { ⑪
            return (
                <a onClick={this.clickHandler}>{this.props.url}</a> ⑫
            );
```

```
        }
    }
}

export class SuperControl extends BaseControl<SuperControlParams,
SuperControlState> {
    protected createParams() { ⑬
        return new SuperControlParams();
    }

    protected createImpl() {
        return new SuperControlImpl(this.props, this.state);
    }
}
```

① Импорт используемых модулей.

② Класс параметров. Объявляем отсутствующее в `BaseControlParams` свойство `url` и непрерываемое событие `onClick`.

③ Название свойства `url` здесь должно быть с маленькой буквы!

④ Интерфейс состояния. Внутренние состояния не используются.

⑤ Класс реализации.

⑥ Инициализируем событие `onClick`.

⑦ Связываем внутренний обработчик с событием `clickHandler`.

⑧ При нажатии ссылки будет выполнен переход по ней, а также вызван обработчик события `onClick`.

⑨ Вызываем обработчик события `onClick`, который указан в Конструкторе web-разметок.

⑩ Возвращаем VirtualDOM элемента управления.

⑪ Если элемент управления невидим (`Visibility` в значении **`FALSE`**), то ничего не возвращаем.

⑫ При нажатии ссылки вызываем внутренний обработчик `clickHandler`

⑬ Интерфейсный класс.

3. Настраиваем клиентское расширение в файле `src/Index.tsx`.

```
import { extensionManager } from "@docsvision/webclient/System/ExtensionManager";
import { SuperControl } from "./Controls/SuperControl";
```

```
extensionManager.registerExtension({
    name: "Client extension with SuperControl",
    version: "1.0",
    controls: [{ controlTypeName: "SuperControl", constructor: SuperControl }]
})
```

4. Собираем клиентское расширение с помощью команды `npm run build`.
5. Полученный файл `extension.js` копируем в папку `{wcd/Content/Modules/SuperControlExtension/` на сервере Web-клиента.
6. Следующие действия выполняются после выполнения всех действий:
    i. Перезапустить **dvwebclient**, чтобы загрузились скрипты с элементом управления.
    ii. Откройте Конструктор Web-разметок и настроить разметку с использованием `Супер элемента управления`.

При настройке элемента управления должны быть доступны свойства **`Url`** и **`Видимость`**, а также событие **`При щелчке`**, для которого можно настроить обработчик.

В веб-браузере `Супер элемент управления` представляет собой ссылку (тег `a`), при нажатии на которую будет выполнен переход по адресу, указанному в **`Url`**, а также вызван обработчик, указанный в событии **`При щелчке`**.

Групповая операция подписания

В данном разделе описан пример реализации групповой операции *Подписание документа*.

Ссылка на пример на GitHub: DocumentSignBatchOperation.

Описание файлов проекта

1. `copy-path.js` — константы с именем расширения и параметрами сборки (в частности, путь, куда копировать файлы после сборки).
2. `gulpfile.js` — конфигурация сборки стилей для Gulp
3. `package.json` — файл конфигурации NPM. Содержит определение команд сборки, а также используемые npm-зависимости.
4. `package-lock.json` — файл, который автоматически генерируется NPM при выполнении команды `npm install`. Это служебный файл NPM, он необходим для того, чтобы зависимости всегда устанавливались в одной и той же последовательности.

5. `rollup.config.js` — файл конфигурации Rollup, который используется для сборки файлов скриптов в один бандл.
6. `tsconfig.json` — файл конфигурации TypeScript
7. `node_modules` — служебная папка NPM, которая появляется после выполнения команды `npm install`. Содержит npm-зависимости проекта, используемые в процессе сборки.
8. `src` — исходные файлы расширения

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS версии v22.14.0 и выше.
- Microsoft Visual Studio 2022.

Начало работы

1. Переименуйте расширение. Например, новое имя будет `MyWebExtension` (замените на уникальное имя своего расширения).

   Для этого:

   - Переименуйте саму папку `TemplateWebExtension` на `MyWebExtension`.
   - Откройте файл `copy-path.js` и замените значение `EXTENSION_NAME` на `MyWebExtension`.
   - Проверьте, что путь `SITE_ROOT` указывает на то место, куда должны копироваться файлы расширения после сборки. Может быть указан только относительный путь (относительно самого файла).
2. Откройте командную строку в корне проекта, выполнить `npm install`. В результате появится папка `node_modules` со всеми необходимыми зависимостями.
3. В командной строке выполните `npm run build`.

   После выполнения команды по пути `SITE_ROOT` (из файла `copy-path.js`) будут подготовлены файлы и папки, которые необходимо скопировать в папку `Site` в установке Web-клиента. В частности:

   - Файл скриптов (`Content/Modules/MyWebExtension/extension.js`).
   - Файл стилей (`Content/Modules/MyWebExtension/extension.css`).

   Файлы в данной папке подключаются автоматически в алфавитном

порядке.

Сборка и установка

1. Откройте консоль в папке `DocumentSignBatchOperationWebExtension` и выполните последовательно команды:

   ```
   npm install
   npm update
   npm run build:prod
   ```

2. Скопируйте каталог `SamplesOutput\Content\Modules\DocumentSignBatchOperationWebExtension` в каталог `Каталог-установки-Web-клиента\Content\Modules`.
3. Скопируйте файл расширения программы Конструктор Web-разметок из папки `DocumentSignBatchOperationDesignerExtension` в папку `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins` в каталоге Web-клиента.
4. Перезапустите **dvappserver**.

Проверка примера

1. Запустите конструктор разметок.
2. Импортируйте решение из файла `DocumentSignBatchOperation.xml`.
3. Сделайте решение `DocumentSignBatchOperation` активным для локации *Папка*. Проверить, что в ЭУ `FolderGridBatchOperationsNode` разметки присутствует дочерний ЭУ `DocumentSignBatchOperation`.
4. Перезапустите **dvappserver**.
5. Откройте Справочник сотрудников в Web-клиенте и назначьте пользователю, которым планируется подписание, сертификат по умолчанию.
6. Авторизуйтесь в Web-клиенте под именем пользователя из предыдущего пункта.

   Убедитесь на любой тестовой карточке документа, что подписание сертификатом из предыдущего пункта проходит корректно.

7. Откройте папку с документами в представлении *Дайджест*. Выделите при помощи флагов документы для подписания. На панели групповых операций нажмите кнопку **Подписать документ**.

8. При необходимости просмотрите подписываемые документы, нажатием на ссылки карточек в открывшемся окне подтверждения. Нажмите **OK**.

   В случае успешного подписания всех документов будет выведено сообщение об успешном подписании.

9. Проверьте на нескольких документах, что подписание прошло корректно.

Чтобы ссылки на карточки отображались корректно в представлении, отличном от *Дайджест*, требуется настроить свойство *Столбцы представления для презентации* ЭУ `DocumentSignBatchOperation` в программе Конструктор Web-разметок. Через запятую укажите `columnName` столбцов представления, выбранных для отображения.

Если в выбор документов в таблице попадет документ, недоступный для подписания по состоянию или настройкам, появится сообщение об ошибке, документ не будет подписан. Для остальных документов процесс подписания будет продолжен.

Дальнейшая разработка

Новые исходные файлы следует добавлять в папку `src`. Структура подчинённых в этой папке не имеет значения. Стили и скрипты могут находиться рядом или быть разнесены — это также неважно.

Единственное обязательное условие — файл `src/Index.ts` должен прямо или косвенно импортировать все остальные файлы расширения. Под косвенным импортированием подразумевается что файл `Index.ts` импортирует файл `A.ts`, а файл `A.ts` в свою очередь импортирует `B.ts` (в этом случае импортировать `B.ts` в файле `Index.ts` не обязательно). Также рекомендуется в файле `Index.ts` располагать код по регистрации обработчиков событий в глобальной области видимости (в объекте window), регистрации ЭУ через `controlFactory`, сервисов и т.д. В противном случае в процессе сборки директивы импорта будут проигнорированы как неиспользуемые.

Для редактирования файлов удобно использовать Visual Studio Code 2022. Достаточно открыть саму папку в редакторе, после чего все файлы будут доступны в дереве проекта. Можно также использовать Visual Studio версии 2022 аналогичным образом.

При использовании сущностей Web-клиента можно использовать функцию авто-импорта. Например, мы можем начать писать вызов функции `publishAsGlobal`, и

Visual Studio Code 2022 автоматически предложит добавить импорт во всплывающем меню.

В процессе разработки можно однажды запустить команду (для этого удобно воспользоваться встроенным терминалом VSCode):

```
npm run watch
```

Данная команда начнет отслеживание изменений в исходных файлах, и при каждом изменении будет повторно собирать расширение и копировать файлы в `SITE_ROOT`. В режиме отслеживания сборка осуществляется значительно быстрее.

```
npm run build
```

Команда `npm run build` собирает скрипты в один бандл `extension.js` без сжатия и с включенным маппингом исходных файлов. То есть, в браузере исходные файлы будут отображаться в том же виде, в каком они есть в папке `src`, что удобно для отладки. При некорректной работе маппинга, можно отключить в браузере маппинг, и использовать в отладке сам файл `extension.js` (т.к. он не минифицирован).

В процессе сборки используется `rollup и gulp`, однако можно использовать любую другую систему сборки (`webpack`, `parcel` и т.д.). Единственное требование — необходимо использовать какую-либо систему `javascript` модулей, поддерживаемую `SystemJS` (`amd`, `commonjs`, `system`).

Обратите внимание, что файлы модулей устанавливаются в папку `Content/Modules/<НазваниеРешения>`

Для сборки расширения в режиме `production` необходимо запустить команду:

```
npm run build:prod
```

Единственное её отличие от `npm run build` в том, что результирующие файлы минифицируются.

Панель отправки на ознакомление

Данный раздел содержит описание примера с элементом управления `Панель`

`отправки на ознакомление`, который демонстрирует запуск бизнес-процесса из карточки документа и создание собственной боковой панели.

Ссылка на пример на GitHub: AcquaintancePanel.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Проект "AcquaintancePanelDesignerExtension"

1. Проект-расширение для конструктора разметок. Содержит описание элемента управления `Пример боковой панели` для программы Конструктор Web-разметок.
2. Демонстрирует использование описания ЭУ в формате `.xml`, задание ему стандартных свойств, реализованных в программе Конструктор Web-разметок и добавление собственного свойства, используя класс `PropretyDescription`.

Проект "AcquaintancePanelServerExtension"

- Проект-расширение для Web-клиент. Содержит бизнес-логику и реализацию элемента управления на клиенте.
- Демонстрирует расширение функционала с помощью добавления новых сервисов, моделей, контроллеров.
- Реализован контроллер `LayoutBusinessProcessController` с методом `SendToAcquaintance`, который вызывает сервис `ILayoutBPService` для отправки документа на ознакомление. В контроллере принудительно записан идентификатор бизнес-процесса.
- Для старта этого бизнес-процесса необходимо передать в качестве параметров "Ознакомителей" (`Performers`), "Документ" (`Document`) и дату "Ознакомиться до" (`EndDate`).

Проект "AcquaintancePanelWebExtension"

- Клиентский ЭУ представляет собой кнопку, при нажатии на которую появляется боковая панель.
- Боковая панель включает в себя стандартные ЭУ Web-клиент:
  - `Сотрудники` для задания списка сотрудников, которым придет задание на ознакомление,

    - Дата/время, дата "Ознакомиться до".
- При нажатии на кнопку **Отправить на ознакомление** происходит проверка параметров, и с помощью объекта requestManager отправляется запрос на сервер.

Сборка и установка

1. Откройте /Samples.sln.
2. Соберите проект Controls > AcquaintancePanel > AcquaintancePanelDesignerExtension.
3. Соберите проект Controls > AcquaintancePanel > AcquaintancePanelServerExtension.
4. Откройте консоль в папке Controls > AcquaintancePanel > AcquaintancePanelWebExtension и выполните команду npm install и npm run build:prod.
5. Скопируйте каталог SamplesOutput\Plugins\AcquaintancePanelDesignerExtension в каталог Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins.
6. Скопируйте каталог SamplesOutput\Extensions\AcquaintancePanelServerExtension в каталог Путь к сайту Web-клиента\Extensions.
7. Скопируйте файл SamplesOutput\Extensions\ru\AcquaintancePanelServerExtension.resources.dll в каталог Путь к сайту Web-клиента\Extensions\ru.
8. Скопируйте файл SamplesOutput\Extensions\uk\AcquaintancePanelServerExtension.resources.dll в каталог Путь к сайту Web-клиента\Extensions\uk.
9. Скопируйте каталог SamplesOutput\Content\Modules\AcquaintancePanel в каталог Путь к сайту Web-клиента\Content\Modules.
10. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Скопируйте любую разметку просмотра.
3. Выберите условия использования этой разметки.
4. Откройте разметку и добавьте в неё новый элемент управления Пример боковой панели.

5. Задайте желаемые параметры элемента управления. Для корректной работы задайте `editOperation = "Send for acquaintance"` или "Отправка на ознакомление".
6. Сохраните разметку.
7. Перезапустите **dvwebclient**.
8. Откройте карточку с этой разметкой.
9. Убедитесь, что появился новый элемент управления, кнопку, при нажатии на которую появляется боковая панель.
10. Заполните необходимые поля и нажмите **Отправить на ознакомление**.
11. Должен запуститься бизнес-процесс, и указанным сотрудникам придет задание на ознакомление со ссылкой на документ.

Пример флага

Данный раздел содержит описание примера реализации элемента управления `Пример флага`.

Ссылка на пример на GitHub: CheckBox.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Проект "CheckBoxDesignerExtension"

1. Проект-расширение для конструктора разметок. Содержит описание элемента управления `Пример флага` для программы Конструктор Web-разметок.
2. Демонстрирует описание и подключение нового ЭУ, используя класс `ControlTypeDescription`, задание ему стандартных свойств, реализованных в программе Конструктор Web-разметок. См. `PropertyFactory.GetNameProperty()` и добавление собственного свойства, используя класс `PropertyDescription`. См. свойство `DefaultValue`.
3. Для свойства `Поле секции`, источник данных, реализован редактор, фильтрующий поля по типу `boolean`, класс `BooleanMetadataEditor`.

Проект "CheckBoxWebExtension"

Проект-расширение клиентской части Web-клиента. Содержит клиентский

скрипт и стили для `CheckBox`.

Сборка и установка

1. Откройте `/Samples.sln`.
2. Соберите проект `Controls > CheckBox > CheckBoxDesignerExtension`.
3. Откройте консоль в папке `Controls > CheckBox > CheckBoxWebExtension` и выполните команду `npm install` и `npm run build:prod`.
4. Скопируйте каталог `SamplesOutput\Plugins\CheckBoxDesignerExtension` в каталог `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins`.
5. Скопируйте каталог `SamplesOutput\Content\Modules\CheckBoxWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
6. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Скопируйте любую разметку для карточки вида *Служебная записка*.
3. Выберите условия использования этой разметки.
4. Откройте разметку и добавить в нее новый элемент управления `Пример флага`.
5. Задайте желаемые параметры элемента управления.
6. Сохраните разметку.
7. Перезапустите **dvwebclient**.
8. Откройте карточку с этой разметкой.
9. Убедитесь, что появился новый элемент управления.

Групповая операция / Скачивание файлов

Данный раздел содержит описание примера реализации элемента управления групповой операции *Групповая операция / Скачивание файлов*.

Ссылка на пример на GitHub: DownloadFilesGroupOperation.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше

- Microsoft Visual Studio 2022.

Проект "DownloadFilesGroupOperationDesignerExtension"

Проект-расширение для конструктора разметок. Содержит описание элемента управления групповой операции *Групповая операция / Скачивание файлов* для программы Конструктор Web-разметок. Демонстрирует описание и подключение нового ЭУ, используя класс `ControlTypeDescription`,

Проект "DownloadFilesGroupOperationServerExtension"

Содержит контроллер, в котором формируется список файлов документа, запрашиваемый элементом управления *Групповая операция / Скачивание файлов*.

Проект "DownloadFilesGroupOperationWebExtension"

Проект-расширение клиентской части Web-клиент. Содержит реализацию элемента управления *Групповая операция / Скачивание файлов*.

Сборка и установка

1. Откройте `/Samples.sln`.
2. Соберите проект `Controls > DownloadFilesGroupOperation > DownloadFilesGroupOperationDesignerExtension`.
3. Соберите проект `Controls > DownloadFilesGroupOperation > DownloadFilesGroupOperationServerExtension`.
4. Откройте консоль в папке `Controls > DownloadFilesGroupOperation > DownloadFilesGroupOperationWebExtension` и выполните команду `npm install` и `npm run build:prod`.
5. Скопируйте каталог `SamplesOutput\Plugins\DownloadFilesGroupOperationDesignerExtension` в каталог `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins`.
6. Скопируйте каталог `SamplesOutput\Extensions\DownloadFilesGroupOperationServerExtension` в каталог `Путь к сайту Web-клиента\Extensions`.
7. Скопируйте каталог `SamplesOutput\Content\Modules\DownloadFilesGroupOperation` в каталог `Путь к сайту Web-клиента\Content\Modules`.
8. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. В разметку **WebFrame** › **Папка** добавьте ЭУ *Групповая операция / Скачивание файлов*.
3. Сохраните разметку.
4. Откройте каталог с документами в Web-клиенте.
5. Включить режим групповых операций и убедиться, что появился новый элемент управления.

Элемент управления "Обменный курс"

Данный раздел содержит описание примера реализации элемента управления `ExchangeRates`. Проект-расширение клиентской части Web-клиента. Содержит клиентский скрипт и стили для `SampleExchangeRates`.

Ссылка на пример на GitHub: ExchangeRates.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Сборка и установка

1. Откройте `/Samples.sln`.
2. Откройте консоль в папке `Controls > ExchangeRates > ExchangeRatesWebExtension` и выполните команду `npm install` и `npm run build:prod`.
3. Скопируйте каталог `SamplesOutput\Plugins\SampleExchangeRatesDesignerExtension` в каталог `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins`.
4. Скопируйте каталог `SamplesOutput\Content\Modules\ExchangeRatesWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
5. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Импортируйте решение из файла `SampleExchangeRatesControlDescriptionLayout.xml`.

3. Сделайте решение `SampleExchangeRatesControlDescriptionLayout` активным для карточки типа Документ вида *ДокументУД/Исходящий*.
4. Откройте разметку `SampleExchangeRatesControlDescriptionLayout`
5. Зарегистрируйте бесплатный аккаунт на сайте fixer.io.
6. Перейдите по ссылке https://fixer.io/quickstart и скопировать ключ доступа к API.
7. Задайте параметру **`Ключ доступа к API`** элемента управления `ExchangeRates` значение вашего ключа из шага выше.
8. Задайте остальные желаемые параметры элемента управления.
9. Сохраните разметку.
10. Перезапустите **dvwebclient**.
11. Откройте карточку с этой разметкой.
12. Убедитесь, что появился новый элемент управления.

Описание ЭУ "SampleExchangeRatesControlDescription.xml"

Демонстрирует добавление нового ЭУ без написания расширения для конструктора разметок. Для этого необходимо описать ЭУ в xml-файле, используя уже подключенные к конструктору свойства и ресурсы.

```
<?xml version="1.0" encoding="utf-8" ?>
<Controls>
  <Control Name="SampleExchangeRates" ControlGroupResourceKey="ControlGroup_Samples"
DisplayName="Sample exchange rates">
    <Properties> ①
      <Property Type="Name" />
      <Property Type="Visibility" />
      <Property Type="StandardCssClass" DefaultValue="sample-exchange-rates" /> ②
      <Property Type="CustomCssClasses" />
    </Properties>
  </Control>
</Controls>
```

① Содержит перечисление свойств ЭУ.

② `DefaultValue` задаёт значение поля по умолчанию.

Элемент управления "Пример изображения"

Данный раздел содержит описание примера элемента управления `Пример`

изображения.

Ссылка на пример на GitHub: Image.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Проект "ImageDesignerExtension"

1. Проект-расширение для конструктора разметок. Содержит описание элемента управления `Пример изображения` для программы Конструктор Web-разметок.
2. Демонстрирует описание и подключение нового ЭУ, используя класс `ControlTypeDescription`, задание ему стандартных свойств, реализованных в программе Конструктор Web-разметок[10] и добавление кастомных свойств, используя класс `PropretyDescription` [11].
3. Для собственного свойства `Slider` реализован свой редактор, `SliderEditor.xaml`.

Проект "ImageServerExtension"

Проект-расширение для Web-клиента. Содержит бизнес-логику и реализацию элемента управления на сервере. Демонстрирует работу с собственным свойством `Slider` — преобразование его значения `string` в рабочую модель, `List` с помощью класса `SliderConverter` и метода `ConvertForDisplay`. Эта модель дублируется на клиенте `ISliderItem[]`. Таким образом реализована связка `List` — `JSON` — `ISliderItem[]` между сервером и клиентом.

С помощью `Slider` можно пролистывать несколько изображений, вешая обработчики `onClick` на соответсвующее элементы разметки.

Проект "ImageWebExtension"

Содержит скрипты и стили ЭУ `Пример изображения`.

Сборка и установка

1. Откройте `/Samples.sln`.
2. Соберите проект `Controls > Image > ImageDesignerExtension`.
3. Соберите проект `Controls > Image > ImageServerExtension`.

4. Откройте консоль в папке `Controls > Image > ImageWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
5. Соберите проект `ControlProperties > Url > UrlPropertyDesignerExtension`.
6. Скопируйте каталог `SamplesOutput\Plugins\ImageDesignerExtension` в каталог `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins`.
7. Скопируйте каталог `SamplesOutput\Extensions\ImageServerExtension` в каталог `Путь к сайту Web-клиента\Extensions`.
8. Скопируйте каталог `SamplesOutput\Content\Modules\ImageWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
9. Скопируйте каталог `SamplesOutput\Plugins\UrlPropertyDesignerExtension` в каталог `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins`.
10. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Скопируйте любую разметку.
3. Выберите условия использования этой разметки.
4. Откройте разметку и добавьте в нее новый элемент управления `Пример изображения`.
5. Задайте желаемые параметры элемента управления.

При указании URL адрес должен быть доступен согласно политике CORS. Большинство серверов не разрешают запросы из веб-приложений, расположенных на других доменах. Политике CORS гарантированно удовлетворяют адреса внутри Web-клиента, например `Content/App/Assets/Images/favorite-empty.svg`.

6. Сохраните разметку.
7. Перезапустите **dvwebclient**.
8. Откройте карточку с этой разметкой.
9. Убедитесь, что появился новый элемент управления.

Элемент управления "Link"

Данный раздел содержит описание примера реализации элемента управления `Пример ссылки`. Проект-расширение клиентской части Web-клиента. Содержит клиентский скрипт и стили для ЭУ `SampleLink`.

Ссылка на пример на GitHub: Link.

Пример требует подключенного расширения программы Конструктор Web-разметок `UrlPropertyDesignerExtension`.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Сборка и установка

1. Откройте `/Samples.sln`.
2. Откройте консоль в папке `Controls > Link > LinkWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
3. Скопируйте каталог `SamplesOutput\Plugins\LinkDesignerExtension` в каталог `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins`.
4. Скопируйте каталог `SamplesOutput\Content\Modules\LinkWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
5. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Импортируйте решение из файла `SampleLinkControlDescriptionLayout.xml`.
3. Сделайте решение `LinkControlDescriptionLayout` активным для карточки типа Документ вида ДокументУД/Исходящий.
4. Откройте разметку `LinkControlDescriptionLayout`.
5. Задайте желаемые параметры элемента управления `Пример ссылки`.
6. Сохраните разметку.
7. Перезапустите **dvwebclient**.

8. Откройте карточку с этой разметкой.
9. Убедитесь, что появился новый элемент управления.

Описание ЭУ "SampleLinkControlDescription.xml"

Демонстрирует добавление нового ЭУ без написания расширения для конструктора разметок. Для этого необходимо описать ЭУ в xml-файле, используя уже подключенные к конструктору свойства и ресурсы.

```
<?xml version="1.0" encoding="utf-16" standalone="no"?>
<Data>
  <SolutionsDirectory CardID="3200C1F1-6CAA-4C73-A169-78F3A12D155C" CardTypeID="3200C1F1-
6CAA-4C73-A169-78F3A12D155C" Description="SolutionsDirectory" CreationDateTime="2018-12-
06T12:18:50" ChangeDateTime="2019-02-21T11:44:59" Template="0" Topic="" Barcode="">
    <Solutions>
      <SolutionsRow RowID="958039AB-72EC-4E16-B25A-D00C97A160DE" OwnServerID="00000000-
0000-0000-0000-000000000000" ChangeServerID="00000000-0000-0000-0000-000000000000"
SolutionID="A925F725-1D89-4FD5-900F-9A097869D479" Name="LinkClientExtensionLayout"
Version="1.0" Mode="0">
        <Layouts>
          <LayoutsRow RowID="56BCF095-8303-4DAB-955C-CAFADBB7CDEB" OwnServerID="00000000-
0000-0000-0000-000000000000" ChangeServerID="00000000-0000-0000-0000-000000000000"
LayoutID="5F515439-7950-4246-8EEF-808B3D894124" Name="LinkClientExtensionLayout"
LayoutFileID="DD1B7EFE-2E29-43F6-BFC9-32144861DADE" LocationID="8e40f327-9517-4a43-998d-
bf2bd619588d" LayoutFileHash="F58E7A92D0A915B0CF90C4AB182A7CEE" LayoutType="1" />
        </Layouts>
      </SolutionsRow>
    </Solutions>
  </SolutionsDirectory>
  <File FileID="DD1B7EFE-2E29-43F6-BFC9-32144861DADE" OwnerCardID="3200C1F1-6CAA-4C73-
A169-78F3A12D155C" Name="LinkClientExtensionLayout" Size="2056" CreationDateTime="2019-
02-21T10:10:01" ChangeDateTime="2019-02-21T11:44:59" StandardAttribs="0"
ExtendedAttribs="0">![CDATA[77u/PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTE2Ij8+PEVsZ
W1lbnQgSWQ9IjZjODA4M2NmLTljY2EtNDczZi05MTAxLWUwMWQyNzQ1YWYzZiIgVHlwZT0iTGF5b3V0Ij48UHJvcG
VydGllcz48UHJvcGVydHkgTmFtZT0iTmFtZSIgVmFsdWU9InJvb3QiIFR5cGU9IlN5c3RlbS5TdHJpbmcsIG1zY29
ybGliIiBJc0F2YWlsYWJsZT0idHJ1ZSIgLz48UHJvcGVydHkgTmFtZT0iQ2FyZE9wZW5pbmciIFZhbHVlPSIiIFR5
cGU9IlN5c3RlbS5TdHJpbmcsIG1zY29ybGliIiBJc0F2YWlsYWJsZT0idHJ1ZSIgLz48UHJvcGVydHkgTmFtZT0iQ
2FyZE9wZW5lZCIgVmFsdWU9IiIgVHlwZT0iU3lzdGVtLlN0cmluZywgbXNjb3JsaWIiIElzQXZhaWxhYmxlPSJ0cn
VlIiAvPjxQcm9wZXJ0eSBOYW1lPSJDYXJkU2F2aW5nIiBWYWx1ZT0iIiBUeXBlPSJTeXN0ZW0uU3RyaW5nLCBtc2N
vcmxpYiIgSXNBdmFpbGFibGU9InRydWUiIC8+PFByb3BlcnR5IE5hbWU9IkNhcmRTYXZlZCIgVmFsdWU9IiIgVHlw
ZT0iU3lzdGVtLlN0cmluZywgbXNjb3JsaWIiIElzQXZhaWxhYmxlPSJ0cnVlIiAvPjxQcm9wZXJ0eSBOYW1lPSJDY
XJkU3RhdGVDaGFuZ2luZyIgVmFsdWU9IiIgVHlwZT0iU3lzdGVtLlN0cmluZywgbXNjb3JsaWIiIElzQXZhaWxhYm
xlPSJ0cnVlIiAvPjxQcm9wZXJ0eSBOYW1lPSJDYXJkTGF5b3V0U3dpdGNoaW5nIiBWYWx1ZT0iIiBUeXBlPSJTeXN
0ZW0uU3RyaW5nLCBtc2NvcmxpYiIgSXNBdmFpbGFibGU9InRydWUiIC8+PC9Qcm9wZXJ0aWVzPjxFbGVtZW50cz48
RWxlbWVudCBJZD0iZDQ0ODg2NDUtZjM1Yi00OGE0LWIxZTYtMzU4NzZiZjY3MGNjIiBUeXBlPSJTYW1wbGVMaW5rI
j48UHJvcGVydGllcz48UHJvcGVydHkgTmFtZT0iTmFtZSIgVmFsdWU9InNhbXBsZUxpbmsxIiBUeXBlPSJTeXN0ZW
```

```
0uU3RyaW5nLCBtc2NvcmxpYiIgSXNBdmFpbGFibGU9InRydWUiIC8+PFByb3BlcnR5IE5hbWU9IlZpc2liaWxpdHk
iIFZhbHVlPSImbHQ7P3htbCB2ZXJzaW9uPSZxdW90OzEuMCZxdW90OyBlbmNvZGluZz0mcXVvdDt1dGYtMTYmcXVv
dDs/Jmd0OyYjeEQ7JiN4QTsmbHQ7RGV2aWNlRGVwZW5kZW50UHJvcGVydHlPZkJvb2xlYW4geG1sbnM6eHNpPSZxd
W90O2h0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW1hLWluc3RhbmNlJnF1b3Q7IHhtbG5zOnhzZD0mcXVvdD
todHRwOi8vd3d3LnczLm9yZy8yMDAxL1hNTFNjaGVtYSZxdW90OyZndDsmI3hEOyYjeEE7ICAmbHQ7RGVza3RvcCZ
ndDt0cnVlJmx0Oy9EZXNrdG9wJmd0OyYjeEQ7JiN4QTsgICZsdDtUYWJsZXQmZ3Q7dHJ1ZSZsdDsvVGFibGV0Jmd0
OyYjeEQ7JiN4QTsgICZsdDtTbWFydHBob25lJmd0O3RydWUmbHQ7L1NtYXJ0cGhvbmUmZ3Q7JiN4RDsmI3hBOyAgJ
mx0O0NhbkNoYW5nZSZndDt0cnVlJmx0Oy9DYW5DaGFuZ2UmZ3Q7JiN4RDsmI3hBOyZsdDsvRGV2aWNlRGVwZW5kZW
50UHJvcGVydHlPZkJvb2xlYW4mZ3Q7IiBUeXBlPSJTeXN0ZW0uU3RyaW5nLCBtc2NvcmxpYiIgUmVzb2x2ZXJUeXB
lPSJEZXZpY2VEZXBlbmRlbnQiIElzQXZhaWxhYmxlPSJ0cnVlIiAvPjxQcm9wZXJ0eSBOYW1lPSJTdGFuZGFyZENz
c0NsYXNzIiBWYWx1ZT0ic2FtcGxlLWxpbmsiIFR5cGU9IlN5c3RlbS5TdHJpbmcsIG1zY29ybGliIiBJc0F2YWlsY
WJsZT0idHJ1ZSIgLz48UHJvcGVydHkgTmFtZT0iVGFiU3RvcCIgVmFsdWU9ImZhbHNlIiBUeXBlPSJTeXN0ZW0uQm
9vbGVhbiwgbXNjb3JsaWIiIElzQXZhaWxhYmxlPSJ0cnVlIiAvPjxQcm9wZXJ0eSBOYW1lPSJDbGljayIgVmFsdWU
9IiIgVHlwZT0iU3lzdGVtLlN0cmluZywgbXNjb3JsaWIiIElzQXZhaWxhYmxlPSJ0cnVlIiAvPjxQcm9wZXJ0eSBO
YW1lPSJNb3VzZU92ZXIiIFZhbHVlPSIiIFR5cGU9IlN5c3RlbS5TdHJpbmcsIG1zY29ybGliIiBJc0F2YWlsYWJsZ
T0idHJ1ZSIgLz48UHJvcGVydHkgTmFtZT0iTW91c2VPdXQiIFZhbHVlPSIiIFR5cGU9IlN5c3RlbS5TdHJpbmcsIG
1zY29ybGliIiBJc0F2YWlsYWJsZT0idHJ1ZSIgLz48L1Byb3BlcnR5PjwvRGVtZW50PjwvRGV
FbGVtZW50Pg==]]</File>
</Data>
```


Пример текста

Данный раздел содержит описание примера реализации элемента управления `Пример текста`.

Ссылка на пример на GitHub: TextBox.

Пример требует подключенного расширения программы Конструктор Web-разметок `UrlProprtyDesignerExtension`.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Проект "TextBoxDesignerExtension"

1. Проект-расширение для конструктора разметок. Содержит описание элемента управления `Пример текста` для программы Конструктор Web-разметок.
2. Демонстрирует описание и подключение нового ЭУ, используя класс `ControlTypeDescription`, задание ему стандартных свойств, реализованных в программе Конструктор Web-разметок[12] и добавление собственного свойства, используя класс `ProprietyDescription` [13].

Проект "CheckBoxWebExtension"

Проект-расширение клиентской части Web-клиента. Содержит клиентский скрипт, формирующий поле ввода текста и картинку (свойство `Url`), и стили для ЭУ `Пример текста`.

Сборка и установка

1. Откройте `/Samples.sln`.
2. Соберите проект `Controls > TextBox > TextBoxDesignerExtension`.
3. Откройте консоль в папке `Controls > TextBox > TextBoxWebExtension` и выполните команду `npm install`, затем `npm update` и в конце `npm run build:prod`.
4. Скопируйте каталог SamplesOutput\Plugins\TextBoxDesignerExtension в каталог `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins`.
5. Скопируйте каталог `SamplesOutput\Content\Modules\TextBoxWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
6. Выполните инструкции по сборке и установке примера `ControlProperties > Url`.
7. Перезапустите **dvwebclient**.

Проверка примера

1. Запустите конструктор разметок.
2. Скопируйте любую разметку редактирования.

Пример рассчитан только на работу в разметках редактирования.

3. Выберите условия использования этой разметки.
4. Откройте разметку и добавить в нее новый элемент управления `Пример текста`.
5. Задайте желаемые параметры элемента управления.

При указании URL адрес должен быть доступен согласно политике CORS. Большинство серверов не разрешают запросы из веб-приложений, расположенных на других доменах. Политике CORS гарантированно удовлетворяют адреса внутри Web-клиента, например `Content/App/Assets/Images/favorite-empty.svg`.

6. Сохраните разметку.
7. Перезапустите **dvwebclient**.
8. Откройте карточку с этой разметкой.
9. Убедитесь, что появился новый элемент управления.

Дополнительная информация

Руководство по разработке решений Web-клиента

Дополнительная информация

Дополнительная информация по разработке примеров для конструктора разметок.

- Получение сервисов в клиентском компоненте
- Использование вложенных элементов управления
- Остановка выполнения операции с помощью прерываемого события элемента управления
- Больше примеров категории в навигационном меню слева…

Получение сервисов в клиентском компоненте

В клиентском компоненте элемента управления можно получить доступ к стандартным (список в справочнике JS API, раздел Services) и разработанным клиентским сервисам.

Ссылка на сервисы автоматически передается при загрузке клиентского компонента. Чтобы получить ссылку, нужно добавить в класс параметров поле services с указанием типов требуемых сервисов:

```
@rw services?: $EditOperationStore & $UrlStore & $RequestManager;
```

В данном примере в services сохраняются сервисы: `$EditOperationStore`, `$UrlStore`, `$RequestManager`.

Обратиться к сервису можно следующим образом:

```
let editOperations = this.state.services.editOperations.getAll(); ①
```

① Получаем список всех операций редактирования с помощью метода `getAll` сервиса `$EditOperations`.

Использование вложенных элементов управления

Стандартные и разработанные элементы управления Web-клиента можно использовать для формирования виртуального DOM другого элемента управления.

Следующий код демонстрирует пример использования стандартного элемента управления `Дата/время` в функции отрисовки элемента управления.

```
renderControl() {
    return (
        <div>
            <DateTimePicker name="someControlNestedControl" editMode={EditMode.Edit}
                    required={true} placeHolder={resources
.AcquaintancePanel_ConsiderationDatePlaceholder} />
        </div>
    );
}
```

Остановка выполнения операции с помощью прерываемого события элемента управления

События, генерируемые элементом управления, могут быть стандартными (непрерываемые) и прерываемыми, с помощью которых обработчик может остановить выполнение операции (например, сохранение карточки).

Чтобы создать прерываемое событие:

1. Добавьте событие в описатель элемента управления:

```
<?xml version="1.0" encoding="utf-8" ?>
<Controls>
  <Control Name="SomeControl" DisplayName="Пример элемента управления">
    <Properties>
      <Property Type="Name" />
    </Properties>
    <Events>
      <Event Name="someEventing" DisplayName="Перед выполнение операции"/> ①
    </Events>
  </Control>
</Controls>
```

① Прерываемое событие.

2. В клиентском компоненте элемента управления в классе параметров для события укажите тип CancelableApiEvent:

```
export class SomeControlParams extends BaseControlParams {
    @apiEvent someEventing?: CancelableApiEvent<IEventArgs>;
}
```

3. В клиентском компоненте элемента управления в классе реализации:

   i. Проинициализируйте прерываемое событие с помощью метода `CancelableEvent.Create`:

```
construct() {
    super.construct();

    this.state.someEventing = CancelableEvent.Create<IEventArgs>(this.state.
wrapper);
}
```

   ii. Во внутреннем обработчике события, вызываемого перед выполнением операции, добавьте код, который будет выполняться при прерывании события внешними обработчиками, и код для продолжения обработки события:

```
protected internalEventHandler = async () => {
    try {
        await CancelableEvent.cast(this.state.someEventing).trigger().promise;

        console.log("Выполнение операции ..."); ①
    } catch {

        console.log("Выполнение операции прервано ..."); ②
    }
}
```

   ① Код, вызываемый при продолжении операции.

   ② Код, вызываемый при отмене операции.

Переопределение стиля элемента управления

Разработчик может изменить стандартный стиль элемента управления с помощью дополнительных CSS-правил.

Изменение внутренних стилей стандартных элементов управления не рекомендуется. Поддержка обратной совместимости верстки элементов управления не гарантируется Docsvision.

Изменение стандартного стиля

Чтобы изменить стандартный стиль элемента управления, выполните следующие действия:

1. Создайте CSS-файл с классом с правилами, переопределяющими стандартный стиль элемента управления.

CSS-правила, используемые стандартным стилем элемента управления, можно определить с помощью инструментов разработчика, предоставляемых веб-браузером. Пример приведён в конце пункта; также смотрите примеры в репозитории Docsvision на Github.

2. Создайте в папке `../../../lib/docsvision/webclient/Content/Modules` каталог для размещения файлов собственного Решения. В названии папки предпочтительно использовать только латиницу, не использовать пробелы и спецсимволы.
3. Скопируйте CSS-файл, полученный на шаге 1, в созданный каталог Решения.
4. Укажите название CSS-класса, реализованного на шаге 1, в программе Конструктор Web-разметок в свойстве Дополнительные css классы изменяемого элемента управления. Сохраните разметку.
5. Перезапустите **dvwebclient** или обновите страницу в браузере, отключив кэширование. Перезапуск необходим только при добавлении нового файла, при изменении — не требуется.

Пример переопределения стиля

В качестве примера далее приведена последовательность действий при изменении цвета фона элемента управления `Строка`.

1. Создадим файл `style.css` со следующим содержимым:

```
.new-textbox input
{
    background-color: yellow !important;
}
```

Класс `new-textbox` определяет правило, которое применяется к элементу `input`, и устанавливает желтый цвет для фоновой заливки.

Класс был сформирован следующим образом:

1. Открыли разметку карточки с элементом управления `Строка`. В данном случае — разметка создания карточки исходящего документа.
2. Открыли код элемента управления `Строка` (в разметке карточки — поле `Тема`) с помощью инструментов разработчика.

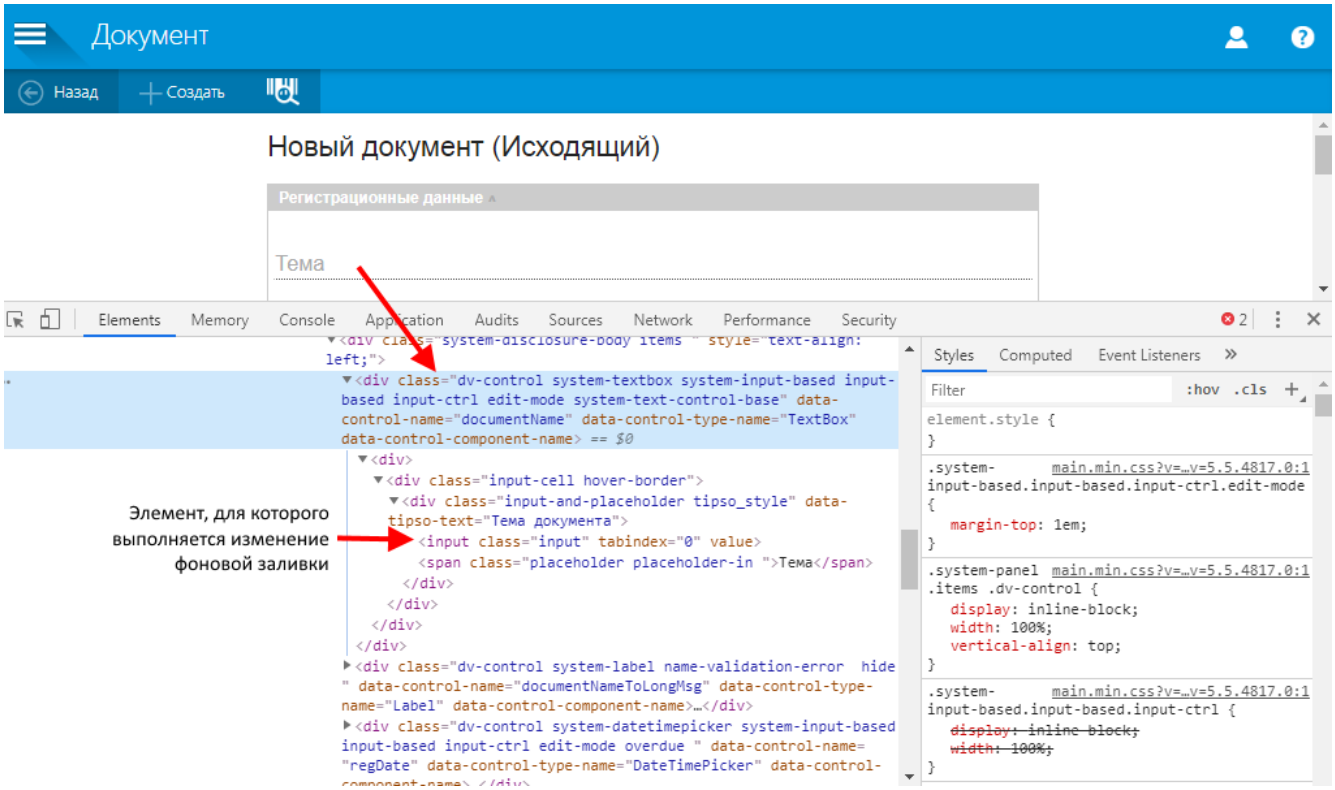


Рисунок 115. Код элемента управления в веб-браузере Chrome

3. Нашли в структуре элемента управления `Строка` элемент, требующий изменения фоновой заливки. Данным элементом является `input` (поле для ввода).

   Для создания селектора CSS-класса, который позволит установить стиль для элемента `input`, достаточно в селекторе указать название тега (`input`) и установить приоритет `important` для свойства `background-color` (фоновая заливка).

4. Создадим для Решения папку `../../../lib/docsvision/webclient/Content/Modules/SuperSolution`.
5. Скопируем файл style.css в папку

../../../lib/docsvision/webclient/Content/Modules/SuperSolution.

6. Укажем название класса new-textbox в свойстве **Дополнительные css классы** элемента управления Строка в программе Конструктор Web-разметок и сохраним разметку.

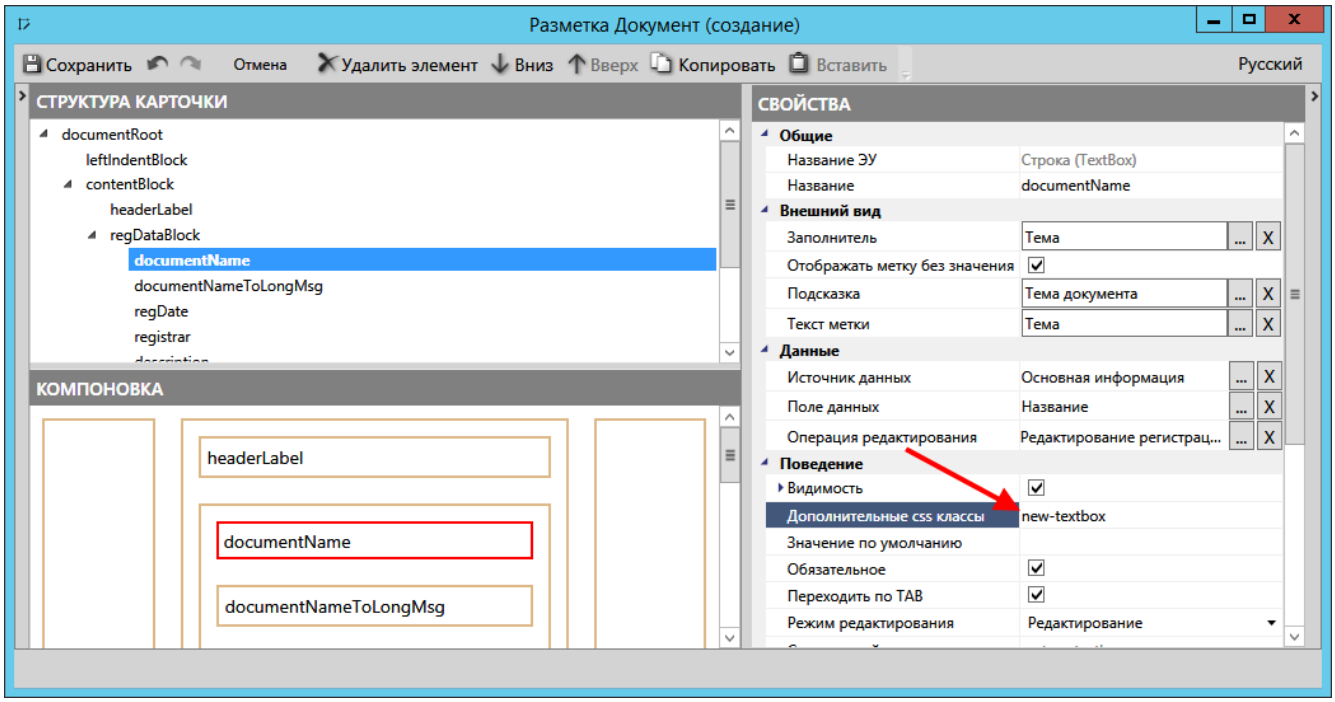


Рисунок 116. Дополнительные CSS-классы в настройках элемента управления TextBox

7. Перезапустим **dvwebclient**.
8. Запустим карточку с настроенной разметкой для проверки применения стиля.

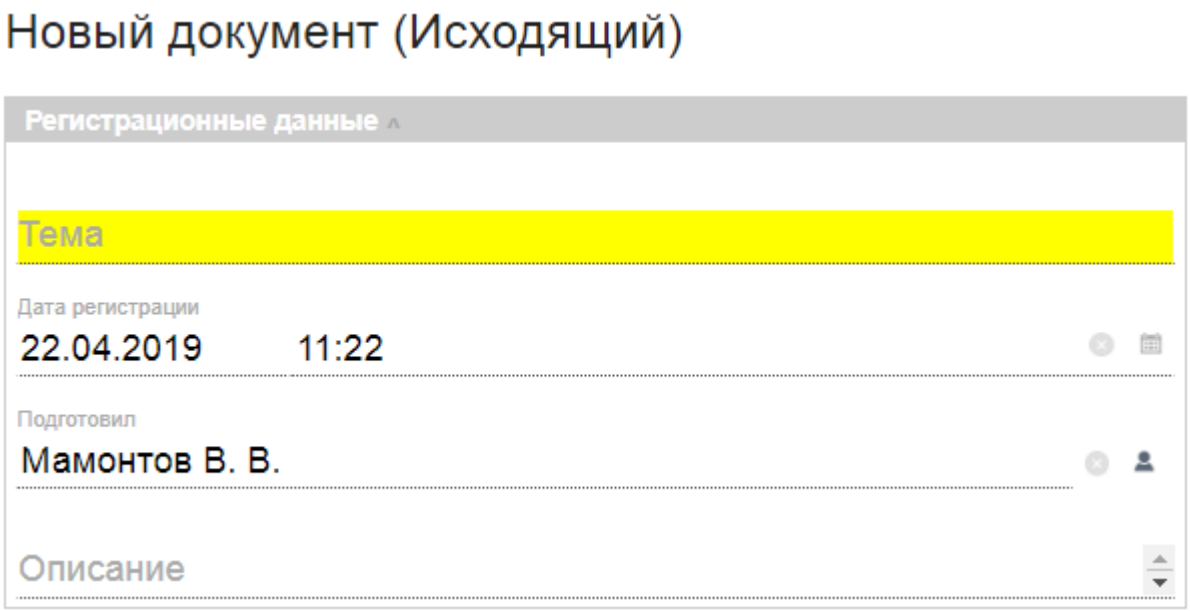


Рисунок 117. Разметка карточки с собственным стилем поля Тема

Переопределение реализации стандартного элемента управления собственной

Стандартная реализация элементов управления Web-клиента может не всегда отвечать требованиям пользователя. В качестве решения, программист может реализовать собственные типы элементов управления и использовать их в разметках, или переопределить стандартную реализацию существующих элементов управления. В последнем случае разметки остаются без изменений, а изменяется клиентская часть элементов управления.

Переопределение реализации стандартных элементов управления упрощает внедрение требуемой клиентской функциональности, но усложняет процесса обновления версии Web-клиента.

Данные способ разработки не позволяет переопределить серверную часть элемента управления и его описатель.

Чтобы переопределить стандартную реализацию элемента управления собственной:

1. Скопируйте файлы `.tsx` с реализацией клиентской части стандартного элемента управления и файлы зависимостей в отдельный каталог.
2. Внесите требуемые изменения в файлы.

   Изменять тип элемента управления не нужно.
3. Сохраните изменения и опубликуйте компоненты в каталоге собственного решения на сервере Web-клиента стандартным образом.
4. При обновлении версии Web-клиента выполните следующие действия:
   a. Проанализируйте исходный код стандартной реализации на наличие изменений[14] относительной версии, для которой создавалось переопределение.
   b. Если изменения затронули настройки или формат данных элемента управления, внесите соответствующие изменения в собственный компонент, переопределяющий стандартный.

Другие примеры

Руководство по разработке решений Web-клиента

Другие примеры

Описание прочих примеров разработки.

- Специальный сервис логики этапа согласования
- Разработка расширения для DVWebTool и DVSupService
- Web-сервис для взаимодействия с платформой
- Больше примеров категории в навигационном меню слева…

Работа с машиночитаемой доверенностью

Данный пример, расположенный в репозитории Docsvision на GitHub, содержит методы работы с МЧД для Web-клиента и для Windows-клиента.

Ссылка на пример на GitHub: PowersOfAttorney.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Модуль Базовые объекты версии 6.1, сборка 176 и выше.
- Модуль Web-клиент версии 6.1, сборка 651 и выше.

Ограничения примера

1. Виды передоверия в примере предназначены только для ознакомления. Для их использования необходима доработка под ситуации, не требующие нотариального удостоверения. По умолчанию создание карточек данных видов запрещено.
2. Формат 5.01 не реализован.
3. Для формата 002 не реализованы следующие сценарии:
   a. Основная доверенность

Доверитель	**Лицо, действующее без доверенности от имени доверителя**	**Представитель**
Российское юридическое лицо	Юридическое лицо	Организация / Индивидуальный предприниматель
Иностранное юридическое лицо	Все типы	Все типы

Доверитель	**Лицо, действующее без доверенности от имени доверителя**	**Представитель**
Индивидуальный предприниматель	Все типы	Все типы
Физическое лицо	Все типы	Все типы

b. Передоверие

Лицо, передавшее полномочия	**Лицо, получившее полномочия**
Российское юридическое лицо	Все типы
Индивидуальный предприниматель	Все типы
Физическое лицо	Организация/ Индивидуальный предприниматель

c. Нотариальное удостоверение доверенностей.

4. Для формата EMCHD_1 (003) не реализованы следующие сценарии:
   a. Основная доверенность

Доверитель	**Единоличный исполнительный орган**	**Представитель**
Российское юридическое лицо	Управляющая компания/ Индивидуальный предприниматель	Юридическое лицо/ Индивидуальный предприниматель/ Филиал (обособленное подразделение) российского юридического лица/ Филиал (аккредитованное представительство) иностранного юридического лица
Иностранное юридическое лицо	Все типы	Все типы

Доверитель	Единоличный исполнительный орган	Представитель
Индивидуальный предприниматель	Все типы	Все типы
Физическое лицо	Все типы	Все типы

b. Передоверие

Лицо, передавшее полномочия	Лицо, получившее полномочия
Российское юридическое лицо	Все типы
Индивидуальный предприниматель	Все типы
Физическое лицо	Юридическое лицо/ Индивидуальный предприниматель/ Филиал (обособленное подразделение) российского юридического лица/ Филиал (аккредитованное представительство) иностранного юридического лица
Филиал (обособленное подразделение) российского юридического лица	Все типы
Филиал (аккредитованное представительство) иностранного юридического лица	Все типы

c. Нотариальное удостоверение доверенностей.

Описание примера

Пример включает компоненты:

- **PowersOfAttorney › PowersOfAttorneyServerExtension** — папка с серверным расширением Web-клиента, в котором реализованы функции создания СКД из демонстрационной карточки доверенностей.
- **PowersOfAttorney › PowersOfAttorneyWebExtension** — папка с клиентским расширением, в котором реализованы обработчики смены

состояния ПКД и управления МЧД, видимостью и обязательностью полей.

- **PowersOfAttorney › PowersOfAttorneyDesignerExtension** — папка с расширением конструктора разметок, в котором реализовано описание элемента управления для группового подписания.
- **PowersOfAttorney › PowersOfAttorney.UserCard.Common** — папка с проектом `PowersOfAttorney.UserCard.Common`, который используется и для Web-клиента и для Windows-клиента.
- **PowersOfAttorney › PowersOfAttorney.Scripts** — содержит файл скрипта для Windows клиента.
- **PowersOfAttorney › Data › PowersOfAttorneySolution** — папка с решением, включающим разметки тестовой карточки доверенности.
- **PowersOfAttorney › Data › SolutionOfPOA.sol** — решение, добавляющее в систему Docsvision новые виды карточки *Документ* (*Доверенность* и *Передоверие*) и необходимые расширенные метаданные.

Сборка проекта

i. Сборка серверной части.

1. Откройте решение **PowersOfAttorney › PowersOfAttorneyServerExtension.sln**.
2. Соберите проект `PowersOfAttorneyServerExtension`.

ii. Сборка клиентской части.

1. Откройте в командной строке папку **PowersOfAttorney › PowersOfAttorneyWebExtension**.
2. Выполните команды:

```
$ npm install
$ npm update
$ npm run build:prod
```

iii. Публикация компонентов на сервере Web-клиент.

1. Скопируйте папку `PowersOfAttorney\SamplesOutput\Content\Modules\PowersOfAttorneyWebExtension` в `Каталог-установки-Web-клиента\Content\Modules`.
2. Скопируйте файл `PowersOfAttorney\SamplesOutput\Extensions\PowersOfAttorneyServerExtension\Power`

sOfAttorneyServerExtension.dll в Каталог-установки-Web-клиента\Extensions.

3. Скопируйте файл PowersOfAttorney\SamplesOutput\Extensions\PowersOfAttorneyServerExtension\PowersOfAttorneyServerExtension.resources.dll в Каталог-установки-Web-клиента\Extensions\ru.
4. Скопируйте файл PowersOfAttorney\SamplesOutput\Extensions\PowersOfAttorney.UserCard.Common\PowersOfAttorney.UserCard.Common.dll в Каталог-установки-Web-клиента\Extensions.
5. Скопируйте файл PowersOfAttorney\SamplesOutput\Extensions\PowersOfAttorney.UserCard.Common\PowersOfAttorney.UserCard.Common.resources.dll в Каталог-установки-Web-клиента\Extensions\ru.

iv. Перезапустите dvappserver.

Проверка примера

Для проверки примера требуется установить версию Backoffice и Web-клиент с поддержкой СКД.

1. В Менеджере решений импортируйте решение *Машиночитаемая доверенность* — PowersOfAttorney\Data\SolutionOfPOA.sol. Рекомендуется использовать версию Менеджера решений версии 6.1 и выше.

   Компоненты данного решения:

   - Справочник видов:
     - *Документ — МЧД Доверенность (версия 002)*, *Передоверие (версия 002)*, *Доверенность (версия EMHCD_1)*, *Передоверие (версия EMHCD_1)*.
     - *Задание — Задание КС — На подписание МЧД*, *На согласование МЧД*.
   - Метаданные: Полномочия, МЧД формат единой формы, Текстовые полномочия (а также все подчиненные).
   - Конструктор правил нумерации: МЧД.
   - Поиск: МЧД (МЧД все, МЧД действующие, МЧД — я автор, МЧД — я подписант, МЧД отозванные, МЧД мои, Поиск МЧД)
   - Представление: МЧД → МЧД единого формата — представление
   - Папка *Доверенность* для хранения маршрутов согласования и подчиненные папки для работы с доверенностями.

- Согласование: маршрут "согласование и подписание МЧД единого формата" — согласование настроено на группу юридического отдела (справочник сотрудников), подписание настроено на группу генеральные директора (справочник сотрудников). Согласование параллельное, без ограничений по времени. Можно убрать участников при отправке на согласование.

2. В справочнике видов (тип: *Согласование*, вид: *Усовершенствованное согласование*) настройте согласование для видов карточек МЧД. В окне *Настройка способа создания карточки* выберите шаблон *Согласование и подписание МЧД единый формат*.
3. В программе Конструктор Web-разметок импортируйте решение *Машиночитаемая доверенность*. Для этого выберите файл `Solution.xml` в папке решения `PowersOfAttorney\Data\PowersOfAttorneySolution\`.
4. Проверьте, что в Справочнике сотрудников заполнены поля для участников МЧД как описано в /dv6/webclient/6.1/user/directories/staff/employee-fields/#attorney[пользовательской документации].
5. В Web-клиенте создайте карточку документ вида *Доверенность (версия 002)*. Заполните обязательные поля.
   Если заполнены не все обязательные поля, при сохранении будет выдано предупреждение.

   Чтобы создать свою разметку, необходимо:

   - Добавить скрипт для отображения в модальном окне незаполненных обязательных полей `showRequiredFields` в качестве обработчика события **`Перед началом сохранения карточки`** `root` для разметок создания и/или редактирования в конструкторе разметок.
   - Добавить скрипт `customizePowerOfAttorneyCardForEditLayout` в качестве обработчика события **`После загрузки всех ЭУ`** для root во всех разметках создания и редактирования карточки доверенности.
   - Добавить скрипт `customizePowerOfAttorneyCardForViewCard` в качестве обработчика события **`После загрузки всех ЭУ`** для root во всех разметках просмотра карточки доверенности.
   - Добавить скрипт `customizeSubstitutionPowerOfAttorneyCardForEditLayout` **`После загрузки всех ЭУ`** для root во всех разметках просмотра карточки передоверия.
   - Добавить скрипт `customizeSubstitutionPowerOfAttorneyCardForViewLayout` в качестве обработчика события **`После загрузки всех ЭУ`** для root во всех

разметках просмотра карточки передоверия.

6. В Web-клиенте создайте карточку документ вида *Доверенность (версия EMCHD_1)*. Заполните обязательные поля. Если заполнены не все обязательные поля, при сохранении будет выдано предупреждение.

   Чтобы создать свою разметку, необходимо:

   - Добавить скрипт для отображения в модальном окне незаполненных обязательных полей `showRequiredFields` в качестве обработчика события **Перед началом сохранения карточки** `root` для разметок создания и/или редактирования в конструкторе разметок.
   - Добавить скрипт `customizeSingleFormatPowerOfAttorneyForEditLayout` в качестве обработчика события **После загрузки всех ЭУ** `root` для разметок создания и редактирования карточки доверенности в конструкторе разметок.
   - Добавить скрипт `customizeSingleFormatPowerOfAttorneyForViewLayout` в качестве обработчика события **После загрузки всех ЭУ** `root` для разметок просмотра, описания и справки карточки доверенности в конструкторе разметок.
   - Добавить скрипт `customizeSingleFormatSPOACardForEditLayout` в качестве обработчика события **После загрузки всех ЭУ** `root` для разметок создания и редактирования карточки передоверия в конструкторе разметок.
   - Добавить скрипт `customizeSingleFormatSPOACardForViewLayout` в качестве обработчика события **После загрузки всех ЭУ** `root` для разметок просмотра, описания и справки карточки передоверия в конструкторе разметок.
   - Добавить скрипт `customizeSingleFormatPowerOfAttorneyForLocationLayout` в качестве обработчика события **После загрузки всех ЭУ** `root` для разметки локация карточки доверенности в конструкторе разметок.
   - Добавить скрипт `customizeSingleFormatPowerOfAttorneyForLocationLayout` в качестве обработчика события **После загрузки всех ЭУ** `root` для разметки локация карточки передоверия в конструкторе разметок.

7. Нажмите кнопку создания МЧД. В результате будет создана МЧД, связанная с текущей карточкой документа.

   Чтобы создать свою разметку, необходимо:

   - Добавить скрипт `createPowerOfAttorney` в качестве обработчика события **При щелчке** для кнопки в разметке просмотра карточки доверенности (версия 002) в конструкторе разметок.
   - Добавить скрипт `createRetrustPowerOfAttorney` в качестве обработчика события **При щелчке** для кнопки в разметках просмотра карточки

передоверия (версия 002) в конструкторе разметок.

- Добавить скрипт `createEMCHDPowerOfAttorney` в качестве обработчика события **При щелчке** для кнопки для разметок просмотра карточки доверенности (версия EMCHD_1) в конструкторе разметок.
- Добавить скрипт `createEMCHDRetrustPowerOfAttorney` в качестве обработчика события **При щелчке** на кнопку для разметок просмотра карточки передоверия (версия EMCHD_1) в конструкторе разметок.

8. Нажмите кнопку экспорта МЧД. На компьютер будет сохранён архив, содержащий файл МЧД в формате XML.

   Чтобы создать свою разметку, необходимо:

   - Добавить скрипт `exportPowerOfAttorneyWithoutSignature` в качестве обработчика события **При щелчке** для кнопки в разметке просмотра карточки доверенности и передоверия в конструкторе разметок.

9. Если планируется отправлять доверенность на регистрацию в распределённый реестр ФНС через провайдеров внешнего ЭДО необходимо выполнить соответствующие настройки.

   - Для работы через Контур.Диадок выполните настройки описанные в документации модуля "Модуль интеграции с операторами ЭДО", раздел "/dv6/edi/6.1/admin/attorney-settings/[Настройка работы с МЧД]".
     - Для этого случая, в примере решения есть преднастроенные разметки: *Доверенность версии 002 (просмотр)_Контур.Диадок* и *Доверенность версии EMCHD_1 (просмотр)_Контур.Диадок*.
   - Для работы через Контур.Доверенность выполните настройки описанные в документации модуля "Интеграции с реестром МЧД", раздел "/dv6/m4d-registry/6.1/admin/connection-settings/[Настройка Модуля интеграции с реестром МЧД]".
     - Для этого случая, в примере решения есть преднастроенные разметки: *Доверенность версии 002 (просмотр)_Контур.Доверенность* и *Доверенность версии EMCHD_1 (просмотр)_Контур.Доверенность*.

10. Нажмите кнопку подписания МЧД. Будет предложено выбрать сертификат подписи, выполнится подписание МЧД.

    - Для создания своей разметки нужно добавить скрипт `signPowerOfAttorney` в качестве обработчика события **При щелчке** на кнопку для разметок просмотра карточки доверенности и передоверия в конструкторе разметок.

- При работе через Контур.Диадок, для подписания и последующей регистрации доверенности по файлу нужно добавить скрипт `signAndSendPowerOfAttorneyToRegistrationAsFile` в качестве обработчика события **При щелчке** на кнопку для разметки просмотра карточки доверенности.
- Если требуется разделить события подписания и отправки, можно воспользоваться скриптом `sendPowerOfAttorneyToRegistrationAsFile` (он выполняет только отправку уже подписанной доверенности в реестр через Контур.Диадок).
- При работе через Контур.Доверенность, для подписания и последующей регистрации доверенности по файлу нужно добавить скрипт `signAndSendForRegistrationToRegistry` в качестве обработчика события **При щелчке** на кнопку для разметки просмотра карточки доверенности.
- Если требуется разделить события подписания и отправки, можно воспользоваться скриптом `sendForRegistrationToRegistry` (он выполняет только отправку уже подписанной доверенности в реестр через Контур.Доверенность).

11. Нажмите кнопку экспорта МЧД. На компьютер будет сохранён архив, содержащий файл МЧД в формате XML и его подпись.

    Чтобы создать свою разметку, необходимо:

    - Добавить скрипт `exportPowerOfAttorneyWithSignature` в качестве обработчика события **При щелчке** для кнопки в разметки просмотра карточки доверенности и передоверия.

12. Нажмите кнопку отзыва МЧД.

    Чтобы создать свою разметку, необходимо:

    - Добавить скрипт `revokePowerOfAttorney` в качестве обработчика события **При щелчке** на кнопку для разметки просмотра карточки доверенности и передоверия (версия 002) в конструкторе разметок.

    Чтобы создать свою разметку с отзывом без заявления, необходимо:

    - Добавить скрипт `revokePowerOfAttorneyWithoutApplication` в качестве обработчика события **При щелчке** на кнопку для разметки просмотра карточки доверенности и передоверия в конструкторе разметок.

    При работе через Контур.Доверенность

    - Для формирования, подписания и последующей отправки заявления на отзыв в ЦПРР ФНС нужно добавить скрипт `revokeAndRecallPowerOfAttorney` в

качестве обработчика **При щелчке** на кнопку отзыва для разметки просмотра карточки доверенности.

Если требуется разделить события формирования заявления и его отправки:

- Можно воспользоваться скриптом `recallPowerOfAttorney`. Скрипт выполняет только отправку уже сформированного и подписанного заявления на отзыв через Контур.Доверенность.

13. Нажмите кнопку экспорта заявления на отзыв для доверенности и передоверия (версия 002). На компьютер будет сохранён архив, содержащий файл заявления на отзыв в формате XML и его подпись.

    Чтобы создать свою разметку, необходимо:

    - Добавить скрипт `exportApplicationForRevocation` в качестве обработчика события **При щелчке** на кнопку для разметки просмотра карточки доверенности (версия 002) и передоверия (версия 002) в конструкторе разметок.

14. Нажмите кнопку удаления пользовательской карточки доверенности.

    Из пользовательской карточки доверенности также удаляется системная карточка доверенности.

    Чтобы создать свою разметку, необходимо:

    - Добавить скрипт `deletePowerOfAttorney` в качестве обработчика события **Перед удалением карточки** для `root`.

15. Доверенность можно подписать в ходе согласования.

    Чтобы подписать доверенность в ходе согласования, необходимо:

    - Отправить Доверенность на согласование как описано в /dv6/webclient/6.1/user/docs-approval/[пользовательской документации].
    - Нажать кнопку **Подписать** в *Задании*. В результате сформируется СКД, затем подпишется доверенность, ПКД перейдет в статус `Подписана`, а задание перейдет в статус `Завершено`.
    - Для создания своей разметки нужно добавить скрипт `signAndSendPowerOfAttorneyToRegistrationAsFileFromTask` в качестве обработчика события `Перед выполнением операции` на ЭУ `Автомат состояния` для подписания и последующей регистрации доверенности по файлу.
      Для подписания без регистрации используйте скрипт `signPowerOfAttorneyFromTask`.

Права доступа к СКД определяются дискреционной моделью и наследуются от ПКД. Права переходят по *Сильной ссылке*, в которую прописывается `InstanceID` СКД благодаря наличию поля `POASysCardId` типа `RefCardID`.

Пример скриптов для Windows клиента

1. Скрипт находится в файле `CardDocumentДоверенность__версия_EMHCD_1_Script.cs` в проекте `PowersOfAttorney.Scripts`
2. Проект нужен только для проверки компилируемости скрипта. Ссылки на сборку `PowersOfAttorney.Scripts.dll` добавлять не надо.
3. Скрипт необходимо скопировать в справочник скриптов для двух видов (для доверенности EMHCD и передоверия EMHCD). Если у родительского вида для этих видов нет скрипта, надо открыть его и сгенерировать для него скрипт по умолчанию.
4. Необходимо закомментировать в файле скрипта для обоих видов строчку

   ```
   using CardDocumentМЧДScript = DocsVision.BackOffice.WinForms.ScriptClassBase;
   ```

   Строка нужна только для компиляции файла скрипта в составе проекта `PowersOfAttorney.Scripts`.

5. Скрипты для этих видов отличаются только названиями классов. Необходимо для каждого вида оставить только одно соответствующее ему название класса (см. комментарий в скрипте).
6. В скриптах необходимо добавить ссылку на сборку `PowersOfAttorney.UserCard.Common.dll`, которую также необходимо распространить на все клиентские рабочие места.
7. В конструкторе разметок необходимо добавить кнопки (например в ленту). Названия кнопок должны соответствовать обработчикам в скрипте.

   В скрипте обработчики выглядят как `имяКнопки__ItemClick();` Если названия кнопки будут соответствовать обработчикам, то обработчики к кнопкам привяжутся автоматически, вручную их создавать не надо.

8. Можно привязать к кнопкам соответствующие операции, чтобы кнопки были доступны только в тех состояниях, когда их нажатие имеет смысл.

API для работы с МЧД

Для работы с МЧД можно использовать перечисленные ниже классы API.

- В `DocsVision.BackOffice.Cards.Web.Model` добавлены новые для работы с машиночитаемыми доверенностями:
  - `CreatePowerOfAttorneyFnsDovBbRequest` с полями:
    - `PowerOfAttorneyFnsDovBbData powerOfAttorneyData` — данные создаваемой доверенности.
    - `Guid representative` — представитель.
    - `Guid signer` — подписант.
    - `Guid parentPowerOfAttorney` — идентификатор родительской доверенности.
  - `SignPowerOfAttorneyRequest` с полями:
    - `Guid PowerOfAttorneyId` — идентификатор доверенности.
    - `byte[] Signature` — данные подписи.
- В контроллер `PowerOfAttorneyApiController` добавлены новые методы:
  - `POST CreatePowerOfAttorney(CreatePowerOfAttorneyFNSDOVBBRequest request)` — вызывает `IPowerOfAttorneyService.CreatePowerOfAttorney` (создание доверенности) с передачей полей из `CreatePowerOfAttorneyFNSDOVBBRequest`.
  - `GET GetMachineReadablePowerOfAttorney(Guid powerOfAttorneyId)` — возвращает МЧД доверенности для подписания.
  - `POST AttachSignatureToPowerOfAttorney(AttachSignatureToPowerOfAttorneyRequest)` — вызывает `IPowerOfAttorneyService.AttachSignature`, загружающий подпись в существующую доверенность и изменяющий статус доверенности.

Подробнее про использование API для работы с МЧД можно узнать из описания веб-API, см. раздел "Как работать с описанием веб-API".

Специальный сервис логики этапа согласования

Данный раздел содержит описание примера реализации специального сервиса логики этапа согласования.

Ссылка на пример на GitHub: CustomStageService.

Сервис выполняет следующие функции:

- Подменяет вычисление согласующих.

- Показывает вариант настройки действий после создания задания на примере дополнения содержания задания.

Пример рассчитан на версии модулей Служба фоновых операций и Windows-клиент версии 6.1 или выше.

Перечень необходимых инструментов:

- Microsoft Visual Studio 2022.

Сборка и установка

1. Откройте `/Samples.sln`.
2. Соберите проект `Others > CustomStageService`.

Пример переопределения сервиса этапа

```
namespace CustomStageService
{
    public class CustomStageService : ApprovalStageService
    {
        public override IEnumerable<ApprovalStageApprover> GetStageApprovers
(ApprovalStage approvalStage, Document document) ①
        {
            List<ApprovalStageApprover> approvers = new List<ApprovalStageApprover>();

            IStaffService staffService = Context.GetService<IStaffService>();
            StaffEmployee employee = staffService.FindEmpoyeeByAccountName
(@"Test\TestEmployee");

            approvers.Add(CreateApprover(employee)); ②

            return approvers;
        }
        protected override void OnTaskCreated(ApprovalStage approvalStage, ApprovalPath
approvalPath, Reconcile reconcileCard, Task task) ③
        {
            task.MainInfo.Content += Environment.NewLine + "Тест расширения"; ④
        }
    }
}
```

① Переопределяем вычисления согласующих для этапа.

Параметры:

- `approvalStage` — текущий этап,

- `document` — документ.

② `CreateApprover` — штатный метод `ApprovalStageService` создающий запись согласующего из сотрудника.

③ Дополняем создание задания согласования.

Параметры:

- `approvalStage` — текущий этап.
- `approvalPath` — маршрут.
- `reconcileCard` — согласование.
- `task` — созданное задание.

④ Дополним содержание задания.

Проверка примера

1. В файле `WebCSamples\Others\CustomStageService\CustomStageService.cs` заменить `Test\TestEmployee` на одну из своих учётных записей, присутствующих в Docsvision.
2. Пересобрать расширение.
3. Добавить на сервере библиотеку `CustomStageService.dll` из каталога `bin\Debug\net6.0`, если в VisualStudio выбрана конфигурация сборки "Debug" или `bin\Release\net6.0`, если выбрана конфигурация "Release".
   - в папку common на сервер, где установлен WorkerService (`/lib/docsvision/common` на Linux)
   - в папку, где установлен Windows-клиент, с которого производится настройка (например `C:\Program Files (x86)\Docsvision\Client`).
4. Открыть шаблон карточки любого этапа (например, "Согласование с рецензированием" из стандартного согласования модуля "Управление документами"), перейти на вкладку "Специальная логика этапа", в поле "Имя сервиса" выбрать библиотеку.
5. Настроить и запустить согласование с данным этапом в маршруте. В результате на данном этапе в качестве согласующего будет выбран пользователь, указанный на шаге 1 (например, если используется этап согласования, то этому пользователю придет задание на согласование). Также в созданном задании в описании появится текст "Тест расширения".

Разработка расширения для DVWebTool и DVSupService

Программа *DVWebTool* устанавливается на клиентскую машину только в ОС Microsoft Windows. Для ОС Linux используется *DVSupService*, см. подробнее /dv6/webclient/6.1/user/prepare-add-components/#dvSupService[Установка и запуск службы DVSupService].

Ссылка на пример на GitHub: Watermark.

Расширения для *DVSupService* и *DVWebTool* можно распространять с помощью пакетов Debian, с подробной инструкцией и необходимыми материалами можно ознакомиться в репозитории на GitHub.

Дополнительным компонентом модуля Web-клиент является программа *DVWebTool*, которая предоставляет функциям Web-клиента доступ к устройствам и программному обеспечению, установленному на компьютере пользователя Web-клиента. Программа *DVWebTool* также может быть использована для обработки произвольных задач на пользовательском компьютере.

Программа *DVWebTool* является приложением для рабочего стола с возможностью расширения функциональности. Расширение *DVWebTool* представляет собой DLL-файл содержащий ожидаемый класс с веб-методами. Данные методы могут вызываться клиентскими компонентами Web-клиента (есть стандартный метод JSAPI для вызова) или другими способами.

В качестве примера будет разработано расширение, реализующее функцию добавления водяных знаков в PDF-файлы карточки *Документ*.

Алгоритм работы расширения:

1. При вызове функции расширения Клиент передаёт идентификаторы PDF файлов карточки *Документ* (идентификаторы карточек *Файл* с версиями, а не идентификаторы собственно файлов) и идентификатор самой карточки *Документ*.
2. Расширение *DVWebTool* загружает PDF-файлы на компьютер пользователя с помощью функций, предоставляемых Web-клиентом.
3. Расширение *DVWebTool* добавляет в полученные PDF-файлы водяные знаки используя функции сторонней библиотеки для работы с PDF.
4. Расширение *DVWebTool* прикрепляет полученные файлы с водяными знаками с помощью функций, предоставляемых Web-клиентом.

Расширение не должно изменять существующие файлы карточки: полученные PDF-файлы с водяными знаками должны быть добавлены в карточку в качестве дополнительных.

Для реализации функций, указанных в пп. 2 и 4, потребуется разработать расширение Web-клиента с соответствующей функциональностью.

Также для примера необходимо разработать клиентское расширение, которое будет вызывать функции расширения *DVWebTool* с передачей им идентификаторов PDF-файлов *Документа*, в которые нужно добавить водяные знаки.

Полный код примера доступен в репозитории на GitHub.

Расширение Web-клиента

Для работы расширения *DVWebTool* необходимо, чтобы Web-клиент по запросу возвращал файл из карточки *Файл* с версиями с переданным идентификатором, а также мог прикреплять переданные файлы к карточке *Документ*.

Требуемая функциональность может быть реализована с помощью расширения Web-клиента.

Ниже приведена часть исходного кода контроллера, реализованного в серверном расширении.

Файл Others/Watermark/WatermarkServerExtension/Controllers/FileOperationsController.cs

```
        public HttpResponseMessage GetFile(Guid fileCardID)
        {
            FileReader fileReader = fileService.GetFileReader(fileCardID);

            if (fileReader.FileID == Guid.Empty)
                return new HttpResponseMessage(HttpStatusCode.NotFound);

            HttpResponseMessage response = CreateResponseForFile(fileReader);
            return response; ①
        }

        [HttpPost]
        public async Task<ActionResult> AddFile([FromForm(Name = "file")]
FormFileCollection formFiles, Guid cardId) ②
        {

            string rootPath = CreateAndGetTempFolder();
```

```
            try
            {
                List<string> files = await SaveFilesFromResponse(formFiles); ③

                await fileService.AddFilesToCard(cardId, files);
                return Ok();

            }
            catch (Exception e)
            {
                return Problem(statusCode: (int)HttpStatusCode.InternalServerError,
detail: e.Message);
            }
        }
```

① Создаёт ответ с файлом.

② Добавляет в карточку файлы из запроса.

③ Добавляемые файлы.

В контроллере используется сервис `IFileService`, реализация которого приведена ниже. Код дополнительных методов смотрите в полном примере на GitHub.

Файл `Others/Watermark/WatermarkServerExtension/Services/FileService.cs`

```
    public class FileService : IFileService
    {
        private readonly ICurrentObjectContextProvider currentObjectContextProvider;
        private Guid DOCUMENT_CARD_ID = new Guid("B9F7BFD7-7429-455E-A3F1-94FFB569C794");

        public FileService(ICurrentObjectContextProvider currentObjectContextProvider) {
            this.currentObjectContextProvider = currentObjectContextProvider;
        }

        public async System.Threading.Tasks.Task<IEnumerable<Guid>> AddFilesToCard(Guid
cardID, List<string> files) ①
        {
            var document = GetDocumentCard(cardID); ②
            ObjectContext objectContext = GetObjectContext();

            ILockService lockService = GetLockService(objectContext);

            if (lockService.IsObjectLockedByAnotherUser(document)) {
                throw new Exception($"Карточка {cardID} заблокирована другим
пользователем");
            }
```

```
            if (lockService.LockObjectBase(document) == false) {
                throw new Exception($"Не удалось заблокировать карточку {cardID}");
            }

            IDocumentService documentService = GetDocumentService(objectContext);

            return await System.Threading.Tasks.Task.Run(() =>
            {
                IEnumerable<Guid> documentFileIds;
                try
                {
                    IEnumerable<DocumentFile> documentsFiles = documentService
.AddAdditionalFiles(document, files);
                    objectContext.AcceptChanges();
                    documentFileIds = documentsFiles.Select(t => t.FileId);
                }
                catch (Exception ex)
                {
                    throw new Exception($"Ошибка при добавлении файлов в карточку {
cardID}\n {ex.Message}");
                }
                finally {
                    lockService.UnlockObject(document);
                }

                return documentFileIds; ③
            });
        }

        public FileReader GetFileReader(Guid fileCardID) ④
        {
            ObjectContext objectContext = GetObjectContext();
            IVersionedFileCardService versionedFileCardService =
GetVersionedFileCardService(objectContext);

            VersionedFileCard fileCard = versionedFileCardService.OpenCard(fileCardID);
⑤

            Guid fileID = fileCard.CurrentVersion.Id; ⑥

            UserSession userSession = GetUserSession();

            if (userSession.FileManager.FileExists(fileID) == false)  ⑦
                return new FileReader();

            var file = userSession.FileManager.GetFile(fileID); ⑧

            return new FileReader() { ⑨
```

```
                FileID = fileID,
                FileName = file.Name,
                Stream = file.OpenReadStream()
            };
        }
```

① Добавляет файлы с файловой системы в карточку `cardID`.

② Получает карточку, к которой прикрепляются файлы.

③ Возвращает идентификаторы добавленных файлов с версиями.

④ Получает указатель для чтения файла из карточки файла с версиями `fileCardID`.

⑤ Получение файла карточки с версиями.

⑥ Получение идентификатора файла из последней версии.

⑦ Если файла нет, возвращается пустой указатель.

⑧ Запрашивается файл текущей версии.

⑨ Возвращается указатель для чтения файла.

Расширение программы DVWebTool

Когда реализовано расширение Web-клиента, предоставляющее и записывающее файлы карточек, может быть реализовано расширение *DVWebTool*, использующее данные функции.

Расширение *DVWebTool* представляет собой сборку, в которой реализован интерфейс `Docsvision.DVWebTool.WebServices.IServiceManager`. Данный интерфейс определяет метод `Register`, который регистрирует контроллеры с необходимыми функциями во внутреннем веб-сервере *DVWebTool*, и поле `DisplayName` с названием расширения.

Ниже приведён код класса, реализующего интерфейс `IServiceManager` в данном примере.

Файл `Others/Watermark/WatermarkWebToolExtension/WatermarkManager.cs`

```
    public class WatermarkManager : IServiceManager
    {
        public string DisplayName => "Watermark to PDF"; ①

        public void Register(WebSocketSharper.Server.WebSocketServer server) ②
        {
            server.AddWebSocketService<WatermarkController>("/Watermark"); ③
```

```
        }
    }
```

① Название расширения для информации в окне "О программе".

② Регистрация контроллера расширения.

③ Регистрация контроллера `PDFWatermarkController` для маршрута `Watermark`.

В данном примере выполняется регистрация контроллера `WatermarkController` для обработки запросов, поступающих по пути `/Watermark`.

Контроллер, передаваемый в `AddWebSocketService`, должен быть производным типа `Docsvision.DVWebTool.WebServices.BaseService`. При его реализации необходимо зарегистрировать в `BaseService.actions` веб-методы, с помощью которых будут вызываться функции *DVWebTool*, предоставляемые расширением. Данные методы будут доступны для вызова по протоколу `WebSocket` по адресу `ws://localhost:/%Адрес контроллера%/%Название метода%`.

Ниже приведена часть реализации контроллера `WatermarkController`, содержащего методы обработки входящих запросов на добавление водяного знака.

Файл `Others/Watermark/WatermarkWebToolExtension/WatermarkController.cs`

```
    public class WatermarkController : BaseService
    {
        private readonly ServiceProvider serviceProvider;
        private static readonly Logger Logger = LogManager.GetCurrentClassLogger();

        private string WATERMARK = "Секретно"; ①

        public WatermarkController()
        {
            serviceProvider = new ServiceProvider();
            Init();
        }

        private void Init() ②
        {
            actions.Add(nameof(AddWatermarkToFiles), AddWatermarkToFiles);
        }

        private WebServiceResponse AddWatermarkToFiles(WebServiceRequest
webServiceRequest, JObject data) ③
        {
            Logger.Info("Получено задание на добавление водяного знака");
```

```
            if (data == null)
            {
                return CreateBadResponse("С клиента не переданы данные для работы");
            }

            AddWatermarkRequest request; ④
            try
            {
                request = data.ToObject<AddWatermarkRequest>();
            }
            catch
            {
                Logger.Error($"Ошибка преобразования полученного сообщения: {data}");
                return CreateBadResponse("Поступивший запрос не соответствует ожидаемому
формату");
            }

            string doneInfo;

            try
            {
                doneInfo = HandleRequest(request).Result;
            }
            catch (Exception ex)
            {
                return CreateBadResponse(ex.Message);
            }

            return CreateEndProcessResponse(request.CardID, $"Водяные знаки добавлены в
файлы:<p/>{doneInfo}");
        }


        private async Task<string> HandleRequest(AddWatermarkRequest request) ⑤
        {
            var connectionService = new ConnectionToWebClient(request.ServerAddress,
request.AccessToken);

            List<string> files = new List<string>();

            foreach (var fileId in request.FileIDs) ⑥
            {
                try
                {
                    string pathToFile = await connectionService.PullFile(fileId);
                    files.Add(pathToFile);
                }
```

```
                catch (Exception ex)
                {
                    Logger.Error(ex.Message);
                    throw new Exception($"Не удалось получить из карточки файл с
идентификатором {fileId}");
                }
            }

            var watermarkService = new WatermarkService(); ⑦

            List<Task<string>> processes = new List<Task<string>>(); ⑧
            string doneInfo = "";

            foreach (var file in files)
            {
                try
                {
                    processes.Add(watermarkService.AddWatermark(file, WATERMARK));
                    doneInfo += $" {Path.GetFileName(file)}<p/>";
                }
                catch (Exception ex)
                {
                    Logger.Error(ex.Message);
                    throw new Exception($"Не удалось добавить водяной знак в файл {Path
.GetFileName(file)}");
                }
            }

            var filesWithWatermark = await Task.WhenAll(processes);

            try
            {
                await connectionService.PushFiles(request.CardID, filesWithWatermark); ⑨
            }
            catch (Exception ex)
            {
                Logger.Error(ex.Message);
                throw new Exception($"Не удалось сохранить файлы в карточке");
            }
            return doneInfo;
        }
```

① Текст водяного знака.

② Регистрация методов контроллера `PDFWatermarkController`. Название метода регистрозависимое.

③ Веб-метод добавления водяного знака. Метод должен принимать два параметра: `WebServiceRequest` и `JObject`. Данные передаются в `data`.

④ Загружаем данные из полученного запроса в модель `AddWatermarkRequest`.

⑤ Обработчик запроса на добавление водяного знака. Получает данные запроса. Возвращает строку с названиями файлов, в которые добавлены запросы.

⑥ Загружаем с Web-клиента файлы, идентификаторы которых переданы в запросе.

⑦ Сервис для работы с водяными знаками.

⑧ Добавление водяных знаков в файлы из списка `files`.

⑨ Отправка запроса на прикрепление файлов `filesWithWatermark` к карточке `request.CardID`.

Контроллер `WatermarkController` использует функции двух сервисов:

- `ConnectionToWebClient` для получения файлов из карточек Файл с версиями и загрузки файлов в карточки Документ. Данный сервис использует функции, реализованного серверного расширения Web-клиента.
- `WatermarkService` для добавления водяных знаков в PDF файлы.

Реализации данных сервисов смотрите в исходных кодах примера на GitHub.

После публикации расширения *DVWebTool*, зарегистрированный в расширении метод `AddWatermarkToFiles` контроллера `Watermark` может быть вызван из клиентского расширения Web-клиента с помощью сервиса `DVWebToolConnection`.

Клиентское расширение

В качестве примера использования функций расширения *Watermark to PDF* программы *DVWebTool* было разработано клиентское расширение, которое выполняет две задачи:

- Получает из текущей открытой карточки Документа идентификаторы основных файлов формата PDF.
- Отправляет идентификаторы файлов расширению *Watermark to PDF*.

Общие требования к реализации клиентских расширений приведены в пункте "Расширение возможностей клиентской части Web-клиента".

Прежде всего реализуем сервис, получающий список идентификаторов файлов, в которые нужно добавить водяной знак и передавать его в метод `AddWatermarkToFiles` расширения *Watermark to PDF*. Ниже приведён исходный код данного сервиса.

Файл `Others/Watermark/WatermarkWebExtension/src/WatermarkService.tsx`

```
export class WatermarkService { ①

    constructor(private services: $RequestManager & $WebServices & $ApplicationSettings
        & $MessageBox & $SiteUrl & $CurrentEmployeeId & $JwtTokenController) { ②
    }

    async AddWatermarkToFiles(cardID: string, fileIDs: string[]): Promise
<IWebServicesResponse<any>> { ③

        const accessToken = await this.services.jwtTokenController.getAccessToken(); ④

        const data: any = { ⑤
            data: {
                cardID: cardID,
                fileIDs: fileIDs,
                userID: this.services.currentEmployeeId,
                serverAddress: this.services.siteUrl,
                accessToken
            },
            action: 'AddWatermarkToFiles', ⑥
            locale: this.services.applicationSettings.culture.twoLetterISOLanguageName ⑦
        };

        return DVWebToolConnection.trySendData("Watermark", data, this.services); ⑧
    }
}

export type $WatermarkService = { watermarkService: WatermarkService }; ⑨
export const $WatermarkService = serviceName((s: $WatermarkService) => s.
watermarkService);
```

① Клиентский сервис, предоставляющий доступ к методу добавления водяных знаков, предоставляемому расширением DVWebTool.

② Метод принимает идентификатор карточки и идентификатор её конвертируемых файлов.

③ Токен доступа, с помощью которого DVWebTool будет выполнять запросы к серверу Web-клиента.

Обратите внимание, время жизни токена ограничено! Подробнее см. ниже.

④ В данных нужно также передать:

- Адрес сервера Web-клиента. DVWebTool должен подключиться к Web-клиенту для получения и сохранения файлов карточки.
- ID пользователя для отправки оповещения о завершении процесса

⑤ Название метода, вызываемого из расширения *DVWebTool*.

⑥ Обязательное для передачи название локали.

⑦ Вызываем метод `AddWatermarkToFiles` из контроллера `Watermark` расширения DVWebTool. Тип `DVWebToolConnection` предоставляет методы для работы с DVWebTool.

⑧ Регистрируем сервис `WatermarkService`.

Сервис `$WatermarkService` предоставляет единственный метод `AddWatermarkToFiles`, который вызывает функцию программы *DVWebTool* с помощью метода `DVWebToolConnection.trySendData`. При вызове данного метода нужно передать название контроллера и данные, в которых должно быть название вызываемого метода контроллера — в поле `action` передаваемых данных.

В качестве метода, использующего сервис `$WatermarkService` реализуем обработчик нажатия кнопки разметки карточки Документ, который будет получать идентификаторы основных PDF-файлов карточки, вызывать метод `$WatermarkService.AddWatermarkToFiles` и после завершения его работы обновлять содержимое элемента управления Список файлов, или отображать ошибку.

Файл `Others/Watermark/WatermarkWebExtension/src/EventHandlers.tsx`

```
export async function addWatermark(sender: LayoutControl) { ①
   showNotify("Запущен процесс добавления водяных знаков");

   let cardId = sender.layout.getService($CardId); ②
   let files = sender.layout.getService($FileService).getFiles();

   let fileIDs = new Array();

   files.forEach((item) => { ③
      if (item.data.isMain && item.data.name.toLowerCase().endsWith(".pdf")) { ④
         fileIDs.push(item.data.fileId);
      }
```

```
    })

    let watermarkService = sender.layout.getService($WatermarkService); ⑤

    let response = await watermarkService.AddWatermarkToFiles(cardId, fileIDs); ⑥

    if (response.success == false) {
       showError(response.errorMessage);
    } else {
       showNotify(response.data.message);

       if (layoutManager.cardLayout == null)
          return;

       let currentCardId = sender.layout.getService($CardId);
       if (currentCardId == cardId) {
          let fileList = layoutManager.cardLayout.controls.get<FileListControl>("fileList
"); ⑦
          await fileList.reloadFromServer();
       }
    }
}
```

① Обработчик для события нажатия иконки добавления водяных знаков.

② Получаем ID текущей карточки и список файлов из элемента `FileList`.

③ Получаем из модели `files` только идентификаторы файлов.

④ Обрабатываем только основные файлы с расширением PDF.

⑤ Получаем реализованный сервис для работы с водяными знаками.

⑥ Вызываем функцию добавления водяных знаков для файлов с ИД из списка `fileIDs`.

⑦ Обновляем список файлов, если есть. Возможно уже открыта другая карточка, но в данном случае это не существенно.

Настройка времени действия токена

Время жизни токена доступа зависит от значений параметров `JwtAccessTokenLifetime` и `JwtClockSkew` в конфигурационном файле модуля.

```
{
"Docsvision": {
    "WebClient": {
      "SettingGroups": {
```

```
        "System": {
          "JwtAccessTokenLifetime": 300, ①
          "JwtClockSkew": 300 ②
        }
      }
    }
  }
}
```

① Время жизни токена. По умолчанию `5` минут (`300` секунд).

② Допустимая погрешность часов при проверке времени жизни токена. Значение прибавляется к `JwtAccessTokenLifetime`. По умолчанию `5` минут (`300` секунд).

Сборка проекта

Сборка серверного расширения Web-клиента и расширения DVWebTool.

1. Откройте решение `Samples.sln`.
2. Соберите проект **Other › Watermark › WatermarkServerExtension**.
3. Соберите проект **Other › WatermarkWebToolExtension**.

Сборка клиентской части.

4. Откройте в командной строке папку **Others › Watermark › WatermarkWebExtension**.
5. Выполните команды:

```
npm install
npm update
npm run build:prod
```

Публикация компонентов на сервере Web-клиент.

6. Остановите **dvwebclient**.
7. Скопируйте папку `SamplesOutput/Content/Modules/WatermarkWebExtension/` в `../../../lib/docsvision/webclient/Content/Modules`.
8. Скопируйте папку `SamplesOutput/Extensions/WatermarkServerExtension` в `../../../lib/docsvision/webclient/Extensions`.
9. Для работы пользователей на Windows скопируйте файлы из папки `SamplesOutput/Site/Content/Tools/DVWebTool/Application Files/` в `Путь к папке с панелью управления Web-клиентом/Site/Content/Tools/DVWebTool/`, например

`C:/Program Files (x86)/Docsvision6/WebClient/Site/Content/Tools/DVWebTool`.

10. Запустите **dvwebclient**.

Регистрация расширения DVWebTool на сервере Web-клиента.

11. Запустите программу Панель управления Web-клиентом.
12. Нажмите кнопку **Собрать DVWebTool**.
13. В диалоге введите адрес публикации Web-клиента, при необходимости установите другие опции, и нажмите кнопку **OK**.
14. Установите собранную утилиту DvWebTool на сервер Web-клиента, см. подробнее "/dv6/webclient/6.1/user/prepare-add-components/#dvWebTool[Установка и запуск программы DVWebTool]".
15. Перезапустите **dvwebclient**.

Сборка и установка пакета для DvSupService

Для работы пользователей на Linux нужно обеспечить копирование файлов расширения в папку установки DvSupService — `/lib/docsvision/dvsupservice`. Рекомендуемый способ это сделать — собрать deb-пакет[15] и распространить его любым удобным способом:

1. Убедитесь, что собран проект `Other > Watermark > WatermarkWebToolExtension`. В результате в папке `Others/Watermark/DebPackage/Build/DvSupServiceExtension` должны оказаться файлы расширения в папках `Assemblies` и `Services`. Параметр `ExtensionsPath` в конфигурационном файле DVSupService позволяет изменять папку для расширений, по-умолчанию это `Assemblies`.
2. Скопировать папку `Others\Watermark\DebPackage` на Linux в каталог, где планируется выполнять сборку пакета (расположение папки значения не имеет).
3. Установите необходимые для сборки компоненты следующей командой:

```
sudo apt-get install make debhelper dpkg-dev
```

4. Откройте консоль в папке `Build` и соберите пакет с помощью следующей команды:

```
cd DebPackage/Build make
```

В результате в папке `DebPackage` появится файл `dvsupservice-extension_1.0_amd64.deb`.

5. Остановите сервис **dvsupservice** командой:

```
sudo systemctl stop dvsupservice
```

6. Распространите и установите на машины пользователей собранный пакет. Для этого можно создать репозиторий, чтобы можно было установить расширение с помощью команды `sudo apt-get install dvsupservice-extension`. См. "/dv6/install-linux/dv6/alternative/[Создание простого репозитория]". Либо просто скопируйте файл на клиентскую машину и выполните команду:

```
sudo dpkg -i dvsupservice-extension_1.0_amd64.deb
```

7. Запустите сервис **dvsupservice** командой:

```
sudo systemctl start dvsupservice
```

Проверка примера

Настройте разметку:

1. В программе Конструктор Web-разметок добавьте элемент **Кнопка** в любую разметку просмотра карточки *Документ*.
2. Укажите для события **При щелчке** обработчик `addWatermark`.
3. Сохраните разметку.
4. Установите или обновите программу *DVWebTool*. См. пункт /dv6/webclient/6.1/user/prepare-add-components/#dvWebTool[Установка и запуск программы DVWebTool] в руководстве пользователя Web-клиент.
5. Запустите программу *DVWebTool*. Убедитесь, что программа *DVWebTool* и Web-клиент будут запущены от имени одного пользователя.

Убедитесь, что все расширения установлены:

8. В Web-клиенте перейдите в раздел *О программе*.

   В разделе Подключенные расширения должны быть указаны расширения:

   - *WatermarkServerExtension* (Сборка <номер сборки>).

- *Watermark to PDF <номер сборки>* — web-расширение.

9. Откройте панель *About* из меню Docsvision *DVWebTool*.

   В списке установленных расширений должно быть расширение:

   - *Watermark to PDF <версия>*.

10. Откройте для просмотра любую карточку с разметкой, настроенной в п. 1.
11. Добавьте один или несколько основных файлов с расширением `.pdf`.
12. Нажмите на добавленную кнопку с обработчиком `addWatermark`.

Появится сообщение `Запущен процесс добавления водяных знаков` — начнётся процесс добавления водяных знаков.

После завершения процедуры появится сообщение `Водяные знаки добавлены в файлы: "список PDF-файлов, в которые добавлены водяные знаки"`.

В карточку будут добавлены дополнительные файлы с постфиксом `_marked`, являющиеся копиями оригинальных файлов с добавленным водяным знаком: слово `Секретно`.

Особенности обновления

При обновлении версии Web-клиента регистрация расширения в программе *DVWebTool* будет отменена (конфигурационные файлы программы перезаписываются).

После установки новой версии Web-клиента необходимо повторно зарегистрировать расширения *DVWebTool* на сервере Web-клиент.

Web-сервис для взаимодействия с платформой

Данный раздел содержит демонстрационный вариант Web-сервиса, содержащего пример взаимодействия с платформой Docsvision.

Ссылка на пример на GitHub: ExternalWebService.

Web-сервис позволяет выполнять следующие действия с платформой Docsvision:

1. Получите информацию о карточке документа по её идентификатору.
2. Создайте карточку документа по переданной модели.
3. Обновите данные карточки документа.
4. Удалите карточку документа по её идентификатору.

5. Измените состояние карточки документа.
6. Прикрепите файл к карточке документа.
7. Получите файл по его идентификатору.
8. Получите результат выполнения расширенного отчета.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- Microsoft Visual Studio 2022.

Сборка и установка

1. Откройте `/Samples.sln`
2. Соберите проект `Others > ExternalWebService > WebService`
3. Соберите проект `Others > ExternalWebService > WebServiceClient`
4. На сервере Docsvision установите Docsvision Resource Kit. С помощью утилиты DVCardManager загрузите в БД Docsvision библиотеку `ReportsLibrary` из каталога `ExternalWebService\CardDefs\ReportsLibrary`.

   Подробное описание процедуры загрузки пользовательской библиотеки карточек содержится в руководстве разработчика Docsvision.

5. Перезапустите **dvwebclient**.

Проверка примера

1. Отредактируйте в файле `SamplesOutput\ExternalWebService\WebService\appsettings.json` параметры `ServerAddress` и `BaseName` так, чтобы они указывали на рабочий сервер Docsvision и базу.
2. Запустите **dvwebclient** из папки `SamplesOutput\ExternalWebService\WebService`.
   a. На Windows запустите `WebService.exe` из под учетной записи Docsvision Web-клиента.
   b. На Linux запустите командой `dotnet WebService.dll`
   c. Для запуска под нужной учетной записью можно задать параметры `SystemUserAccount` и `SystemUserPassword` в `appsettings.json`.
3. Запустите приложение `SamplesOutput\ExternalWebService\WebServiceClient\WebServiceClient.exe`. В выводе

консоли отобразится результат выполнения операций.

Каталог "CardDefs"

Содержит описание библиотеки `ReportsLibrary` с одним расширенным отчетом, который возвращает описание карточки (поле `description`) по её идентификатору.

```
<?xml version="1.0" encoding="utf-8"?>
<CardLibrary xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ID="D030376C-1F99-445B-9455-
01538CB11461" Alias="ReportsLibrary" ControlInfo="" IconFile="" Version="1"
MsiProductCode="" MsiPackageName="">
  <Name>
    <LocalizedString Language="en">ReportsLibrary</LocalizedString>
    <LocalizedString Language="ru-RU">ReportsLibrary</LocalizedString>
  </Name>
  <Reports>
    <Report Alias="TestReport" ID="85701F4B-61FF-4A33-ADD1-959B471E7558" SQLFile="">
      <Name>
        <LocalizedString Language="en">TestReport</LocalizedString>
      </Name>
      <Parameter Type="uniqueidentifier" Name="CardId" />
      <SqlFiles>
        <SqlFile ServerType="mssql">Sql\TestReport.sql</SqlFile>
        <SqlFile ServerType="pgsql">PgSql\TestReport.sql</SqlFile>
      </SqlFiles>
      <Result>
        <ResultSet Name="dvtable_report_result_{85701f4b-61ff-4a33-add1-959b471e7558}">
          <Column Name="Description" DataType="String" Size="255" Nullable="true" />
        </ResultSet>
      </Result>
    </Report>
  </Reports>
  <Log>
    <Name />
  </Log>
</CardLibrary>
```

Проект "WebService.Interfaces"

Проект содержит общие модели для веб-сервиса и клиентского приложения.

Проект "WebService"

Предоставляет API для взаимодействия с сервером Docsvision. Содержит сервис

`DocumentService` для работы с карточками типа `Document`.

Также включает в себя `ReportService`, содержащий метод для выполнения расширенного отчета.

Проект "WebServiceClient"

Содержит пример консольной утилиты для взаимодействия с веб-сервисом посредством WebApi.

Интеграция с сервисом Контур.Фокус

Данный раздел содержит описание примера интеграции с сервисом Контур.Фокус.

Ссылка на пример на GitHub: KonturIntegration.

Пример включает два компонента:

- `KonturServerExtension` — папка с серверным расширением Web-клиента, в котором реализовано проксирование запросов к Контур.Фокус (для обхода ограничений CORS в браузере).
- `KonturWebExtension` — папка с клиентским расширением, в котором реализовано открытие отчета о контрагенте и загрузка информации.
- `KonturSolution.xml` — решение с разметкой для организации контрагента, содержащее разметку с необходимыми обработчиками и элементами управления.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- Microsoft Visual Studio 2022.

Сборка и установка

1. Сборка серверной части.
   a. Откройте решение `Samples.sln`.
   b. Соберите проект `Other > Kontur > Kontur`.
2. Сборка клиентской части.
   a. Откройте в командной строке папку `Others > KonturIntergration > KonturWebExtension`.

b. Выполните команды:

```
npm install
npm update
npm run build:prod
```

3. Публикация компонентов на сервере Web-клиента.
   a. Скопируйте папку `SamplesOutput\Content\Modules\KonturWebExtension\` в `Путь к сайту Web-клиента\Content\Modules`.
   b. Скопируйте папку `SamplesOutput\Extensions\KonturServerExtension` в `Путь к сайту Web-клиента\Extensions`.
   c. Перезапустите Web-сервис.

Проверка примера

При создании контрагента

1. В программе Конструктор Web-разметок импортируйте решение `KonturSolution.xml`.
2. В `appsettings.json` Web-клиента добавьте в корневой элемент настройку с ключом доступа к API Контур.Фокус:

```
{
"KonturSettings": {
  "ApiKey": "..."
}
}
```

3. Откройте разметку с ЭУ `PartnerDepartment`, например, создание исходящего документа, откройте окно справочника и нажмите кнопку добавления организации.
4. В открывшейся разметке введите ИНН организации, например, `6663003127`.
5. Нажмите кнопку **Просмотреть отчет о контрагенте** — откроется окно с информацией о контрагенте.
6. Нажмите на кнопку **Загрузить информацию** — значения элементов управления на разметке заполнятся данными из Контур.Фокус.

В карточке документа

1. Добавьте разметку карточки типа документ кнопку, назначьте ей обработчик `openBriefReportAndAttachFile`.
2. В ту же разметку добавьте ЭУ с именем `INN` типа `TextBox`. Настройте его таким образом, что бы он содержал актуальный ИНН контрагента, например, через расширенные источники данных.
3. Откройте документ и нажмите на кнопку, откроется отчет в модальном окне и к документу будет прикреплен файл отчета в качестве дополнительного файла.

Компонент "KonturServerExtension"

```
namespace Kontur
{
    public class KonturServerExtension : WebClientExtension ①
    {

        public KonturServerExtension(IServiceProvider serviceProvider) ②
            : base()
        {
        }

        public override string ExtensionName ③
        {
            get { return Assembly.GetAssembly(typeof(KonturServerExtension)).GetName
().Name; }
        }

        public override Version ExtensionVersion ④
        {
            get { return new Version(FileVersionInfo.GetVersionInfo(Assembly
.GetExecutingAssembly().Location).FileVersion); }
        }

        #region WebClientExtension Overrides

        public override void InitializeServiceCollection(IServiceCollection services)
        {
            services.AddSingleton<IKonturRequestService, KonturRequestService>();
            services.AddOptions<KonturSettings>().BindConfiguration(KonturSettings.Key);

        }

        protected override List<ResourceManager> GetLayoutExtensionResourceManagers() ⑤
```

```
        {
            return new List<ResourceManager>
            {

            };
        }

        #endregion
    }
}
```

① Задаёт описание расширения для Web-клиента, которое задано в текущей сборке.

② Создаёт новый экземпляр, см. `KonturServerExtension`.

- `serviceProvider` — сервис-провайдер.

③ Получить название расширения.

④ Получить версию расширения.

⑤ Получает менеджеры ресурсов для расширения конструктора разметок.

Отправка оповещений пользователям

Для отправки оповещений пользователям Web-клиента могут быть использованы функции SignalR, реализованные в сервисе `IRealtimeCommunicationService`. Данный сервис позволяет отправлять оповещения всем пользователям или определённому пользователю.

Ссылка на пример на GitHub: SignalForUsers.

Не следует путать оповещение с функцией отображения диалоговых окон (пункт "Отображение сообщений, запросов, предупреждений").

Текстовые оповещения отображаются в правом нижнем углу окна Web-клиента.

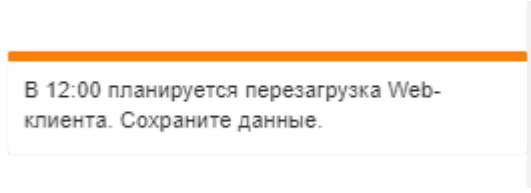


Рисунок 118. Пример оповещения

На стороне клиента также предоставляется доступ к механизму оповещения, позволяющий передавать сообщения между вкладками браузера и на сервер Web-клиента. Данная

> функциональность реализована в клиентском сервисе `$RealtimeCommunicationService` (см. описание в JsDocApi).

В качестве примера работы с сервисом IRealtimeCommunicationService реализуем функцию рассылки сообщений всем пользователям, доступную на стороне клиента.

Полный код примера доступен в репозитории на GitHub.

Реализация серверного метода для рассылки оповещений

Во-первых, нужно реализовать серверный метод, использующий функции сервиса `IRealtimeCommunicationService` для рассылки оповещений.

Для этого разработаем серверное расширение с web-функцией. Ниже приведён код такой функции, реализованной в контроллере.

Файл `Controllers\ServiceController.cs`

```
using DocsVision.Platform.WebClient.Models.RealTimeCommunication;
using DocsVision.Platform.WebClient.Models.RealTimeCommunication.NotificationMessage;
using DocsVision.Platform.WebClient.Services;
using System;
using Microsoft.AspNetCore.Mvc;

namespace SignalForUsersExtension
{
    public class ServiceController : ControllerBase
    {
        readonly IRealtimeCommunicationService communicationService;

        public ServiceController(IRealtimeCommunicationService communicationService) ①
        {
            this.communicationService = communicationService;
        }

②
        [HttpPost]
        public void SendAll([FromBody]SendMessageRequest request)
        {
            var commMessage = new RealtimeCommunicationMessage
<NotificationRealtimeMessage>();
            var messageData = new NotificationRealtimeMessage()
            {
                NotificationType = request.MessageType,
                Message = request.Message
```

```
            };
            commMessage.Initialize(NotificationRealtimeMessage.MessageType, messageData);
            communicationService.SendAll(commMessage);
        }
    }
}
```

① Сервис IRealtimeCommunicationService получаем из конструктора.

② Отправляет оповещение всем пользователям
<param name="request">Сообщение</param>

Для удобства использования методов серверного расширения разработаем клиентское расширение с сервисом, который будет вызывать web-методы.

Файл `src\MessageService.tsx`

```
import { serviceName } from "@docsvision/webclient/System/ServiceUtils";
import { $RequestManager } from "@docsvision/webclient/System/$RequestManager";
import { urlStore } from "@docsvision/webclient/System/UrlStore";

export class MessageService { ①
    constructor(private services: $RequestManager) {
    }

    sendAll(message: string, messageType: MessageType) { ②
        let url = urlStore.urlResolver.resolveApiUrl("SendAll", "Service");

        this.services.requestManager.post(url, JSON.stringify({ message, messageType }));
    }
}

export type $MessageService = { messageService: MessageService };
export const $MessageService = serviceName((s: $MessageService) => s.messageService);
```

① Клиентский сервис, предоставляющий доступ к серверным методам работы с подсистемой оповещений.

② Отправляет сообщение message всем пользователям. Тип сообщения `messageType` влияет на оформление сообщения.

Здесь `messageType` — ограниченный список значений, которые может принимать тип сообщения:

```
export enum MessageType {
    Alert = 0,
```

```
    Success = 1,
    Warning = 2,
    Error = 3,
    Info = 4
}
```

Осталось добавить функцию, которая будет обрабатывать, к примеру, нажатие кнопки разметки и вызывать метод `MessageService.sendAll`:

Файл `src\EventHandlers.tsx`

```
import { IEventArgs } from "@docsvision/webclient/System/IEventArgs";
import { Layout } from "@docsvision/webclient/System/Layout";
import { $MessageService, MessageType, MessageService } from "./MessageService";
import { layoutManager } from "@docsvision/webclient/System/LayoutManager";

export function sendMessage(sender: Layout, e: IEventArgs) { ①
   let messageService = layoutManager.cardLayout.getService<MessageService>
($MessageService);
   return messageService.sendAll("В 12:00 планируется перезагрузка {wc}а. Сохраните
данные.", MessageType.Warning);
}
```

① Функция использует сервис `$MessageService` (проект) для отправки оповещения всем пользователям.

Функция `sendMessage` может быть указана в качестве обработчика нажатия кнопки или другого события.

Реализация собственного штампа электронной подписи

Реализация не поддерживается в текущей версии системы.

Устаревшие примеры

Руководство по разработке решений Web-клиента

Устаревшие примеры

Данные примеры помечены как устаревшие и неактуальны.

- Пример текста
- Пример. Элемент управления для выбора из Справочника номенклатуры дел

- Больше примеров категории в навигационном меню слева...

Пример текста

Данный раздел содержит описание примера реализации элемента управления `HyperComments`.

Платформа `HyperComments` перестала поддерживать бесплатный тариф. Необходимо приобретение лицензии.

Пример требует подключенного расширения программы Конструктор Web-разметок `UrlProprtyDesignerExtension`.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022.

Проект "HyperCommentsWebExtension"

Проект-расширение клиентской части Web-клиента. Содержит пример одного из способов интеграции со сторонними веб-приложениями.

Проект "HyperCommentsDesignerExtension"

Проект-расширение для конструктора разметок. Содержит описание элемента управления `HyperComments` для программы Конструктор Web-разметок. Демонстрирует описание и подключение нового ЭУ, используя класс `ControlTypeDescription`, задание ему стандартных свойств, реализованных в программе Конструктор Web-разметок[16] и добавление собственного свойства, используя `PropretyDescription`, класс[17].

Сборка и установка

1. Откройте `/Samples.sln`.
2. Соберите проект `Controls > HyperComments > HyperCommentsDesignerExtension`.
3. Откройте консоль в папке `Controls > HyperComments > ExchangeRatesWebExtension` и выполните команду `npm install` и `npm run build:prod`.
4. Скопируйте каталог `SamplesOutput\Plugins\HyperCommentsDesignerExtension` в

каталог `Путь к директории с исполняемым файлом конструктора Web-разметок\Plugins`.

5. Скопируйте каталог `SamplesOutput\Content\Modules\HyperCommentsWebExtension` в каталог `Путь к сайту Web-клиента\Content\Modules`.
6. Перезапустите **dvwebclient**.

Регистрация виджета "HyperComments"

1. Перейдите на сайт https://www.hypercomments.com.
2. Нажмите на кнопку **Войти**.
3. Авторизоваться под аккаунтом Google.
4. Выбрать лицензию и следовать инструкциям.
5. Нажмите на кнопку **Установить**.
6. Ввести адрес сайта (Допустимо имя сайта в локальной сети).
7. Выбрать тип виджета - HTML Custom code.
8. Нажмите **Далее**.
9. Нажмите **Готово**.
10. Перейдите на страницу *Виджет — Общее*.
11. Под заголовком указан идентификатор виджета ID: `######`, его следует запомнить, он потребуется далее.
12. Установите флаг напротив настройки `Get-параметры`. Учитывайте GET параметры и сгенерируйте новый виджет комментирования для отдельной страницы.
13. Нажмите на кнопку **Сохранить**.

Проверка примера

1. Запустите конструктор разметок.
2. Скопируйте любую разметку для карточки вида *Служебная записка*.
3. Выберите условия использования этой разметки.
4. Откройте разметку и добавить в нее новый элемент управления `Hyper comments`.
5. Укажите в свойстве `Widget id` идентификатор из пункта выше.
6. Сохраните разметку.

7. Перезапустите **dvwebclient**.
8. Откройте карточку с этой разметкой.
9. Убедитесь, что появился новый элемент управления.

Пример. Элемент управления для выбора из Справочника номенклатуры дел

В данном пункте рассмотрено создание элемента управления для выбора дела из Справочника номенклатуры дел.

В примере рассмотрены следующие вопросы разработки элементов управления:

- Создание расширения программы Конструктор Web-разметок, решающего следующие задачи:
  - Ограничение выбора поля карточки со значением элемента управления — поле должно содержать ссылку на строку *Справочника номенклатуры дел*.
  - Возможность ограничить для пользователя область выбора дел — настройки элемента управления должны содержать параметр для выбора *Раздела*, из которого можно выбирать дела.
- Создание серверного расширения Web-клиент, который должен предоставлять:
  - Сервис для работы с данными Справочника номенклатуры дел (с использованием базового API Docsvision).
  - Конвертер значений, возвращающий отображаемое значение элемента управления.
- Создание клиентского расширения с несколькими функциями:
  - Реализация пользовательского интерфейса, одним из элементов которого должно быть дерево разделов Справочника номенклатуры дел.
  - Реализация функции ограничения выбора дел из раздела Справочника номенклатуры дел 5, указанного в программе Конструктор Web-разметок.
  - Предоставление возможности быстрого поиска дел.

Полный исходный код примера приведён в репозитории Docsvision Samples на GitHub, проект `Controls/RefCases`.

Создание описателя элемента управления и расширения программы Конструктор Web-разметок

Прежде всего определим перечень настроек разрабатываемого элемента управления, которые должны быть доступны пользователю программы

Конструктор Web-разметок:

- Стандартный набор свойств: текст метки, подсказка, обязательные и пр..
- Свойства для выбора источника данных.
- Специальное свойство для ограничения области выбора дел.
- Стандартный набор событий (при наведении/отведении курсора и при смене данных).

Описатель элемента управления

Для описания данного элемента управления сформируем текстовый описатель со следующим содержимым:

Файл `RefCategoriesDesignerExtension\xml\RefCasesControlDescription.xml`

```
<?xml version="1.0" encoding="utf-8" ?>
<Controls>
  <Control Name="RefCases" ControlGroupResourceKey="ControlGroup_Directories"
ResourceKey="Control_RefCases">
    <Properties>
①
      <Property Type="Name" /> ②
      <Property Type="PlaceHolder" /> ③
      <Property Type="Tip" /> ④
      <Property Type="LabelText" />
      <Property Type="ShowEmptyLabel" />
      <Property Type="Visibility" />
      <Property Type="StandardCssClass" DefaultValue="ref-cases" />
      <Property Type="CustomCssClasses" />
      <Property Type="TabStop" DefaultValue="True" />
      <Property Type="Required" />

⑤
      <Property Type="ExtendedDataSource" /> ⑥
      <Property Type="DataSource" /> ⑦
      <Property Type="DataField" Editor="RefCasesFieldMetadataEditor" />
      <Property Type="Binding" BindingConverter="RefCasesConverter" /> ⑧
      <Property Type="EditOperation" /> ⑨

      <Property Name="RootSection" ResourceKey="Control_RefCases_Section"
                Category="BehaviorCategory" DataType="System.Guid"
                Editor="RefCasesSectionEditor" /> ⑩
    </Properties>
    <Events> ⑪
      <Event Name="Click" ResourceKey="ControlTypes_ClickEventProperty" />
```

```
      <Event Name="MouseOver" ResourceKey="ControlTypes_MouseOverEventProperty" /> ⑫
      <Event Name="MouseOut" ResourceKey="ControlTypes_MouseOutEventProperty" /> ⑬
      <Event Name="DataChanged" ResourceKey="ControlTypes_DataChangedEventProperty" /> ⑭
    </Events>
  </Control>
</Controls>
```

① Стандартный набор свойств:

② **Название**;

③ **Заполнитель**;

④ **Подсказка**.

⑤ Свойства для настройки источника данных и операции редактирования:

⑥ `ExtendedDataSource` — выбор расширенного источника данных;

⑦ `DataSource`, `DataField` — выбор секции и поля с данными элемента управления. Редактор `RefCasesFieldMetadataEditor` (будет разработан далее) реализует заявленную функциональность ограничения полей карточки, доступных для выбора;

⑧ Binding — системное свойство, с которым на клиент передаётся связь с источником данных.

Значение элемента управления — идентификатор строки дела, выбранного в *Справочнике номенклатуры дел*. Пользователю предпочтительно показывать название дела из справочника, а не идентификатор.

Такое преобразование данных осуществляется с помощью конвертера, название которого указывается в атрибуте `BindingConverter`. В данном примере название конвертера `RefCasesConverter` (будет разработан далее);

⑨ `EditOperation` — выбор операции редактирования.

⑩ Новое свойство `RootSection` обеспечивает заявленную возможность выбора разделе *Справочника номенклатуры дел*, из которого пользователю будет разрешено выбирать дела.

Значение свойства будет устанавливаться с помощью специального редактора `RefCasesSectionEditor` (будет разработан далее в серверном расширении Web-клиента).

⑪ Также для элемента управления определены события:

⑫ При наведении курсора;

⑬ При отведении курсора;

⑭ При смене данных.

Расширение программы Конструктор Web-разметок

В описателе элемента управления определено несколько нестандартных редакторов, а также ресурсы с текстовыми значениями. Данные сущности нужно добавить в Конструктор Web-разметок с помощью расширения, код которого приведён ниже.

Файл RefCategoriesDesignerExtension\Extension\RefCasesDesignerExtension.cs

```
using DocsVision.Platform.Tools.LayoutEditor.Extensibility;
using RefCasesDesignerExtension.Editors;
using System;
using System.Collections.Generic;
using System.Resources;

①
namespace RefCasesDesignerExtension.Extension
{
    class RefCasesDesignerExtension : WebLayoutsDesignerExtension
    {
        public RefCasesDesignerExtension(IServiceProvider provider)
            : base(provider)
        {
        }

        protected override Dictionary<string, Type> GetEditors()
        {
            return new Dictionary<string, Type>
            {
                { "RefCasesFieldMetadataEditor", typeof(RefCasesFieldMetadataEditor)}, ②
                { "RefCasesSectionEditor", typeof(RefCasesSectionEditor)} ③
            };
        }

        protected override List<ResourceManager> GetResourceManagers()
        {
            return new List<ResourceManager>
            {
                Resources.ResourceManager
            };
        }
    }
}
```

① В данном расширении регистрируется два редактора, которые указаны в текстовом описателе:

② `RefCasesFieldMetadataEditor` — редактор для выбора поля карточки, содержащего значение элемента управления;

③ `RefCasesSectionEditor` — редактор для выбора раздела *Справочника номенклатуры дел*, из которого пользователям будет разрешено выбирать дела.

Редактор "RefCasesFieldMetadataEditor"

Редактор `RefCasesFieldMetadataEditor` является стандартным редактором редактором для выбора поля карточки, реализованном в классе `Docsvision.BackOffice.WebLayoutsDesigner.Editors.FieldMetadataEditor`.

Файл `RefCategoriesDesignerExtension\Editors\RefCasesFieldMetadataEditor.cs`

```
using System.Windows;
using DocsVision.Platform.Data.Metadata.CardModel;
using System;
using Xceed.Wpf.Toolkit.PropertyGrid;
using Xceed.Wpf.Toolkit.PropertyGrid.Editors;
using DocsVision.BackOffice.WebLayoutsDesigner.Editors;
using DocsVision.Platform.WebLayoutsDesigner.NewEditors;

namespace RefCasesDesignerExtension.Editors
{
    public class RefCasesFieldMetadataEditor : ITypeEditor ①
    {
        public FrameworkElement ResolveEditor(PropertyItem propertyItem)
        {
            var refCasesID = new Guid("246197EA-846A-44DA-9EA3-0BCAE5500388");
            var sectionCasesID = new Guid("56AF8231-B918-42D4-AC15-90EC2E9A0725");

            var editor = new FieldMetadataEditor();

            editor.FieldFilter = (field) => { ②
                return (field.FieldType == FieldType.RefId
                        && field.LinkedCardTypeId == refCasesID
                        && field.LinkedSectionId == sectionCasesID);
            };

            return editor.ResolveEditor(propertyItem);
        }

    }
```

```
}
```

① Редактор для выбора поля карточки, ссылающегося на Дело в Справочнике номенклатуры дел.

② Устанавливаем фильтр для выбора полей только из справочника.

Ограничение возможности выбора полей карточки включено с помощью фильтра `FieldFilter`, в котором проверяется тип поля (`field.FieldType`), которое должно быть ссылочным полем (`FieldType.RefId`), ссылающимся на секцию *Дела* (`field.LinkedSectionId == sectionCasesID`) *Справочника номенклатуры дел* (`field.LinkedCardTypeId == refCasesID`).

В стандартной реализации приложения *Делопроизводство 5* поле карточки, используемое для хранения ссылки на *Дело*, не является ссылочным, поэтому для него не подходит фильтр, использованный в данном примере, при настройке разметки данное поле будет недоступно для выбора. Если в Web-клиенте нужно повторить настройки разметки Windows-клиента, фильтр нужно изменить следующим образом:

```
var editor = new FieldMetadataEditor();

editor.FieldFilter = (field) => {
    return (field.FieldType == FieldType.RefId
        && field.LinkedCardTypeId == refCasesID
        && field.LinkedSectionId == sectionCasesID);
};
```

Редактор "RefCasesSectionEditor"

Редактор `RefCasesSectionEditor` имеет более сложную реализацию (по сравнению с `RefCasesFieldMetadataEditor`), из-за необходимости отображения дерева *Разделов Справочника номенклатуры дел 5*.

У данного редактора есть две составляющие:

- Графическая — предоставляет форму для выбора Разделов.
- Функциональная — предоставляет функции, загружающие дерево Разделов и выполняющие сопутствующие операции.

В простейшем случае можно обойтись без сложного редактора, и предоставить возможность непосредственно вводить идентификатор требуемого *Раздела* справочника в значение

> настройки элемента управления, или отказаться от данной настройки.

Графическая и функциональные составляющие также распределяются между двумя компонентами:

- Основной компонент редактора с реализацией интерфейса `ITypeEditor`.
- Форма с деревом Разделов.

Далее приведён код основного компонента редактора без графической составляющей (см. полный исходный код примера на GitHub).

```
using DocsVision.Platform.Tools.LayoutEditor;
using DocsVision.Platform.Tools.LayoutEditor.PropertiesEditor;
using DocsVision.Platform.WebClient;
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using Xceed.Wpf.Toolkit.PropertyGrid;
using Xceed.Wpf.Toolkit.PropertyGrid.Editors;

namespace RefCasesDesignerExtension.Editors
{
    public partial class RefCasesSectionEditor : UserControl, ITypeEditor
    {
        public static readonly DependencyProperty ValueProperty = DependencyProperty
.Register("Value", typeof(Guid), typeof(RefCasesSectionEditor),①
        new FrameworkPropertyMetadata(Guid.Empty, FrameworkPropertyMetadataOptions
.BindsTwoWayByDefault));
        public static readonly DependencyProperty TextProperty = DependencyProperty
.Register("Text", typeof(string), typeof(RefCasesSectionEditor), new
FrameworkPropertyMetadata(string.Empty));

        public Guid Value ②
        {
            get { return (Guid)GetValue(ValueProperty); }
            set { SetValue(ValueProperty, value); }
        }

        public string Text ③
        {
            get { return (string)GetValue(TextProperty); }
            set {
                SetValue(TextProperty, value);
```

```
                Clear.Visibility = value != ""? Visibility.Visible: Visibility.Collapsed;
// Кнопка очистки значения ④
            }
        }

        private IServiceProvider serviceProvider;
        private SessionContext sessionContext;

        public RefCasesSectionEditor()
        {
            InitializeComponent();
        }

        public FrameworkElement ResolveEditor(PropertyItem propertyItem) ⑤
        {
            var bindingObject = (IControlPropertiesObject)propertyItem.Instance;

            this.serviceProvider = bindingObject.ServiceProvider; ⑥
            var currentObjectContextProvider = this.serviceProvider.GetService(typeof
(ICurrentObjectContextProvider)) as ICurrentObjectContextProvider;
            this.sessionContext = currentObjectContextProvider
.GetOrCreateCurrentSessionContext();

            Binding binding = new Binding("Value"); ⑦
            binding.Source = propertyItem;
            binding.Mode = propertyItem.IsReadOnly ? BindingMode.OneWay : BindingMode
.TwoWay;
            BindingOperations.SetBinding(this, RefCasesSectionEditor.ValueProperty,
binding);

            if (this.Value != Guid.Empty) ⑧
                this.Text = new RefCasesUtils(sessionContext).GetSectionTitle(this.
Value);

            return this;
        }

        private void ShowSections_Click(object sender, RoutedEventArgs e) ⑨
        {
            var sectionTree = new SectionsTree(sessionContext);
            if (sectionTree.ShowDialog() == true)
            {
                this.Value = sectionTree.SelectedNodeID;
                this.Text = sectionTree.SelectedNodeText;
            }
        }

        private void Clear_Click(object sender, RoutedEventArgs e) ⑩
```

```
        {
            this.Value = Guid.Empty;
            this.Text = "";
        }
    }
}
```

① Объявляем свойства зависимости для связывания со значением настройки (идентификатор *Раздела*) и отображаемым значением.

② Идентификатор выбранного *Раздела* справочника — является значение настройки.

③ Название выбранного *Раздела* справочника, отображаемое в строке настройки.

④ Кнопка очистки значения.

⑤ Реализация метода `ITypeEditor.ResolveEditor`.

⑥ Получаем поставщика сервисов из элемента управления.

⑦ Связываем значение компонента с `ValueProperty`.

⑧ Получаем отображаемое значение выбранного *Раздела* при загрузке элемента.

⑨ Открываем форму для выбора *Раздела*.

⑩ Очищаем значение настройки.

Основные функции прокомментированы в коде. При реализации нового редактора (в данном случае не используются готовые реализации, как в редакторе `RefCasesFieldMetadataEditor`) особое внимание следует обратить на необходимость связывания значение настройки — `Value` — со свойством зависимости (в данном примере — `ValueProperty`).

Метод `ShowSections_Click` вызывается при нажатии кнопки выбора значения свойства. Данный метод открывает форму `SectionsTree` с деревом Разделов *Справочника номенклатуры дел 5* (будет разработан далее).

Метод `Clear_Click` вызывается при нажатии кнопки очистки значения настройки.

Код вспомогательного метода `RefCasesUtils.GetSectionTitle`, возвращающего текстовое описание для Раздела справочника, идентификатор которого передан в метод:

Файл RefCategoriesDesignerExtension\Editors\RefCasesUtils.cs

```
public string GetSectionTitle(Guid sectionId)
{
    SectionData sectionSesction = cardManager.GetDictionaryData(refCasesId).Sections
[sectionsSectionID]; ① ② ③

    if (sectionSesction.RowExists(sectionId) == false) ④
        return "Ошибка!";

    RowData sectionRow = sectionSesction.GetRow(sectionId);
    RowData yearRow = sectionRow.SubSection.ParentRow;

    if (yearRow != null)
        return string.Format("{0}. {1}", yearRow["Year"], sectionRow["Name"].ToString());

    return "Ошибка!";
}
```

① `cardManager` — Менеджер карточек (базовое API Docsvision).

② `refCasesId` — идентификатор *Справочника номенклатуры дел 5*.

③ Возможно раздел был удалён.

④ `sectionsSectionID` — идентификатор секции "Разделы" справочника.

Далее приведён код компонента дерева *Разделов* без графической составляющей (см. полный исходный код примера на GitHub).

Файл RefCategoriesDesignerExtension\Editors\SectionsTree.xaml.cs

```
using DocsVision.Platform.WebClient;
using System;
using System.Windows;

namespace RefCasesDesignerExtension.Editors
{

    public partial class SectionsTree : Window
    {
        public string SelectedNodeText = "";
        public Guid SelectedNodeID = Guid.Empty;
        private RefCasesUtils refCasesUtils;

        public SectionsTree(SessionContext sessionContext)
        { ①
            InitializeComponent();
```

```
            refCasesUtils = new RefCasesUtils(sessionContext);

            Years.ItemsSource = refCasesUtils.GetYears(); ②
        }

        private void Years_SelectionChanged(object sender, System.Windows.Controls
.SelectionChangedEventArgs e) ③
        {
            if (Years.SelectedIndex == -1 || !(Years.SelectedItem is Year))
                return;

            Year selectedYear = Years.SelectedItem as Year;

            Sections.ItemsSource = refCasesUtils.GetSections(selectedYear.ID); ④
        }

        private void Accept_Click(object sender, RoutedEventArgs e) ⑤
        {
            if (Sections.SelectedItem == null)
                return;
            var selectedNode = Sections.SelectedItem as Node;
            var selectedYear = Years.SelectedItem as Year;

            this.SelectedNodeText = string.Format("{0}. {1}", selectedYear.Value,
selectedNode.Name);
            this.SelectedNodeID = selectedNode.ID;

            this.DialogResult = true;
            this.Close();
        }

        private void Cancel_Click(object sender, RoutedEventArgs e) ⑥
        {
            this.Close();
        }
    }
}
```

① Для формирования дерева *Разделов* нужно получить данные из *Справочника номенклатуры дел 5*. Для этого объявляем необходимость передачи контекста сессии в конструкторе класса.

② Получаем список лет из Справочника номенклатуры дел 5.

③ При выборе Года формируем дерево Разделов для данного года.

④ Получаем список Разделов из Справочника номенклатуры дел 5.

⑤ Обработка нажатия кнопки сохранения выбора.

⑥ Обработка нажатия отмены.

Основная "работа" здесь выполняется методами `RefCasesUtils.GetYears`, `RefCasesUtils.GetSections`.

Метод `RefCasesUtils.GetYears` получает все строки из секции Года *Справочника номенклатуры дел 5*:

Файл `RefCategoriesDesignerExtension\Editors\RefCasesUtils.cs`

```
public IEnumerable<Year> GetYears()
{
    refCasesData = cardManager.GetDictionaryData(refCasesId);
    SectionData yearSection = refCasesData.Sections[yearsSectionId];

    return yearSection.Rows.Select<RowData, Year>(row => new Year() { ID = row.Id, Value
= row["Year"].ToString() });
}
```

Метод `RefCasesUtils.GetSections` получает дерево строк для секции Разделы *Справочника номенклатуры дел 5*.

Файл `RefCategoriesDesignerExtension\Editors\RefCasesUtils.cs`

```
public List<Node> GetSections(Guid yearID)
{
    var yearSection = refCasesData.Sections[yearsSectionId];

    if (yearSection.RowExists(yearID)) {
        RowDataCollection sectionRows = yearSection.GetRow(yearID).ChildSections
[sectionsSectionID].Rows;

        return GetNodesFromRows(sectionRows);
    }

    return new List<Node>();
}

List<Node> GetNodesFromRows(RowDataCollection rows) ①
{
    var nodes = new List<Node>();

    foreach (var row in rows)
    {
```

```
        var node = new Node() { ID = row.Id, Name = row["Name"].ToString() };
        if (row.HasChildRows)
            node.Nodes = GetNodesFromRows(row.ChildRows);

        nodes.Add(node);
    }
    return nodes;
}
```

1. Возвращает список Разделов для строк секции справочника

Создание серверного расширения Web-клиента

В описателе элемента управления был объявлен конвертер для значения элемента — `RefCasesConverter` (свойство `Binding`), который должен приводить настоящее значение (идентификатор строки *Дела* в *Справочнике номенклатуры дел 5*) к виду, удобному для пользователя — в данном примере к строке вида: `Год. Название разделов. Название дела`.

Также для работы клиентской части понадобятся методы, предоставляющие список лет, дерево разделов и список дел из *Справочника номенклатуры дел 5*, а также функцию для поиска дела по названию (если нужно реализовать функцию поиска на клиенте).

Сервис для работы со Справочником номенклатуры дел 5

Прежде всего необходимо реализовать сервис для работы со *Справочником номенклатуры дел 5*, методы которого должны:

- Возвращать список лет.
- Возвращать год для определённой секции — нужно, если в программе Конструктор Web-разметок был выбран раздел, из которого выбираются дела.
- Возвращать все разделы для определённого года, а если требуется, только разделы из корневого раздела, выбранного в программе Конструктор Web-разметок.
- Возвращать все дела из определённого раздела.
- Возвращать отображаемое название дела — для конвертера значений `RefCasesConverter`.
- Возвращать список дел с искомым названием, а если требуется, только дела из корневого раздела, выбранного в программе Конструктор Web-разметок.

Ниже представлен интерфейс сервиса, описывающего установленные требования. Первые два условия для сервиса объединены в один метод `GetYears`.

Файл `RefCasesServerExtension\Services\IRefCasesService.cs`

```
using RefCasesServerExtension.Models;
using System;
using System.Collections.Generic;

namespace RefCasesServerExtension.Services
{
    public interface IRefCasesService
    {
        List<Year> GetYears(Guid? rootSection = null);
        List<Section> GetSections(Guid yearID, Guid? rootSection = null);
        List<Case> GetCases(Guid sectionID);
        string GetCaseTitle(Guid caseID);
        List<CaseClientModel> SearchCases(string caseName, Guid? rootSection = null);
    }
}
```

И реализация сервиса `IRefCasesService`:

Файл `RefCasesServerExtension\Services\RefCasesService.cs`

```
using DocsVision.Platform.ObjectManager;
using DocsVision.Platform.WebClient;
using DocsVision.Platform.WebClient.Managers;
using RefCasesServerExtension.Models;
using System;
using System.Collections.Generic;
using System.Linq;

namespace RefCasesServerExtension.Services
{
    public class RefCasesService : IRefCasesService
    {

        private Guid refCasesId = new Guid("246197EA-846A-44DA-9EA3-0BCAE5500388"); ①
        private Guid yearsSectionId = new Guid("BD44C786-6E41-450E-BD60-66919657E51B");
②
        private Guid sectionsSectionID = new Guid("319E425D-543C-45DB-BD51-
955B58476EDB"); ③
        private Guid caseSectionID = new Guid("56af8231-b918-42d4-ac15-90ec2e9a0725"); ④

⑤
        private const string Year = "Year";
```

```
        private const string DisplayIndex = "DisplayIndex";
        private const string Case_Name = "Case_Name";
        private const string Case_Index = "Case_Index";
        private const string Case_SectionDisplayIndex = "Case_SectionDisplayIndex";
        private const string Name = "Name";

        ICurrentObjectContextProvider objectContextProvider;
        private AdvancedCardManager cardManager => objectContextProvider
.GetOrCreateCurrentSessionContext().AdvancedCardManager;

        public RefCasesService(ICurrentObjectContextProvider objectContextProvider) =>
this.objectContextProvider = objectContextProvider;

        public List<Year> GetYears(Guid? rootSection) ⑥
        {
            CardData refCasesData = cardManager.GetDictionaryData(refCasesId);
            SectionData yearSection = refCasesData.Sections[yearsSectionId];

            if (rootSection.HasValue && refCasesData.Sections[sectionsSectionID
].RowExists(rootSection.Value)) ⑦
            {
                var sectionRow = refCasesData.Sections[sectionsSectionID].GetRow
(rootSection.Value);
                var yearRow = sectionRow.SubSection.ParentRow;

                return new List<Year>() {
                    new Year() { ID = yearRow.Id, DisplayValue = yearRow[Year].ToString()
}
                };
            }

            return yearSection.Rows.Select<RowData, Year>(row => new Year() ⑧
            {
                ID = row.Id,
                DisplayValue = row[Year].ToString()
            }).ToList();
        }

        public List<Section> GetSections(Guid yearID, Guid? rootSection) ⑨
        {
            CardData refCasesData = cardManager.GetDictionaryData(refCasesId);
            SectionData sectionsSection = refCasesData.Sections[sectionsSectionID];

            if (rootSection.HasValue && sectionsSection.RowExists(rootSection.Value)) ⑩
            {
                RowData row = sectionsSection.GetRow(rootSection.Value);
                return GetSectionsFromRows(new List<RowData>() { row });
            }
```

```
            SectionData yearSection = refCasesData.Sections[yearsSectionId];

            if (yearSection.RowExists(yearID)) ⑪
            {
                RowDataCollection sectionRows = yearSection.GetRow(yearID).ChildSections
[sectionsSectionID].Rows;
                return GetSectionsFromRows(sectionRows);
            }

            return new List<Section>();
        }

        public List<Case> GetCases(Guid sectionID) ⑫
        {
            CardData refCasesData = cardManager.GetDictionaryData(refCasesId);
            SectionData sectionsSection = refCasesData.Sections[sectionsSectionID];

            if (sectionsSection.RowExists(sectionID))
            {
                RowDataCollection sectionRows = sectionsSection.GetRow(sectionID
).ChildSections[caseSectionID].Rows;
                return GetCasesFromRows(sectionRows);
            }

            return new List<Case>();
        }

        public string GetCaseTitle(Guid caseID) ⑬
        {
            CardData refCasesData = cardManager.GetDictionaryData(refCasesId);
            SectionData section = refCasesData.Sections[caseSectionID];

            if (section.RowExists(caseID))
            {
                RowData caseRow = section.GetRow(caseID); ⑭
                var sectionRow = caseRow.SubSection.ParentRow;
                var yearRow = sectionRow.SubSection.ParentRow;

                return string.Format("{0}, {1}, {2}", yearRow[Year], sectionRow
[DisplayIndex], caseRow[Case_Name]);
            }

            return "Ошибка!";
        }

        public List<CaseClientModel> SearchCases(string caseName, Guid? rootSection =
null) ⑮
```

```
        {
            CardData refCasesData = cardManager.GetDictionaryData(refCasesId);

            RowDataCollection allRows;

            if (rootSection.HasValue && refCasesData.Sections[sectionsSectionID
].RowExists(rootSection.Value)) ⑯
                allRows = refCasesData.Sections[caseSectionID].GetAllRows(rootSection
.Value, true);
            else
                allRows = refCasesData.Sections[caseSectionID].GetAllRows();

            var results = new List<CaseClientModel>();

            foreach (var caseRow in allRows)
            {
                if (caseRow[Case_Name].ToString().IndexOf(caseName, StringComparison
.InvariantCultureIgnoreCase) > -1 ||
                    (caseRow[Case_SectionDisplayIndex].ToString() + "-" + caseRow
[Case_Index].ToString()).IndexOf(caseName, StringComparison.InvariantCultureIgnoreCase) >
-1) ⑰
                {
                    var sectionRow = caseRow.SubSection.ParentRow;
                    var yearRow = sectionRow.SubSection.ParentRow;

                    results.Add(new CaseClientModel()
                    {
                        Id = caseRow.Id,

                        Name = string.Format("{0}, {1}, {2}", yearRow[Year], sectionRow
[DisplayIndex], caseRow[Case_Name]) ⑱
                    });
                }
            }

            return results;
        }


        List<Section> GetSectionsFromRows(IEnumerable<RowData> rows) ⑲
        {
            var nodes = new List<Section>();

            foreach (var row in rows)
            {
                var node = new Section() { ID = row.Id, DisplayValue = row[Name].
ToString() };
                if (row.HasChildRows)
```

```
                    node.Sections = GetSectionsFromRows(row.ChildRows);

                nodes.Add(node);
            }
            return nodes;
        }

        List<Case> GetCasesFromRows(RowDataCollection rows) ⑳
        {
            var nodes = new List<Case>();

            foreach (var row in rows)
            {
                var node = new Case() { ID = row.Id, DisplayValue = row[Case_Name
].ToString() };
                if (row.HasChildRows)
                    node.Cases = GetCasesFromRows(row.ChildRows);

                nodes.Add(node);
            }
            return nodes;
        }
    }
}
```

① Справочник номенклатуры дел 5.

② Секция *Года* справочника.

③ Секция *Разделы* справочника.

④ Секция *Дела* справочника.

⑤ Названия полей.

⑥ Возвращает список лет из *Справочника номенклатуры дел 5*.

⑦ Если выбран корневой раздел, нужно вернуть только его *Год*.

⑧ Иначе возвращаем все *Года*.

⑨ Возвращает дерево разделов указанного *Года* из *Справочника номенклатуры дел 5*.

⑩ Если выбран корневой раздел, вернуть только его подразделы.

⑪ Иначе вернуть все разделы указанного *Года*.

⑫ Вернуть все дела указанного *Раздела*.

⑬ Вернуть отображаемое название *Дела*.

⑭ Получаем для дела родительские строки раздела и *Года*.

⑮ Поиск дела по названию и по идентификатору *Дела*.

⑯ Если указан коневой *Раздел*, поиск только в нём, иначе — во всех *Разделах*.

⑰ Проверяем название *Раздела* — поле `Case_Name`.

⑱ Возвращаем сразу отображаемое название.

⑲ Возвращаем список *Разделов* для строк секции справочника.

⑳ Возвращаем список *Дел* для строк справочника.

Конвертер "RefCasesConverter"

Как уже было сказано ранее, конвертер нужен для формирования отображаемого значения элемента управления, показываемого при инициализации элемента. Иначе в элементе будет показан идентификатор выбранного дела.

Для разрабатываемого элемента управления название конвертера (`RefCasesConverter`) было объявлено в описателе элемента управления. Теперь его нужно реализовать и зарегистрировать в серверном расширении Web-клиента.

В данном пример конвертер использует метод `IRefCasesService.GetCaseTitle` для получения отображаемого названия Дела.

Файл `RefCasesServerExtension\BindingConverters\RefCasesConverter.cs`

```
using DocsVision.WebClientLibrary.Layout.IL;
using DocsVision.WebClientLibrary.ObjectModel;
using DocsVision.WebClientLibrary.ObjectModel.Services.BindingConverters;
using DocsVision.WebClientLibrary.ObjectModel.Services.LayoutModel;
using RefCasesServerExtension.Models;
using RefCasesServerExtension.Services;
using System;

namespace RefCasesServerExtension.BindingConverters
{
    public class RefCasesConverter : BaseBindingConverter ①
    {
        private IRefCasesService refCasesService;

        public RefCasesConverter(IServiceProvider serviceProvider, IRefCasesService
refCasesService) : base(serviceProvider, "RefCasesConverter") ②
        {
            this.refCasesService = refCasesService; ③
        }
```

```
        public override BindingResult ConvertForDisplay(ControlContext controlContext,
LayoutBinding binding, BindingResult bindingResult) ④
        {
            var itemId = bindingResult.Value != null ? (Guid)bindingResult.Value : Guid
.Empty;
            var name = itemId == Guid.Empty ? "" : refCasesService.GetCaseTitle(itemId);

            var model = new CaseClientModel() { Id = itemId, Name = name }; ⑤

            return bindingResult.Clone(model);
        }
    }
}
```

① Класс конвертера может быть производным от `BaseBindingConverter` или полностью реализовывать интерфейс `IBindingConverter`.

② В базовый класс нужно передать название конвертера, указанного в описателе `RefCasesConverter`.

③ Получаем реализованный ранее сервис для работы со *Справочником номенклатуры дел 5*.

④ Основной метод, возвращающий отображаемое значение (а точнее модель) для значения элемента управления — `bindingResult.Value`.

⑤ Клиент ожидает модель, включающую идентификатор и название *Дела*.

Веб-методы серверного расширения

Клиентской части элемента управления нужны данные из *Справочника номенклатуры дел 5*: для показа списка лет, разделов, а также дел. Предоставить доступ к данным можно с помощью веб-методов серверного расширения.

Далее приведён код контроллера, предоставляющего такие веб-методы.

Файл `RefCasesServerExtension\Controllers\RefCasesOperationController.cs`

```
using RefCasesServerExtension.Models;
using RefCasesServerExtension.Services;
using System;
using System.Linq;
using DocsVision.Platform.WebClient.Models;
using DocsVision.Platform.WebClient.Models.Generic;
using Microsoft.AspNetCore.Mvc;
```

```
namespace RefCasesServerExtension.Controllers
{
    public class RefCasesOperationController : ControllerBase
    {
        private readonly IRefCasesService refCasesService;

        public RefCasesOperationController(IRefCasesService refCasesService) ①
        {
            this.refCasesService = refCasesService;
        }

        [HttpPost]
        public Year[] GetYears([FromQuery]Guid? rootSectionID) ②
        {
            return refCasesService.GetYears(rootSectionID).ToArray();
        }

        [HttpPost]
        public Section[] GetSections([FromQuery]Guid yearID, [FromQuery]Guid?
rootSectionID) { ③
            return refCasesService.GetSections(yearID, rootSectionID).ToArray();
        }

        [HttpPost]
        public Case[] GetCases([FromQuery]Guid sectionID) ④
        {
            return refCasesService.GetCases(sectionID).ToArray();
        }

        [HttpPost]
        public string GetCaseDisplayName([FromQuery]Guid caseID) ⑤
        {
            return refCasesService.GetCaseTitle(caseID);
        }

        [HttpPost]
        public CaseSearchResult SearchCase([FromQuery]string caseName, [FromQuery]int
skipCount, [FromQuery]int maxCount, [FromQuery]Guid? rootSectionID) ⑥
        {
            var rows = refCasesService.SearchCases(caseName, rootSectionID); ⑦

            var result = new CaseSearchResult
            {
                Items = rows.Skip(skipCount).Take(maxCount).ToArray(), ⑧ ⑨

                HasMore = rows.Count > skipCount + maxCount
            }; ⑩
```

```
            return result;
        }
    }
}
```

① В конструкторе контроллера получаем ссылку на реализованный сервис для работы со Справочником номенклатуры дел 5.

② Возвращает список лет.

③ Возвращает список разделов.

④ Возвращает список дел.

⑤ Возвращает отображаемое название дела.

⑥ Поиск дела по имени.

⑦ Получаем все подходящие дела.

⑧ Оставляем только количество запрошенных клиентом.

⑨ Т. к. количество результатов поискового запроса `SearchCase` может быть достаточно большим, клиенту предоставлена возможность ограничивать его с помощью параметров `skipCount` (количество пропускаемых результатов) и `maxCount` (максимальное количество результатов, принимаемых клиентом за один раз).

⑩ Устанавливаем флаг наличия дополнительных результатов.

Ядро серверного расширения

Реализованные сущности необходимо зарегистрировать в серверном расширении.

Файл `RefCasesServerExtension\RefCasesServerExtension.cs`

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Reflection;
using System.Resources;
using Autofac;
using Autofac.Extras.Ordering;
using DocsVision.WebClient.Extensibility;
using DocsVision.WebClient.Helpers;
using DocsVision.WebClientLibrary.ObjectModel.Services.BindingConverters;
using Microsoft.Extensions.DependencyInjection;
using RefCasesServerExtension.BindingConverters;
using RefCasesServerExtension.Services;
```

```
namespace RefCasesServerExtension
{
    public class RefCasesServerExtension : WebClientExtension
    {
        public RefCasesServerExtension(IServiceProvider serviceProvider)
            : base()
        {
        }

        public override string ExtensionName
        {
            get { return Assembly.GetAssembly(typeof(RefCasesServerExtension)).GetName
().Name; }
        }

        public override Version ExtensionVersion
        {
            get { return new Version(FileVersionInfo.GetVersionInfo(Assembly
.GetExecutingAssembly().Location).FileVersion); }
        }

        #region WebClientExtension Overrides

        public override void InitializeServiceCollection(IServiceCollection services) ①
        {
            services.AddSingleton<IBindingConverter, RefCasesConverter>(); ②
            services.AddSingleton<IRefCasesService, RefCasesService>(); ③
        }

        protected override List<ResourceManager> GetLayoutExtensionResourceManagers() ④
        {
            return new List<ResourceManager>
            {
                { Resources.ResourceManager}
            };
        }

        #endregion
    }
}
```

① Регистрация компонентов, реализованных в расширении.

② Регистрируем конвертер.

③ Регистрируем сервисы

④ Также для примера добавлено несколько локализованных текстовых ресурсов, которые будут использоваться в клиентском расширении.

Контроллеры регистрировать не требуется.

Создание клиентского расширения с реализацией элемента управления

Непосредственно элемент управления для Web-клиента реализуется в клиентском расширении, и состоит из нескольких частей (см. пункт Разработка и публикация клиентского компонента элемента управления): класс параметров, интерфейс состояния, интерфейсный класс и класс реализации.

С точки зрения пользовательского интерфейса реализуемый элемент управления будет представлять собой поле для ввода текста, в котором по умолчанию будет отображаться значение из поля карточки, связанного со строкой *Дела Справочника номенклатуры дел*. Особенностью элемента управления является необходимость реализации дополнительного графического компонента, в котором будут отображаться данные для выбора значения из *Справочника номенклатуры дел 5*, а также возможность быстрого поиска дел по справочнику.

Клиентский сервис для работы со "Справочником номенклатуры дел"

Реализуемому элементу управления потребуется доступ к *Справочнику номенклатуры дел*, который предоставляют веб-методы, разработанные в серверном расширении. Взаимодействовать с веб-методами можно вызывая их напрямую в основном коде элемента или используя отдельный клиентский сервис.

Ниже представлен код такого сервиса, который будет использоваться далее.

Файл `RefCasesWebExtension\src\Services\RefCasesServices.ts`

```
import { $RequestManager } from "@docsvision/webclient/System/$RequestManager";
import { urlStore } from "@docsvision/webclient/System/UrlStore";
import { Models } from "../Controls/RefCases/Data/CaseModel";
import { serviceName } from "@docsvision/webclient/System/ServiceUtils";

export class RefCasesService {
    constructor(private services: $RequestManager) {
    }

    getYears(rootSectionID?: string): Promise<Models.RefCasesYearModel[]> { ①
        let url = urlStore.urlResolver.resolveApiUrl("GetYears", "RefCasesOperation");
```

```
        url = url + "?rootSectionID=" + rootSectionID;

        return this.services.requestManager.post<Models.RefCasesYearModel[]>(url, "");
    }

    getSections(yearID?: string, rootSectionID?: string): Promise<Models
.RefCasesSectionModel[]> { ②
        let url = urlStore.urlResolver.resolveApiUrl("GetSections", "RefCasesOperation");
        url = url + "?yearID=" + yearID + "&rootSectionID=" + rootSectionID;

        return this.services.requestManager.post<Models.RefCasesSectionModel[]>(url, "");
    }

    getCases(sectionID?: string): Promise<Models.RefCasesCaseModel[]> { ③
        let url = urlStore.urlResolver.resolveApiUrl("GetCases", "RefCasesOperation");
        url = url + "?sectionID=" + sectionID;

        return this.services.requestManager.post<Models.RefCasesCaseModel[]>(url, "");
    }

    getCaseTitleName(caseID?: string): Promise<string> { ④
        let url = urlStore.urlResolver.resolveApiUrl("GetCaseDisplayName",
"RefCasesOperation");
        url = url + "?caseID=" + caseID;

        return this.services.requestManager.post<string>(url, "");
    }

    searchCase(caseName: string, skipCount: number, maxCount: number, rootSectionID?:
string): Promise<Models.CaseSearchResult> { ⑤
        let url = urlStore.urlResolver.resolveApiUrl("SearchCase", "RefCasesOperation");
        url = url + "?caseName=" + caseName + "&skipCount=" + skipCount + "&maxCount=" +
maxCount + "&rootSectionID=" + rootSectionID;

        return this.services.requestManager.post<Models.CaseSearchResult>(url, "");
    }
}

export type $RefCasesService = { refCasesService: RefCasesService };
export const $RefCasesService = serviceName((s: $RefCasesService) => s.refCasesService);
```

① Возвращает модель списка лет.

② Возвращает модель списка (дерева) разделов.

③ Возвращает модель списка дел.

④ Возвращает отображаемое название дела.

⑤ Возвращает результат поиска дела по названию. В `skipCount` и `maxCount` будут передаваться ограничения для количества результатов, возвращаемых сервером.

Класс параметров

В классе параметров объявляются свойства и события элемента управления, указанные в его описателе.

Данный элемент управления представляет собой поле для ввода текста, основанный на элементе `InputBasedControl`, в котором уже объявлены стандартные свойства подобного элемента. Единственным новым свойством является свойство `RootSection`, содержащее идентификатор корневого раздела справочника.

Ниже приведён код класса параметров, наследующий основной набор свойств от `InputBasedControlParams` — класса параметров элемента `InputBasedControl`. Значение элемента управления описывается моделью `Models.RefCasesCaseDisplayModel`.

Файл `RefCasesWebExtension\src\Controls\RefCases\RefCases.tsx`

```
export class RefCasesParams extends InputBasedControlParams<Models
.RefCasesCaseDisplayModel> {

    @r standardCssClass?: string = "ref-cases"; ①

    @rw rootSection?: string; ②

    @rw services?: $LayoutDirectoryDesignerController & $EditOperationStore & $LayoutInfo
& $RefCasesService; ③
}
```

① Стандартный CSS класс со стилями элемента управления Необходимо переопределить с собственным названием CSS класса.

② Секция справочника, из которой разрешено выбирать дела.

Настраивается в программе Конструктор Web-разметок. Может быть пустым (ограничение отсутствует).

③ Клиентские сервисы. `$RefCasesService` — собственный сервис для работы с данными *Справочника номенклатуры дел*.

Интерфейс состояния

В интерфейсе состояния нужно объявить свойства, описывающие внутреннее

состояние элемента управления. В частности во внутреннем состоянии будет храниться "биндинг" элемента управления.

Файл RefCasesWebExtension\src\Controls\RefCases\RefCasesImpl.tsx

```
①
export interface RefCasesState extends RefCasesParams, InputBasedControlState<Models
.RefCasesCaseDisplayModel> {
    binding: IBindingResult<Models.RefCasesCaseDisplayModel>;  ②

    dialog: ModalWindow; ③

    requestHelper: RequestHelper; ④

    inputKeyDown: SimpleEvent<React.KeyboardEvent<any>>; ⑤
}
```

① @internal.

② Используется для хранения биндинга.

③ Диалоговое окно выбора *Дела* из справочника.

④ Вспомогательный компонент, предоставляющий метод для обработки длительных операций (будет использоваться при получении данных из справочника).

⑤ Событие ввода значения в поле элемента управления.

Интерфейсный класс

В интерфейсном классе необходимо создать экземпляр реализации элемента управления, загрузить "биндинг", а также реализовать метод getBindings, возвращающий "биндинг" с текущим значением.

Файл RefCasesWebExtension\src\Controls\RefCases\RefCases.tsx

```
export class RefCases extends InputBasedControl<Models.RefCasesCaseDisplayModel,
RefCasesParams, RefCasesState>
{
    protected getServices(): $LayoutInfo {
        return this.state.services;
    }

    protected createParams(): RefCasesParams { ①
        return new RefCasesParams();
    }
```

```
    @handler("binding") ②
    protected set binding(binding: IBindingResult<Models.RefCasesCaseDisplayModel>) {
        this.value = binding && binding.value;
        this.state.canEdit = editOperationAvailable(this.state.services, binding);
        this.state.binding = binding;
    }

    protected getBindings() { ③
        let binding = cloneObject(this.state.binding);
        return [getBindingResult(binding, this.params.value && this.params.value.id ||
null, () => at(RefCasesParams).labelText)];
    }

    protected createImpl() { ④
        return new RefCasesImpl(this.props, this.state);
    }
}
```

① Инициализация параметров элемента управления.

② Загружаем биндинг при инициализации элемента управления.

③ Возвращаем биндинги.

④ Предоставляем экземпляр реализации элемента управления Справочник номенклатуры дел.

Класс реализации

В данном примере элемент управления представляет собой поле для ввода текста, дополненное функцией быстрого поиска, реализуемой с использованием компонента Typeahead.

Реализация класса в данном примере является достаточно большой: здесь код будет приведён частично.

Файл `RefCasesWebExtension\src\Controls\RefCases\RefCasesImpl.tsx`

```
①
export class RefCasesImpl extends InputBasedControlImpl<Models.RefCasesCaseDisplayModel,
RefCasesParams, RefCasesState>
{
    private typeahead: Typeahead;

    constructor(props: RefCasesParams, state: RefCasesState) {
        super(props, state);

        this.state.requestHelper = new RequestHelper(() => this.forceUpdate()); ②
```

```
        this.state.inputKeyDown = new SimpleEvent<React.KeyboardEvent<any>>(this);

        this.findItems = this.findItems.bind(this);
        this.showDictionary = this.showDictionary.bind(this);
        this.onSelected = this.onSelected.bind(this);
        this.attachTypeahead = this.attachTypeahead.bind(this); ③
    }

    async showDictionary() { ④
        if (this.state.dialog && this.state.dialog.IsOpened) {
            return;
        }

        let controlInModal: RefCasesSelectDialog; ⑤

        let params = new ModalWindowParams(); ⑥
        params.headerText = resources.RefCases_SelectFromDirectory;
        params.content = "";
        params.buttonOkShow = true;
        params.buttonOkText = resources.Navigator_ButtonSelect;

        let okFunction = () => { ⑦
            let selectedCase = cloneObject(controlInModal.selectedCase);

            if (selectedCase) { ⑧
                this.state.services.refCasesService.getCaseTitleName(selectedCase
.uniqueId).then((title) => {

                    let displayValue = {
                        id: selectedCase.uniqueId,
                        name: title
                    } as Models.RefCasesCaseDisplayModel;

                    this.setValue(displayValue, true);
                });

                if (this.state.dialog) {
                    this.state.dialog.Hide();
                    this.state.dialog = null;
                }
            }
        };

        params.buttonOkFunction = okFunction; ⑨

        const value = this.getValue(); ⑩
        this.state.dialog = new ModalWindow(params);
```

```
        renderModalContent(this.state.dialog, ( ⑪
            <RefCasesSelectDialog key={this.state.name + "_Modal"} ref={(el) =>
controlInModal = el}
                rootSectionId={this.state.rootSection}
                services={this.state.services}

                nodeSelected={(node) => { ⑫
                    if (this.state.dialog) {
                        if (node) this.state.dialog.OkButtonElement.classList.remove(
"disabled");
                        else this.state.dialog.OkButtonElement.classList.add("disabled");
                    }
                }}

                nodeAccepted={okFunction} /> ⑬
        ));

        this.state.dialog.Show();
        this.state.dialog.OkButtonElement.classList.add("disabled");
    }


    protected renderInputWithPlaceholder(): React.ReactNode { ⑭

        let buttons: IBoxWithButtonsButtonInfo[] = [ ⑮
            {
                onClick: this.showDictionary,
                name: "open-dictionary",
                iconClassName: "open-dictionary-button-icon dv-ico dv-ico-dictionary",
                visible: this.editAvailable,
                title: resources.RefCases_SelectFromDirectory,
                disabled: !this.editAvailable,
                tabIndex: this.getTabIndex(),
            }
        ];

        return ( ⑯
            <Typeahead className={"universal-directory-box"} extraButtons={buttons}
                findItems={this.findItems}
                clearButton={this.hasValue()}
                searchText={this.state.inputText}
                afterOpenCallback={() => this.afterOpenCallback()}
                popoverClassName={this.state.standardCssClass}
                popoverAttributes={{ "data-control-name": this.state.name }}
                inputKeyDown={this.state.inputKeyDown}
                onSelected={this.onSelected}
                disabled={!this.editAvailable}
```

```
                ref={this.attachTypeahead}>
                {super.renderInputWithPlaceholder()}
            </Typeahead>
        );
    }



    protected findItems(typeaheadQuery: ITypeaheadSearchQuery): Promise
<ITypeaheadSearchResult> { ⑰
        return new Promise<ITypeaheadSearchResult>((resolve, reject) => {
            this.state.services.refCasesService.searchCase(typeaheadQuery.searchText,
typeaheadQuery.skipCount, typeaheadQuery.maxCount, this.state.rootSection).then(response
=> {
                let result = {
                    items: response.items.map(item => new CaseTypeaheadVariant(item)), ⑱

                    hasMore: response.hasMore ⑲
                } as ITypeaheadSearchResult;
                resolve(result);
            }).catch(reject);
        });
    }
}
```

① @internal.

② Инициализация компонента для выполнения длительных операций.

③ Связывание обработчиков с контекстом.

④ Отрисовка основного элемента управления: поле для ввода текста, к которому добавляется стандартная кнопка выбора значения из справочника.

⑤ Кнопка открытия справочника. Отключается, если нет прав на операцию редактирования.

⑥ Формируем элемент с быстрым поиском.

⑦ Метод, отображающий диалоговое окно выбора *Дела* из справочника (реализовано в отдельном компоненте).

⑧ Компонент диалогового окна выбора из справочника.

⑨ Устанавливаем параметры диалогового окна.

⑩ Обработчик нажатия кнопки **OK** в диалоговом окне.

⑪ Если выбрано *Дело*, его модель (`RefCasesCaseDisplayModel`) устанавливается в значение элемента управления.

⑫ Устанавливается обработчик нажатия кнопки **ОК**.

⑬ Получаем текущее значение элемента управления.

⑭ Формируем диалоговое окно.

⑮ Кнопка **ОК** включается при выборе *Дела*.

⑯ Обработчик для двойного щелчка по делу — аналогично нажатию кнопки **ОК**.

⑰ Обработчик события быстрого поиска по справочнику. Результаты поиска отображаются в списке, выводимом под окном поля ввода.

⑱ Результаты поиска, должны быть приведены к `ITypeaheadSearchResult`.

⑲ Флаг, сообщающий о наличии результатов, не включенных в данный ответ.

В данном компоненте три ключевых функции:

- `renderInputWithPlaceholder` — формирует поле для ввода текста с возможностью быстрого поиска, которая реализована в компоненте `Typeahead`.
- `showDictionary` — открывает диалоговое окно с элементами для выбора дела из справочника.
- `findItems` — реализует функцию получения результатов поиска дел по справочнику.

Компонент `Typeahead` умеет ограничивать кол-во результатов, получаемых за один раз. Данные ограничения регулируются с помощью его параметров:

- `firstPageSize` — максимальное кол-во результатов при первом запросе, по умолчанию — 8.
- `nextPageSize` — максимальное кол-во результатов при следующих запросах, по умолчанию — 15.

При выполнении первого запроса в функцию `findItems` передаётся аргумент `ITypeaheadSearchQuery` со значениями `skipCount = 0` и `maxCount = firstPageSize`. При следующем запросе — при нажатии пользователем кнопки **Показать ещё**, в `findItems` передаются значения `skipCount = firstPageSize` и `maxCount = nextPageSize`. Таким образом клиент может запросить у сервера недостающие данные, если после первого запроса сервер вернул флаг `hasMore = true`.

Компонент диалогового окна выбора дела из справочника

Диалоговое окно выбора дела из *Справочника номенклатуры дел* представляет собой панель с тремя элементами управления:

- Раскрывающийся список для выбора года,
- Дерево разделов,
- Дерево дел.

> В модуле *Делопроизводство 5* секция дел является древовидной, функция вложенных дел не используется. Тем не менее в данном примере для совместимости используется элемент, поддерживающий древовидную структуру данных.
>
> С целью уменьшения количества исходного кода в данном примере умышленно не используется функция выбора текущего значения элемента управления в окне выбора Дела. Данная возможность может быть реализована самостоятельно.

В приведённом далее коде пропущены участки с классом параметров и интерфейсом состояния диалогового окна. Полный исходный код доступен на GitHub.

Файл `RefCasesWebExtension\src\Controls\RefCases\RefCasesSelectDialog\RefCasesSelectDialog.tsx`

```
export class RefCasesSelectDialog extends React.Component<IRefCasesSelectDialogProps,
IRefCasesSelectDialogState> {

①
    state: IRefCasesSelectDialogState;

    constructor(props: IRefCasesSelectDialogProps) {
        super(props);

        this.state = {} as IRefCasesSelectDialogState;
        this.state.requestHelper = new RequestHelper(() => this.forceUpdate());


        this.collectYearsList = this.collectYearsList.bind(this);
        this.loadSectionsTree = this.loadSectionsTree.bind(this);
        this.loadCasesTree = this.loadCasesTree.bind(this);

        this.onSectionNodeSelected = this.onSectionNodeSelected.bind(this);
        this.onCaseNodeSelected = this.onCaseNodeSelected.bind(this);
        this.onCaseNodeAccepted = this.onCaseNodeAccepted.bind(this); ②

        this.collectYearsList(); ③
    }
```

```
    public get selectedCase() {
        return this.state.selectedCaseNode;
    }

    protected collectYearsList() { ④
        this.props.services.refCasesService.getYears(this.props.rootSectionId).then
((items) => { ⑤
            this.state.years = items.map(x => ({
                id: x.id,
                title: x.displayValue
            } as IComboBoxElement));

            this.setState({ showYearsList: true }); ⑥
        });
    }

    protected loadSectionsTree(): Promise<IDynamicTreeNodeData[]> { ⑦
        return new Promise<IDynamicTreeNodeData[]>((resolve, reject) => {
            this.state.requestHelper.send(
                () => this.props.services.refCasesService.getSections(this.state
.selectedYearID, this.props.rootSectionId),
                items => {
                    let nodes = RefCasesSectionTreeNode.Create(items);
                    resolve(nodes);
                },
                reject);
        });
    }

    protected loadCasesTree(): Promise<IDynamicTreeNodeData[]> { ⑧
        return new Promise<IDynamicTreeNodeData[]>((resolve, reject) => {
            this.state.requestHelper.send(
                () => this.props.services.refCasesService.getCases(this.state
.selectedSectionID),
                items => {
                    let nodes = RefCasesCaseTreeNode.Create(items);
                    resolve(nodes);
                },
                reject);
        });
    }

    protected onSectionNodeSelected(node: TreeNode) { ⑨
        this.state.selectedSectionID = node.uniqueId;

        this.state.selectedCaseNode = null;
        this.props.nodeSelected && this.props.nodeSelected(null);
```

```
        this.setState({ showCasesTree: false }, () => this.setState({ showCasesTree: true
}));
    }

    protected onCaseNodeSelected(node: TreeNode) { ⑩
        this.state.selectedCaseNode = node.data as RefCasesCaseTreeNode;
        this.props.nodeSelected && this.props.nodeSelected(node.data as
RefCasesCaseTreeNode);
    }

    protected onCaseNodeAccepted(node: TreeNode) { ⑪
        this.state.selectedCaseNode = node.data as RefCasesCaseTreeNode;
        this.props.nodeAccepted && this.props.nodeAccepted(node.data as
RefCasesCaseTreeNode);
    }

    render(): React.ReactNode { ⑫

        let yearsList = <div>{resources.RefCases_Years} ⑬
            <CommonComboBox elements={this.state.years} selectedID={this.state
.selectedYearID}
            onChange={(selectedElement: IComboBoxElement) => { ⑭
                this.setState({ selectedYearID: selectedElement.id });
                this.setState({ showSectionsTree: false }, () => this.setState({
showSectionsTree: true })); ⑮
                this.setState({ showCasesTree: false });
                this.forceUpdate();
            }}

            renderElementList={(elements, expanded) =>
                <PopoverComboBoxBodyContent mode={PopoverMode.BottomDropdown} isOpen=
{expanded} className="combobox-helper">
                    {elements}
                </PopoverComboBoxBodyContent>
            } />
            </div>;

        let sectionsTree = <div className="ref-cases-dialog__tree"> ⑯
            <div className="tree-block">{resources.RefCases_Sections}
                <DynamicTree loadNodes={this.loadSectionsTree} treeHeight={300}
                nodeSelected={this.onSectionNodeSelected} > ⑰
                </DynamicTree>
            </div>
        </div>;

        let casesTree = <div className="ref-cases-dialog__tree"> ⑱
            <div className="tree-block">{resources.RefCases_Cases}
```

```
                <DynamicTree loadNodes={this.loadCasesTree} treeHeight={300}
                nodeSelected={this.onCaseNodeSelected} nodeAccepted={this
.onCaseNodeAccepted} > ⑲
                </DynamicTree>
            </div>
        </div>;

        return (
            <div>
                {this.state.showYearsList && yearsList}
                {this.state.showSectionsTree && sectionsTree}
                {this.state.showCasesTree && casesTree}
            </div>
        );
    }
}
```

① @internal

② Связываем функции и обработчиков событий с контекстом.

③ Загружаем список лет из справочника.

④ Загружаем из *Справочника номенклатуры дел* список лет в `state.years`, который является источником данных для элемента управления `CommonComboBox`.

⑤ Если установлен раздел, из которого возможен выбор *Дел* — `rootSectionId`, будет возвращен только год с данным разделом.

⑥ Показываем элемент со списком лет.

⑦ Возвращает список разделов из Справочника номенклатуры дел.

⑧ Загружает список дел после выбора раздела.

⑨ Сохраняет *Дело* в `selectedNode` после его выбора в списке дел.

⑩ Сохраняет *Дело* в `selectedNode` после его выбора в списке дел и нажатия кнопки **Выбрать**.

⑪ Сохраняет *Дело* в `selectedNode` после его выбора в списке дел.

⑫ Инициализация интерфейса.

⑬ Список лет.

⑭ При выборе года из списка инициализируем дерево разделов.

⑮ Элемент для выбора *Раздела* перемонтируется, элемент для выбора *Дела* отмонтируется.

⑯ Дерево *Разделов*.

⑰ При выборе *Раздела* из дерева разделов вызываем метод, загружающий дерево *Дел*.

⑱ Список дел.

⑲ При выборе *Дела*, сохраняем значение узла, а при двойном щелчке вызываем обрабочик `onCaseNodeAccepted`.

Пример стиля для дерева дел и дерева разделов

Файл `RefCasesWebExtension/src/Controls/RefCases/RefCasesSelectDialog/RefCasesSelectDialog.scss`

```
.ref-cases-dialog__tree {
    border-bottom: 1px solid lightgray;
    padding-top: 10px;
    display: inline-block;
    width: 50%;
}
```

Дополнительно

Руководство по разработке решений Web-клиента

Дополнительно

Дополнительная информация по разработке решений Web-клиента.

- Стандартные свойства и события элементов управления
- Нестандартные встроенные редакторы свойств
- Стандартные стили
- Больше информации в навигационном меню слева…

Стандартные свойства и события элементов управления

Далее приведено описание стандартных свойств и событий элементов управления, которое включает: тип значения свойства, метод получения описателя свойства, а также ключ ресурса с локализованным названием — для событий.

Свойства

Name / Название

Тип: String

Метод получения описателя: PropertyFactory.GetNameProperty

Width / Ширина в процентах

Тип: Double

Метод получения описателя: PropertyFactory.GetWidthProperty

Header / Заголовок

Тип: String

Метод получения описателя: PropertyFactory.GetHeaderProperty

Binding / Свойство, содержащее механизм получения данных

Тип: String

Метод получения описателя свойства: PropertyFactory.GetBindingProperty

DataSource / Источник данных

Тип: Guid

Метод получения описателя: PropertyFactory.GetDataSourceProperty

EditOperation / Операция редактирования

Тип: Guid

Метод получения описателя: PropertyFactory.GetEditOperationProperty

ExtendedDataSource / Расширенный источник данных

Тип: Guid

Метод получения описателя: PropertyFactory.GetExtendedDataSourceProperty

StandardCssClass / Стандартный css класс

Тип: String

Метод получения описателя: PropertyFactory.GetStandardCssClassProperty

CustomCssClasses / Дополнительные css классы

Тип: String

Метод получения описателя: PropertyFactory.GetCustomCssClassesProperty

TabStop / Переходить по TAB

Тип: Boolean

Метод получения описателя: PropertyFactory.GetTabStopProperty

Visibility / Видимость

Тип: Boolean

Метод получения описателя: PropertyFactory.GetVisibilityProperty

Default / Значение по умолчанию

Тип: String

Метод получения описателя: PropertyFactory.GetDefaultProperty

Required / Обязательное

Тип: Boolean

Метод получения описателя: PropertyFactory.GetRequiredProperty

EditMode / Режим редактирования

Тип: Int32

Метод получения описателя: PropertyFactory.GetEditModeProperty

PlaceHolder / Заполнитель

Тип: String

Метод получения описателя: PropertyFactory.GetPlaceHolderProperty

Tip / Подсказка

Тип: String

Метод получения описателя: PropertyFactory.GetTipProperty

LabelText / Текст метки

Тип: String

Метод получения описателя: PropertyFactory.GetLabelTextProperty

ShowEmptyLabel / Отображать метку без значения

Тип значения: `Boolean`

Метод получения описателя: `PropertyFactory.GetShowEmptyLabelProperty`

MinWidth / Минимальная ширина

Тип: `Double`

Метод получения описателя: `PropertyFactory.GetMinWidthProperty`

ButtonText / Текст кнопки

Тип: `String`

Метод получения описателя: `PropertyFactory.GetButtonTextProperty`

События

Все события имеют тип `String`.

Click / При щелчке

Метод получения описателя: `PropertyFactory.GetClickEvent`

Ключ ресурса: `ControlTypes_ClickEventProperty`

MouseOver / При наведении курсора

Метод получения описателя: `PropertyFactory.GetMouseOverEvent`

Ключ ресурса: `ControlTypes_MouseOverEventProperty`

MouseOut / При отведении курсора

Метод получения описателя: `PropertyFactory.GetMouseOutEvent`

Ключ ресурса: `ControlTypes_MouseOutEventProperty`

Focus / При получении фокуса

Метод получения описателя: `PropertyFactory.GetFocusEvent`

Ключ ресурса: `ControlTypes_FocusEventProperty`

Blur / При потере фокуса

Метод получения описателя: `PropertyFactory.GetBlurEvent`

Ключ ресурса: `ControlTypes_BlurEventProperty`

Collapsed / При сворачивании

Метод получения описателя: `PropertyFactory.GetCollapsedEvent`

Ключ ресурса: `ControlTypes_CollapsedEventProperty`

Collapsing / Перед сворачиванием

Метод получения описателя свойства: PropertyFactory.GetCollapsingEvent`

Ключ ресурса: `ControlTypes_CollapsingEventProperty`

Expanded / При разворачивании

Метод получения описателя: `PropertyFactory.GetExpandedEvent`

Ключ ресурса: `ControlTypes_ExpandedEventProperty`

Expanding / Перед разворачиванием

Метод получения описателя: `PropertyFactory.GetExpandingEvent`

Ключ ресурса: `ControlTypes_ExpandingEventProperty`

DataChanged / При смене данных

Метод получения описателя: `PropertyFactory.GetDataChangedEvent`

Ключ ресурса: `ControlTypes_DataChangedEventProperty`

Нестандартные встроенные редакторы свойств

1. Редактор локализуемых текстовых свойств.

Рисунок 119. Редактор локализуемых текстовых свойств

Пример использования в коде:

```
private PropertyDescription GetLocalizableMessagePropertyDescription () {
    var propertyDescription = new PropertyDescription {
        Type = typeof (string),
        Name = "LocalizableMessage",
        Category = DocsVision.Platform.Tools.LayoutEditor.Infrostructure.
PropertyCategoryConstants.DataCategory,
        DisplayName = "Localizable message",
        Localized = true,
        Editor = typeof (DocsVision.Platform.Tools.LayoutEditor.Editors
```

```
.LocalizedPropertyEditor)
    };
    return propertyDescription;
}
```

Для использования редактора нужно:

a. Установить свойство `Type` в `string`.

b. Установить свойство `Localized` в `true`.

c. Установить свойство `Editor` в `Docsvision.Platform.Tools.LayoutEditor.Editors.LocalizedPropertyEditor`.

2. Редактор перечислений.

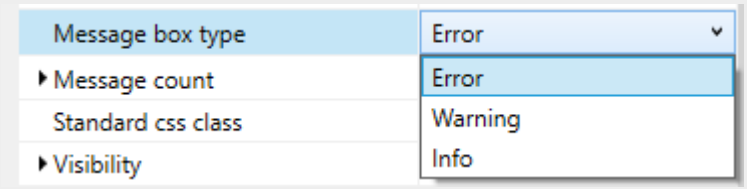


Рисунок 120. Редактор перечислений

Пример использования в коде:

```
enum MessageBoxType {
    Error = 0,
    Warning = 1,
    Info = 2
}

class MessageBoxTypeSource : Xceed.Wpf.Toolkit.PropertyGrid.Attributes
.IItemsSource { ①
    private readonly System.IServiceProvider serviceProvider;

    public MessageBoxTypeSource (System.IServiceProvider serviceProvider) {
        this.serviceProvider = serviceProvider;
    }

    public Xceed.Wpf.Toolkit.PropertyGrid.Attributes.ItemCollection GetValues ()
{ ②
        var itemCollection = new Xceed.Wpf.Toolkit.PropertyGrid.Attributes
.ItemCollection ();

        itemCollection.Add ((int) MessageBoxType.Error, MessageBoxType.Error
.ToString ());
        itemCollection.Add ((int) MessageBoxType.Warning, MessageBoxType.Warning
```

```
.ToString ());
        itemCollection.Add ((int) MessageBoxType.Info, MessageBoxType.Info
.ToString ());

        return itemCollection;
    }
}

class MessageBoxTypeEditor : Xceed.Wpf.Toolkit.PropertyGrid.Editors.ITypeEditor {
③
    public FrameworkElement ResolveEditor (PropertyItem propertyItem) {

        var editor = new ItemsSourceEditor (typeof (MessageBoxTypeSource)); ④
        return editor.ResolveEditor (propertyItem);
    }
}

var propertyDescription = new PropertyDescription { ⑤
    Type = typeof (int),
    Name = "MessageBoxType",
    Category = PropertyCategoryConstants.BehaviorCategory,
    DisplayName = Resources.Property_MessageBoxType_DisplayName,
    DefaultValue = 0,
    Editor = typeof (MessageBoxTypeEditor) ⑥
};
```

① Класс, содержащий данные списка. Должен быть унаследован от IItemsSource.

② Метод `GetValues` должен вернуть значения. Тип значений `ItemCollection`.

③ Класс редактора значения свойства должен содержать значения.

④ Редактор значения свойства создаётся из класса `ItemsSourceEditor`, в который передается класс с данными списка.

⑤ Свойство для выбора значения из списка.

⑥ Разработанный редактор значения свойства.

3. Редактор устройствозависимых свойств.

Значение свойств данного вида зависят от типа устройства, на котором открыт Web-клиент.

Пример использования в коде:

```
private PropertyDescription GetDeviceDependentPropertyDescription () {
    var propertyDescription = new PropertyDescription {
        Type = typeof (int),
        Name = "DeviceDependent",
        Category = PropertyCategoryConstants.DataCategory,
        DisplayName = "Device Dependent",
        DeviceDependent = true,
        DefaultValue = DeviceDependentHelper.GetDeviceDependentDefaultValue(1)
    };
    return propertyDescription;
}
```

Стандартные стили

Модуль предоставляет возможность при разработке собственного Решения использовать стандартные стили, что позволяет сохранить общую цветовую схему.

Стандартные стили:

- `.card-type-background-color` — определяет фон элемента типа карточки;
- `.card-type-background-color-hover` — определяет фон элемента типа карточки при наведении курсора;
- `.card-type-background-color-light` — определяет светлый фон элемента типа карточки;
- `.card-type-background-color-light-hover` — определяет светлый фон элемента типа карточки при наведении мыши;
- `.card-type-background-color-disabled` — определяет заблокированный фон элемента типа карточки (например, когда кнопка недоступна для нажатия);
- `.card-type-foreground-color` — определяет цвет шрифта типа карточки.

Пример использования стилей:

```
<div class="card-type-background-color">
 <a href="/opencard" class="card-type-foreground-color">Открыть</a>
</div>
```

Описание проекта TemplateWebExtension

Проект `TemplateWebExtension` является частью репозитория WebClient-Samples.

Данный проект предназначен для разработки и сборки клиентских расширений и скриптов.

Описание файлов проекта

В папке "TemplateWebExtension" расположены файлы:

- `copy-path.js` — содержит константы с названием расширения и параметрами сборки; в данном файле указывается каталог, в который выгружаются файлы при сборке.
- `gulpfile.js` — файл конфигурации Gulp (используется для сборки стилей);
- `package.json` — файл конфигурации NPM с командами сборки и списком зависимостей.
- `package-lock.json` — служебный файл NPM.
- `rollup.config.js` — файл конфигурации Rollup. Используется для сборки скриптов в один файл — "бандл".
- `tsconfig.json` — файл конфигурации TypeScript.
- `node_modules` — служебная папка NPM, в которую при выполнении команды npm `install` загружаются npm-зависимости.
- `src` — папка для размещения исходных файлы расширения.

Папка src содержит файлы:

- `EventHandlers.tsx` — пример файла с обработчиком события.
- `Index.ts`-- файл с обязательной функцией регистрации клиентского расширения.
- `Template.scss` — пример файла стилей.

Пример рассчитан на версию Web-клиента dv6 или выше.

Перечень необходимых инструментов:

- NodeJS v22.14.0 и выше
- Microsoft Visual Studio 2022. == Описание файла Index.ts

Содержимое файла `Index.ts`:

```
import * as EventHandlers from "./EventHandlers";
import { extensionManager } from "@docsvision/webclient/System/ExtensionManager";

extensionManager.registerExtension({ ①
```

```
    name: "Template web extension",
    version: "1.0",
    globalEventHandlers: [ EventHandlers ]
})
```

① Регистрация расширения позволяет корректно установить все обработчики событий, сервисы и прочие сущности веб-приложения.

Команда `import * as EventHandlers from "./EventHandlers"` импортирует все публичные сущности (с `export`) из файла `EventHandlers` (`.tsx`).

Чтобы импортировать функции из другого файла, необходимо добавить новую команду импорта, указав относительный путь к файлу:

```
import * as EventHandlers from "./EventHandlers";
import * as NewEventHandlers from "./Scripts/NewEventHandlers"; ①
```

① Добавлена строчка импорта из файла /Scripts/NewEventHandlers.

Команда `import { extensionManager } from "@docsvision/webclient/System/ExtensionManager"` импортирует из модуля "@docsvision/webclient/System/ExtensionManager" сущность extensionManager (менеджер расширений).

Команда `extensionManager.registerExtension({ ⋯ })` регистрирует клиентское расширение с названием `Template web extension`, которое предоставляет глобальные обработчики (`globalEventHandlers`).

Чтобы добавить в расширение обработчики из нового файла, добавьте их в массив `globalEventHandlers`:

```
globalEventHandlers: [ EventHandlers, NewEventHandlers] ①
```

① Добавлены обработчики из `NewEventHandlers`.

Описание файла EventHandlers.tsx

Содержимое файла:

```
import { IEventArgs } from "@docsvision/webclient/System/IEventArgs";
import { Layout } from "@docsvision/webclient/System/Layout";

export function someHandler(sender: Layout, e: IEventArgs) { ①
    ...
}
```

① Все функции, классы и переменные, используемые за пределами модуля (т.е. файла), должны экспортироваться (содержать ключевое слово export в объявлении).

Команды `import` в начале файла выполняют загрузку из модулей Web-клиента сущностей, используемых в коде обработчика.

Функция `someHandler` является примером обработчика события разметки (тип `sender` — `Layout`). Ключевое слово `export` указывает, что данная функция доступна для импорта. Чтобы экспортировать другую функцию, необходимо также указывать ключевое слово `export`:

```
export function someHandler(sender: Layout, e: IEventArgs) {

}

export function newHandler(sender: Employee, e: IEventArgs) { ①
}
```

① Добавлена новая функция для обработки события в ЭУ с типом `Employee`.

Для создания клиентского скрипта достаточно добавить функции-обработчики в файл `EventHandlers.tsx`. Например, следующий код, вызванный при изменении значения элемента управления `Сотрудник`, отобразит информационное окно с ФИО выбранного сотрудника:

```
import { Employee } from "@docsvision/webclient/BackOffice/Employee";
import { IEventArgs } from "@docsvision/webclient/System/IEventArgs";
import { MessageBox } from "@docsvision/webclient/Helpers/MessageBox/MessageBox"; ①

export function changeEmployeeHandler(sender: Employee, e: IEventArgs) { ②
  MessageBox.ShowInfo(sender.params.value.displayName);
}
```

① Импорт необходимых сущностей из модулей Web-клиента.

② Реализация функции-обработчика.

Функции-обработчики должны экспортироваться — сопровождаться ключевым словом `export`.

Сервисы получения моделей объектов

Следующие сервисы предназначены для получения на клиенте моделей объектов для программного изменения значений элементов управления.

- `$DirectoryDesignerRowController` — предоставляет методы для получения объектов *Конструктора справочников*:
  - `getDirectoryDesignerRow(rowId: string)` — получает объект *Строка справочника* для строки с идентификатором `rowId`.
  - `getDirectoryDesignerRows(rows: GenModels.CollectionRequestModelOf<string>)` — получает объекты *Строка справочника* для строк с идентификаторами rows.
- `$CardKindController` — предоставляет методы для получения объектов *Справочника видов карточек*:
  - `getCardKind(cardKindId: string)` — получает объект *Вид карточки* для вида с идентификатором `cardKindId`.
  - `getCardKinds(cardKinds: GenModels.CollectionRequestModelOf<string>)` — получает объекты *Вид карточки* для видов с идентификаторами `cardKinds`.
- `$DepartmentController` — предоставляет методы для получения объектов *Справочника сотрудников* и *Справочника контрагентов*:
  - `getStaffDepartment(departmentId: string)` — получает объект *Подразделение* для подразделения с идентификатором `departmentId`.
  - `getPartnerDepartment(departmentId: string)` — получает объект *Подразделение контрагента* для подразделения с идентификатором `departmentId`.
  - `getStaffDepartments(departments: GenModels.CollectionRequestModelOf<string>)` — получает объекты *Подразделение* для подразделений с идентификаторами `departments`.
  - `getPartnerDepartments(departments: GenModels.CollectionRequestModelOf<string>)` — получает объекты *Подразделение контрагента* для подразделений с идентификаторами `departments`.
- `$EmployeeController` — предоставляет методы для получения объектов *Справочника сотрудников*:
  - `getEmployee(employeeId: string, options?: RequestOptions)` — получает объект *Сотрудник* для сотрудника с идентификатором `employeeId`.

    - `getEmployees(employees: GenModels.CollectionRequestModelOf<string>, options?: RequestOptions)` — получает объекты *Сотрудник* для сотрудников с идентификаторами `employees`.

- `$PartnerEmployeeController` — предоставляет методы для получения объектов *Справочника контрагентов*:
    - `getPartnerEmployee(employeeId: string, options?: RequestOptions)` — получает объект Сотрудник контрагента для сотрудника с идентификатором `employeeId`.
    - `getPartnerEmployees(employees: GenModels.CollectionRequestModel, options?: RequestOptions)` — получает объекты *Сотрудник контрагента* для сотрудников с идентификаторами `employees`.

Следующий код демонстрирует пример получения объекта *Подразделение* с идентификатором `6832216B-FED9-4D20-B1D4-0097C73D446E` при помощи сервиса `$DepartmentController`.

```
export function superHandler(sender: CustomButton, e: IEventArgs) {
    let departmentControllerService = sender.layout.getService($DepartmentController);
    let departmentModel = departmentControllerService.getStaffDepartment("6832216B-FED9-
4D20-B1D4-0097C73D446E");
}
```

Подробное описание сервисов приведено в справочнике по JS API.

Список специальных адресов Web-клиента

Адреса, перечисленные в данном разделе, позволяют вызвать специальные функции Web-клиента из адресной строки браузера.

Работа со строками справочников

- Открыть окно просмотра строки справочника: `#/RowView/%cardId%/%sectionId%/%rowId%`.
- Открыть окно создания строки справочника: `#/NewRow/%cardId%/%sectionId%/%parentRowId%/%parentTreeRowID%`.
- Открыть окно редактирования строки справочника: `#/RowEdit/%cardId%/%sectionId%/%rowId%`.

Здесь:

- `%cardId%` — идентификатор карточки.
- `%sectionId%` — идентификатор секции карточки cardId.
- `%rowId%` — идентификатор редактируемой/открываемой строки.
- `%parentRowId%` — идентификатор родительской строки родительской секции (если секция находится внутри другой секции).
- `%parentTreeRowID%` — идентификатор родительской строки текущей секции (в случае, если секция древовидная).

Работа со Справочником контрагентов

Добавление:

- Открыть окно добавления корневой организации: `#/NewOrganizationRow/65ff9382-17dc-4e9f-8e93-84d6d3d8fe8c/c78abded-db1c-4217-ae0d-51a400546923/00000000-0000-0000-0000-000000000000/00000000-0000-0000-0000-000000000000`.
- Открыть окно добавления дочерней организации: `#/NewOrganizationRow/65ff9382-17dc-4e9f-8e93-84d6d3d8fe8c/c78abded-db1c-4217-ae0d-51a400546923/00000000-0000-0000-0000-000000000000/%parent-id%`.

  Здесь `%parent-id%` — идентификатор родительской организации.

- Открыть окно добавления подразделения: `#/NewDepartmentRow/65ff9382-17dc-4e9f-8e93-84d6d3d8fe8c/c78abded-db1c-4217-ae0d-51a400546923/00000000-0000-0000-0000-000000000000/%parent-id%`.

  Здесь `%parent-id%` — идентификатор организации.

- Открывает окно добавления сотрудника: `#/NewRow/65ff9382-17dc-4e9f-8e93-84d6d3d8fe8c/1a46bf0f-2d02-4ac9-8866-5adf245921e8/%parent-id%/00000000-0000-0000-0000-000000000000`.

  Здесь `%parent-id%` — идентификатор организации/подразделения.

Просмотр:

- Открывает окно просмотра сотрудника: `#/RowView/65ff9382-17dc-4e9f-8e93-84d6d3d8fe8c/1a46bf0f-2d02-4ac9-8866-5adf245921e8/%id%`.

  Здесь `%id%` — идентификатор сотрудника.

- Открывает окно просмотра организации: `#/OrganizationRowView/65ff9382-17dc-4e9f-8e93-84d6d3d8fe8c/c78abded-db1c-4217-ae0d-51a400546923/%id%`.

  Здесь `%id%` — идентификатор организации.

- Открывает окно просмотра подразделения: `#/DepartmentRowView/65ff9382-17dc-4e9f-8e93-84d6d3d8fe8c/c78abded-db1c-4217-ae0d-51a400546923/%id%`.

  Здесь `%id%` — идентификатор подразделения.

Редактирование:

- Открывает окно редактирования сотрудника: `#/RowEdit/65ff9382-17dc-4e9f-8e93-84d6d3d8fe8c/1a46bf0f-2d02-4ac9-8866-5adf245921e8/%id%`.

  Здесь `%id%` — идентификатор сотрудника.

- Открывает окно редактирования организации: `#/OrganizationRowEdit/65ff9382-17dc-4e9f-8e93-84d6d3d8fe8c/c78abded-db1c-4217-ae0d-51a400546923/%id%`.

  Здесь `%id%` — идентификатор организации.

- Открывает окно редактирования подразделения: `#/DepartmentRowEdit/65ff9382-17dc-4e9f-8e93-84d6d3d8fe8c/c78abded-db1c-4217-ae0d-51a400546923/%id%`.

  Здесь `%id%` — идентификатор подразделения.

Механизм внедрения зависимостей на клиенте

В данном разделе рассмотрены особенности передачи клиентских сервисов с использованием механизма внедрения зависимостей.

У сервиса имеется имя и интерфейс. Например, сервис для получения информации о текущей карточке имеет имя `$CardInfo`, и интерфейс `GenModels.CardInfoModel`. Получить сервис можно по имени при помощи метода `getService` у объекта элемента управления, разметки или глобального объекта `app`. Например, таким образом можно получить сервис `$CardInfo` в обработчике события:

```
export function onCustomButtonClick(sender: CustomButton, args: IEventArgs) {
    const cardInfo = sender.getService($CardInfo);
}
```

Сервисы, как правило, размещаются в контейнерах — специальных объектах. Имя сервиса — это ключ, по которому данный сервис можно получить в сервис-контейнере. Получить сервис-контейнер можно вызвав `getService` без параметров. Так альтернативный способ получения сервиса по ключу из контейнера будет выглядеть так:

```
export function onCustomButtonClick(sender: CustomButton, args: IEventArgs) {
     const cardInfo = sender.getService()[$CardInfo] as GenModels.CardInfoModel; ①
     const cardInfo2 = sender.getService<$CardInfo>().cardInfo; ②
}
```

① Получаем либо по ключу

② Либо иным образом

Сервис-контейнер уровня приложения доступен через глобальный объект app, однако его не рекомендуется использовать в большинстве случаев.

Сервис контейнер в элементах управления также доступен через параметр `services` (он же `props` при отрисовке элемента управления как React-компонента).

Все стандартные сервисы Web-клиента (приведены в JSDoc API).

Создание сервисов

Определение сервиса включает в себя три составляющих:

- Описание интерфейса сервиса и его имени.
- Определение реализации.
- Регистрация реализации.

1. Первое, что нужно сделать — выбрать название сервиса. Допустим, наш сервис будет называться `Greeting`. Тогда описание интерфейса и имени сервиса будет выглядеть следующим образом:

   ```
   export interface IGreetingService {
        showGreeting(name: string): Promise<void>;
   }
   export type $Greeting = { greeting: IGreetingService };
   export const $Greeting = serviceName<$Greeting, IGreetingService>(x => greeting);
   ```

2. Реализацию сервиса рекомендуется определять в отдельном от интерфейса файле (чтобы не создавать лишние зависимости). Реализацией

будет определение класса, реализующего интерфейс `IGreetingService`.

```
export class GreetingSevice implements IGreetingService {
      constructor(private services: $MessageWindow) {
      }
      showGreeting(name: string): Promise<void>{
            return this.services.messageWindow.showInfo("Добро пожаловать, " +
name);
      }
 }
```

3. Регистрируются сервисы в файле `Index.ts` веб-расширения через вызов `registerExtension`. Если сервис имеет отношение к разметке, то следует передавать его в поле `layoutServices`, иначе в `globalServices`.

```
extensionManager.registerExtension({
      name: "Greeting",
      version: "1.0",
      layoutServices: [
           Service.fromFactory($Greeting,
                (services: $StandardServices & $StandardControllers) =>
                      new GreetingSevice(services))
      ]
})
```

Стандартные web-сервисы

В данном разделе рассмотрены основы работы с сервисами, а также основные web-сервисы, доступные для использования в web-расширениях.

Сервисы представляют основной API взаимодействия с системой в части функциональности, не связанной с конкретными элементами управления. Они позволяют получить информацию о текущем контексте выполнения, выполнять запросы к серверу, показывать модальные окна, взаимодействовать с разметкой и использовать множество других функций системы.

Сервисы информации о текущей карточке/строке/разметке

Информацию о текущей разметке можно получить с помощью следующих сервисов:

- `$CardId` — идентификатор текущей карточки текущей разметки.

- `$RowId` — идентификатор текущей строки текущей разметки (если открыта разметка строки, а не карточки)
- `$CardTimestamp` — `timestamp` текущей карточки текущей разметки.
- `$CardInfo` — дополнительная информация о текущей карточке, текущей разметки. Например, там можно найти тип и вид карточки, ее состояние и другие данные.
- `$RowInfo` — дополнительная информация о текущей строке, текущей разметки (если открыта разметка строки).
- `$LayoutInfo` — дополнительная информация о текущей разметке.
- `$Layout` — экземпляр текущей разметки. Содержит множество различных методов, среди которых сохранение (`save`), перезагрузка (`reloadFromServer`), смены состояния карточки (`changeState`) и другие.
- `$ControlStore` — предоставляет доступ к элементам управления на разметке по имени через методы `get` и `tryGet`. Первый метод генерирует ошибку, в случае отсутствия элемента управления с таким именем, второй просто возвращает `null`.
- `$EditOperationStore` — позволяет получить информацию о доступности операции редактирования.

Сервисы информации об окружении

Информацию в целом о текущем контексте работы приложения можно с помощью следующих сервисов:

- `$CurrentEmployeeId` — идентификатор текущего пользователя.
- `$CurrentEmployeeAccountName` — имя учетной записи текущего пользователя.
- `$CurrentEmployee` — расширенная информация о текущем пользователе. Состав полей можно найти в описании типа `CurrentEmployeeInfo`.
- `$ApplicationTimestamp` — строка, изменяющаяся при перезапуске веб-сервиса.
- `$ApplicationSettings` — системные настройки (см. `GenModels.ApplicationSettings`).
- `$Locale` — двухбуквенное обозначение текущей выбранной локализации веб-клиента (например: `ru`, `en`).
- `$CurrentFolder` — информация о текущей открытой папке (см. `ICurrentFolderService`).

- `$DeviceType` — обозначение текущего выбранного типа устройства (компьютер/телефон/планшет). См. `GenModels.DeviceType`.

Сервисы для выполнения запросов

Выполнение запросов к серверу осуществляется через сервис `$RequestManager`. Сформировать URL запроса поможет сервис `$UrlResolver`.

Подробнее см. "Сервисы для выполнения запросов".

Сервисы для отображения сообщений

Показать на экране простое информационное сообщение можно с помощью сервиса `$MessageWindow`. Основные методы этого сервиса: `showInfo`, `showWarning`, `showError`. Они различаются только цветом оформления и отображают диалоговое окно.

Подробнее см. "Сервисы для выполнения запросов".

Сервис навигации

Для перехода между страницами существует сервис `$RouterNavigation`. Он предоставляет следующие методы:

```
export interface IRouterNavigation {
    refresh(): Promise<void>;
    goTo(route: string, refresh?: boolean): Promise<void>;
    openUrl(url: string, mode: GenModels.OpenMode): Promise<void>;
    back(): Promise<void>;
}
```

- `refresh` — перезагружает содержимое текущей страницы. Например, если открыта карточка, то перезагружается разметка карточки.
- `goTo` — открывает страницу по указанному хеш-адресу. Для получения корректного адреса можно использовать следующие функции:
  - `getCardViewRoute`, `getCardEditRoute`, `getCardCreateRoute` - для отображения разметки карточки.
  - `makeFolderUrl` - для открытия папки
  - `getViewDirectoryRoute` - для открытия справочника
  - `getRowEditRoute`, `getRowViewRoute`, `getRowCreateRoute` — для отображения разметок строки. Опциональный параметр refresh определяет поведение, в

случае если переданный ардес совпадает с текущим. Если `refresh = true`, то в такой ситуацию будет произведено обновление страницы.

- `openUrl` — расширение метода `goTo`, добавляющее возможность открывать адрес в новой вкладке или новом окне (параметр `mode`). При открытии в текущей вкладке работает также, как `goTo`.
- `back` — возврат на предыдущую страницу.

Настройка роутинга осуществляется с помощью сервиса `$Router`. В нем присутствуют методы для регистрации своих обработчиков. Подробнее об этом сервисе будет рассказано в другой статье, посвященной реализации собственных страниц.

Изменение основного шрифта Web-клиента

Основным шрифтом Web-клиента является "Roboto". По умолчанию поддерживаются два уровня жирности шрифта: стандартный (`400`) и жирный (`700`).

Если требуется, можно изменить настройки шрифта по умолчанию:

1. Добавьте новые шрифты или шрифт *Roboto* с требуемой жирностью в подпапку своего *Решения* в `{Каталог установки Web-клиент}\Content`.
2. Откройте для редактирования файл `{Каталог установки Web-клиент}\Content\fonts.html`.

```
<link rel="preload" href="Content/App/Assets/Fonts/Roboto-Regular.ttf" as="font"
type="font/ttf" crossorigin>
<style type="text/css" media="screen, print">
    @font-face {
        font-family: "Roboto";
        src: url("Content/App/Assets/Fonts/Roboto-Regular.ttf") format("truetype"),
            url('Content/App/Assets/Fonts/Roboto-Bold.ttf') format("truetype");
    }

    body {
        font-family: "Roboto";
        font-style: normal;
        font-weight: 400;
        font-display: swap;  /* IE11 */
    }
</style>
```

3. Измените настройки шрифта:
    - Если нужно добавить шрифты "Roboto" с новой жирностью — укажите их в `@font-face`:

```
@font-face {
        font-family: "Roboto";
        src: url("Content/App/Assets/Fonts/Roboto-Regular.ttf") format("truetype"),
            url('Content/App/Assets/Fonts/Roboto-Bold.ttf') format("truetype"),
            url('Content/Modules/SomeSolution/Fonts/Roboto-Black.ttf') format
("truetype");
    }
```

    - Если нужно изменить шрифт по умолчанию:
        a. Добавьте новое правило `@font-face` со своим шрифтом аналогично правилу для шрифта *Roboto*.
        b. Укажите название шрифта в разделе `body`.
4. Перезапустите **dvwebclient**.

При обновлении Web-клиента файл `fonts.html` будет перезаписан — после обновления нужно повторить изменения.

Библиотека классов

Руководство по разработке решений Web-клиента

Библиотека классов

Описание API Web-клиента Docsvision.

- AdvancedCardManager — класс
- ControlTypeDescription — класс
- CommonResponse — класс
- Подробнее описание библиотеки классов API Web-клиента в навигационном меню слева…

AdvancedCardManager — класс

Класс `AdvancedCardManager` предоставляет собственные методы Web-клиента для работы с карточками с использованием API Docsvision.

Пространство имён: `DocsVision.Platform.WebClient.Managers`

Сборка: `DocsVision.Platform.WebClient.dll`

Синтаксис

```
public class AdvancedCardManager
```

Методы

Имя	Описание
`CardReadAccessCheck(Guid)`	Проверяет наличие прав на чтение карточки `cardId`.
`ChangeState(IServiceProvider, Guid, Guid, CardOperationArguments)`	Изменяет состояние карточки `cardId` при применении операции `operationId`. Если при изменении состояния была обработанная ошибка, возвращает её описание.
`CheckCardExists(Guid)`	Проверяет существование карточки `cardId`.
`CheckCardTimestamp(Guid, System.Int64)`	Проверяет актуальность штампа времени `timestamp` для карточки `cardId`.
`CheckFieldNullValueAllowed(Guid, Guid, String)`	Проверяет возможность установки пустого значения в поле `fieldAlias`, секции `sectionId`, типа карточки `cardTypeId`.
`CheckRowTimestamp(Guid, Guid, System.Int64)`	Проверяет актуальность штампа времени `timestamp` для строки `rowId` секции `sectionId`.
`CreateCard(Guid, Guid?)`	Создаёт карточку типа `cardTypeId`, вида `cardKindId`. Возвращает идентификатор новой карточки.

Имя	Описание
`CreateCard(Guid)`	Создаёт карточку по шаблону `templateId`. Возвращает идентификатор новой карточки.
`DeleteCard(Guid, Boolean)`	Удаляет карточку `cardId` и все её дочерние карточки. Для удаления в корзину параметр `permanently` должен быть `false`.
`DeleteRow(Guid, Guid, Guid)`	Удаляет строку `rowId` из секции sectionId карточки `cardId`.
`DisableReadOnlyDictionary(Guid)`	Добавляет справочник cardTypeId в список справочников, для которых запрещен режим открытия "только для чтения". Проверка осуществляется методом `IsReadonlyDictionaryEnabled`.
`GetCard<T>(Guid, Boolean)`	Возвращает карточку `cardId` в формате объекта `T`. При установленном флаге `refresh` обновляет кэш для получения актуальных данных.
`GetCardCreateDate(Guid)`	Возвращает дату создания карточки `cardId`.
`GetCardData(Guid, Boolean)`	Возвращает данные карточки `cardId`. При установленном флаге `refresh` обновляет кэш для получения актуальных данных.
`GetCardTimestamp(Guid, Boolean)`	Возвращает штамп времени карточки `cardId`. При установленном флаге `refresh` обновляет кэш для получения актуальных данных.
`GetCardTimestampModel(Guid, Boolean)`	Возвращает штамп времени карточки `cardId` в формате модели `CommonResponseTimestamp`. При установленном флаге `refresh` обновляет кэш для получения актуальных данных.

Имя	Описание
`GetCardTypeId(Guid)`	Возвращает тип карточки `cardId`.
`GetDictionary<T>(Guid, Boolean)`	Возвращает справочник `cardTypeId` в формате объекта `T`.
`GetDictionaryAsReadonly<T>(Guid, Boolean)`	Возвращает справочник `cardTypeId` в формате объекта `T` и режим чтения. Если данные справочника будут изменяться, получайте его методом `GetDictionary`.
`GetDictionaryData(Guid, Boolean)`	Возвращает данные справочника `cardTypeId`.
`GetDictionaryDataAsReadonly(Guid, Boolean)`	Возвращает данные справочника `cardTypeId` в формате объекта `T` и режим чтения. Если данные справочника будут изменяться, получайте их методом `GetDictionaryData`.
`GetFieldDescription(Guid, Guid, String)`	Возвращает отображаемое локализованное название поля `fieldAlias` секции `sectionId` карточки `cardTypeId`.
`GetFieldValue(Guid, Guid, String)`	Возвращает значение поля `fieldAlias` секции `sectionId` карточки `cardTypeId`.
`GetFieldValue<T>(Guid, Guid, String)`	Возвращает значение поля `fieldAlias` секции `sectionId` карточки `cardTypeId` в формате объекта `T`.
`GetFieldValue(CardData, Guid, String)`	Возвращает значение поля `fieldAlias` секции `sectionId` из данных карточки `cardData`.
`GetOperations(Guid)`	Возвращает список операций (разрешённых и запрещённых) карточки в формате `OperationModel`.
`GetRow(Guid, Guid, Guid)`	Возвращает данные строки rowId секции `sectionId` карточки `cardId`.

Имя	Описание
`GetRowData(Guid, Guid, Guid, Boolean)`	Возвращает данные строки `rowId` секции `sectionId` карточки `cardId`. При установленном флаге `refresh` обновляет кэш для получения актуальных данных.
`GetRowTimestamp(Guid, Guid)`	Возвращает штамп времени для строки `rowId` секции `sectionId`.
`GetSection(Guid, Guid)`	Возвращает данные секции `sectionId` карточки `cardId`.
`IsOperationAllowed(Guid, BuiltInOperation)`	Проверяет возможность выполнения операции `builtInOperation` для карточки `cardId`.
`MarkAsRead(Guid)`	Устанавливает на карточку `cardId` признак "прочитана".
`SetFieldValue(Guid, Guid, String, Object)`	Устанавливает для поля `fieldAlias` секции `sectionId` карточки `cardId` значение `value`.
`SetWasRead(Guid, Boolean)`	Устанавливает на карточку `cardId` признак "прочитана"/"не прочитана" (`setRead`).
`TryGetCardSystemInfo(Guid)`	Возвращает данные системной секции карточки `cardId`.
`UpdateDigest(Guid, String)`	Обновляет дайджест карточки. Для карточек Базовые объекты и производных дайджест формируется в соответствии с настройками формирования дайджеста в Конструкторе разметок или устанавливает значение по умолчанию `defaultDigest`. Для других типов карточек в дайджест записывается название типа карточки.

ControlTypeDescription — класс

Класс `ControlTypeDescription` содержит данные описателя элемента управления.

Пространство имён: `DocsVision.Platform.Tools.LayoutEditor.ObjectModel.Descriptions`

Сборка: `DocsVision.LayoutEditor.ObjectModel.dll`

Синтаксис

```
public class ControlTypeDescription
```

Конструкторы

Имя	Описание
`ControlTypeDescription(String)`	Инициализирует новый экземпляр класса `ControlTypeDescription` с заданным названием типа элемента управления.

Свойства

Имя	Описание
`CheckAsChild`	Задаёт или возвращает функцию, которая проверяет возможность добавления данного элемента управления в другой элемент управления.
`CheckAsParent`	Задаёт или возвращает функцию, которая проверяет возможность добавления одного элемента управления в другой.
`ControlResolverType`	Задаёт или возвращает название типа резолвера для получения модели элемента управления.
`ControlTypeName`	Задаёт или возвращает тип элемента управления.

Имя	Описание
`ControlValidator`	Задаёт или возвращает функцию валидации элемента управления при сохранении разметки.
`DisplayName`	Задаёт или возвращает отображаемое название элемента управления.
`GetAllowedOperations`	Задаёт или возвращает функцию, которая проверяет возможность совершения операции с элементом управления при настройке разметки.
`IsBlockLevel`	Задаёт или возвращает признак того, что элемент управления относится к элементам разметки.
`IsParentConrol`	Задаёт или возвращает признак того, что элемент управления может являться родительским для других элементов управления.
`OnChildAdd`	Задаёт или возвращает функцию, которая выполняется при добавлении дочернего элемента управления.
`OnPropertyChange`	Задаёт или возвращает функцию обработчик изменения настроек элемента управления.
`PropertyDescriptions`	Возвращает коллекцию описателей свойств элемента управления.
`Validator`	Задаёт или возвращает функцию валидации разметки, вызываемую при сохранении разметки.
`Visualizer`	Задаёт или возвращает класс компонента, представляющего графический интерфейс элемента управления в редакторе разметок.

Методы

Имя	Описание
`CreateProperties(String)`	Заполняет свойства экземпляра элемента управления из свойств типа ЭУ при добавлении ЭУ на разметку.

CommonResponse — класс

Класс `CommonResponse` содержит ответ на вызов метода контроллера.

Пространство имён: `DocsVision.Platform.WebClient.Models`

Сборка: `DocsVision.Platform.WebClient.dll`

Синтаксис

```
public class CommonResponse
```

Свойства

Имя	Описание
`Success`	Признак завершения запроса: `true` — запрос завершен, `false` — запрос не выполнен, произошла ошибка.
`Timestamp`	Новый штамп времени, если при выполнении запроса была изменена карточка.
`TimestampCardId`	Идентификатор карточки, если при выполнении запроса была изменена карточка.
`Message`	Содержимое сообщения об ошибке, возникшей при выполнении запроса, или предупреждения — при завершении запроса.
`ExtendedDataSourceInfos`	Информация о дополнительных источниках данных.

Методы

Имя	Описание
`CreateSuccess`	Создаёт экземпляр объекта `CommonResponse` в состоянии Выполнено. В перегрузках метода в `CommonResponse` могут передаваться значения: `timestamp` — штамп времени. `sessionContext` — контекст сессии. `cardId` — идентификатор карточки. `extendedDataSourceInfoModels` — информация о дополнительных источниках данных. `data` — данные, возвращаемые запросом. `sectionId`, `rowId` — идентификаторы сессии и строки.
`CreateError`	Создаёт экземпляр объекта `CommonResponse` в состоянии Ошибка. В message передаётся информация об ошибке.
`InitializeSuccess`	Устанавливает `CommonResponse` в состояние Выполнено. В перегрузке метода также задаётся значение `Timestamp` и `TimestampCardId`.
`InitializeError`	Устанавливает `CommonResponse` в состояние Ошибка.

Пример

Ниже приведён код метода контроллера, в котором выполняется изменение карточки.

```
[HttpPost]
public CommonResponse AddReference([FromBody]AddReferenceRequestModel model)
{
    var currentObjectContextProvider = ServiceUtil.GetService
<ICurrentObjectContextProvider>(this.serviceProvider);
    var sessionContext = currentObjectContextProvider.GetOrCreateCurrentSessionContext();

    var responseModel = new CommonResponse(); ①

    var currentCardTimestamp = sessionContext.AdvancedCardManager.GetCardTimestamp(model
.CardId); ②

    if (currentCardTimestamp == model.CardTimestamp) ③
    {
④

        var commonResponseTimestamp = new CommonResponseTimestamp ⑤
        {
            CardId = model.CardId,
            Timestamp = sessionContext.AdvancedCardManager.GetCardTimestamp(model.CardId)
        };
        responseModel.InitializeSuccess(commonResponseTimestamp);
    }
    else
    {
        responseModel.InitializeError("Сообщение клиенту об ошибке"); ⑥
    }

    return responseModel;
}
```

① Создание модели `CommonResponse`.

② Получение текущего штампа времени.

③ Проверка актуальности данных загруженной карточки.

④ Здесь выполняются действия с карточкой.

⑤ Обновляется штамп времени в ответе.

⑥ Возврат ошибки.

В приведённом коде запрос со стороны клиента включает штамп времени изменяемого объекта (`model.CardTimestamp`), что позволяет убедиться в актуальности клиентских данных.

Если данные на клиенте не актуальны (например, карточка была изменена

другим пользователем) — клиенту возвращается ответ (`CommonResponse`) с состоянием `Ошибка`.

Если данные на клиенте соответствуют данным на сервере — выполняется операция с карточкой. При этом в ответе клиенту отправляется новый штамп времени (`CommonResponseTimestamp`) — на клиентской стороне хранится в поле `cardInfo.timestamp` (см. JSDocAPI).

NotificationRealtimeMessage — класс

Класс NotificationRealtimeMessage содержит данные сообщения, передаваемого с помощью сервиса IRealtimeCommunicationService.

Пространство имён: `DocsVision.Platform.WebClient.Models.RealTimeCommunication.NotificationMessage`

Сборка: `DocsVision.Platform.WebClient.dll`

Синтаксис

```
public class NotificationRealtimeMessage
```

Конструкторы

Имя	Описание
`NotificationRealtimeMessage()`	Инициализирует новый экземпляр класса `NotificationRealtimeMessage`.

Свойства

Имя	Описание
`NotificationType`	Тип передаваемого сообщения.
`Message`	Возвращает менеджер для работы с блокировками.

PropertyCategoryConstants — класс

Класс `PropertyCategoryConstants` предоставляет названия стандартных категорий для списка свойств элемента управления.

Пространство имён: `DocsVision.Platform.Tools.LayoutEditor.Infrostructure`

Сборка: `WebLayoutsDesigner.exe`

Синтаксис

```
public static class PropertyCategoryConstants
```

Поля

Имя	Описание
`AppearanceCategory`	Внешний вид.
`BehaviorCategory`	Поведение.
`DataCategory`	Данные.
`DefaultValueCategory`	Значение по умолчанию.
`EventsCategory`	События.
`GeneralCategory`	Общие.

PropertyDescription — класс

Класс `PropertyDescription` содержит описание свойства элемента управления в расширении программы Конструктор Web-разметок.

Пространство имён: `DocsVision.Platform.Tools.LayoutEditor.ObjectModel.Descriptions`

Сборка: `DocsVision.LayoutEditor.ObjectModel.dll`

Синтаксис

```
public class PropertyDescription : ICloneable
```

Конструкторы

Имя	Описание
`PropertyDescription(String)`	Инициализирует новый экземпляр класса `PropertyDescription`.

Свойства

Имя	Описание
`BindingResolverType`	Зарезервировано.
`Category`	Задаёт или возвращает категорию свойства.
`ChangeLocalizaionKeys`	Задаёт или возвращает функцию обновления ключей локализации при копировании свойства с сериализованным значением.
`CheckAvailablity`	Задаёт или возвращает функцию проверки доступности свойства для настройки в текущем контексте (к примеру, в текущем типе разметки).
`CollectLocalizationKeys`	Задаёт или возвращает функцию получения нужных ключей локализации при копировании свойства с сериализованным значением.
`DefaultValue`	Задаёт или возвращает значение по умолчанию.
`DeviceDependent`	Задаёт или возвращает флаг, указывающий, зависит ли значение свойства от типа клиентского устройства.
`DisplayName`	Задаёт или возвращает строковый ресурс, предоставляющий локализованное название элемента управления.
`Editor`	Задаёт или возвращает тип компонента редактирования (редактора) значения свойства. Редактор является компонентом UserControl с реализованным интерфейсом `Xceed.Wpf.Toolkit.PropertyGrid.Editors.ITypeEditor` (сборка `Xceed.Wpf.Toolkit.Fixed.dll`).

Имя	Описание
`ExcludeFromIL`	Задаёт или возвращает флаг, указывающий, будет ли данное свойство исключено из пакета свойств элемента управления, передаваемых в клиентскую часть Web-клиента. Свойство с установленным флагом `ExcludeFromIL` остается настраиваемым в программе Конструктор Web-разметок.
`GenerateDefaultValue`	Задаёт или возвращает метод, генерирующий для свойства значение по умолчанию.
`GenerateValueOnPaste`	Задаёт или возвращает метод, генерирующий значение свойства при копировании элемента управления, содержащего данное свойство.
`GetBindingLoader`	Задаёт или возвращает функцию, реализующую механизм получения и сохранения значения свойства.
`Hidden`	Задаёт или возвращает флаг, определяющий, является ли свойство скрытым. Скрытое свойство не отображается при настройке элемента управления.
`IsTemporarilyReadOnly`	Задаёт или возвращает функцию, проверяющую является ли элемент управления доступным только на чтение.
`ItemsSource`	Задаёт или возвращает тип компонента, который предоставляет список готовых значений свойства, из которого значение свойства выбирается.

Имя	Описание
`Localized`	Задаёт или возвращает флаг, указывающий, является ли свойство локализуемым. Свойство должно быть строкового типа. Для редактирования значения локализуемого свойства в Editor должен быть указан тип редактора `LocalizedPropertyEditor`.
`Name`	Задаёт или возвращает уникальное название свойства.
`Priority`	Задаёт или возвращает порядок отображения свойства в текущей категории свойств. Чем выше порядок, тем выше будет отображено свойство.
`PropertyResolverType`	Задаёт или возвращает название компонента, предоставляющего дополнительную возможность настройки свойства: `DeviceDependent` (зависящего от типа устройства), `Localized` (зависящего от локализации). Рекомендуется использовать вместо `PropertyResolverType` флаги `Localized` и `DeviceDependent` класса `PropertyDescription`.
`ReadOnly`	Задаёт или возвращает флаг, указывающий, является ли значение элемента управления доступным только на чтение.
`RefreshProperties`	Задаёт или возвращает флаг, указывающий, требуется ли обновлять отображаемое значение свойства при изменении значения. В `true` следует устанавливать, если для изменения свойства используется отдельное окно редактирования.

Имя	Описание
`Type`	Задаёт или возвращает тип значения свойства.
`Validate`	Задаёт или возвращает функцию валидации значения свойства.

Примеры

Следующий код демонстрирует пример свойства `Slider`, добавляемого в категорию *Данные*. Для свойства назначается специализированный редактор значения — `SliderEditor`.

```
private PropertyDescription GetSliderPropertyDescription()
{
 return new PropertyDescription ①
 {
  Type = typeof(string), ②
  Name = Constants.SampleImage.Slider, ③
  Category = PropertyCategoryConstants.DataCategory, ④
  DisplayName = Resources.ControlTypes_SliderProperty, ⑤
  DefaultValue = String.Empty, ⑥
  ExcludeFromIL = true, ⑦
  Editor = typeof(SliderEditor), ⑧

  CollectLocalizationKeys = args => ⑨
  {
   var value = (string)args.ControlInfo.GetProperty(args.PropertyDescription.Name); ⑩
   var items = SliderModeHelper.StringToItems(value); ⑪
   var keys = args.LocalizationKeys; ⑫
   foreach (var item in items) ⑬
   {
    if (!string.IsNullOrEmpty(item.DescriptionLocalizationKey))
     keys.Add(item.DescriptionLocalizationKey);
   }
  },
  ChangeLocalizationKeys = args => ⑭
  {
   var value = (string)args.ControlInfo.GetProperty(args.PropertyDescription.Name);

   var items = SliderModeHelper.StringToItems(value);
   var mapping = args.LocalizationMapping;
   foreach (var item in items)
   {
    if (!string.IsNullOrEmpty(item.DescriptionLocalizationKey))
```

```
    {
     string newKey;
     if (mapping.TryGetValue(item.DescriptionLocalizationKey, out newKey)) ⑮
      item.DescriptionLocalizationKey = newKey;
     }
    }
    return SliderModeHelper.ItemsToString(items); ⑯
   }
  };
 }
```

① Создаем экземпляр класса описания свойства .

② Определяем строковый тип данного свойства ЭУ. Реальное значение сериализуемое.

③ Указывается уникальное название.

④ Категория для размещения свойства Данные.

⑤ Отображаемое название свойства.

⑥ Пустое значение по умолчанию.

⑦ Свойство не должно передаваться на клиентскую сторону Web-клиента.

⑧ Указываем специфический редактор значения свойства.

⑨ Определяем метод, который будет вызываться при копировании свойства. Метод получает ключи локализации DescriptionLocalizationKey из объекта свойства args.PropertyDescription.Name и заполняет ими коллекцию args.LocalizationKeys.

⑩ Получаем значение требуемого свойства.

⑪ Десериализуем значение свойства в нужный класс.

⑫ Получаем из параметров (передается в args) элемента управления текущей разметки.

⑬ Заполняем keys ключами из описания свойства.

⑭ Определяем метод, который будет вызываться при копировании элемента управления и, соответственно, его свойств.

⑮ Создаем связь.

⑯ Сериализуем значение свойства.

SessionContext — класс

Класс `SessionContext` содержит методы и свойства контекста сессии.

Пространство имён: DocsVision.Platform.WebClient

Сборка: DocsVision.Platform.WebClient.dll

Синтаксис

```
public sealed class SessionContext : IServiceProvider
```

Конструкторы

Имя	Описание
SessionContext(IRequestContextProvider)	Инициализирует новый экземпляр класса SessionContext.

Свойства

Имя	Описание
AdvancedCardManager	Возвращает менеджер для работы с карточками.
AdvancedLockManager	Возвращает менеджер для работы с блокировками.
BaseAlias	Возвращает псевдоним базы данных, к которой выполнено подключение.
FullTextSearchEnabled	Возвращает флаг, указывающий, что включено полнотекстовое индексирование.
ObjectContext	Возвращает контекст объектов.
Session	Возвращает пользовательскую сессию.
UserInfo	Возвращает информацию о текущем пользователе.

Методы

Имя	Описание
RegisterObjectContextCreateHandler (Action<ObjectContext>)	Добавляет дополнительный обработчик для события инициализации контекста объектов.

Имя	Описание
`Close`	Освобождает объекты контекста объектов и пользовательской сессии.
`DisposeObjectContext`	Освобождает объект контекста объектов.
`GetService(Type)`	Зарезервирован.
`ResetRolesCache(ObjectBase)`	Вызывает метод `ResetRolesCache` системы Docsvision для сброса кэша ролевой модели.
`RollbackChanges`	Отменяет несохранённые изменения контекста объектов.

UserInfo — класс

Класс `UserInfo` содержит данные пользователя.

Пространство имён: `DocsVision.Platform.WebClient`

Сборка: `DocsVision.Platform.WebClient.dll`

Синтаксис

```
public class UserInfo
```

Поля

Имя	Описание
`AccountName`	Возвращает имя учетной записи пользователя.
`AccountSID`	Возвращает идентификатор безопасности пользователя.
`Employee`	Возвращает модель данных сотрудника.
`EmployeeId`	Возвращает идентификатор сотрудника.
`Roles`	Возвращает роли сотрудника.

Имя	Описание
`UserProfileCardId`	Возвращает идентификатор карточки настроек пользователя.

WebClientExtension — класс

Класс `WebClientExtension` является базовым классом расширения программы Конструктор Web-разметок.

Пространство имён: `DocsVision.WebClient.Extensibility`

Сборка: `DocsVision.WebClient.Extensibility.dll`

Синтаксис

```
public abstract class WebClientExtension
```

Конструкторы

Имя	Описание
`WebClientExtension(IServiceProvider)`	Инициализирует новый экземпляр класса WebClientExtension.

Свойства

Имя	Описание
`BindingConverters`	Возвращает коллекцию зарегистрированных в расширении конвертеров значений элементов управления. **Свойство устарело.**
`BindingResolvers`	Возвращает коллекцию зарегистрированных в расширении "резолверов" (`resolvers`) значений элементов управления и выполнения обратной процедуры. **Свойство устарело.**

Имя	Описание
`CardFactories`	Возвращает коллекцию зарегистрированных в расширении фабрик карточек. **Свойство устарело.**
`ControlResolvers`	Возвращает коллекцию зарегистрированных в расширении "резолверов" (`resolvers`) элементов управления. **Свойство устарело.**
`ExtensionName`	Возвращает название расширения.
`ExtensionVersion`	Возвращает версию расширения.
`LayoutExtensionResourceManagers`	Возвращает зарегистрированные в расширении диспетчеры ресурсов, ресурсы которых могут быть получены клиентскими компонентами Web-клиента.
`Namespace`	Возвращает пространство имён расширения. **Свойство устарело.**
`NavigatorExtension`	Возвращает информацию о данном серверном расширении. **Свойство устарело. Используйте класс `WebClientExtension`, свойства `ExtensionName` и `ExtensionVersion`.**
`PropertyResolvers`	Возвращает коллекцию зарегистрированных в расширении "резолверов" (`resolvers`) свойств элементов управления. **Свойство устарело.**

Имя	Описание
`RegisteredResourceManagers`	Возвращает зарегистрированные в расширении диспетчеры ресурсов, ресурсы которых могут быть получены в расширении Web-клиента.
`ServiceActivators`	Возвращает активаторы для реализованных в расширении сервисов. **Свойство устарело.**
`ServiceProvider`	Возвращает экземпляр сервис-провайдера. **Свойство устарело. Используйте метод с перезагрузкой `ILifetimeScope`.**

Методы

Имя	Описание
`OnLoad(IServiceProvider)`	Обработчик загрузки серверного расширения. Вызывается после загрузки расширения. Метод помечен устаревшим, работает. **Используйте метод с перезагрузкой `ILifetimeScope`.**
`OnLoad(ILifetimeScope lifetimeScope)`	Обработчик загрузки серверного расширения. Вызывается после загрузки расширения. Описание типа аргумента метода на странице https://autofac.org/apidoc/html/BA5D3489.htm. Метод устарел, не работает и не требуется, регистрация осуществляется в `InitializeServiceCollection`.

Имя	Описание
`OnObjectContextCreate(ObjectContext)`	Обработчик для события инициализации контекста объектов. Переопределение метода можно использовать для добавления новых фабрик библиотек карточек и фабрик сервисов в используемый Web-клиентом контекст объектов.
`GetApiControllerActivators(IServiceProvider)`	Предоставляет активаторы для реализованных в расширении контроллеров WebApi. Все контроллеры сборки регистрируются автоматически при загрузке расширения.
`GetBindingConverters`	Предоставляет методы получения значений элементов управления для передачи в их веб-компоненты. Через переопределение метода можно добавить функцию изменения значения элемента управления до его передачи клиенту (в браузер). **Устарел. Используйте IoC-контейнер.**

Имя	Описание
`GetBindingResolvers`	Предоставляет методы разрешения значений элементов управления и выполнения обратной процедуры. Через переопределение метода можно добавить функцию изменения значения элемента управления до его передачи клиенту (при получении), а также изменения значения перед его передачей серверу Web-клиента (при сохранении). **Устарел. Используйте IoC-контейнер.**
`GetCardFactories`	Предоставляет фабрики карточек, которые реализуют методы создания новых экземпляров карточек заданных типов. **Функция `GetCardFactories` устарела**: используйте `GetCardLifeCycles`.
`GetCardLifeCycles`	Предоставляет методы создания карточек, проверки данных карточки перед сохранением. **Устарел. Используйте IoC-контейнер.**
`GetControllerActivators(IServiceProvider)`	Предоставляет активаторы для реализованных в расширении контроллеров MVC. Все контроллеры сборки регистрируются автоматически при загрузке расширения.

Имя	Описание
`GetControlResolvers`	Предоставляет методы формирования моделей данных для веб-компонентов элементов управления. Через переопределение метода можно внести изменения в стандартный процесс загрузки свойств и значений свойств элемента управления. **Устарел. Используйте IoC-контейнер.**
`GetLayoutExtensionResourceManagers`	Предоставляет диспетчеры ресурсов, ресурсы которых могут быть получены клиентскими компонентами (клиентскими скриптами и веб-компонентами элементов управления).
`GetNavigatorExtension`	Предоставляет информацию о данном расширении. Используйте класс `WebClientExtension`, свойства `ExtensionName` и `ExtensionVersion`.
`GetPropertyResolvers`	Предоставляет методы получения значений свойств элементов управления. Через переопределение метода можно внести изменения в стандартный процесс загрузки значений свойств. **Устарел. Используйте IoC-контейнер.**
`GetRegisteredResourceManagers`	Предоставляет диспетчеры ресурсов, ресурсы которых могут быть получены в расширении Web-клиента.

Имя	Описание
`GetRowLifeCycles`	Предоставляет методы создания строк секций, проверки данных строки перед сохранением. **Устарел. Используйте IoC-контейнер.**
`GetServiceActivators(IServiceProvider)`	Предоставляет активаторы для реализованных в расширении сервисов. **Устарел. Используйте IoC-контейнер.**

WebLayoutsDesignerExtension — класс

Класс `WebLayoutsDesignerExtension` является базовым классом расширения программы Конструктор Web-разметок.

Пространство имён: `DocsVision.Platform.Tools.LayoutEditor.Extensibility`

Сборка: `WebLayoutsDesigner.exe`

Синтаксис

```
public abstract class WebLayoutsDesignerExtension : IWebLayoutsDesignerExtension
```

Конструкторы

Имя	Описание
`WebLayoutsDesignerExtension(IServiceProvider)`	Инициализирует новый экземпляр класса `WebLayoutsDesignerExtension`.

Свойства

Имя	Описание
AllowedOperations	Возвращает коллекцию функций, определяющих возможность совершения в программе Конструктор Web-разметок определённой операции с элементом управления.
ControlTypeDescriptions	Возвращает коллекцию описаний элементов управления, добавляемых в программы Конструктор Web-разметок данным расширением.
Editors	Возвращает коллекцию редакторов значений свойств элементов управления.
PropertyDescriptions	Возвращает коллекцию свойств и событий, добавляемых данным расширением.
ResourceManagers	Возвращает коллекцию ссылок на файлы ресурсов, предоставляемых данным расширением.
Visualizers	Возвращает коллекцию компонентов, содержащих графическое представление для отображения ЭУ в программе Конструктор Web-разметок.
PropertyFactory	Возвращает ссылку на фабрику свойств и событий элемента управления.
ServiceContainer	Возвращает ссылку на контейнер сервисов.

Методы

Имя	Описание
GetAllowedOperations	Метод должен возвращать коллекцию функций, определяющих возможность совершения в программе Конструктор Web-разметок определённой операции с элементом управления.

Имя	Описание
`GetControlTypeDescriptions`	Метод должен возвращать коллекцию описаний элементов управления, добавляемых в программы Конструктор Web-разметок данным расширением.
`GetEditors`	Метод должен возвращать коллекцию редакторов значений свойств элементов управления.
`GetPropertyDescriptions`	Метод должен возвращать коллекцию свойств и событий, добавляемых данным расширением.
`GetResourceManagers`	Метод должен возвращать коллекцию ссылок на файлы ресурсов, предоставляемых данным расширением.
`GetVisualizers`	Метод должен возвращать коллекцию компонентов, содержащих графическое представление для отображения ЭУ в программе Конструктор Web-разметок.

IApplicationTimestampService — интерфейс

Интерфейс `IApplicationTimestampService` определяет методы сервиса для работы со штампами времени приложения.

Пространство имён: `DocsVision.WebClient.Services`

Сборка: `DocsVision.BackOffice.WebClient.dll`

Синтаксис

```
public interface IApplicationTimestampService
```

Свойства

Имя	Описание
`ApplicationTimestamp`	Возвращает текущий штамп времени.

Методы

Имя	Описание
UpdateApplicationTimeStamp	Обновляет штамп времени, вызывающий обновление пользовательского кэша приложения.

Примечание

Реализация метода UpdateApplicationTimeStamp используется при вызове веб-метода GET http://Адрес-сервера-Web-клиента/WebClient/Navigator/ResetClientCache. Если требуется, стандартная реализация сервиса может быть переопределена собственной.

Стандартная реализация

```
public class ApplicationTimestampService : IApplicationTimestampService
{
    private string applicationTimestamp;

    public string ApplicationTimestamp =>
        applicationTimestamp ?? (applicationTimestamp = WebClientApplication
.ApplicationStartTimestamp);

    public void UpdateApplicationTimeStamp()
    {
        applicationTimestamp = DateTime.Now.Ticks.ToString();
    }
}
```

ICardLifeCycleEx — интерфейс

Интерфейс ICardLifeCycleEx определяет методы для работы с карточками из разметок.

Пространство имён: DocsVision.WebClientLibrary.ObjectModel.Services.EntityLifeCycle

Сборка: DocsVision.WebClientLibrary.ObjectModel.dll

Синтаксис

```
public interface ICardLifeCycleEx
```

Методы

Имя	Описание
`CardTypeId { get }`	Идентификатор целевой карточки для текущего жизненного цикла. **Тип:** `Guid`.
`Create(SessionContext, CardCreateLifeCycleOptions)`	Создаёт новую карточку и возвращает её идентификатор. **Тип:** `Guid`. **Параметры:** `sessionContext` — контекст сессии; `options` — параметр формируется самостоятельно. Обычно `OperationContext` является экземпляром `CreateRowOperationContext`.
`Validate(SessionContext, CardValidateLifeCycleOptions, out List<ValidationResult>)`	Проверяет данные карточки перед сохранением изменений. Возвращает `true`, если ошибки не обнаружены, иначе — `false`. **Тип:** `Boolean`. **Параметры:** `sessionContext` — контекст сессии; `options` — параметр формируется самостоятельно; `validationResults` — массив для формирования сообщения о результатах проверки.
`OnSave(SessionContext, CardSaveLifeCycleOptions)`	Обрабатывает событие сохранения карточки. **Параметры:** `sessionContext` — контекст сессии; `cardId` — идентификатор карточки.

Имя	Описание
`CanDelete(SessionContext, CardDeleteLifeCycleOptions, out String)`	Проверяет возможность удаления карточки. Возвращает `true`, если удаление возможно, иначе — `false`. **Тип:** `Boolean`. **Параметры:** `sessionContext` — контекст сессии; `options` — параметр формируется самостоятельно; `message` — сообщение о невозможности удаления.
`OnDelete(SessionContext, CardDeleteLifeCycleOptions)`	Обрабатывает событие удаления карточки. **Параметры:** `sessionContext` — контекст сессии; `options` — параметр формируется самостоятельно.
`GetDigest(SessionContext, CardDigestLifeCycleOptions);`	Метод генерации дайджеста. **Тип:** `String`. **Параметры:** `sessionContext` — контекст сессии; `options` — параметр формируется самостоятельно.

ICardsPresentationExtension — интерфейс

Интерфейс `ICardsPresentationExtension` представляет метод получения описания представления карточки.

Пространство имён: `DocsVision.WebClientLibrary.ObjectModel.Services.EntityLifeCycle`

Сборка: `DocsVision.WebClientLibrary.ObjectModel.dll`

Синтаксис

```
public interface ICardsPresentationExtension
```

Методы

Имя	Описание
`ExtensionName { get; set; }`	Идентификатор целевой карточки для текущего жизненного цикла. **Тип:** `String`.
`GetPresentation(SessionContext, CardsPresentationRequest);`	Получает представление карточки. **Тип:** `CardPresentation`. **Параметры:** `sessionContext` — контекст сессии; `request` — возвращает представление.

IImageGenerator — интерфейс

Для данной версии системы интерфейс недействителен.

ILinksService — интерфейс

Интерфейс `ILinksService` определяет методы сервиса для работы со ссылками карточек — сервиса ссылок.

Пространство имён: `DocsVision.BackOffice.WebClient.Links`

Сборка: `DocsVision.BackOffice.WebClient.dll`

Синтаксис

```
public interface ILinksService
```

Методы

Имя	Описание
`AddReference(SessionContext, Guid, Guid, Guid, Boolean)`	Добавляет ссылку на карточку `referencedCardId` с типом ссылки `linkTypeId` в список ссылок `referenceListCardId`. Если `hardLink` в значении `true`, будет добавлена жесткая ссылка.
`AddUrlReference(SessionContext, Guid, String, Guid)`	Добавляет ссылку на адрес `url` в список ссылок `referenceListCardId`. В `linkTypeId` может быть указан тип ссылки.
`GetAllowedCreateLinkKinds(SessionContext)`	Возвращает модели видов карточек, на которые можно создать ссылки.
`GetAllowedLinkCardTypes`	Возвращает идентификаторы типов карточек, на которые можно создать ссылки.
`GetLinkModels(SessionContext, Guid, Guid?, LinksDescriptionColumnViewMode, String)`	Возвращает модели ссылок для списка ссылок `referenceListCardId`. В параметре `descriptionColumnViewMode` указывается способ формирования описания ссылок, а в `descriptionColumnMethodKey` название метода формирования описания ссылок.
`GetLinkTypes(SessionContext)`	Возвращает модели типов ссылок.
`GetOrCreateReferenceListCardId(SessionContext , Guid, Guid, String)`	Возвращает или создаёт список ссылок карточки `cardId`. Идентификатор карточки Список ссылок может быть получен из поля `fieldAlias` секции `sectionId`.
`GetReferenceListCardId(SessionContext, Guid, Guid, String)`	Возвращает идентификатор карточки Список ссылок карточки `cardId`. Идентификатор карточки Список ссылок может быть получен из поля `fieldAlias` секции `sectionId`.

Имя	Описание
`RegisterDescriptionColumnGenerator(String, DescriptionColumnGeneratorDelegate)`	Регистрирует метод генерации описаний ссылок. В `key` должно быть указано название метода, по которому он может быть запрошен элементом Ссылки, в `delegateFunction` — делегат на метод генерации описания ссылки.
`RemoveReference(SessionContext, Guid, Guid, Guid)`	Удаляет ссылку `referenceId` из списка ссылок `referenceListCardId` карточки `cardId`. Возвращает true, если ссылка была удалена, иначе — `false`.
`UpdateReferenceDescription(SessionContext, Guid, String)`	Изменяет описание ссылки `referenceId` на `description`.

IRealtimeCommunicationService — интерфейс

Интерфейс `IRealtimeCommunicationService` определяет методы для работы с подсистемой оповещений Web-клиента.

Пространство имён: `DocsVision.Platform.WebClient.Services`

Сборка: `DocsVision.Platform.WebClient.dll`

Синтаксис

```
public interface IRealtimeCommunicationService
```

Методы

Имя	Описание
`GetCurrentSessionId`	Возвращает идентификатор текущей сессии в ASP NET. **Тип:** `String`.
`Send<T>(String, RealtimeCommunicationMessage<T>)`	Отправляет сообщение пользователю с идентификатором сессии sessionId в ASP NET.

Имя	Описание
`Send<T>(Guid, RealtimeCommunicationMessage<T>)`	Отправляет сообщение пользователю с идентификатором employeeId в *Справочнике сотрудников*.
`SendAll<T>(RealtimeCommunicationMessage<T>)`	Отправляет оповещение `message` всем подключенным пользователям (с активной и неактивной сессией) Web-клиента.
`SendToCurrentSession<T>(RealtimeCommunicationMessage<T>)`	Отправляет оповещение `message` текущему пользователю.

События

Имя	Описание
`Connected`	Вызывается при подключении пользователя. **Тип:** `EventHandler<RealtimeConnectionEventArgs>`.
`Disconnected`	Вызывается при отключении пользователя. **Тип:** `EventHandler<RealtimeConnectionEventArgs>`.
`MessageReceived`	Вызывается при получении оповещения. **Тип:** `EventHandler<RealtimeMessageEventArgs>`.
`MessageSent`	Вызывается при отправке оповещения. **Тип:** `EventHandler<RealtimeCommunicationMessage>`.

Примечание

Подсистема оповещений основана на библиотеке функций реального времени SignalR.

Отправленное сообщение будет показано в правом нижнем углу. Цветовое оформление сообщения зависит от типа сообщения, указанного в `RealtimeCommunicationMessage`.

В Web-клиент реализован единственный тип сообщения, передаваемого с помощью SignalR — `NotificationRealtimeMessage`.

На клиентской стороне аналогичную сервису `IRealtimeCommunicationService` функциональность предоставляет клиентский сервис `$RealtimeCommunicationService`.

Примеры

Следующий код демонстрирует пример реализации в серверном расширении функции отправки сообщений всем пользователям.

```
public void SendAll()
{
    var commMessage = new RealtimeCommunicationMessage<NotificationRealtimeMessage>(); ①
    var messageData = new NotificationRealtimeMessage()
    {
        NotificationType = NotificationType.Info,
        Message = message
    };
    commMessage.Initialize(NotificationRealtimeMessage.MessageType, messageData);
    communicationService.SendAll(commMessage);
}
```

① Создаём объект-оповещение. Тип содержимого оповещения — `NotificationRealtimeMessage` (текстовое оповещение).

Возможные типы оповещений перечислены в описании перечисления NotificationType.

IPropertyFactory — класс

Интерфейс `IPropertyFactory` определяет методы фабрики свойств.

Пространство имён: `DocsVision.Platform.Tools.LayoutEditor.Infrostructure`

Сборка: `WebLayoutsDesigner.exe`

Синтаксис

```
public interface IPropertyFactory
```

Методы

Имя	Описание
`CanCreate(String)`	Проверяет возможность получения описателя для свойства с указанным названием. Возвращает `true`, если свойство может быть зарегистрировано, иначе — `false`. **Тип:** `Boolean`. **Параметры:** `propertyName` — название свойства.
`Create(String)`	Возвращает экземпляр описателя свойства с указанным названием. Свойство должно быть зарегистрировано в фабрике свойств. **Тип:** `PropertyDescription`. **Параметры:** propertyName — название свойства.

Имя	Описание
`CreateEvent(String, String, String, String)`	Возвращает экземпляр описателя свойства с указанным названием. Предназначен для получения описателя свойства-события. В отличие от метода `Create`, `CreateEvent` не получает свойство из зарегистрированных в фабрике свойств, а создаёт новый экземпляр описателя свойства. В возвращаемом описателе свойства установлены категория свойства (в "События") и тип значения (в `String`), свойственные свойству-событию элемента управления. **Тип:** `PropertyDescription`. **Параметры:** `propertyName` — название свойства; `resourceKey` — ключ ресурса, предоставляющего локализованное отображаемое название элемента управления; `displayName` — отображаемое название элемента управления — используется, если не указан `resourceKey`; `defaultValue` — значение по умолчанию для свойства.
`Register(String, PropertyDescription)`	Регистрирует новое свойство в фабрике свойств. **Параметры:** `propertyName` — название свойства; `propertyDescription` — описатель регистрируемого свойства.

IRowLifeCycleEx — интерфейс

Интерфейс `IRowLifeCycleEx` определяет методы для работы со строками секций из

разметок.

Пространство имён: `DocsVision.WebClientLibrary.ObjectModel.Services.EntityLifeCycle`

Сборка: `DocsVision.WebClientLibrary.ObjectModel.dll`

Синтаксис

```
public interface IRowLifeCycleEx
```

Методы

Имя	Описание
`SectionId { get }`	Идентификатор целевой секции для текущего жизненного цикла. **Тип:** `Guid`.
`Create(SessionContext, RowCreateLifeCycleOptions)`	Создаёт новую строку в секции и возвращает её идентификатор. **Тип:** `Guid`. **Параметры:** `sessionContext` — контекст сессии; `options` — параметр формируется самостоятельно. Обычно `OperationContext` является экземпляром `CreateRowOperationContext`.

Имя	Описание
`Validate(SessionContext, RowValidateLifeCycleOptions, out List<ValidationResult>)`	Проверяет данные строки перед сохранением изменений. Возвращает `true`, если ошибки не обнаружены, иначе — `false`. **Тип:** `Boolean`. **Параметры:** `sessionContext` — контекст сессии; `options` — параметр формируется самостоятельно; `validationResults` — массив для формирования сообщения о результатах проверки.
`OnSave(SessionContext, RowValidateLifeCycleOptions)`	Обрабатывает событие сохранения строки секции. **Параметры:** `sessionContext` — контекст сессии; `options` — параметр формируется самостоятельно.
`CanDelete(SessionContext, RowDeleteLifeCycleOptions, out string)`	Проверяет возможность удаления строки из секции. Возвращает `true`, если удаление возможно, иначе — `false`. **Тип:** `Boolean`. **Параметры:** `sessionContext` — контекст сессии; `options` — параметр формируется самостоятельно; `message` — сообщение о невозможности удаления.

Имя	Описание
`OnDelete(SessionContext, RowDeleteLifeCycleOptions)`	Обрабатывает событие удаления строки из секции. **Параметры:** `sessionContext` — контекст сессии; `options` — параметр формируется самостоятельно.

ISelectedLayoutService — интерфейс

Интерфейс `ISelectedLayoutService` определяет методы для работы с разметками.

Пространство имён: `DocsVision.Platform.Tools.LayoutEditor.Infrostructure`

Сборка: `WebLayoutsDesigner.exe`

Синтаксис

```
public interface ISelectedLayoutService
```

Методы

Имя	Описание
`SelectedLibraryId`	Возвращает идентификатор библиотеки карточек. **Тип:** `Guid`.
`SelectedCardTypeId`	Возвращает идентификатор типа карточек. **Тип:** `Guid`.
`SelectedKindId`	Возвращает идентификатор вида карточек. **Тип:** `Guid`.
`LayoutInfo`	Возвращает информацию о разметке. **Тип:** `ILayoutInfo`.

AllowedOperationsFlag — перечисление

Перечисление `AllowedOperationsFlag` определяет операции, которые могут быть выполнены с элементом управления при настройке разметки.

Пространство имён: `DocsVision.Platform.Tools.LayoutEditor.ObjectModel.Descriptions`

Сборка: `DocsVision.LayoutEditor.ObjectModel.dll`

Синтаксис

```
[Flags]
public enum AllowedOperationsFlag
```

Члены

Имя	Описание
`None`	Все операции запрещены. **Значение:** `0x00`.
`Create`	Элемент управления может быть добавлен. **Значение:** `0x01`.
`Delete`	Элемент управления может быть удалён. **Значение:** `0x02`.
`ChangeOrder`	Элемент управления может быть перемещен в пределах своего родительского элемента. **Значение:** `0x04`.
`Move`	Элемент управления может быть перемещен к другому родителю. **Значение:** `0x08`.

Имя	Описание
`AddChildControl`	В элемент управления может быть добавлен дочерний элемент управления. **Значение:** `0x10`.
`All`	Разрешены все операции с элементом управления. **Значение:** `0x1F`.

NotificationType — перечисление

Перечисление `NotificationType` определяет типы оповещений, которые могут быть отправлены с помощью сервиса IRealtimeCommunicationService.

Пространство имён: `DocsVision.Platform.WebClient.Models.RealTimeCommunication.NotificationMessage`

Сборка: `DocsVision.Platform.WebClient.dll`

Синтаксис

```
public enum NotificationType
```

Члены

Имя	Описание
`Alert`	Предупреждение. **Значение:** `0`.
`Success`	Успех. **Значение:** `1`.
`Warning`	Внимание. **Значение:** `2`.

Имя	Описание
`Error`	Ошибка. **Значение:** `3`.
`Info`	Информационное сообщение. **Значение:** `4`.

Информация

Тип оповещения влияет на цветовое оформление блока, в котором отображается оповещение.

DescriptionColumnGeneratorDelegate — делегат

Делегат `DescriptionColumnGeneratorDelegate` устарел, вместо него используйте `ICardsPresentationExtension`.

[1] Рекомендуется использовать Visual Studio Code 2022.

[2] Рекомендуется использовать Visual Studio Code 2022.

[3] Рекомендуется использовать Visual Studio Code 2022.

[4] Рекомендуется использовать Visual Studio Code 2022.

[5] Рекомендуется использовать Visual Studio Code 2022.

[6] Рекомендуется использовать Visual Studio Code 2022.

[7] Рекомендуется использовать Visual Studio Code 2022.

[8] Рекомендуется использовать Visual Studio Code 2022.

[9] Рекомендуется использовать Visual Studio Code 2022.

[10] См. `PropertyFactory.GetNameProperty()`

[11] См. свойство `Slider, ImageHeight)`

[12] См. `PropertyFactory.GetNameProperty()`

[13] См. свойство `Url`

[14] Перечень изменений в реализации исходного кода стандартных компонентов не предоставляется.

[15] Репозиторий Docsvision с примерами пакетов Debian расположен на GitHub.

[16] См. `PropertyFactory.GetNameProperty()`

[17] См. свойство `WidgetId`

Руководство разработчика Модуля интеграции с операторами ЭДО

Разработка компонентов модуля Модуля интеграции с операторами ЭДО

Модуль предоставляет возможность разрабатывать собственные компоненты, представленные на странице *Компоненты* в *Справочнике настроек операторов ЮЗДО*.

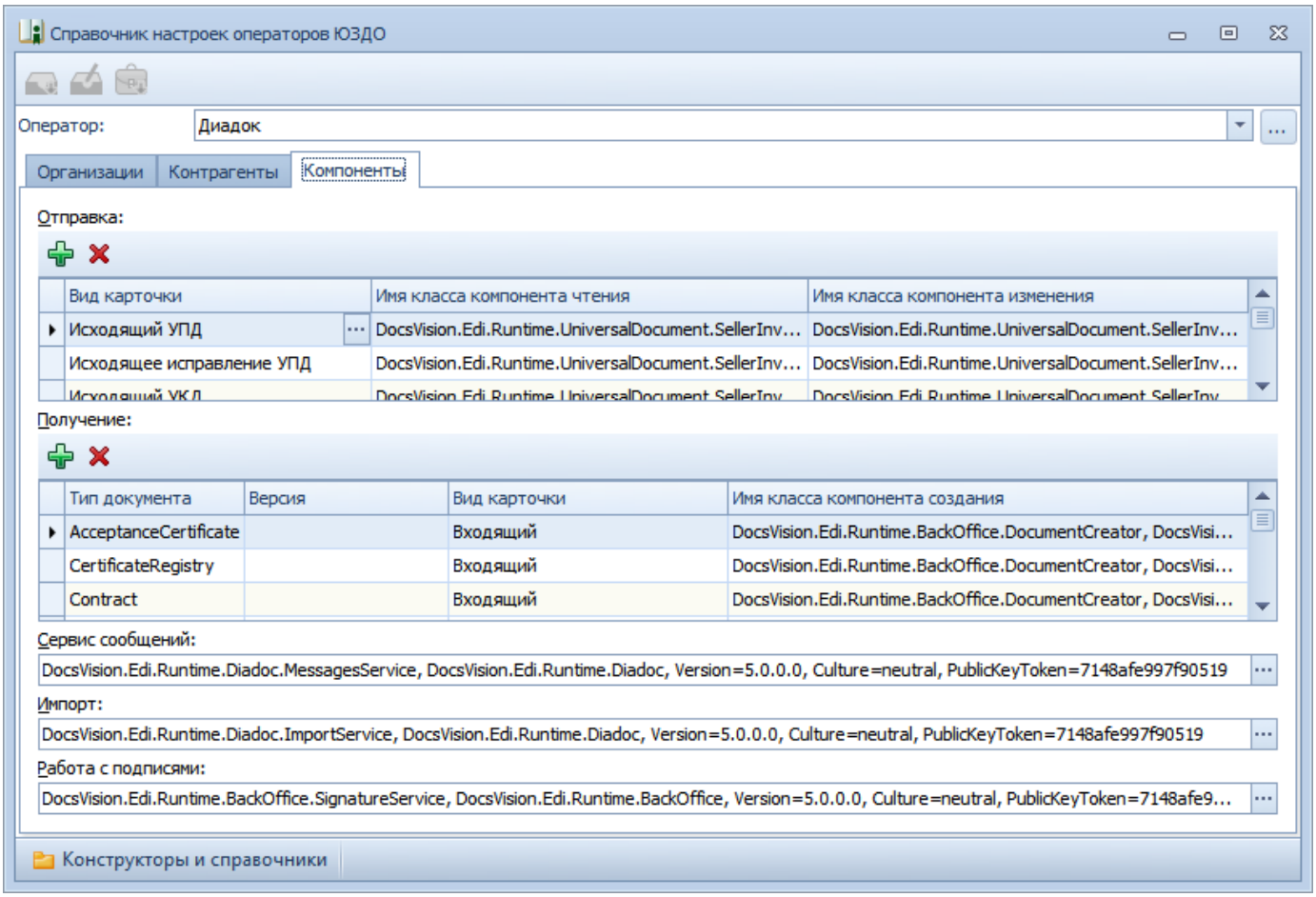


Рисунок 121. Настройки компонентов в Справочнике настроек операторов ЮЗДО

Вынесенная в компоненты функциональность модуля позволяет абстрагировать механизм отправки электронных документов от видов карточек и механизмов взаимодействия с операторами ЭДО.

Разрабатываемые компоненты делятся на две группы:

- Компоненты для работы с карточками Docsvision: компоненты чтения и изменения (данных карточек), а также компонент создания документов.

- Компоненты коннектора к оператору ЭДО: компоненты сервиса сообщений, импорта и работы с электронными подписями.

Разработка компонента чтения данных карточки

Компонент чтения данных предоставляет модулю не зависящий от типа карточки слой получения данных из карточки, необходимых для формирования *Карточки обмена сообщениями*, а также вспомогательные методы, связанные с получением данных из карточек определенного типа.

Компонент должен реализовывать программный интерфейс `IDocumentDataReader`.

Рекомендуется создавать свой компонент на основе абстрактного класса `BaseDataReader` (сборка `DocsVision.Edi.Runtime.UniversalDocument.dll`). Далее приведена часть класса `BaseDataReader` с описанием.

```
public abstract class BaseDataReaders : IDocumentDataReader
{

    public void Initialize(ObjectContext objectContext) ①
    {
        ObjectContext = objectContext;

        UserSession = ObjectContext.GetService<UserSession>();

        documentService = ObjectContext.GetService<IDocumentService>();
    }

    public string GetDigest(Guid cardId) ②
    {
        Document document = GetDocument(cardId);
        if (document == null)
        {
            throw Error.InvalidOperation(Resources.DocumentNotExists, cardId);
        }

        return document.Description;
    }

    public string GetFileDigest(Guid cardId, Guid fileId) ③
    {
        Document document = GetDocument(cardId);
        if (document == null)
        {
            throw Error.InvalidOperation(Resources.DocumentNotExists, cardId);
```

```
        }

        DocumentFile documentFile = document.Files.FirstOrDefault(item => item
.FileVersionRowId == fileId);
        if (documentFile == null)
        {
            throw Error.InvalidOperation(Resources.FileNotExists, fileId, cardId);
        }

        return GetUniversalDocumentDescription(documentFile.FileVersionRowId,
MessageFileType.File) ?? documentFile.FileName;
    }

    public string GetFileContent(Guid cardId, Guid fileId) ④
    {
        Document document = GetDocument(cardId);
        if (document == null)
        {
            throw Error.InvalidOperation(Resources.DocumentNotExists, cardId);
        }

        DocumentFile documentFile = document.Files.FirstOrDefault(item => item
.FileVersionRowId == fileId);
        if (documentFile == null)
        {
            throw Error.InvalidOperation(Resources.FileNotExists, fileId, cardId);
        }

        if (!UserSession.FileManager.FileExists(documentFile.FileVersionRowId))
        {
            return null;
        }

        FileData fileData = UserSession.FileManager.GetFile(documentFile.
FileVersionRowId);
        using (Stream stream = fileData.OpenReadStream())
        {
            using (TextReader textReader = new StreamReader(stream))
            {
                return textReader.ReadToEnd();
            }
        }
    }

    public string GetPrintFormContent(Guid cardId) ⑤
    {
        Document document = GetDocument(cardId);
        if (document == null)
```

```
        {
            throw Error.InvalidOperation(Resources.DocumentNotExists, cardId);
        }

        return GetUniversalDocumentPrintContent(document);
    }

    public string GetReplyFileDigest(Guid cardId, Guid fileId, MessageFileType fileType)
⑥
    {
        return GetUniversalDocumentDescription(fileId, fileType);
    }

    public string GetReplyFileComment(Guid cardId, Guid fileId, MessageFileType fileType)
    {
        if (!UserSession.FileManager.FileExists(fileId))
        {
            return null;
        }

        try
        {
            FileData fileData = UserSession.FileManager.GetFile(fileId);
            using (Stream stream = fileData.OpenReadStream())
            {
                switch (fileType)
                {
                    case MessageFileType.InvoiceCorrectionRequest:
                    case MessageFileType.SignatureRejection:
                        return InvoiceCorrectionNotice.GetComment(stream);
                }
            }
        }
        catch
        {
        }

        return null;
    }

    public ItemCollection<DocumentRecipientData> GetAllRecipients(Guid cardId) ⑦
    {
        Document document = GetDocument(cardId);
        if (document == null)
        {
            throw Error.InvalidOperation(Resources.DocumentNotExists, cardId);
        }
```

```
        ItemCollection<DocumentRecipientData> recipients = new ItemCollection
<DocumentRecipientData>();

        var contractSection = document.GetSection(CardDocument.Contract.ID);
        if (contractSection.Count == 0)
        {
            return recipients;
        }

        BaseCardSectionRow contractRow = (BaseCardSectionRow)contractSection[0];
        Guid companyId = contractRow.GetGuid(CardDocument.Contract.PartnerCompany);
        if (companyId == Guid.Empty)
        {
            return recipients;
        }

        PartnersCompany company = ObjectContext.GetObject<PartnersCompany>(companyId);
        if (company != null)
        {
            recipients.Add(new DocumentRecipientData(ObjectContext.GetObjectRef(company
).Id, company.Name));
        }

        return recipients;
    }

    public ItemCollection<DocumentSignature> GetActualSignatures(Guid cardId) ⑧
    {
        return null;
    }

    public ItemCollection<DocumentFileData> GetAllFiles(Guid cardId) ⑨
    {
        Document document = GetDocument(cardId);
        if (document == null)
        {
            throw Error.InvalidOperation(Resources.DocumentNotExists, cardId);
        }

        return new ItemCollection<DocumentFileData>(document.Files.Where(CheckFile)
            .Select(item => new DocumentFileData(item.FileVersionRowId, item.FileName,
item.FileType == DocumentFileType.Main)));
    }

    public ItemCollection<DocumentFileData> GetLastSignedFiles(Guid cardId) ⑩
    {
        return GetSignedFiles(cardId, Guid.Empty);
    }
```

```
    public ItemCollection<DocumentFileData> GetSignedFiles(Guid cardId, Guid
signatureGroupId) ⑪
    {
        Document document = GetDocument(cardId);
        if (document == null)
        {
            throw Error.InvalidOperation(Resources.DocumentNotExists, cardId);
        }

        if (document.MainInfo.SignatureList == null || !document.MainInfo.SignatureList
.Signatures.Any())
        {
            return null;
        }

        BaseCardSignature lastSignature = signatureGroupId != Guid.Empty
            ? document.MainInfo.SignatureList.Signatures.FirstOrDefault(item =>
ObjectContext.GetObjectRef(item).Id == signatureGroupId)
            : document.MainInfo.SignatureList.Signatures.OrderByDescending(item => item
.TimeStamp).FirstOrDefault();

        if (lastSignature == null)
        {
            return null;
        }

        ItemCollection<DocumentFileData> result = new ItemCollection<DocumentFileData>();

        foreach (DocumentFile documentFile in document.Files.Where(CheckFile))
        {
            if (UserSession.CardManager.GetCardState(documentFile.FileId) != ObjectState
.Existing)
            {
                continue;
            }

            VersionedFileCard verFileCard = (VersionedFileCard)UserSession.CardManager
.GetCard(documentFile.FileId);

            BaseCardSignaturePart filePart = lastSignature.Parts ⑫
                .FirstOrDefault(item => item.FileVersion == verFileCard.CurrentVersion
.VersionId &&
                                        ((documentFile.FileType ==
DocumentFileType.Main &&
                                            documentService
.IsDocumentSignaturePartMainFile(item)) ||
                                            (documentFile.FileType ==
```

```
DocumentFileType.Additional &&
                                                    documentService
.IsDocumentSignaturePartAttachments(item))));
            if (filePart == null)
            {
                continue;
            }

            result.Add(new DocumentFileData(documentFile.FileVersionRowId, documentFile
.FileName,
                documentFile.FileType == DocumentFileType.Main, filePart.Signature));
        }

        return result;
    }

    public MessageData PrepareMessageData(Guid cardId, string documentType, string
documentVersion) ⑬
    {
        Document document = GetDocument(cardId);

        if (document.MainInfo.SignatureList == null || !document.MainInfo.SignatureList
.Signatures.Any())
        {
            throw Error.InvalidOperation(Resources.DocumentSignaturesNotExists, cardId);
        }

        MessageData messageData = new MessageData(cardId, documentType, documentVersion,
document.MainInfo.Name);

        DateTime? regDate = document.MainInfo[CardDocument.MainInfo.RegDate] as
DateTime?;
        messageData.DocumentDate = regDate ?? document.CreateDate;
        messageData.DocumentComment = document.MainInfo[CardDocument.MainInfo.Content] as
string;

        Guid numberId = document.MainInfo.GetGuid(CardDocument.MainInfo.RegNumber);
        if (numberId != Guid.Empty)
        {
            BaseCardNumber number = document.Numbers.FirstOrDefault(item =>
ObjectContext.GetObjectRef(item).Id == numberId);
            if (number != null)
            {
                messageData.DocumentNumber = number.Number;
            }
        }

        return messageData;
```

```
    }

    public MessageFile PrepareFileData(Guid cardId, Guid fileId, Guid signatureId, bool
isMain, string tempFolder) ⑭
    {
        return null;
    }

    // Реализация метода проверки подписи
    public SignatureValidation ValidateSignature(Guid cardId, Guid fileId, Guid
signatureId) ⑮
    {
        if (cardId == Guid.Empty)
        {
            throw Error.ArgumentNull("cardId");
        }

        if (fileId == Guid.Empty)
        {
            throw Error.ArgumentNull("fileId");
        }

        if (signatureId == Guid.Empty)
        {
            throw Error.ArgumentNull("signatureId");
        }

        if (!UserSession.FileManager.FileExists(fileId))
        {
            return null;
        }

        byte[] signatureData = null;
        try
        {
            signatureData = UserSession.AccessManager.GetCryptObject(signatureId);
        }
        catch
        {
        }
        if (signatureData == null)
        {
            return null;
        }

        string signerName = null;
        X509Certificate2 certificate = null;
        bool isValid = false;
```

```
        bool isCertificateValid = false;
        string validationError = null;

        try
        {
            FileData fileData = UserSession.FileManager.GetFile(fileId);
            using (Stream stream = fileData.OpenReadStream())
            {
                using (BinaryReader reader = new BinaryReader(stream))
                {
                    ContentInfo contentInfo = new ContentInfo(reader.ReadBytes((int
)reader.BaseStream.Length));
                    SignedCms signedCms = new SignedCms(contentInfo, true);
                    signedCms.Decode(signatureData);

                    if (signedCms.SignerInfos.Count == 0 || signedCms.Certificates.Count
== 0)
                    {
                        return null;
                    }

                    certificate = signedCms.Certificates[0];
                    signerName = CertificateHelper.GetCertificateSignerName(certificate);

                    signedCms.CheckSignature(false);
                    isValid = true;

                    signedCms.CheckSignature(new X509Certificate2Collection(certificate),
true);
                    isCertificateValid = true;
                }
            }
        }
        catch (Exception ex)
        {
            validationError = ex.Message;
        }

        return new SignatureValidation(isValid, isCertificateValid, certificate,
signerName, validationError);
    }

    public string GetInvoiceReplyData(Guid cardId) ⑯
    {
        Document document = GetDocument(cardId);
        if (document == null)
        {
            throw Error.InvalidOperation(Resources.DocumentNotExists, cardId);
```

```
        }

        return GetInvoiceReplyData(document);
    }

    protected virtual bool CheckFile(DocumentFile documentFile) ⑰
    {
        return false;
    }

    protected virtual string GetUniversalDocumentDescription(Guid fileId, MessageFileType
fileType) ⑱
    {
        return null;
    }

    protected virtual string GetUniversalDocumentPrintContent(Document document) ⑲
    {
        return null;
    }

    protected virtual string GetInvoiceReplyData(Document document) ⑳
    {
        return null;
    }
}
```

① Инициализируем компонент чтения. При инициализации получаем сервисы API Docsvision, которые потребуются в дальнейшем.

② Реализация метода получения дайджеста карточки.

③ Реализация метода получения дайджест файла карточки.

④ Реализация метода получения содержимого файла карточки.
Для неформализованного документа реализация функциональности не требуется.
В стандартной реализации для УПД метод возвращает содержимое приложенного файла XML.

⑤ Реализация метода получения содержимое карточки для печати.
Для неформализованного документа реализация функциональности не требуется.
В стандартной реализации для УПД используется XSLT-преобразование данных карточки.
К печатной форме требуемого вида, отображаемой в карточке УПД.

⑥ Реализация метода получения дайджеста для файла ответа.

⑦ Реализация метода, возвращающего получателей электронного документа.
В качестве получателей выбираются все контрагенты из карточки *Документ*.

⑧ Реализация метода получения актуальных подписей (действительная подпись и действительный сертификат) карточки.

⑨ Реализация метода получения всех приложенных к карточке файлов.

⑩ Реализация метода получения последних подписанных файлов.

⑪ Реализация метода получения файлов карточки, подписанных указанной подписью.

⑫ Только подписанные отдельной подписью файлы.

⑬ Реализация метода подготовки данных для создания сообщения электронного обмена.

⑭ Реализация метода подготовки файла сообщения электронного обмена.
Если метод возвращает null будет использована стандартная реализация функции получения MessageFile.
Если требуется, может быть реализован собственный метод получения `MessageFile`, см. подробнее MessageFile — класс.

⑮ По умолчанию выгружаем файл по ID непосредственно при создании сообщения.

⑯ Реализация метода формирования содержимого для ответного сообщения.
Используется при формировании ответа на полученный формализованный документ.

⑰ В своей реализации нужно переопределить метод, добавив алгоритм проверки ЭП файла.

⑱ В своей реализации можно переопределить метод, добавив алгоритм получения описания для файла `fileId` в зависимости от типа сообщения.

⑲ В своей реализации нужно переопределить метод, добавив алгоритм формирования печатной формы для документа `document`.

⑳ В своей реализации нужно переопределить метода, добавив алгоритм формирования содержимого для ответного сообщения на полученный формализованный документ.
Пример реализации в классе `DocsVision.Edi.Runtime.UniversalDocument.BuyerInvoiceDataReader` (сборка `DocsVision.Edi.Runtime.UniversalDocument.dll`).

Разработка компонента изменения данных карточки

Компонент изменения данных предоставляет модулю не зависящий от типа карточки слой изменения данных карточки, обеспечивающий возможность внесения изменения при получении ответа от контрагента.

Компонент должен реализовывать программный интерфейс `IDocumentUpdater`.

Рекомендуется создавать свой компонент на основе абстрактного класса `BaseUpdater` (сборка `DocsVision.Edi.Runtime.UniversalDocument.dll`). Далее приведена часть класса BaseDataReader с описанием.

```
public abstract class BaseUpdater : IDocumentUpdater
{

    public void Initialize(ObjectContext objectContext) ①
    {
        ObjectContext = objectContext;
        UserSession = objectContext.GetService<UserSession>();

        documentService = objectContext.GetService<IDocumentService>();
        baseCardService = objectContext.GetService<IBaseCardService>();
        staffService = objectContext.GetService<IStaffService>();
    }

    public virtual Guid AddSignatureToDocument(Guid cardId, Guid fileId, MessageFile
messageFile, string partnerName) ②
    {
        Document document = GetDocument(cardId);

        DocumentFile documentFile = document.Files.FirstOrDefault(item => item
.FileVersionRowId == fileId);
        if (documentFile == null)
        {
            return Guid.Empty;
        }

        byte[] signatureData = null;
        if (messageFile.SignatureData != null && messageFile.SignatureData.Length > 0) ③
        {
            signatureData = messageFile.SignatureData;
        }
        else if (!string.IsNullOrEmpty(messageFile.SignatureFilePath))
        {
            signatureData = File.ReadAllBytes(messageFile.SignatureFilePath);
        }
```

```
        if (signatureData == null)
        {
            return Guid.Empty;
        }

        SignedCms signedCms = new SignedCms();
        signedCms.Decode(signatureData);

        if (signedCms.SignerInfos.Count == 0 || signedCms.Certificates.Count == 0)
        {
            return Guid.Empty;
        }

        BaseCardSignature cardSignature = document.MainInfo.SignatureList.Signatures
.FirstOrDefault(item => item.SignedFromConfirmation == partnerName); ④
        if (cardSignature == null)
        {
            cardSignature = baseCardService.AddSignature(document.MainInfo.SignatureList,
signedCms.Certificates[0], document.Description, document.SystemInfo.State);
            cardSignature.Signer = staffService.GetCurrentEmployee();
            cardSignature.SignedFromConfirmation = partnerName;
            cardSignature.Certificate = signedCms.Certificates[0].Thumbprint;
            cardSignature.Imported = true;
        }

⑤
        BaseCardSignature newSignature = baseCardService.AddSignaturePart(cardSignature,
MainFileSignaturePartType, signatureData,
            string.Format(Resources.MainFilePartnerSignature, documentFile.FileName),
documentFile.FileId, documentFile.FileVersionId, document);

        ObjectContext.SaveObject(document);

        return ObjectContext.GetObjectRef(newSignature).Id;
    }

    public virtual Guid CreateNewSignatures(Guid cardId, X509Certificate2 certificate,
bool signAdditionalFiles) ⑥
    {
        Document document = GetDocument(cardId);

        if (document.MainInfo.SignatureList == null) ⑦
        {
            SignatureList signatureList = documentService.CreateSignatureList();
            ObjectContext.SaveObject(signatureList);
            document.MainInfo.SignatureList = signatureList;
        }
```

```
        foreach (DocumentFile documentFile in document.Files)
        {
            UpdateInvoiceFile(document, documentFile, certificate);
            string validateResult = ValidateInvoiceFile(document, documentFile); ⑧
            if (!string.IsNullOrEmpty(validateResult))
            {
                throw new IncompleteSignatureDataException(string.Format(Resources
.UnableToValidateInvoice, documentFile.FileName, validateResult));
            }
        }

        DocumentSetting documentSetting = documentService.GetKindSettings(document
.SystemInfo.CardKind);

        BaseCardSignature baseCardSignature = documentService.AddSignature(document,
certificate, false, documentSetting.DocumentSignature.Fields); ⑨

        ObjectContext.SaveObject(document);

        return ObjectContext.GetObjectRef(baseCardSignature).Id;
    }

    public virtual string CheckCertificate(Guid cardId, X509Certificate2 certificate) ⑩
    {
        return CheckDocumentSignerForCertificate(GetDocument(cardId), certificate);
    }

    public virtual bool ChangeDocumentState(Guid cardId, Guid newStateId) ⑪
    {
        if (cardId == Guid.Empty)
        {
            throw Error.ArgumentNull("cardId");
        }

        Document document = GetDocument(cardId);

        IStateService stateService = ObjectContext.GetService<IStateService>();
        StatesStateMachineBranch branch = stateService.FindLineBranchesByStartState
(document.SystemInfo.State).FirstOrDefault(item => item.EndState.BuiltInState ==
newStateId);
        if (branch == null)
        {
            return false;
        }

        if (!stateService.IsOperationAllowedFull(branch.Operation, document))
        {
```

```
            return false;
        }

        stateService.ChangeState(document, branch, true, out string processErrors);

        return true;
    }

    public virtual Guid GetNewStateId(MessageFileType messageFileType) ⑫
    {
        switch (messageFileType)
        {
            case MessageFileType.InvoiceReply:
            case MessageFileType.ReplySignature:
                return SignedStateId;

            case MessageFileType.SignatureRejection:
                return RejectedStateId;

            case MessageFileType.InvoiceCorrectionRequest:
                return CorrectionRequiredStateId;

            case MessageFileType.RevocationRequest:
                return RevocationRequestStateId;

            case MessageFileType.RevocationReply:
                return RevocationReplyStateId;

            case MessageFileType.RevocationRejection:
                return RevocationRejectionStateId;
        }

        return Guid.Empty;
    }

    public void UpdateDocumentDataFromFile(Guid cardId, Guid fileId) ⑬
    {
        if (cardId == Guid.Empty)
        {
            throw Error.ArgumentNull("cardId");
        }

        if (fileId == Guid.Empty)
        {
            throw Error.ArgumentNull("fileId");
        }

        Document document = GetDocument(cardId);
```

```
        DocumentFile documentFile = document.Files.FirstOrDefault(item => item
.FileVersionRowId == fileId);
        if (documentFile == null)
        {
            throw Error.InvalidOperation(Resources.FileNotExists, fileId, cardId);
        }

        UpdateDocumentDataFromFile(document, documentFile);
    }

    protected virtual void UpdateInvoiceFile(Document document, DocumentFile
documentFile, X509Certificate2 certificate) ⑭
    {
    }

    protected virtual string ValidateInvoiceFile(Document document, DocumentFile
documentFile) ⑮
    {
        return null;
    }

    protected virtual string CheckDocumentSignerForCertificate(Document document,
X509Certificate2 certificate) ⑯
    {
        CardData documentData = UserSession.CardManager.GetCardData(ObjectContext
.GetObjectRef(document).Id);
        RowData signerRow = documentData.Sections[documentData.Type.AllSections[CardDefs
.UniversalDocumentSignature.Alias].Id].FirstRow;
        Guid employeeId = signerRow.GetGuid(CardDefs.UniversalDocumentSignature.Signer
).GetValueOrDefault();
        StaffEmployee employee = (employeeId != Guid.Empty) ? ObjectContext.GetObject
<StaffEmployee>(employeeId) : null;

        if (employee == null)
        {
            return Resources.NoSignerData;
        }

        string certificateName = CertificateHelper.GetCertificateSignerName(certificate);
        if (string.IsNullOrEmpty(certificateName))
        {
            return Resources.NoCertificateSignerData;
        }

        if (string.Compare($"{employee.LastName} {employee.FirstName} {employee
.MiddleName}", certificateName, StringComparison.InvariantCultureIgnoreCase) != 0)
        {
```

```
            return Resources.DifferentSignerData;
        }

        return null;
    }

⑰
    protected virtual void UpdateDocumentDataFromFile(Document document, DocumentFile
documentFile)
    {
    }
}
```

① Инициализируем компонент отправки.
При инициализации получаем сервисы API Docsvision, которые потребуются в дальнейшем.

② Реализация метода добавления подписи к указанному файлу карточки.

③ Получаем данные подписи из сообщения.
Подпись может быть в сообщении или в файле, указанном в сообщении.

④ Получаем карточку подписей.

⑤ Добавляем подпись в документ.

⑥ Реализация метода подписания карточки.

⑦ Если в документе не списка подписей — создаём.

⑧ Обновляем информацию в титуле продавца.

⑨ Добавляем подпись в документ.

⑩ Реализация метода проверка сертификата подписи в карточке.
Для неформализованного документа реализация функциональности не требуется.
В стандартной реализации для УПД выполняется сравнение данных сотрудника, подписавшего УПД с данными сотрудника-владельца указанного сертификата.

⑪ Реализация метода изменения состояния карточки.

⑫ Реализация метода получения идентификатора состояния карточки.
Для неформализованного документа реализация функциональности не требуется.
В стандартной реализации для УПД метод возвращает идентификатор одного из встроенных состояний карточки УПД из Конструктора состояний.

⑬ Реализация метода обновления данных карточки данными из файла титула продавца.
Для неформализованного документа реализация функциональности не требуется.
В стандартной реализации для УПД метод получает содержимое файла (XML-формата) и устанавливает значения полей карточки.

⑭ В своей реализации нужно переопределить метод, добавив алгоритм обновления титула продавца.
Пример реализации в классе `DocsVision.Edi.Runtime.UniversalDocument.SellerInvoiceUpdater` (сборка `DocsVision.Edi.Runtime.UniversalDocument.dll`).

⑮ В своей реализации нужно переопределить метод, добавив алгоритм проверки файла титула продавца.

⑯ В своей реализации нужно переопределить метод, добавив алгоритм проверки подписанта.

⑰ В своей реализации нужно переопределить метод, добавив алгоритм загрузки данных в карточку из приложенного файла титула продавца.
Пример реализации в классе `DocsVision.Edi.Runtime.UniversalDocument.BuyerInvoiceUpdater` (сборка `DocsVision.Edi.Runtime.UniversalDocument.dll`).

Разработка компонента создания документов

Компонент создания документов предоставляет модулю функцию создания карточек в системе Docsvision по поступившим от оператора ЭДО электронным документам.

Компонент должен реализовывать программный интерфейс IDocumentCreator.

Рекомендуется создавать свой компонент на основе класса DocumentCreator (сборка `DocsVision.Edi.Runtime.BackOffice.dll`). Далее приведена часть класса DocumentCreator с описанием.

```
public class DocumentCreator : IDocumentCreator
{

    public void Initialize(ObjectContext objectContext, Guid defaultKindId) ①
    {
        ObjectContext = objectContext;
        this.defaultKindId = defaultKindId;
```

```
        UserSession = ObjectContext.GetService<UserSession>();

        documentService = ObjectContext.GetService<IDocumentService>();
        staffService = ObjectContext.GetService<IStaffService>();
        baseCardService = ObjectContext.GetService<IBaseCardService>();

        documentFilesHelper = new DocumentFilesHelper(UserSession, documentService,
baseCardService, staffService);
        parentLinksCreator = new ParentLinksCreator(ObjectContext, UserSession);
    }


    public virtual bool CreateDocument(MessageData messageData, Guid partnerId, Guid
partnerDepartmentId) ②
    {
        bool isSaved = false;
        Document document = null;

        try
        {
            KindsCardKind newCardKind = GetNewDocumentKind(messageData, partnerId); ③
            if (newCardKind == null)
            {
                return false;
            }

            document = documentService.CreateDocument(null, newCardKind); ④

            if (messageData.DocumentDate.HasValue)
            {
                document.MainInfo.DeliveryDate = messageData.DocumentDate.Value;
            }

            document.MainInfo[CardDocument.MainInfo.ExternalNumber] = messageData
.DocumentNumber;
            document.MainInfo[CardDocument.MainInfo.Content] = messageData
.DocumentComment;
            document.MainInfo[CardDocument.MainInfo.RegDate] = DateTime.Now;
            document.MainInfo.Name = messageData.DocumentName;

            var senderRows = document.GetSection(CardDocument.SenderPartner.ID);
            BaseCardSectionRow senderRow;
            if (senderRows.Count == 0)
            {
                senderRow = new BaseCardSectionRow();
                senderRows.Add(senderRow);
            }
```

```
            else
            {
                senderRow = (BaseCardSectionRow)senderRows[0];
            }

            PartnersCompany partnersCompany = ObjectContext.GetObject<PartnersCompany
>(partnerId);

            senderRow[CardDocument.SenderPartner.SenderOrg] = partnerId;
            senderRow[CardDocument.SenderPartner.SenderDep] = partnerDepartmentId;

            StaffEmployee employee = GetNewDocumentAuthor(messageData, partnerId);
            Guid employeeId = employee == null ? Guid.Empty : ObjectContext.GetObjectRef
(employee).Id;

            if (employee != null)
            {
                document.MainInfo.Registrar = employee;
            }

            StaffUnit recipientDepartment = null;

⑤
            StaffEmployee recipient = null;
            StaffGroup recipientGroup = null;
            if (!string.IsNullOrEmpty(messageData.Recipient.DepartmentId))
            {
                OperatorsDepartment operatorsDepartment = ObjectContext.FindObject
<OperatorsDepartment>(
                    new QueryObject(RefOperators.Departments.DepartmentId, messageData
.Recipient.DepartmentId));
                if (operatorsDepartment != null)
                {
                    recipientDepartment = operatorsDepartment.Department;
                    recipient = operatorsDepartment.Recipient;
                    recipientGroup = operatorsDepartment.RecipientGroup;
                }
            }

            if (recipientDepartment == null)
            {
                OperatorsBox organizationBox = ObjectContext.FindObject<OperatorsBox>(
                    new QueryObject(RefOperators.Boxes.BoxId, messageData.Recipient.
BoxId));
                if (organizationBox != null)
                {
                    recipientDepartment = organizationBox.Unit.Unit;
                    recipient = organizationBox.Unit.Recipient;
```

```
                    recipientGroup = organizationBox.Unit.RecipientGroup;
                }
            }

            if (recipientDepartment != null)
            {
                document.MainInfo[CardDocument.MainInfo.ResponsDepartment] =
ObjectContext.GetObjectRef(recipientDepartment).Id;
            }

            if (recipientGroup != null)
            {
                foreach (StaffGroupItem groupItem in recipientGroup.GroupItems)
                {
                    BaseCardSectionRow recRow = new BaseCardSectionRow();
                    recRow[CardDocument.ReceiversStaff.ReceiverStaff] = groupItem
.EmployeeId;
                    document.GetSection(CardDocument.ReceiversStaff.ID).Add(recRow);
                }
            }
            else if (recipient != null)
            {
                BaseCardSectionRow recRow = new BaseCardSectionRow();
                recRow[CardDocument.ReceiversStaff.ReceiverStaff] = ObjectContext
.GetObjectRef(recipient).Id;
                document.GetSection(CardDocument.ReceiversStaff.ID).Add(recRow);
            }

            if (!string.IsNullOrEmpty(messageData.DocumentType))
            {
                SetDocumentType(messageData, document);
            }

            ObjectContext.SaveObject(document);
            isSaved = true;


            foreach (MessageFile messageFile in messageData.DocumentFiles) ⑥
            {
                AddFileToDocument(document, messageFile, employeeId, partnersCompany.
Name);
            }

            SetAdditionalAttributes(messageData, document);

            ObjectContext.SaveObject(document);
```

```
            parentLinksCreator.CreateParentLinks(document, ⑦
                messageData.DocumentFiles.Select(item => item.OperatorParentEntityId
).Where(item => !string.IsNullOrEmpty(item)).Distinct().ToList());

            Guid cardId = ObjectContext.GetObjectRef(document).Id;

            document.Description = baseCardService.GenerateDigest(document, UserSession
.CardManager.GetCardData(cardId), null);

            ObjectContext.SaveObject(document);

            messageData.CardId = cardId;

            return true;
        }
        catch (Exception)
        {
            SafeRollback();
            try
            {
                if (isSaved)
                {
                    ObjectContext.DeleteObject(document);
                    ObjectContext.AcceptChanges();
                }
            }
            catch
            {
            }

            throw;
        }
    }


    public virtual void SetDocumentType(MessageData messageData, Document document) ⑧
    {
        CardSection dataSection = UserSession.CardManager.CardTypes[CardDocument.ID
].AllSections ⑨
            .FirstOrDefault(item => string.Equals(item.Alias, CardDefs
.UniversalDocumentData.Alias));
        if (dataSection == null)
        {
            return;
        }

        if (!dataSection.Fields.Contains(CardDefs.UniversalDocumentData.DocumentType))
        {
```

```
            return;
        }

        Field docTypeField = dataSection.Fields[CardDefs.UniversalDocumentData
.DocumentType];
        var invoiceRows = document.GetSection(dataSection.Id);
        BaseCardSectionRow invoiceRow;
        if (invoiceRows.Count == 0)
        {
            invoiceRow = new BaseCardSectionRow();
            invoiceRows.Add(invoiceRow);
        }
        else
        {
            invoiceRow = (BaseCardSectionRow)invoiceRows[0];
        }

        invoiceRow[CardDefs.UniversalDocumentData.DocumentType] = docTypeField.EnumValues
            .FirstOrDefault(item => string.Equals(messageData.DocumentType, item.Alias,
StringComparison.OrdinalIgnoreCase))?.Value;
    }

    public virtual void AddReplyFilesToDocument(MessageData messageData, Guid partnerId)
⑩
    {
        try
        {
            StaffEmployee employee = GetNewDocumentAuthor(messageData, partnerId);
            Guid employeeId = employee == null ? Guid.Empty : ObjectContext.GetObjectRef
(employee).Id;

            PartnersCompany partnersCompany = ObjectContext.GetObject<PartnersCompany
>(partnerId);

            Document document = ObjectContext.GetObject<Document>(messageData.CardId);
            foreach (MessageFile messageFile in messageData.DocumentFiles)
            {
                AddFileToDocument(document, messageFile, employeeId, partnersCompany.
Name);
            }

            ObjectContext.SaveObject(document);
        }
        catch
        {
            SafeRollback();
            throw;
        }
```

```
    }


    public virtual KindsCardKind GetNewDocumentKind(MessageData messageData, Guid
partnerId) ⑪
    {
        KindsCardKind cardKind = null;
        if (defaultKindId != Guid.Empty)
        {
            cardKind = ObjectContext.GetObject<KindsCardKind>(defaultKindId);
        }

        return cardKind ?? ObjectContext.GetObject<KindsCardKind>(
IncomingDocumentKindId);
    }


    public virtual StaffEmployee GetNewDocumentAuthor(MessageData messageData, Guid
partnerId) ⑫
    {
        return staffService.GetCurrentEmployee();
    }


    public virtual void SetAdditionalAttributes(MessageData messageData, Document
document) ⑬
    {
    }


    public virtual void AddFileToDocument(Document document, MessageFile messageFile,
Guid authorId, string partnerName) ⑭
    {
        documentFilesHelper.AddSignedFileToDocument(document, messageFile, authorId,
partnerName, true);
    }
}
```

① Инициализация компонента.

② Реализация метода создания карточки входящего документа по полученному от оператора ЭДО сообщению.

③ Получаем настройки вида карточек.

④ Создаём документ.

⑤ Группа из настроек подразделения или организации (указывается в

получателях).

⑥ Добавление файлов из полученного сообщения в создаваемый документ.

⑦ Добавление ссылки на родительскую карточку, если поступило исправление формализованного документа.

⑧ Добавляет в карточку документа значение типа поступившего документа.
В собственной реализации метод может записывать типы документов по другой логике.

⑨ Секции и поля может не быть, если не грузили схему УПД.

⑩ Реализация метода переноса файлов из ответа контрагента в карточку.

⑪ Предоставляет вид создаваемого документа.
В собственной реализации метод может вычислять вид документа по другой логике.

⑫ Предоставляет автора создаваемого документа.
В собственной реализации метод может вычислять автора документа по другой логике.

⑬ Добавляет дополнительные атрибуты в создаваемый документ.
В собственной реализации метод может добавлять в карточку требуемые данные из полученного сообщения.

⑭ Добавляет в создаваемый документ файл из полученного сообщения.
В собственной реализации метод может добавлять файл, следуя иной логике.

Разработка компонента загрузки событий в существующей карточке

События отдельных документов Модуля интеграции с операторами ЭДО и коннекторах по умолчанию не синхронизируются, так как исходящие события от оператора ЮЗДО не запрашиваются. Данная возможность может быть реализована с помощью скриптов.

Ниже приведён пример загрузки событий произвольного документа в Windows-клиенте с помощью скриптов. Можно загрузить как уже существующий, так и отсутствующий в системе документ. Скрипт может быть назначен на кнопку, добавлен в БП (с другим вариантом ILogWriter) или использован иным способом.

```
    private void LoadDocumentById(Guid boxRowId, string documentId, bool isOutgoing) ①
    {
        try
        {
```

```
            OperatorsBox operatorsBox = objectContext.GetObject<OperatorsBox>(boxRowId);
            BoxesService boxesService = new BoxesService(objectContext, false);
            LogWriter logWriter = new LogWriter();
            boxesService.LoadDocumentById(operatorsBox, documentId, isOutgoing,
logWriter);
            MessageBox.Show(logWriter.LogText);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
    }

    internal class LogWriter : ILogWriter
    {
        private readonly StringBuilder logBuilder = new StringBuilder();

        public void WriteInformation(string message, params object[] args)
        {
            logBuilder.AppendLine(string.Format(message, args));
        }

        public void WriteError(string message, params object[] args)
        {
            logBuilder.AppendLine(string.Format(message, args));
        }

        public void WriteError(string message, Exception ex, params object[] args)
        {
            logBuilder.AppendLine(string.Format(message, args));
            logBuilder.AppendLine(ex.ToString());
        }

        public string LogText => logBuilder.ToString();
    }
```

① Загрузка документа по ID.

Параметры:

- `boxRowId` — ID строки ящика из справочника операторов ЮЗДО.
- `documentId` — ID документа Диадок, то что у них в параметре `letterId` в адресе.
- `isOutgoing` — `true` — загрузить как исходящий, `false` — как входящий.

Следующий фрагмент кода демонстрирует загрузку недостающих событий в уже существующий документ при помощи скрипта Windows-клиента.

```
 private void RefreshDocumentEvents_ItemClick(System.Object sender, DevExpress.XtraBars
.ItemClickEventArgs e)
  {
    try

    {
      var searchQuery = CardControl.Session.CreateSearchQuery();
      var messageTypeQuery = searchQuery.AttributiveSearch.CardTypeQueries.AddNew
(CardEdiMessage.ID);
      var messageMainInfoQuery = messageTypeQuery.SectionQueries.AddNew(CardEdiMessage
.MainInfo.ID);
      messageMainInfoQuery.ConditionGroup.Operation = ConditionGroupOperation.And;
      messageMainInfoQuery.ConditionGroup.Conditions.AddNew(CardEdiMessage.MainInfo
.CardId, FieldType.UniqueId, ConditionOperation.Equals, CardControl.CardData.Id);

      CardDataCollection foundCards = CardControl.Session.CardManager.FindCards
(searchQuery.GetXml());
      if (foundCards.Count == 0)
      {
        CardControl.ShowMessage("У документа нет связанных карточек сообщений ЭДО.");
      return;
      }

      EdiMessage ediMessage = CardControl.ObjectContext.GetObject<EdiMessage>(foundCards
.Last().Id);

      if (string.IsNullOrEmpty(ediMessage.MainInfo.OperatorMessageId))
      {
        CardControl.ShowMessage("Карточка обмена сообщениями не связана с сообщением
оператора.");
        return;
      }

      LogWriter logWriter = new LogWriter();
      BoxesService boxesService = new BoxesService(CardControl.ObjectContext);
      boxesService.LoadDocumentById(ediMessage.MainInfo.OrganizationBox, ediMessage
.MainInfo.OperatorMessageId, !ediMessage.MainInfo.IncomingMessage, logWriter);
      CardControl.ShowMessage(logWriter.LogText);
      }
        catch (Exception ex)
      {
        CardControl.ShowMessage(ex.Message, "Документ", ex.ToString(), MessageType.Error,
MessageButtons.Ok);
      }
    }
```

```
    internal class LogWriter : ILogWriter
    {
      private readonly StringBuilder logBuilder = new StringBuilder();

      public void WriteInformation(string message, params object[] args)
      {
        logBuilder.AppendLine(string.Format(message, args));
      }

      public void WriteError(string message, params object[] args)
      {
        logBuilder.AppendLine(string.Format(message, args));
      }

      public void WriteError(string message, Exception ex, params object[] args)
      {
        logBuilder.AppendLine(string.Format(message, args));
        logBuilder.AppendLine(ex.ToString());
      }

      public string LogText
      {
        get{return logBuilder.ToString();}
      }
  }
```

Для работы скрипта потребуется указать дополнительные пространства имен:

```
using DocsVision.Edi.Runtime;
using DocsVision.Edi.CardLib.CardDefs;
using DocsVision.Platform.ObjectManager.SearchModel;
using DocsVision.Platform.ObjectManager.Metadata;
using DocsVision.Edi.ObjectModel;
using DocsVision.Edi.Interfaces;
using System.Linq;
using System.Text;
```

Также для компиляции следует подключить дополнительные сборки из папки `C:\Program Files (x86)\Docsvision\Edi\` на Windows-машине с серверными компонентами Docsvision:

```
DocsVision.Edi.ObjectModel.dll
DocsVision.Edi.Interfaces.dll
DocsVision.Edi.Runtime.dll
```

При использовании скрипта для новых видов документов, если в нём не было загрузки библиотек ЭДО и обращений к `EdiScriptHelper`, в инициализацию необходимо добавить следующие строки, например, для `CardActivated`:

```
EdiMapperFactory.RegisterFactory(cardControl.ObjectContext);
EdiServiceFactory.RegisterFactory(cardControl.ObjectContext);
```

А также дополнить сборки:

```
using DocsVision.Edi.ObjectModel.Mapping;
using DocsVision.Edi.ObjectModel.Services;
```

При использовании скрипта существует ограничение, связанное с отображением кнопок предыдущего состояния. При смене состояния документа в Диадок, например, подтверждении или отказе от аннулирования, состояние документа в Docsvision не изменится, а в карточке останутся доступны кнопки предыдущего состояния (запроса аннулирования).

Разработка коннектора

Модуль позволяет добавлять возможность обмениваться электронными документами с контрагентами через различных операторов ЭДО.

Чтобы добавить возможность обмениваться через оператора ЭДО, нужно реализовать три компонента:

- Компонент сервиса сообщений,.
- Компонент импорта.
- Компонент сервиса подписей.

Для каждого оператора ЭДО разрабатывается свой комплект компонентов коннектора. Компоненты для работы с карточками могут быть общими для всех операторов ЭДО или для каждого оператора может использоваться собственный набор компонентов.

Далее приведены требования и примеры реализации компонентов коннектора к оператору ЭДО, а также инструкция по подключению разработанного коннектора к модулю.

Разработка компонента сервиса сообщений

Компонент сервиса сообщений предоставляет модулю не зависящий от оператора ЭДО слой отправки и получения электронных сообщений.

Компонент данного типа непосредственно взаимодействует с оператором ЭДО. При взаимодействии выполняется передача и получение сообщений электронного обмена, в т.ч. содержащих электронные документы (входящие и исходящие).

Данный компонент должен реализовывать программный интерфейс IMessageService. Методы реализуемого интерфейса вызываются при получении и отправке сообщений оператору ЭДО.

Вариант реализации данного компонента содержится в сборке DocsVision.Edi.Runtime.Diadoc.dll (добавляется при установке "Коннектор к Диадок").

Фактическая реализация компонента сервиса сообщений сильно зависит от метода взаимодействия с оператором ЭДО и предоставляемого им API.

Далее приведена часть исходного кода (только публичные методы) примера реализации интерфейса IMessageService для работы с оператором *Диадок*.

```
public class MessagesService: IMessageService ①
{
    private DiadocSession DiadocSession ②
    {
        get
        {
            if (diadocSession != null && !diadocSession.IsAlive())
            {
                diadocSession = null;
            }

            if (diadocSession == null)
            {
                if (!settings.ContainsKey(DiadocSettings.ApiUrl))
                {
                    throw Error.InvalidOperation(Resources.NoConnectionData);
                }

                if (!settings.ContainsKey(DiadocSettings.Login))
                {
                    throw Error.InvalidOperation(Resources.NoConnectionData);
```

```
                }

                diadocSession = new DiadocSession(
                    settings[DiadocSettings.ApiUrl],
                    settings[DiadocSettings.Login],
                    settings[DiadocSettings.Password],
                    settings.ContainsKey(DiadocSettings.ProxyUrl) ? settings
[DiadocSettings.ProxyUrl] : null,
                    settings.ContainsKey(DiadocSettings.ProxyLogin) ? settings
[DiadocSettings.ProxyLogin] : null,
                    settings.ContainsKey(DiadocSettings.ProxyPassword) ? settings
[DiadocSettings.ProxyPassword] : null);
            }

            return diadocSession;
        }
    }
③
    public void Initialize(Dictionary<string, string> settings) ④
    {
        this.settings = settings;
    }

    public void SendMessage(MessageData messageData) ⑤
    {
        DiadocSession.SendMessage(messageData);
    }

    public void SendSignatureReply(MessageData messageData) ⑥
    {
        DiadocSession.SendSignatureReply(messageData);
    }

    public void SendReceipt(MessageData messageData) ⑦
    {
        DiadocSession.SendReceipt(messageData);
    }

    public void SendRevocation(MessageData messageData) ⑧
    {
        DiadocSession.SendRevocation(messageData);
    }

    public void SendCorrection(MessageData messageData) ⑨
    {
        DiadocSession.SendCorrection(messageData);
    }
```

```
    public int GetNewEventsCount(string boxId, string lastEventId) ⑩
    {
        return DiadocSession.GetNewEventsCount(boxId, lastEventId);
    }

    public string GetLastEventId(string boxId) ⑪
    {
        return DiadocSession.GetLastEventId(boxId, null);
    }

    public bool EventExists(string boxId, string eventId) ⑫
    {
        return DiadocSession.EventExists(boxId, eventId);
    }

    public ItemCollection<MessageData> GetNewEvents(string boxId, string lastEventId,
DateTime? fromDate) ⑬
    {
        return DiadocSession.GetNewEvents(boxId, lastEventId, fromDate);
    }

    public MessageFile GenerateInvoiceReceipt(string boxId, string messageId, string
entityId, X509Certificate2 certificate, string positionName) ⑭
    {
        return DiadocSession.GenerateInvoiceReceipt(boxId, messageId, entityId,
certificate, positionName);
    }

    public MessageFile GenerateRevocationRequest(string boxId, string messageId, string
entityId, ⑮
        X509Certificate2 certificate, string positionName, string comment)
    {
        return DiadocSession.GenerateRevocationRequest(boxId, messageId, entityId,
certificate, positionName, comment);
    }

    public MessageFile GenerateInvoiceReply(string boxId, string messageId, string
entityId, X509Certificate2 certificate, string documentType, string replyData) ⑯
    {
        return DiadocSession.GenerateInvoiceReply(boxId, messageId, entityId,
certificate, documentType, replyData);
    }

    public MessageFile GenerateSignatureRejection(string boxId, string messageId, string
entityId, ⑰
        X509Certificate2 certificate, string positionName, string comment)
    {
        return DiadocSession.GenerateSignatureRejection(boxId, messageId, entityId,
```

```
certificate, positionName, comment);
    }

    public MessageFile GenerateCorrectionRequest(string boxId, string messageId, string
entityId, ⑱
        X509Certificate2 certificate, string positionName, string comment)
    {
        return DiadocSession.GenerateCorrectionRequest(boxId, messageId, entityId,
certificate, positionName, comment);
    }
}
```

① Компонент сервиса сообщение.

② Точка доступа к API Диадок.

③ Реализация интерфейса IMessageService.

④ Инициализация компонента.

⑤ Реализация метода отправки сообщения оператору ЭДО.

⑥ Реализация метода отправки ответной подписи.

⑦ Реализация метода отправки квитанции.

⑧ Реализация метода отправки запроса на аннулирование.

⑨ Реализация метода отправки запроса на уточнение.

⑩ Реализация метода получения количества новых событий.

⑪ Реализация метода получения идентификатора последнего события в ящике организации.

⑫ Реализация метода проверки наличия события в ящике.

⑬ Реализация метода получения новых событий из ящика.

⑭ Реализация метода формирования извещения.

⑮ Реализация метода формирования запроса на аннулирование.

⑯ Реализация метода формирования ответной подписи для поступившего документа.

⑰ Реализация метода формирования сообщения с отказом в подписании.

⑱ Реализация метода формирования запроса на уточнение.

Далее приведена часть исходного кода класса `DiadocSession`, в котором показан принцип взаимодействия с оператором ЭДО через его API на примере получения идентификатора последнего сообщения.

```
public class DiadocSession
{
    private readonly DiadocApi DiadocApi; ①
    private readonly string AuthToken;

       public DiadocSession(string apiUrl, string login, string password, string
proxyUrl, string proxyLogin, string proxyPassword)
    {
        WinApiCrypt diadocCrypt = new WinApiCrypt();
        DiadocApi = new DiadocApi(DefaultClientId, apiUrl, diadocCrypt);

        if (!string.IsNullOrEmpty(proxyUrl))
        {
            DiadocApi.DisableSystemProxyUsage();
            DiadocApi.SetProxyUri(proxyUrl);
            if (!string.IsNullOrEmpty(proxyLogin))
                DiadocApi.SetProxyCredentials(proxyLogin, proxyPassword);
        }

        AuthToken = DiadocApi.Authenticate(login, password);
    }

       public string GetLastEventId(string boxId, string lastEventId) ②
    {
        while (true)
        {
            BoxEventList eventList = DiadocApi.GetNewEvents(AuthToken, boxId,
lastEventId);
            if (eventList == null || eventList.TotalCount == 0)
                return lastEventId;

            lastEventId = eventList.Events[eventList.Events.Count -- 1].EventId;
            if (eventList.TotalCount == eventList.Events.Count)
                return lastEventId;
        }
    }

}
```

① DiadocApi — объект API Диадок, проксирующий работу с веб-сервисом Диадок. Описание API Диадок приведено на странице с документацией Диадок API.

② Получаем идентификатор последнего события в ящике организации.

Разработка компонента сервиса ответных сообщений

Далее приведена часть исходного кода (только публичные методы) примера

реализации класса `OutgoingMessagesService` для отправки ответа на неформализованный документ.

```
OutgoingMessagesService outgoingMessagesService = new OutgoingMessagesService(
cardControl.ObjectContext); ①

EdiMessage messageCard = outgoingMessagesService.FindIncomingMessageCards(cardId
).FirstOrDefault(); ②

③


DocumentUpdaterService documentUpdaterService = new DocumentUpdaterService(
objectContext); ④

Guid newSignatureGroupId = documentUpdaterService.CreateNewSignatures(cardId,
certificate, false); ⑤

⑥

DataReaderService dataReaderService = new DataReaderService(objectContext); ⑦

ItemCollection<DocumentFileData> signedFiles = dataReaderService.GetSignedFiles(cardId,
newSignatureGroupId); ⑧

if (outgoingMessagesService.CreateOutgoingSignatureReplies(cardId, signedFiles).Any())
{

} ⑨
else
{
⑩
}
```

① Сервис отправки исходящих событий.

② Поиск карточки сообщений, для которой нужен ответ, для документа `cardId`.

③ Данный момент можно пропустить, если уже есть готовая ответная подпись. Будьте внимательны, при пропуске существует риск отправить оригинальную подпись контрагента как ответную.

④ Сервис изменения документа.

⑤ `X509Certificate2 certificate` на входе — это сертификат, которым делаем новые подписи.

⑥ Конец создания новой подписи.

⑦ Сервис чтения данных из карточек.

⑧ Получаем свежеподписанные файлы.
При наличии готовой ответной подписи, `newSignatureGroupId` — это `ID` подписи из списка подписей.

⑨ Все хорошо, подписи отправили.
Ответные подписи успешно отправлены.

⑩ По карточке сообщения уже некому отправлять ответ — отработал другой скрипт, кто-то отправил руками и т.п.
Не существует связанных писем в ящике ЮЗДО, ожидающих ответных действий по ним. Отправка произведена не будет.

Далее приведён пример отправки отказа от подписи, сервисы те же, что в предыдущем примере:

```
outgoingMessagesService.CreateOutgoingRejectionReplies(cardId, commentText, certificate);
①
```

① `commentText` — причина отказа, `X509Certificate2 certificate` — сертификат.

Следующий пример демонстрирует отправку положительного ответа на УПД или УКД:

```
string invoiceReplyData = dataReaderService.GetInvoiceReplyData(cardId); ①

if (outgoingMessagesService.CreateOutgoingInvoiceReplies(cardId, certificate,
invoiceReplyData).Any()) ②
{
③
}
else
{
④
}
```

① Получение данных ответа для ответа из карточки.

② Генерация файла ответа, подписание его сертификатом `X509Certificate2 certificate`, отправка.

③ Отправили успешно.

④ Не отправили, то же, что и для неформализованных.

Разработка компонента импорта настроек организации из системы оператора ЭДО

Компонент импорта предоставляет модулю зависящий не от оператора ЭДО слой получения списка подразделений организации, контрагентов и ящиков, зарегистрированных в системе оператора ЭДО. Информация, предоставляемая данным компонентом, требуется для определения связей между контрагентами, зарегистрированными в системе Docsvision, и контрагентами организации в системе оператора ЭДО.

Данный компонент должен реализовывать программный интерфейс IImportService. Методы реализуемого интерфейса вызываются из компонента *Справочника настроек операторов ЮЗДО* при загрузке подразделений и ящиков организации, контрагентов и ящиков контрагентов.

Реализация данного компонента, получающего информацию от оператора Диадок, содержится в сборке `DocsVision.Edi.Runtime.Diadoc.dll` (добавляется при установке *Коннектор к Диадок*).

Фактическая реализация компонента импорта сильно зависит от метода взаимодействия с оператором ЭДО и предоставляемого им API.

Далее приведена часть исходного кода (только публичные методы) примера реализации интерфейса `ImportService` для работы с оператором *Диадок*.

```
public class ImportService : IImportService
{
    private DiadocSession DiadocSession ①
    {
        get
        {
            if (diadocSession != null && !diadocSession.IsAlive())
                diadocSession = null;

            if (diadocSession == null)
            {
                if (!settings.ContainsKey(DiadocSettings.ApiUrl))
                    throw Error.InvalidOperation(Resources.NoConnectionData);
                if (!settings.ContainsKey(DiadocSettings.Login))
                    throw Error.InvalidOperation(Resources.NoConnectionData);

                diadocSession = new DiadocSession(
```

```
                    settings[DiadocSettings.ApiUrl],
                    settings[DiadocSettings.Login],
                    settings[DiadocSettings.Password],
                    settings.ContainsKey(DiadocSettings.ProxyUrl) ? settings
[DiadocSettings.ProxyUrl] : null,
                    settings.ContainsKey(DiadocSettings.ProxyLogin) ? settings
[DiadocSettings.ProxyLogin] : null,
                    settings.ContainsKey(DiadocSettings.ProxyPassword) ? settings
[DiadocSettings.ProxyPassword] : null);
            }

            return diadocSession;
        }
    }


②

    public void Initialize(Dictionary<string, string> settings) ③
    {
        this.settings = settings; ④
    }


    public ItemCollection<Unit> ImportUnits() ⑤
    {
        return DiadocSession.ImportUnits();
    }

    public ItemCollection<Partner> ImportPartners() ⑥
    {
        return DiadocSession.ImportPartners();
    }
}
```

① Точка подключения к API Диадок.

② Реализация интерфейса `IImportService`.

③ Реализация метода инициализации компонента.

④ Загрузка настроек.

⑤ Реализация метода получения от оператора ЭДО списка организаций.

⑥ Реализация метода получения от оператора ЭДО списка контрагентов.

Далее приведена часть исходного кода класса `DiadocSession`, показывающая принцип взаимодействия с оператором ЭДО через его API на примере загрузки

данных организации.

```
internal class DiadocSession {
 private readonly DiadocApi diadocApi;
 private readonly string authToken;

 public DiadocSession(string apiUrl, string login, string password) {

①
  WinApiCrypt diadocCrypt = new WinApiCrypt();
  diadocApi = new DiadocApi(DefaultClientId, apiUrl, diadocCrypt);
  authToken = diadocApi.Authenticate(login, password);
 }

②
 public ItemCollection<Unit> ImportUnits() {
  ItemCollection<Unit> units = new ItemCollection<Unit>();

③
  units.AddRange(diadocApi.GetMyOrganizations(authToken).Organizations.Select(GetUnit));
  return units;
 }

④
}
```

① Подключаемся и авторизуемся в *Диадок*.

② Получаем данные подразделений организации.

③ Получаем подразделения организации с помощью метода `GetMyOrganizations` API Диадок метод.

④ Часть кода пропущена.

Разработка компонента сервиса подписей

Компонент сервиса подписей предоставляет модулю зависящий от оператора ЭДО слой подписание отправляемых электронных сообщений.

Данный компонент должен реализовывать программный интерфейс ISignatureService. Методы реализуемого интерфейса вызываются при подписании электронных сообщений, отправляемых оператору ЭДО.

Вариант реализации данного компонента содержится в сборке `DocsVision.Edi.Runtime.BackOffice.dll`.

Далее приведена часть исходного кода стандартной реализации компонента сервиса подписей.

```
public class SignatureService : ISignatureService
{
    public void Initialize(ObjectContext objectContext)
    {
        this.objectContext = objectContext;

        baseCardService = this.objectContext.GetService<IBaseCardService>();
    }

    public Guid CreateCommentSignature(string commentText, X509Certificate2 certificate)
①
    {
        using (MemoryStream stream = new MemoryStream(Encoding.UTF8.GetBytes(
commentText)))
        {
            byte[] signature = baseCardService.ComputeSignature(stream, certificate);
            return objectContext.GetService<UserSession>().AccessManager.StoreSignature
(Guid.NewGuid(), signature, certificate.Thumbprint);
        }
    }

    public void CreateFileSignature(MessageFile messageFile, X509Certificate2
certificate) ②
    {

        if (messageFile.FileData != null && messageFile.FileData.Length > 0) ③
        {
            using (MemoryStream stream = new MemoryStream(messageFile.FileData))
            {
                byte[] signature = baseCardService.ComputeSignature(stream, certificate);
                messageFile.SignatureData = signature;
            }
        }
        else if (!string.IsNullOrEmpty(messageFile.FilePath))
        {
            using (Stream fileStream = new FileStream(messageFile.FilePath, FileMode.
Open))
            {
                byte[] signature = baseCardService.ComputeSignature(fileStream,
certificate);
                messageFile.SignatureData = signature;
            }
        }
    }
```

```
}
```

① Реализация метода подписания текстового комментария.

② Реализация метода подписания файла сообщения.

③ Подписываемые данные могут быть получены непосредственно из файла сообщения (`messageFile.FileData`) или с файловой системы (`messageFile.FilePath`).

Регистрация коннектора к оператору ЭДО

После разработки компонентов коннектора к оператору ЭДО, необходимо добавить в Docsvision нового оператора, который будет использовать данные компоненты.

Чтобы добавить нового оператора ЭДО:

1. Откройте карточку *Справочник настроек операторов ЮЗДО* в программе "Docsvision Explorer" (входит в *Resource Kit*).

Инструкция по работе с утилитой *Docsvision Explorer* приведена в документации *Resource Kit*, раздел "/dv6/resource-kit/6.1/dvexplorer/util/[DVExplorer]".

2. Добавьте в секцию *Операторы* новую строку.
3. В поле *Name*, добавленной строки, введите название оператора, которое будет отображаться при настройке в *Справочнике настроек операторов ЮЗДО*.

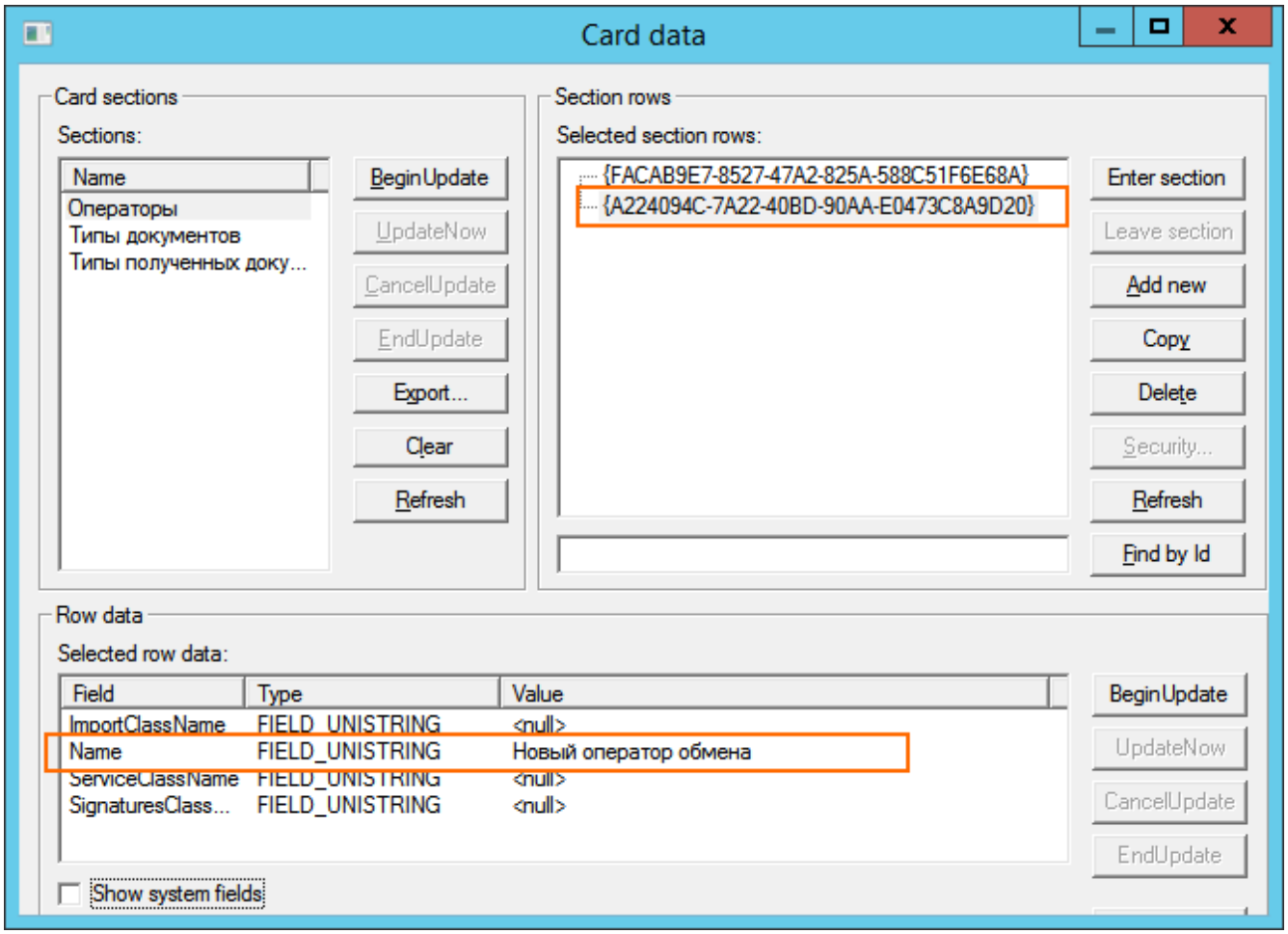


Рисунок 122. Секция "Операторы" карточки "Справочника настроек операторов ЮЗДО", в Docsvision Explorer

Следующие шаги необходимо выполнить, чтобы добавить настраиваемые из *Справочника настроек операторов ЮЗДО* параметры нового оператора ЭДО. В частности, можно добавить настраиваемые параметры соединения с оператором ЭДО.

4. Откройте строку секции добавленного оператора ЭДО.
5. Добавьте в секцию *Настройки* новую строку.
6. В поле *Name*, добавленной строки, введите название настраиваемого параметра. Значение параметра можно будет установить в пользовательском интерфейсе *Справочника настроек операторов ЮЗДО*.

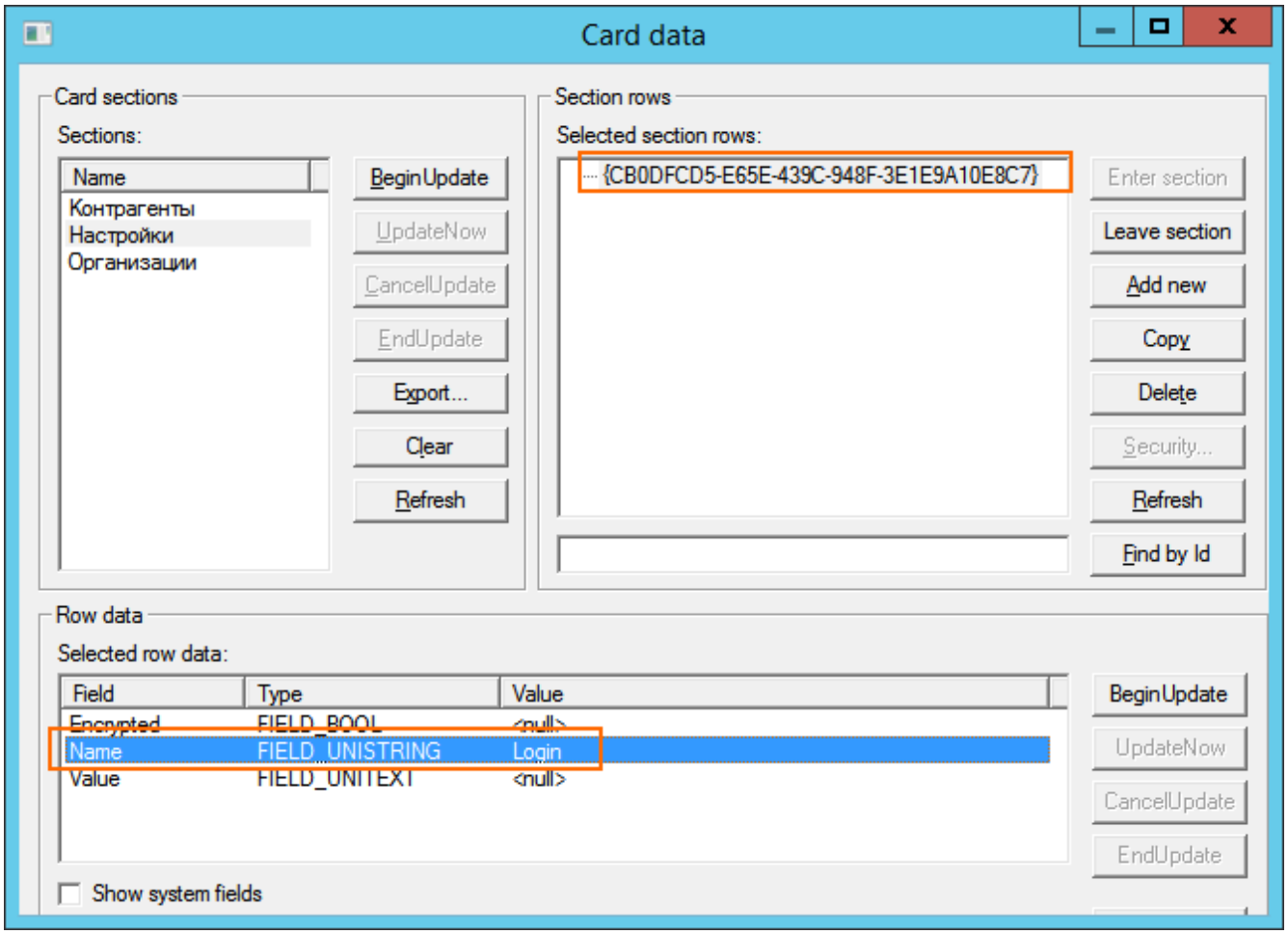


Рисунок 123. Секция "Настройки" карточки "Справочника настроек операторов ЮЗДО", открытая в Docsvision Explorer

7. Повторите шаги 5-6, чтобы добавить все необходимые настраиваемые параметры.

Добавьте разработанные компоненты в конфигурацию оператора ЭДО в *Справочнике настроек операторов ЮЗДО*:

8. Зарегистрируйте разработанные компоненты в GAC на сервере Docsvision.
9. Откройте *Справочнике настроек операторов ЮЗДО* и выберите нового оператора ЭДО.

   Если оператор недоступен, перезагрузите сервер Docsvision.

10. Выберите разработанные компоненты на странице "Компоненты".

Объектная модель модуля

Модуль предоставляет собственную объектную модуль для реализации компонентов и работы с данными карточек модуля.

В данном разделе перечислены основные члены объектной модули модуль.

Box — класс

Содержит данные ящика электронного обмена.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class Box
```

Конструкторы

Имя	Описание
`Box(String, String)`	Инициализирует новый экземпляр класса `Box`. *Параметры:* • `id` — идентификатор ящика. • `name` — название ящика.

Свойства

Имя	Описание
`Id`	Задает или возвращает идентификатор ящика.
`Name`	Задает или возвращает название ящика.

OperatorsOperator — класс

Содержит данные оператора ЮЗДО.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public class OperatorsOperator : BaseCardSectionRow
```

Свойства

Имя	Описание
Name	Задает или возвращает имя **Тип**: string
ServiceClassName	Задает или возвращает имя класса сервиса **Тип**: string
SignaturesClassName	Задает или возвращает имя классов подписей **Тип**: string
ImportClassName	Задает или возвращает имя класса. **Тип**: string
LoaderClassName	Задает или возвращает имя класса загрузившего **Тип**: string
InvitationClassName	Задает или возвращает имя класса приглашения **Тип**: string
OperatorCertificate	Задает или возвращает сертификат оператора **Тип**: bool
Units	Задает или возвращает подразделения **Тип**: OperatorsUnit

Имя	Описание
`Partners`	Задает или возвращает контрагентов **Тип**: `OperatorsPartner`
`Settings`	Задает или возвращает настройки **Тип**: `OperatorsSetting`

Extension — класс

Расширение модуля ЮЗДО для Storage Server.

- **Пространство имен:** `DocsVision.Edi.ServerExtension`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public class Extension : StorageServerExtension
```

Методы

Имя	Описание
`public void ResetServices(Guid serviceId)`	Сбрасывает сервисы
`public void ResetAllServices()`	Сбрасывает все сервисы
`public void InitializeMessageService(Guid serviceId, string serviceClassName, string settingsXml)`	Инициализирует сервис сообщений
`public void InitializeImportService(Guid serviceId, string serviceClassName, string settingsXml)`	Инициализирует сервис импорта
`public void InitializeInvitationService(Guid serviceId, string serviceClassName, string settingsXml)`	Инициализирует сервис приглашений
`public void SendMessage(Guid serviceId, string messageDataXml)`	Отправляет сообщение

Имя	Описание
`public void SendSignatureReply(Guid serviceId, string messageDataXml)`	Отправляет ответные подписи или отказ подписания
`public void SendReceipt(Guid serviceId, string messageDataXml)`	Отправляет квитанцию
`public void SendRevocation(Guid serviceId, string messageDataXml)`	Отправляет отказ
`public void SendCorrection(Guid serviceId, string messageDataXml)`	Отправляет исправление
`public int GetNewEventsCount(Guid serviceId, string boxId, string lastEventId)`	Получить счётчик новых событий
`public string GetLastEventId(Guid serviceId, string boxId)`	Получить идентификатор последнего события ящика
`public bool EventExists(Guid serviceId, string boxId, string eventId)`	Проверяет, существует ли событие ящика
`public string GetNewEvents(Guid serviceId, string boxId, string lastEventId)`	Получить новые события
`public string GetNewEventsFromDate(Guid serviceId, string boxId, string lastEventId, DateTime fromDate)`	Получит новые события с указанной даты
`public string GenerateInvoiceReceipt(Guid serviceId, string boxId, string messageId, string entityId, string certificateBytes, string positionName)`	Генерация формализованной квитанции
`public string GenerateRevocationRequest(Guid serviceId, string boxId, string messageId, string entityId, string certificateBytes, string positionName, string comment)`	Сгенерировать запрос на отказ
`public string GenerateInvoiceReply(Guid serviceId, string boxId, string messageId, string entityId, string certificateBytes, string documentType, string replyData)`	Сгенерировать ответный счёт

Имя	Описание
`public string GenerateSignatureRejection(Guid serviceId, string boxId, string messageId, string entityId, string certificateBytes, string positionName, string comment)`	Сгенерировать запрос подписи
`public string GenerateCorrectionRequest(Guid serviceId, string boxId, string messageId, string entityId, string certificateBytes, string positionName, string comment)`	Сгенерировать запрос исправления
`public string ImportUnits(Guid serviceId)`	Получить список зарегистрированных подразделений
`public string ImportPartners(Guid serviceId)`	Получить список зарегистрированных контрагентов
`public string ImportPartnersFromId(Guid serviceId, string unitId, string lastPartnerId)`	Получить список зарегистрированных контрагентов из идентификатора
`public string ImportPartner(Guid serviceId, string inn, string kpp)`	Получить одного контрагента
`public void SendInvitation(Guid serviceId, string messageDataXml)`	Отправить приглашение
`public void AcceptInvitation(Guid serviceId, string messageDataXml)`	Принять приглашение
`public void RejectInvitation(Guid serviceId, string messageDataXml)`	Отклонить приглашение
`public void CheckInvitationStatus(Guid serviceId, string messageDataXml)`	Проверяет статус приглашения
`public string GetNewInvitations(Guid serviceId, string boxId, string lastEventId)`	Получает новые приглашения

Имя	Описание
`public string GetNewInvitationsFromDate(Guid serviceId, string boxId, string lastEventId, DateTime fromDate)`	Получает новые приглашения с указанной даты
`public string GetLastError(string errorKey)`	Получает последнюю ошибку

BoxesService — класс

`BoxesService` — основной сервис модуля для выполнения всех стандартных операций.

- **Пространство имен:** `DocsVision.Edi.Runtime`
- **Сборка:** `DocsVision.Edi.Runtime.dll`

Синтаксис

```
public class BoxesService
```

Методы

Не рекомендуется самостоятельно вызывать методы обработки очередей сообщений (например, `ProcessAllNewEvents`) без необходимости, т.к. это может привести к конфликтам со штатными БП модуля.

Имя	Описание
`public void ResetExtensionServices(OperatorsOperator)`	Сбрасывает сервисы серверного расширения.
`public bool ImportUnits(OperatorsOperator)`	Импортирует подразделения.
`public bool ImportPartners(OperatorsOperator)`	Импортирует контрагентов.
`public bool ImportPartner(OperatorsOperator, string, string)`	Импортирует одного контрагента по ИНН и КПП.

Имя	Описание
`public bool ImportPartner(OperatorsOperator, PartnersCompany)`	Импортирует одного контрагента по организации контрагента.
`public string GetLastEventId(OperatorsBox)`	Получает идентификатор последнего события.
`public int GetNewEventsCount(OperatorsBox)`	Получает количество новых событий ящика.
`public bool BoxLastEventExists(OperatorsBox)`	Проверяет существование последнего события ящика.
`public bool ProcessAllNewEvents(ILogWriter)`	Находит и обрабатывает все входящие события.
`public int ProcessNewEvents(OperatorsBox, ILogWriter)`	Обрабатывает все события в ящике.
`public bool ProcessAllNewMessagesToSend(ILogWriter)`	Находит и обрабатывает все новые исходящие сообщения.
`public int ProcessNewMessagesToSend(OperatorsOperator, ILogWriter)`	Находит и обрабатывает новые исходящие сообщения для оператора.
`public bool ProcessAllNewTransferMessages(ILogWriter)`	Находит и обрабатывает все новые сообщения о передаче файлов.
`public MessageFile GenerateInvoiceReceipt(OperatorsOperator, string, string, string, X509Certificate2, string)`	Генерирует получение счёт-фактуры. *Параметры:* • ID ящика • ID сообщения • ID сущности • Наименование должности

Имя	Описание
`public MessageFile GenerateRevocationRequest(OperatorsOperator, string, string, string, X509Certificate2, string, string)`	Генерирует запрос на аннулирование. *Параметры:* • ID ящика • ID сообщения • ID сущности • Наименование должности • Комментарий
`public MessageFile GenerateInvoiceReply(OperatorsOperator, string, string, string, X509Certificate2, string, string)`	Генерирует ответный счёт-фактуру *Параметры:* • ID ящика • ID сообщения • ID сущности • Тип документа • Ответные данные
`public MessageFile GenerateSignatureRejection(OperatorsOperator, string, string, string, X509Certificate2, string, string)`	Генерирует запрос на подписание. *Параметры:* • ID ящика • ID сообщения • ID сущности • Наименование должности • Комментарий
`public MessageFile GenerateCorrectionRequest(OperatorsOperator, string, string, string, X509Certificate2, string, string)`	Генерирует запрос на исправление. *Параметры:* • ID ящика • ID сообщения • ID сущности • Наименование должности • Комментарий

Имя	Описание
`public string LoadIncomingDocuments(OperatorsBox, DateTime, DateTime, string, ILogWriter)`	Загружает входящие документы. *Параметры:* • Дата с … • Дата по … • ID последнего документа
`public string LoadOutgoingDocuments(OperatorsBox, DateTime, DateTime, string, ILogWriter)`	Загружает исходящие документы. *Параметры:* • Дата с … • Дата по … • ID последнего документа
`public bool ProcessAllNewInvitations(ILogWriter)`	Находит и обрабатывает все новые входящие приглашения.
`public bool ProcessAllNewInvitationsToSend(ILogWriter)`	Находит и обрабатывает все новые исходящие приглашения.
`public DateTime? CheckInvitationsStatus(ILogWriter, DateTime?)`	Проверяет статус приглашений с указанной даты.
`public bool ProcessAllNewPowersOfAttorneyToSend(ILogWriter logWriter)`	Находит и обрабатывает все новые исходящие доверенности.
`public DateTime? CheckPowersOfAttorneyStatus(ILogWriter logWriter, DateTime? dateFrom)`	Проверяет статус доверенности

DataReaderService — класс

Клас выполняет чтение данных карточек, обычно для чтения получателей и подписанных файлов.

Класс вызывает экземпляры `IDataReader` по привязке к виду или типу карточки. Можно создать свой `IDataReader` с нуля или на основе существующего и прописать его в Справочнике настроек операторов ЮЗДО.

`DataReaderService` можно использовать в своих компонентах или скриптах для получения данных при отправке.

- **Пространство имен:** `DocsVision.Edi.Runtime`
- **Сборка:** `DocsVision.Edi.Runtime.dll`

Синтаксис

```
public class DataReaderService
```

Методы

Имя	Описание
`public string GetDigest(Guid)`	Получает дайджест карточки по её ID
`public string GetFileDigest(Guid, Guid)`	Получает дайджест файла документа по ID карточки и ID файла
`public string GetFileContent(Guid, Guid)`	Получает содержимое файла документа по ID карточки и ID файла
`public string GetPrintFormContent(Guid)`	Получает печатную форму документа из содержимого по ID карточки
`public string GetReplyFileDigest(Guid, Guid, MessageFileType)`	Получает дайджест ответного файла по ID карточки и ID файла
`public string GetReplyFileComment(Guid, Guid, MessageFileType)`	Получает комментарий к ответному файлу по ID карточки и ID файла
`public ItemCollection<DocumentRecipientData> GetAllRecipients(Guid)`	Получает всех получателей для документа по ID карточки
`public ItemCollection<DocumentSignature> GetActualSignatures(Guid)`	Получает действующие подписи документа по ID карточки
`public ItemCollection<DocumentFileData> GetAllFiles(Guid)`	Получает все файлы для документа по ID карточки
`public ItemCollection<FileValidationResult> ValidateFiles(Guid)`	Проверяет структуру файлов, например, действительно ли файл является счёт-фактурой в формате XML по ID карточки

Имя	Описание
`public ItemCollection<DocumentFileData> GetLastSignedFiles(Guid)`	Получает последние подписанные файлы для документа по ID карточки
`public ItemCollection<DocumentFileData> GetSignedFiles(Guid, Guid)`	Получает подписанные файлы для документа по ID карточки и ID группы подписи
`public ItemCollection<Guid> GetLinkedCards(Guid)`	Получает ссылочные файлы для документа по ID карточки
`public MessageData PrepareMessageData(Guid)`	Подготавливает данные сообщения для документа по ID карточки
`public MessageFile PrepareFileData(Guid, Guid, Guid, bool, string)`	Подготавливает данные файла для документа по ID карточки, основного файла, подписи и временной папки.
`public SignatureValidation ValidateSignature(Guid, Guid, Guid)`	Проверяет действительность подписи ID: карточки, файла, подписи
`public string GetInvoiceReplyData(Guid)`	Получает ответный счёт-фактуру на документ по ID карточки

Department — класс

Содержит данные подразделения организации или контрагента, полученные от оператора ЭДО.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class Department
```

Конструкторы

Имя	Описание
`Department(String, String, String)`	Инициализирует новый экземпляр класса `Department`. *Параметры:* • `id` — идентификатор подразделения у оператора ЭДО • `name` — название подразделения у оператора ЭДО • `kpp` — КПП подразделения у оператора ЭДО

Свойства

Имя	Описание
`Departments`	Возвращает коллекцию подчиненных подразделений. **Тип**: `ItemCollection<Department>`
`Id`	Задает или возвращает идентификатор подразделения в системе оператора ЭДО. **Тип**: `System.String`
Kpp	Задает или возвращает КПП подразделения в системе оператора ЭДО. **Тип**: `System.String`
Name	Задает или возвращает название подразделения в системе оператора ЭДО. **Тип**: `System.String`

DocumentFileData — класс

Содержит данные файла карточки *Документ*.

- **Пространство имен:** DocsVision.Edi.Entities
- **Сборка:** DocsVision.Edi.Interfaces.dll

Синтаксис

```
public class DocumentFileData
```

Конструкторы

Имя	Описание
DocumentFileData(Guid, String)	Инициализирует новый экземпляр класса DocumentFileData. *Параметры:* • id — идентификатор карточки файла. • name — название файла.
DocumentFileData(Guid, String, Boolean)	Инициализирует новый экземпляр класса DocumentFileData. *Параметры:* • id — идентификатор карточки файла. • name — название файла. • isMain — флаг, определяющий, является ли файл основным.

Имя	Описание
`DocumentFileData(Guid, String, Boolean, Guid)`	Инициализирует новый экземпляр класса `DocumentFileData`. *Параметры:* • `id` — идентификатор карточки файла. • `name` — название файла. • `isMain` — флаг, определяющий, является ли файл основным. • `signatureId` — идентификатор подписи файла.
`DocumentFileData(Guid, String, Boolean, Guid, FileContentType)`	Инициализирует новый экземпляр класса `DocumentFileData`. *Параметры:* • `id` — идентификатор карточки файла. • `name` — название файла. • `isMain` — флаг, определяющий, является ли файл основным. • `signatureId` — идентификатор подписи файла. • `fileContentType` — тип содержимого файла.

Свойства

Имя	Описание
`FileContentType`	Задает или возвращает тип содержимого файла. **Тип**: `FileContentType`

Имя	Описание
`Id`	Возвращает идентификатор карточки версионного файла. **Тип**: `System.Guid`
`IsMain`	Возвращает флаг, определяющий, является ли файл основным: `true` — основной, `false` — дополнительный **Тип**: `Boolean`
`Name`	Возвращает название файла. **Тип**: `System.String`
`SignatureId`	Задает или возвращает идентификатор подписи файла. **Тип**: `System.Guid`

DocumentRecipientData — класс

Содержит данные получателя документа.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class DocumentRecipientData
```

Конструкторы

Имя	Описание
`DocumentRecipientData(Guid, String)`	Инициализирует новый экземпляр класса `DocumentRecipientData`. *Параметры:* • `id` — идентификатор получателя. • `name` — название получателя.

Свойства

Имя	Описание
`Id`	Возвращает идентификатор получателя — идентификатор контрагента в Справочнике контрагентов. **Тип**: `System.Guid`
`Name`	Возвращает название получателя — название котрагента. **Тип**: `System.String`

DocumentSignature — класс

Содержит данные подписи документа.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class DocumentSignature
```

Конструкторы

Имя	Описание
`DocumentSignature(Guid, DateTime, String, String, X509Certificate2)`	Инициализирует новый экземпляр класса `DocumentSignature`. *Параметры:* • `id` — идентификатор подписи в Docsvision. • `date` — штамп времени подписи. • `signerName` — отображаемое имя подписавшего. • `label` — локализованное название метки подписи. • `certificate` — сертификат подписи.

Свойства

Имя	Описание
`Certificate`	Возвращает сертификат подписи. **Тип**: `X509Certificate2`
`Date`	Возвращает штамп времени подписи. **Тип**: `DateTime`
`Id`	Возвращает идентификатор подписи в Docsvision. **Тип**: `System.Guid`
`Label`	Возвращает локализованное название метки подписи. **Тип**: `System.String`
`SignerName`	Возвращает отображаемое имя подписавшего. **Тип**: `System.String`

DocumentUpdaterService — класс

Класс запись данных в документ, обычно при получении ответов. Вызывает экземпляры `IDocumentUpdater` по привязке к виду или типу карточки.

Можно создать собственный свой `IDocumentUpdater` с нуля или на основе нашего и прописать его в Справочнике настроек операторов ЮЗДО. Тут для своих скриптов или компонентов можно использовать метод CreateNewSignatures для создания новых подписей в документе, будет делать новую подпись с учетом вида документа, для примера, для неформализованного это будет просто подпись файла, для формализованного - генерация ответного титула и уже его подпись.

- **Пространство имен:** `DocsVision.Edi.Runtime`
- **Сборка:** `DocsVision.Edi.Runtime.dll`

Синтаксис

```
public class DocumentUpdaterService
```

Методы

Имя	Описание
`public Guid AddReplySignatureToDocument(Guid cardId, Guid fileId, MessageFile messageFile, string partnerName)`	Добавляет к документу ответную подпись по ID карты и ID файла
`public Guid CreateNewSignatures(Guid cardId, X509Certificate2 certificate, bool signAdditionalFiles)`	Создаёт новые подписи по ID карты
`public string CheckCertificate(Guid cardId, X509Certificate2 certificate)`	Проверяет сертификат для карточки по ID карты
`public bool ChangeDocumentState(Guid cardId, Guid newStateId)`	Изменяет статус документа по ID карты и ID нового статуса
`public Guid GetNewStateId(Guid cardId, MessageFileType messageFileType)`	Получает ID нового статуса по ID карты

Имя	Описание
`public void UpdateDocumentDataFromFile(Guid cardId, Guid fileId)`	Обновляет данные документа из файла по ID карты и ID файла
`public bool ChangePowerOfAttorneyState(Guid cardId, Guid powerOfAttorneyCardId, MessageFileType messageFileType)`	Изменяет статус МЧД

EdiMessage — класс

Содержит данные *Карточки обмена сообщениями*.

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public class EdiMessage : BaseCard
```

Свойства

Имя	Описание
`Files`	Возвращает файлы сообщения. **Тип**: `ObjectCollection<EdiMessageFile>`
`Log`	Возвращает записи их журнал обмена. **Тип**: `ObjectCollection<EdiMessageLogItem>`
`MainInfo`	Возвращает основную информацию. **Тип**: EdiMessageMainInfo
Processes	*Не используется.*
`SignedComments`	Подписанные комментарии. **Тип**: `ObjectCollection<EdiMessageSignedComment>`

Имя	Описание
SignedFiles	Подписанные файлы. **Тип**: ObjectCollection<EdiMessageSignedFile>
SystemInfo	*Не используется.*
TransferData	Данные для переноса. ObjectCollection<EdiMessageTransferData>

Поля

Имя	Описание
FilesProperty	Определяет свойство *Файлы*.
LogProperty	Определяет свойство ""*Журнал обмена*"".
MainInfoProperty	Определяет свойство *Основная информация*.
SignedCommentsProperty	Определяет свойство *Подписанные комментарии*.
SignedFilesProperty	Определяет свойство *Подписанные файлы*.
TransferDataProperty	Определяет свойство *Файлы для переноса*.

EdiMessageFile — класс

Содержит данные файла из *Карточки обмена сообщениями*.

- **Пространство имен:** DocsVision.Edi.ObjectModel
- **Сборка:** DocsVision.Edi.ObjectModel.dll

Синтаксис

```
public class EdiMessageFile : BaseCardSectionRow
```

Свойства

Имя	Описание
ContentType	Задает или возвращает тип содержимого файла. **Тип**: EdiMessageFileContentType
FileId	Задает или возвращает идентификатор файла. **Тип**: System.Guid
FileName	Задает или возвращает название файла. **Тип**: System.String
FileState	Задает или возвращает статус файла. **Тип**: EdiMessageFileState
FileType	Задает или возвращает тип файла. **Тип**: EdiMessageFileType
OperatorFileId	Задает или возвращает идентификатор файла у оператора ЭДО. **Тип**: System.String
OperatorSignatureId	Задает или возвращает идентификатор подписи у оператора ЭДО. **Тип**: System.String
RejectionComment	Задает или возвращает комментарий к файлу, если пользователь отказывает в подписании. **Тип**: EdiMessageSignedComment
ReplyFiles	Возвращает **Тип**: EdiMessageReplyFile

Имя	Описание
`ReplySignatureId`	Задает или возвращает идентификатор ответной подписи. **Тип**: `System.Guid`
`SignatureId`	Задает или возвращает идентификатор подписи. **Тип**: `System.Guid`

Поля

Имя	Описание
`ContentTypeProperty`	Определяет свойство *Тип содержимого*.
`FileIdProperty`	Определяет свойство *Файл*.
`FileNameProperty`	Определяет свойство *Имя файла*.
`FileStateProperty`	Определяет свойство *Состояние файла*.
`FileTypeProperty`	Определяет свойство *Тип файла*.
`OperatorFileIdProperty`	Определяет свойство *Идентификатор файла у оператора*.
`OperatorSignatureIdProperty`	Определяет свойство *Идентификатор подписи у оператора*.
`RejectionCommentProperty`	Определяет свойство *Комментарий при отказе от подписи*.
`ReplyFilesProperty`	Определяет свойство *Ответные файлы*.
`ReplySignatureIdProperty`	Определяет свойство *Ответная подпись*.
`SignatureIdProperty`	Определяет свойство *Подпись*.

EdiMessageLogItem — класс

Содержит данные записи в журнале из *Карточки обмена сообщениями*.

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public class EdiMessageLogItem : BaseCardSectionRow
```

Свойства

Имя	Описание
Comment	Задает или возвращает текст комментария. **Тип**: System.String
Date	Задает или возвращает дату записи. **Тип**: System.DateTime
Description	Задает или возвращает текст сообщения. **Тип**: System.String
EventType	Задает или возвращает тип события. **Тип**: EdiMessageEventType
FileName	Задает или возвращает название файла, с которым связано событие. **Тип**: System.String
SignedFile	Задает или возвращает подписанный файл. **Тип**: EdiMessageSignedFile

Поля

Имя	Описание
CommentProperty	Определяет свойство *Комментарий*.
DateProperty	Определяет свойство *Дата*.
DescriptionProperty	Определяет свойство *Описание*.
EventTypeProperty	Определяет свойство *Тип события*.

Имя	Описание
`FileNameProperty`	Определяет свойство *Имя файла*.
`SignedFileProperty`	Определяет свойство *Подписанный файл*.

EdiMessageMainInfo — класс

Содержит основную информацию *Карточки обмена сообщениями*.

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public class EdiMessageMainInfo : BaseCardSectionRow
```

Свойства

Имя	Описание
`CardId`	Задает или возвращает идентификатор карточки, с которой связано сообщение. **Тип**: `System.Guid`
`MessageState`	Задает или возвращает состояние сообщения. **Тип**: `EdiMessageState`
`OperatorMessageId`	Задает или возвращает идентификатор сообщения у оператора. **Тип**: `System.String`
`OrganizationBox`	Задает или возвращает ящик организации. **Тип**: `OperatorsBox`

Имя	Описание
`OrganizationDepartment`	Задает или возвращает подразделение организации. **Тип**: `OperatorsDepartment`
`Partner`	Задает или возвращает контрагента. **Тип**: `PartnersCompany`
`PartnerBox`	Задает или возвращает ящик контрагента. **Тип**: `OperatorsPartnerBox`
`PartnerDepartment`	Задает или возвращает подразделение контрагента. **Тип**: `OperatorsPartnerDepartment`

Поля

Имя	Описание
`CardIdProperty`	Определяет свойство *Карточка*.
`MessageStateProperty`	Определяет свойство *Состояние*.
`OperatorMessageIdProperty`	Определяет свойство *Идентификатор сообщения у оператора*.
`OrganizationBoxProperty`	Определяет свойство *Ящик организации*.
`OrganizationDepartmentProperty`	Определяет свойство *Подразделение организации*.
`PartnerBoxProperty`	Определяет свойство *Ящик контрагента*.
`PartnerDepartmentProperty`	Определяет свойство *Подразделение контрагента*.
`PartnerProperty`	Определяет свойство *Контрагент*.

EdiMessageReplyFile — класс

Содержит данные ответного файла, полученного от контрагента, в карточке обмена сообщениями.

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public class EdiMessageReplyFile : BaseCardSectionRow
```

Свойства

Имя	Описание
`NeedReceipt`	Задает или возвращает флаг, указывающий на необходимость подтверждения получения. **Тип**: `System.Boolean`
`ParentMessageFile`	Возвращает ссылку на родительское сообщение электронного обмена. **Тип**: `EdiMessageFile`
`ReplyFile`	Задает или возвращает ответный файл. **Тип**: `EdiMessageSignedFile`
`ReplyFileType`	Задает или возвращает тип ответного файла. **Тип**: `EdiMessageReplyFileType`
`ReplyReceipt`	Задает или возвращает ссылку на ответный файл (квитанцию). **Тип**: `EdiMessageSignedFile`

Поля

Имя	Описание
`NeedReceiptProperty`	Определяет свойство *Требуется извещение*.
`ReplyFileProperty`	Определяет свойство *Ответный файл*.
`ReplyFileTypeProperty`	Определяет свойство *Тип ответного файла*.
`ReplyReceiptProperty`	Определяет свойство *Ответная квитанция*.

EdiMessageSignedComment — класс

Содержит данные подписанного сообщения из *Карточки обмена сообщениями*.

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public class EdiMessageSignedComment : BaseCardSectionRow
```

Свойства

Имя	Описание
`Comment`	Задает или возвращает содержимое комментария.
`SignatureId`	Задает или возвращает идентификатор подписи, которой подписан комментарий.

Поля

Имя	Описание
`CommentProperty`	Определяет свойство *Комментарий*.
`SignatureIdProperty`	Определяет свойство *Подпись*.

EdiMessageSignedFile — класс

Содержит данные подписанного файла в карточке обмена сообщениями.

- **Пространство имен:** DocsVision.Edi.ObjectModel
- **Сборка:** DocsVision.Edi.ObjectModel.dll

Синтаксис

```
public class EdiMessageSignedFile : BaseCardSectionRow
```

Свойства

Имя	Описание
FileId	Задает или возвращает идентификатор файла в Docsvision.
FileName	Задает или возвращает название файла.
OperatorFileId	Задает или возвращает идентификатор файла у оператора ЭДО.
OperatorSignatureId	Задает или возвращает идентификатор подписи файла у оператора ЭДО.
SignatureId	Задает или возвращает идентификатор подписи файла в Docsvision.

Поля

Имя	Описание
FileIdProperty	Определяет свойство *Файл*.
FileNameProperty	Определяет свойство *Имя файла*.
OperatorFileIdProperty	Определяет свойство *Идентификатор файла у оператора*.
OperatorSignatureIdProperty	Определяет свойство *Идентификатор подписи у оператора*.
SignatureIdProperty	Определяет свойство *Подпись*.

EdiMessageTransferData — класс

Содержит полученные от контрагента файлы, которые должны быть загружены в карточку УПД, а также новое состояние для карточки УПД.

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public class EdiMessageTransferData : BaseCardSectionRow
```

Свойства

Имя	Описание
`Date`	Задает или возвращает дату получения данных от контрагента.
`NewStateId`	Задает или возвращает новое состояние для карточки УПД.
`TransferFiles`	Задает или возвращает список файлов, которые должны быть загружены в карточку УПД.

Поля

Имя	Описание
`DateProperty`	Определяет свойство "Дата".
`NewStateIdProperty`	Определяет свойство "Новое состояние карточки".
`TransferFilesProperty`	Определяет свойство "Файлы для переноса".

EdiPowerOfAttorney — класс

Машиночитаемая доверенность ЮЗДО.

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public class EdiPowerOfAttorney : BaseCard
```

Свойства

Имя	Описание
MainInfo	Задает или возвращает основную информацию о доверенности. **Тип**: EdiPowerOfAttorneyMainInfo
Log	Задает или возвращает запись журнала МЧД **Тип**: EdiPowerOfAttorneyLogItem
PowerOfAttorneyData	Задает или возвращает данные МЧД **Тип**: EdiPowerOfAttorneyData

EdiPowerOfAttorneyData — класс

Данные машиночитаемой доверенности ЮЗДО.

- **Пространство имен:** DocsVision.Edi.ObjectModel
- **Сборка:** DocsVision.Edi.ObjectModel.dll

Синтаксис

```
public class EdiPowerOfAttorneyData : BaseCardSectionRow
```

Свойства

Имя	Описание
RegistrationNumber	Задает или возвращает регистрационный номер доверенности. **Тип**: Guid

Имя	Описание
`IssuerInn`	Задает или возвращает ИНН доверителя **Тип**: `string`
`RepresentativeInn`	ИНН доверенного лица **Тип**: `string`
`IsDefault`	Задает или возвращает признак использования по умолчанию **Тип**: `bool`

EdiPowerOfAttorneyEventType — перечисление

Тип события МЧД

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public enum EdiPowerOfAttorneyEventType
```

Члены

Имя	Описание
`Information = 0,`	Информация
`Error = 1,`	Ошибка
`Reply = 2,`	Ответ
`FromDocsvision = 3,`	От Docsvision
`FromOperator = 5`	От оператора ЮЗДО

EdiPowerOfAttorneyLogItem — класс

Запись в журнале МЧД

- **Пространство имен:** DocsVision.Edi.ObjectModel
- **Сборка:** DocsVision.Edi.ObjectModel.dll

Синтаксис

```
public class EdiPowerOfAttorneyLogItem : BaseCardSectionRow
```

Свойства

Имя	Описание
EventType	Задает или возвращает тип события **Тип**: EdiPowerOfAttorneyEventType
Date	Задает или возвращает дату события **Тип**: DateTime
Description	Задает или возвращает описание события **Тип**: string
Comment	Задает или возвращает комментарий к событию **Тип**: string
FileName	Задает или возвращает имя файла **Тип**: string

EdiPowerOfAttorneyMainInfo — класс

Основная информация о доверенности

- **Пространство имен:** DocsVision.Edi.ObjectModel
- **Сборка:** DocsVision.Edi.ObjectModel.dll

Синтаксис

```
public class EdiPowerOfAttorneyMainInfo : BaseCardSectionRow
```

Свойства

Имя	Описание
CardId	Задает или возвращает идентификатор карточки **Тип**: Guid
MessageState	Задает или возвращает состояние сообщения. **Тип**: EdiPowerOfAttorneyState
OrganizationBox	Задает или возвращает ящик организации **Тип**: OperatorsBox
OrganizationDepartment	Задает или возвращает подразделение организации **Тип**: OperatorsDepartment
OperatorMessageId	Задает или возвращает идентификатор сообщения оператора ЮЗДО **Тип**: string
DocumentType	Задает или возвращает тип документа **Тип**: string
DocumentVersion	Задает или возвращает версию документа **Тип**: string

Имя	Описание
`IncomingMessage`	Задает или возвращает входящее сообщение оператора ЮЗДО **Тип**: `bool`
`IsRecalled`	Задает или возвращает признак отозванности **Тип**: `bool`
`PowerOfAttorneyCardId`	Задает или возвращает идентификатор карточки доверенности **Тип**: `Guid`
`SendAsFile`	Отправляет МЧД как файл. **Тип**: `bool`
`OperatorFileId`	Задает или возвращает идентификатор файла у оператора ЭДО **Тип**: `string`
`OperatorSignatureId`	Задает или возвращает идентификатор подписи оператора ЮЗДО **Тип**: `string`

EdiPowerOfAttorneyState — перечисление

Состояние доверенности

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public enum EdiPowerOfAttorneyState
```

Свойства

Имя	Описание
`Creating = 0`	Создание
`ToSend = 1`	Готово к отправке
`SentToOperator = 2`	Отправлена оператору
`Registered = 3`	Зарегистрирована
`AssignedToEmployee = 4`	Передана сотруднику
`Error = 5`	Ошибка
`Revoked = 6`	Отозвана

MessageAddress — класс

Содержит данные отправителя или получателя сообщения электронного обмена.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class MessageAddress
```

Конструкторы

Имя	Описание
`MessageAddress(String)`	Инициализирует новый экземпляр класса `MessageAddress`. *Параметры:* • `boxId` — идентификатор ящика электронного обмена

Имя	Описание
`MessageAddress(String, String)`	Инициализирует новый экземпляр класса `MessageAddress`. *Параметры:* • `boxId` — идентификатор ящика электронного обмена • `name` — название ящика электронного обмена

Свойства

Имя	Описание
`BoxId`	Задает или возвращает идентификатор ящика электронного обмена. **Тип**: `System.String`
`DepartmentId`	Задает или возвращает идентификатор подразделения отправителя или получателя. **Тип**: `System.String`
`Name`	Задает или возвращает название отправителя или получателя. **Тип**: `System.String`

MessageData — класс

Содержит данные сообщения электронного обмена с оператором ЭДО.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class MessageData
```

Конструкторы

Имя	Описание
`MessageData(Guid)`	Инициализирует новый экземпляр класса `MessageData`. *Параметры:* • `cardId` — идентификатор карточки
`MessageData(Guid, String)`	Инициализирует новый экземпляр класса `MessageData`. *Параметры:* • `cardId` — идентификатор карточки • `documentName` — название документа
`MessageData(String, String, Boolean)`	Инициализирует новый экземпляр класса `MessageData` по данным входящего сообщения. *Параметры:* • `operatorEventId` — идентификатор события у оператора ЭДО • `operatorMessageId` — идентификатор сообщения у оператора ЭДО • `isPatch` — является дополнением к имеющемуся сообщению

Свойства

Имя	Описание
`AditonalAttributes`	Возвращает коллекцию дополнительных атрибутов. **Тип**: `ItemCollection<MessageAditonalAttribute>`

Имя	Описание
`CardId`	Задает или возвращает идентификатор карточки, по которой получено сообщение. **Тип**: `System.Guid`
`DocumentName`	Задает или возвращает название документа. **Тип**: `System.String`
`DocumentNumber`	Задает или возвращает регистрационный номер документа. **Тип**: `System.String`
`DocumentDate`	Задает или возвращает дату регистрации или дату создания документа. **Тип**: `DateTime?`
`DocumentComment`	Задает или возвращает содержание документа. **Тип**: `System.String`
`OperatorMessageId`	Задает или возвращает идентификатор сообщения у оператора ЭДО. **Тип**: `System.String`
`OperatorEventId`	Задает или возвращает идентификатор события у оператора ЭДО. **Тип**: `System.String`
`OperatorDate`	Задает или возвращает штамп времени оператора ЭДО. **Тип**: `DateTime`

Имя	Описание
`IsPatch`	Задает или возвращает флаг, указывающий является ли данное сообщение дополнением к имеющемуся сообщению: `true` — является, `false` — не является. **Тип**: `Boolean`
`DocumentFiles`	Возвращает коллекцию файлов сообщения. **Тип**: `ItemCollection<MessageFile>`
`Sender`	Задает или возвращает данные отправителя сообщения. **Тип**: `MessageAddress`
`Recipient`	Задает или возвращает данные получателя сообщения. **Тип**: `MessageAddress`
`InvitationPartnerData`	Задает или возвращает данные о приглашении партнёра к обмену. **Тип**: `InvitationPartnerData`
`PowerOfAttorneyData`	Задает или возвращает данные МЧД. **Тип**: `PowerOfAttorneyData`
`AdditionalAttributes`	Задает или возвращает дополнительные атрибуты сообщения. **Тип**: `MessageAdditionalAttribute`

MessageFile — класс

Содержит данные файла электронного обмена с оператором ЭДО.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class MessageFile
```

Конструкторы

Имя	Описание
`MessageFile(Guid, String)`	Инициализирует новый экземпляр класса `MessageFile`. *Параметры:* • `fileId` — идентификатор файла. • `fileName` — название файла.
`MessageFile(String, String, String)`	Инициализирует новый экземпляр класса `MessageFile` с указанием идентификатора файла у оператора ЭДО, названием файла и идентификатором подписи у оператора ЭДО. *Параметры:* • `operatorFileId` — идентификатор файла у оператора ЭДО. • `fileName` — название файла. • `operatorSignatureId` — идентификатор подписи файла у оператора ЭДО.

Имя	Описание
`MessageFile(MessageFileType, String, String, String)`	Инициализирует новый экземпляр класса `MessageFile` с указанием типа сообщения, идентификатора файла у оператора ЭДО, названием файла и идентификатором подписи у оператора ЭДО. *Параметры:* • `fileType` — тип содержимого. • `operatorFileId` — идентификатор файла у оператора ЭДО. • `fileName` — название файла. • `operatorParentEntityId` — идентификатор сущности, к которой относится файл у оператора ЭДО.

Свойства

Имя	Описание
`FileComment`	Задает или возвращает комментарий к файлу. **Тип**: `System.String`
`FileData`	Задает или возвращает бинарные данные файла. **Тип**: `Byte`
`FileId`	Задает или возвращает идентификатор файла в Docsvision. **Тип**: `System.Guid`
`FileName`	Задает или возвращает название файла. **Тип**: `System.String`

Имя	Описание
`FilePath`	Задает или возвращает путь, по которому выгружен файл. **Тип**: `System.String`
`FileType`	Задает или возвращает тип содержимого. **Тип**: `MessageFileType`
`OperatorFileId`	Задает или возвращает идентификатор файла у оператора ЭДО. **Тип**: `System.String`
`OperatorParentEntityId`	Задает или возвращает идентификатор сущности, к которой относится файл у оператора ЭДО. **Тип**: `System.String`
`OperatorSignatureId`	Задает или возвращает идентификатор подписи у оператора ЭДО. **Тип**: `System.String`
`SignatureData`	Задает или возвращает бинарные данные подписи файла. **Тип**: `Byte`
`SignatureFilePath`	Задает или возвращает путь, по которому выгружена подпись файла. **Тип**: `System.String`

MessageFileType — перечисление

Определяет возможные типы содержимого в `MessageFile`.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public enum MessageFileType
```

Члены

Имя члена	Описание
`File`	Файл.
`SignatureConfirmation`	Ответная подпись.
`ReplySignature`	Подпись файла.
`DeliveryFailureReport`	Не удалось доставить.
`FileComment`	Текстовый комментарий.
`SignatureRejection`	Отказ в формировании запрошенной подписи.
`Invoice`	Извещение.
`InvoiceConfirmation`	Подтверждение получения УПД.
`InvoiceReceipt`	Извещение о получении УПД.
`InvoiceReply`	Ответный титул УПД.
`InvoiceCorrectionRequest`	Запрос на уточнение УПД.

Partner — класс

Содержит данные контрагента, полученные от оператора ЭДО.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class Partner
```

Конструкторы

Имя	Описание
`Partner(String, String, String)`	Инициализирует новый экземпляр класса Partner. *Параметры:* • `id` — идентификатор контрагента у оператора ЭДО. • `name` — название контрагента у оператора ЭДО. • `inn` — ИНН контрагента у оператора ЭДО.

Свойства

Имя	Описание
`Departments`	Возвращает коллекцию подразделений контрагента. **Тип**: `ItemCollection<Department>`
`Id`	Задает или возвращает идентификатор контрагента в системе оператора ЭДО. **Тип**: `System.String`
Inn	Задает или возвращает ИНН контрагента в системе оператора ЭДО. **Тип**: `System.String`
`LinkedBoxes`	Возвращает коллекцию связанных ящиков контрагента. **Тип**: `ItemCollection<Box>`
`Name`	Задает или возвращает название контрагента в системе оператора ЭДО. **Тип**: `System.String`

Имя	Описание
`PartnerBoxes`	Возвращает коллекцию ящиков контрагента. **Тип**: `ItemCollection<Box>`

SignatureValidation — класс

Содержит данные с результатами проверки электронной подписи.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class SignatureValidation
```

Конструкторы

Имя	Описание
`SignatureValidation(Boolean, Boolean, X509Certificate2, String, String)`	Инициализирует новый экземпляр класса `SignatureValidation` с использованием переданных данных.

Свойства

Имя	Описание
`Certificate`	Возвращает сертификат подписи, с помощью которого выполнялась проверка.
`IsCertificateValid`	Возвращает флаг, указывающий что сертификат является действительным.
`IsValid`	Возвращает флаг, указывающий что подпись является действительной.
`SignerName`	Возвращает имя владельца сертификата.

Имя	Описание
`ValidationError`	Возвращает сообщение об ошибке, если она была при проверке подписи.

Unit — класс

Содержит данные организации, полученные от оператора ЭДО.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class Unit
```

Конструкторы

Имя	Описание
`Unit(String, String, String)`	Инициализирует новый экземпляр класса `Unit`. *Параметры:* • `id` — идентификатор организации у оператора ЭДО. • `name` — название организации у оператора ЭДО. • `inn` — ИНН организации у оператора ЭДО.

Свойства

Имя	Описание
`Boxes`	Возвращает коллекцию ящиков организации. **Тип**: `ItemCollection<Box>`

Имя	Описание
`Departments`	Возвращает коллекцию подразделений организации. **Тип**: `ItemCollection<Department>`
`Id`	Задает или возвращает идентификатор организации в системе оператора ЭДО. **Тип**: `System.String`
`Inn`	Задает или возвращает идентификатор контрагента в системе оператора ЭДО. **Тип**: `System.String`
Name	Задает или возвращает название организации в системе оператора ЭДО. **Тип**: `System.String`

IDocumentCreator — интерфейс

Интерфейс `IDocumentCreator` определяет методы, которые должен реализовывать компонент создания документов.

- **Пространство имен:** `DocsVision.Edi.Interfaces`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public interface IDocumentCreator
```

Свойства

Имя	Описание
`AddReplyFilesToDocument(MessageData, Guid)`	Переносит файлы из электронного сообщения в целевую карточку. *Параметры:* • `messageData` — электронное сообщение, полученное от контрагента • `partnerId` — идентификатор контрагента (в Справочнике сотрудников), от которого получено сообщение В стандартной конфигурации модуля метод вызывается бизнес-процессом "Перенос файлов из сообщений ЮЗДО".

Имя	Описание
`CreateDocument(MessageData, Guid, Guid)`	Создаёт новую карточку по полученному от оператору ЭДО сообщению. Если карточка успешно создана, возвращает `True`. Тип возвращаемого значения: `Boolean` *Параметры:* • `messageData` — сообщение от оператора ЭДО с типом MessageData. • `partnerId` — идентификатор отправившего электронный документ контрагента в системе Docsvision. • `partnerDepartmentId` — идентификатор подразделения контрагента, если электронный документ отправлен подразделением контрагента. В стандартной конфигурации модуля метод вызывается, когда поступил новый электронный документ.
`Initialize(ObjectContext, Guid)`	Инициализирует экземпляр компонента создания документов. *Параметры:* • `objectContext` — контекст объектов. • `defaultKindId` — идентификатор вида создаваемого документа.

IDocumentDataReader — интерфейс

Интерфейс `IDocumentDataReader` определяет методы, которые должен реализовывать компонент чтения данных карточки.

- **Пространство имен:** `DocsVision.Edi.Interfaces`

- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public interface IDocumentDataReader
```

Методы

Имя	Описание
`GetActualSignatures(Guid)`	Возвращает актуальные подписи (действительные с действующим сертификатом) карточки. Тип возвращаемого значения: `ItemCollection<DocumentSignature>` *Параметры:* • `cardId` — идентификатор карточки, подписи которой возвращаются. В стандартной конфигурации модуля метод вызывается при открытии формы отправки электронного документа.
`GetAllFiles(Guid)`	Возвращает основные и дополнительные файлы карточки. Тип возвращаемого значения: `ItemCollection<DocumentFileData>` *Параметры:* • `cardId` — идентификатор карточки, файлы которой возвращаются. В стандартной конфигурации модуля метод вызывается при отправке электронного документа.

Имя	Описание
`GetAllRecipients(Guid)`	Возвращает список получателей электронного документа. Тип возвращаемого значения: `ItemCollection<DocumentRecipientData>`. *Параметры:* • `cardId` — идентификатор карточки, из которой запрашиваются получатели. В стандартной конфигурации модуля метод вызывается при отправке электронного документа.
`GetDigest(Guid)`	Предоставляет описание (дайджест) карточки. **Тип возвращаемого значения**: `String`. *Параметры:* • `cardId` — идентификатор карточки. В стандартной конфигурации модуля метод вызывается при открытии формы отправки электронного документа.
`GetFileContent(Guid, Guid)`	Предоставляет содержимое файла карточки. **Тип возвращаемого значения**: `System.String`. *Параметры:* • `cardId` — идентификатор карточки. • `fileId` — идентификатор файла, для которого возвращается содержимое. В стандартной конфигурации модуля метод вызывается при открытии формы отправки электронного документа.

Имя	Описание
`GetFileDigest(Guid, Guid)`	Предоставляет описание (дайджест) файла карточки. **Тип возвращаемого значения**: `System.String`. *Параметры:* • `cardId` — идентификатор карточки. • `fileId` — идентификатор файла, для которого возвращается описание. В стандартной конфигурации модуля метод вызывается при открытии формы отправки электронного документа.

Имя	Описание
`GetInvoiceReplyData(Guid)`	Возвращает строку с ответом на сообщение, полученное от оператора ЭДО. Тип возвращаемого значения: `System.String`. *Параметры:* • `cardId` — идентификатор карточки. В стандартной конфигурации модуля метод вызывается при формировании ответной подписи. Метод `GetInvoiceReplyData` должен вернуть строку с данными карточки в определенном формате, который ожидается методом `IMessageService.GenerateInvoiceReply` коннектора к оператору ЭДО. В реализации коннектора к Диадок для ответа по УПД метод `GetInvoiceReplyData` должен вернуть сериализованный объект `DocsVision.Edi.Entities.BuyerReplyData`, заполненный данными карточки УПД, по которой формируется ответ.
`GetLastSignedFiles(Guid)`	Возвращает последние (по времени) подписанные файлы карточки. **Тип возвращаемого значения**: `ItemCollection<DocumentFileData>`. *Параметры:* • `cardId` — идентификатор карточки.

Имя	Описание
`GetPrintFormContent(Guid)`	Возвращает содержимое карточки, подготовленное для печати. **Тип возвращаемого значения**: `String`. *Параметры:* • `cardId` — идентификатор карточки. В стандартной конфигурации модуля метод вызывается печати УПД.
`GetReplyFileComment(Guid, Guid, MessageFileType)`	Формирует строку с комментарием к файлу отправленного электронного сообщения. **Тип возвращаемого значения**: `System.String`. *Параметры:* • `cardId` — идентификатор карточки. • `fileId` — идентификатор файла, для которого нужен комментарий. • `fileType` — ответ контрагента на переданный в электронном сообщении файл. В стандартной конфигурации модуля метод вызывается при формировании сообщений для журнала обмена ЮЗДО.

Имя	Описание
`GetReplyFileDigest(Guid, Guid, MessageFileType)`	Формирует строку с дайджестом файла отправленного электронного сообщения. **Тип возвращаемого значения**: `System.String`. *Параметры:* • `cardId` — идентификатор карточки. • `fileId` — идентификатор файла, для которого нужен дайджест. • `fileType` — ответ контрагента на переданный в электронном сообщении файл. В стандартной конфигурации модуля метод вызывается при формировании сообщений для журнала обмена ЮЗДО.
`GetSignedFiles(Guid, Guid)`	Возвращает файлы карточки, которые подписаны указанной подписью. **Тип возвращаемого значения**: `ItemCollection<DocumentFileData>`. *Параметры:* • `cardId` — идентификатор карточки. • `signatureGroupId` — идентификатор ЭЦП. В стандартной конфигурации модуля метод вызывается при открытии формы отправки электронного документа.
`Initialize(ObjectContext)`	Инициализирует экземпляр компонента отправки электронного документа. *Параметры:* • `objectContext` — контекст объектов.

Имя	Описание
`PrepareFileData(Guid, Guid, Guid, Boolean, String)`	Создаёт экземпляр файла сообщения электронного обмена. **Тип возвращаемого значения**: `MessageFile`. *Параметры:* • `cardId` — идентификатор карточки, содержащий файл. • `fileId` — идентификатор файла, передаваемого в сообщении. • `signatureId` — идентификатор подписи, которой подписан файл. • `isMain` — файл является основным. • `tempFolder` — временный каталог, для выгрузки файла. В стандартной конфигурации модуля метод вызывается при отправке оператору ЭДО сообщения электронного обмена с новым документом. В большинстве случаев полностью реализовывать данный метод не требуется, т.к. принцип подготовки файла электронного сообщения стандартный. Чтобы использовать стандартный механизм формирования, достаточно вернуть из метода значение `null`.

Имя	Описание
`PrepareMessageData(Guid, String, String)`	Создаёт экземпляр электронного сообщения электронного обмена. **Тип возвращаемого значения**: `MessageData`. *Параметры:* • `cardId` — идентификатор карточки, для которой формируется сообщение электронного обмена для отправки оператору ЭДО. • `documentType` — тип документа. • `documentVersion` — версия документа. В стандартной конфигурации модуля метод вызывается при отправке оператору ЭДО сообщения электронного обмена с новым документом.

Имя	Описание
`ValidateSignature(Guid, Guid, Guid)`	Проверяет подпись файла и возвращает объект, содержащий результаты проверки. Тип возвращаемого значения: `SignatureValidation` *Параметры:* • `cardId` — идентификатор карточки с проверяемым файлом. • `fileId` — идентификатор файла карточки, подпись которого проверяется. • `signatureId` — идентификатор подписи. В стандартной конфигурации модуля метод вызывается при работе с Журналом обмена ЮЗДО, если в процессе обмена УПД от контрагента поступили подписанные файлы.

IDocumentUpdater — интерфейс

Интерфейс `IDocumentUpdater` определяет методы, которые должен реализовывать компонент изменения данных карточки.

- **Пространство имен:** `DocsVision.Edi.Interfaces`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public interface IDocumentUpdater
```

Методы

Имя	Описание
`AddSignatureToDocument(Guid, Guid, MessageFile, String)`	Добавляет подпись файла в карточку. **Тип возвращаемого значения**: `System.Guid`. *Параметры:* • `cardId` — идентификатор карточки. • `fileId` — идентификатор подписываемого файла. • `messageFile` — поступившее сообщение электронного обмена, содержащее подпись файла. • `partnerName` — название контрагента. В стандартной конфигурации модуля метод вызывается, когда от оператора ЭДО поступило сообщение с ответной подписью.

Имя	Описание
`CreateNewSignatures(Guid, X509Certificate2, Boolean)`	Подписывает карточку переданным сертификатом. **Тип возвращаемого значения**: `System.Guid`. *Параметры:* • `cardId` — идентификатор подписываемой карточки. • `certificate` — сертификат, которым подписывается карточка. • `signAdditionalFiles` — подписывать дополнительные файлы: `true` — подписывать; `false` — не подписывать. В стандартной конфигурации модуля метод вызывается, когда пользователь подписывает карточку отправляемого электронного документа.
`CheckCertificate(Guid, X509Certificate2)`	Проверяет соответствие подписи в указанной карточке переданному сертификату и возвращает результат проверки. **Тип возвращаемого значения**: `System.String`. *Параметры:* • `cardId` — идентификатор проверяемой карточки. • `certificate` — сертификат, для которого выполняется проверка. В стандартной конфигурации модуля метод вызывается при отправке УПД пользователем.

Имя	Описание
`ChangeDocumentState(Guid, Guid)`	Изменяет состояние карточки. Если изменение было успешным, метод возвращает `true`, иначе — `false`. **Тип возвращаемого значения**: `System.Boolean`. *Параметры:* • `cardId` — идентификатор карточки. • `newStateId` — идентификатор нового состояния. В стандартной конфигурации модуля метод вызывается при обработке входящих сообщений бизнес-процессом *Перенос файлов из сообщений ЮЗДО*.
`GetNewStateId(MessageFileType)`	Возвращает идентификатор встроенного состояния для карточки УПД, сопоставленного типу ответа контрагента на отправленный ранее УПД. **Тип возвращаемого значения**: `System.Guid`. *Параметры:* • `messageFileType` — Тип файла сообщения В стандартной конфигурации модуля метод вызывается, когда пользователь подписывает карточку отправляемого электронного документа.

Имя	Описание
`Initialize(ObjectContext)`	Инициализирует экземпляр компонента создания документов. *Параметры:* • `objectContext` — контекст объектов.
`UpdateDocumentDataFromFile(Guid, Guid)`	Обновляет содержимое карточки данными из файла. *Параметры:* • `cardId` — идентификатор обновляемой карточки. • `fileId` — идентификатор файла, содержащего данные для обновления карточки. В стандартной конфигурации модуля метод вызывается, когда пользователь использует команду **Заполнить данные из файла** в карточке УПД.
`AddLinkToMessageCard(Guid, Guid);`	Добавляет ссылку на карточку сообщения ЮЗДО *Параметры:* • `messageCardId` — Идентификатор карточки сообщения • `documentCardId` — Идентификатор карточки документа
`UpdatePowerOfAttorneyStatus(Guid, PowerOfAttorneyData)`	Устанавливает или снимает флаг `Использовать по умолчанию` *Параметры:* • `cardId` — Идентификатор карточки • `powerOfAttorneyData` — Данные МЧД

Имя	Описание
`ChangePowerOfAttorneyState(Guid, Guid, MessageFileType);`	Изменяет состояние документа и МЧД *Параметры:* • `cardId` — Идентификатор карточки • `powerOfAttorneyCardId` — Идентификатор карточки доверенности • `messageFileType` — Тип файла сообщения Возвращаемое значение: true, если состояние изменилось

IEdiPowerOfAttorneyService — интерфейс

Сервис для работы с машиночитаемой доверенностью ЮЗДО

- **Пространство имен:** `DocsVision.Edi.ObjectModel.Services`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public interface IEdiPowerOfAttorneyService
```

Методы

Имя	Описание
`CreateEdiPowerOfAttorney(Guid, OperatorsBox)`	Создаёт Карточку обмена сообщениями для МЧД: *Параметры:* • `cardId` — Идентификатор карточки • `organizationBox` — Ящик организации

Имя	Описание
`AddLogMessage(EdiPowerOfAttorney, EdiPowerOfAttorneyEventType, string)`	Добавляет сообщение в журнал *Параметры:* • `message` — Карточка сообщения • `eventType` — Тип события • `description` — Описание
`AddLogMessage(EdiPowerOfAttorney, EdiPowerOfAttorneyEventType, string, string)`	Добавляет сообщение в журнал *Параметры:* • `message` — Карточка сообщения • `eventType` — Тип события • `description` — Описание • `comment` — Комментарий
`AddLogMessage(EdiPowerOfAttorney, EdiPowerOfAttorneyEventType, string, string, string)`	Добавляет сообщение в журнал *Параметры:* • `message` — Карточка сообщения • `eventType` — Тип события • `description` — Описание • `comment` — Комментарий • `fileName` — Имя файла

IMessageService — интерфейс

Интерфейс `IMessageService` определяет методы, которые должен реализовывать компонент сервиса сообщений.

- **Пространство имен:** `DocsVision.Edi.Interfaces`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public interface IMessageService
```

Методы

Имя	Описание
`EventExists(String, String)`	Проверяет наличие события с указанным идентификатором в ящике организации. Возвращает True, если событие есть. **Тип возвращаемого значения**: `Boolean`. *Параметры:* • boxId — идентификатор ящика организации. • eventId — идентификатор события. В стандартной конфигурации модуля метод вызывается при получении новых сообщений от оператора ЭДО.

Имя	Описание
`GenerateCorrectionRequest(String, String, String, X509Certificate2, String, String)`	Формирует запрос на уточнение. **Тип возвращаемого значения**: `MessageFile`. *Параметры:* • `boxId` — идентификатор ящика организации. • `messageId` — идентификатор сообщения у оператора обмена. • `entityId` — идентификатор сущности у оператора ЭДО, по которому отправляется запрос на уточнение. • `certificate` — сертификат, которым подписывается запрос. • `positionName` — должность подписывающего запрос сотрудника. • `comment` — комментарий к запросу. В стандартной конфигурации модуля метод вызывается, когда пользователь запрашивает уточнение по поступившему документу.

Имя	Описание
`GenerateInvoiceReceipt(String, String, String, X509Certificate2, String)`	Формирует ответную подпись на полученный документ. **Тип возвращаемого значения**: `MessageFile`. *Параметры:* • `boxId` — идентификатор ящика организации. • `messageId` — идентификатор сообщения у оператора обмена. • `entityId` — идентификатор сущности у оператора ЭДО, извещение о получении которой отправляется. • `certificate` — сертификат, которым подписывается извещение. • `positionName` — должность подписывающего извещение сотрудника. В стандартной конфигурации модуля метод вызывается, когда пользователь вызывает функцию отправки извещения.

Имя	Описание
`GenerateInvoiceReply(String, String, String, X509Certificate2, String, String)`	Формирует ответную подпись для поступившего документа. **Тип возвращаемого значения**: `MessageFile`. *Параметры:* • `boxId` — идентификатор ящика организации. • `messageId` — идентификатор сообщения у оператора обмена. • `entityId` — идентификатор сущности у оператора ЭДО, по которой формируется подпись. • `certificate` — сертификат, которым подписывается запрос. • `documentType` — тип документа. • `replyData` — содержимое ответа. В стандартной конфигурации модуля метод вызывается, когда пользователь подписывает поступивший документ.

Имя	Описание
`GenerateRevocationRequest(String, String, String, X509Certificate2, String, String)`	Формирует запрос на аннулирование. **Тип возвращаемого значения**: `MessageFile`. *Параметры:* • `boxId` — идентификатор ящика организации. • `messageId` — идентификатор сообщения у оператора обмена. • `entityId` — идентификатор сущности у оператора ЭДО, по которому отправляется запрос на аннулирование. • `certificate` — сертификат, которым подписывается запрос. • `positionName` — должность подписывающего запрос сотрудника. • `comment` — комментарий к запросу. В стандартной конфигурации модуля метод вызывается, когда пользователь запрашивает аннулирование.

Имя	Описание
`GenerateSignatureRejection(String, String, String, X509Certificate2, String, String)`	Формирует отказ в подписании. **Тип возвращаемого значения**: `MessageFile`. *Параметры:* • `boxId` — идентификатор ящика организации. • `messageId` — идентификатор сообщения у оператора обмена. • `entityId` — идентификатор сущности у оператора ЭДО, по которому отправляется отказ в подписании. • `certificate` — сертификат, которым подписывается ответ. • `positionName` — должность подписывающего запрос сотрудника. • `comment` — комментарий к отказу. В стандартной конфигурации модуля метод вызывается, когда пользователь отказывает в подписании поступившего документа.
`GetLastEventId(String)`	Возвращает идентификатор последнего события в ящике организации. **Тип возвращаемого значения**: `String`. *Параметры:* • `boxId` — идентификатор ящика организации. В стандартной конфигурации модуля метод используется при активации ящика организации.

Имя	Описание
`GetNewEvents(String, String, DateTime?)`	Получает новые события из ящика организации. **Тип возвращаемого значения**: `ItemCollection<MessageData>`. *Параметры:* • `boxId` — идентификатор ящика организации. • `lastEventId` — идентификатор последнего полученного события (может отсутствовать). • `fromDate` — дата минимальная дата создания сообщения. Если значение указано, то должны возвращаться сообщения, созданные после указанной даты. В стандартной конфигурации модуля метод вызывается при получении новых сообщений от оператора ЭДО.
`GetNewEventsCount(String, String)`	Возвращает количество новых событий. **Тип возвращаемого значения**: `Int32`. *Параметры:* • `boxId` — идентификатор ящика организации. • `lastEventId` — идентификатор последнего полученного события (может отсутствовать). В стандартной конфигурации модуля метод вызывается при активации ящика организации.

Имя	Описание
`Initialize(Dictionary<String, String>)`	Создаёт экземпляр электронного сообщения электронного обмена. *Параметры:* • `settings` — коллекция настроек оператора ЭДО из *Справочника настроек операторов ЮЗДО*. Каждая запись содержит ключ и значение, например: `Login` и Docsvision@diadoc.ru.
`SendCorrection(MessageData)`	Отправляет запрос на уточнение. *Параметры:* • `messageData` — сообщение типа `MessageData`, содержащее запрос на уточнение. В стандартной конфигурации модуля метод используется отправке новых сообщений оператору ЭДО.
`SendMessage(MessageData)`	Отправляет новое сообщение оператору ЭДО. *Параметры:* • `messageData` — сообщение типа `MessageData`. В стандартной конфигурации модуля метод используется отправке новых сообщений оператору ЭДО.
`SendReceipt(MessageData)`	Отправляет квитанцию. *Параметры:* • `messageData` — сообщение типа `MessageData`.

Имя	Описание
`SendRevocation(MessageData)`	Отправляет запрос на аннулирование. *Параметры:* • `messageData` — сообщение типа `MessageData`, содержащее запрос на аннулирование В стандартной конфигурации модуля метод используется отправке новых сообщений оператору ЭДО.
`SendSignatureReply(MessageData)`	Отправляет ответную подпись на полученное сообщение. *Параметры:* • `messageData` — сообщение типа `MessageData`. В стандартной конфигурации модуля метод используется при отправке оператору ЭДО ответа на полученное сообщение.

InvitationService — класс

Сервис работы с приглашениями, может быть использован для замены стандартных БП модуля.

Вызывает экземпляры `IInvitationService` для соответствующего оператора обмена.

- **Пространство имен:** `DocsVision.Edi.Runtime`
- **Сборка:** `DocsVision.Edi.Runtime.dll`

Синтаксис

```
public class InvitationService
```

Методы

Имя	Описание
`public void ProcessInvitationToSend(EdiInvitation, ILogWriter)`	Обрабатывает карточку приглашения
`public void CheckInvitationStatus(EdiInvitation, ILogWriter)`	Проверяет статус нового приглашения
`public ItemCollection<MessageData> GetNewInvitations(string, string, DateTime?)`	Получает новые приглашения с указанной даты по ID ящика и ID последнего события
`public void ProcessEvent(MessageData, ILogWriter)`	Обрабатывает событие

IPowerOfAttorneyService — интерфейс

Интерфейс `IPowerOfAttorneyService` определяет методы для работы с МЧД.

- **Пространство имен:** `DocsVision.Edi.Interfaces`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public interface IPowerOfAttorneyService
```

Методы

Имя	Описание
`RegisterPowerOfAttorney(MessageData);`	Регистрирует доверенность *Параметры:* • `messageData`

Имя	Описание
`RevokePowerOfAttorney(MessageData);`	Отвязывает МЧД от сотрудника в Диадок, в реестре ФНС доверенность не отзывается *Параметры:* • `messageData`
`CheckPowerOfAttorneyStatus(MessageData);`	Проверяет статус доверенности *Параметры:* • `messageData`

OutgoingPowerOfAttorneyData — класс

Содержит данные исходящей доверенности.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class OutgoingPowerOfAttorneyData
```

Свойства

Имя	Описание
`CardId`	Задает или возвращает идентификатор карточки. **Тип**: `Guid`
`PowerOfAttorneyCardId`	Задает или возвращает идентификатор карточки доверенности. **Тип**: `Guid`
`SendAsFile`	Отправляет МЧД как файл. **Тип**: `bool`

Имя	Описание
`PowerOfAttorneyData`	Задает или возвращает данные доверенности. **Тип**: `PowerOfAttorneyData`

OutgoingPowerOfAttorneyService — класс

Сервис создания исходящей доверенности

- **Пространство имен:** `DocsVision.Edi.Runtime`
- **Сборка:** `DocsVision.Edi.Runtime.dll`

Синтаксис

```
public class OutgoingPowerOfAttorneyService
```

Методы

Имя	Описание
`CreateOutgoingPowerOfAttorney(OperatorsBox, OperatorsDepartment, OutgoingPowerOfAttorneyData, out string)`	Создаёт исходящую доверенность **Тип возвращаемого значения**: `EdiPowerOfAttorney` *Параметры:* • `organizationBox` — Ящик организации • `organizationDepartment` — подразделение организации • `powerOfAttorneyData` — Данные МЧД • `sendErrors` — отправлять ошибки

Имя	Описание
`CreateOutgoingPowerOfAttorneyRecall(OperatorsBox, OperatorsDepartment, OutgoingPowerOfAttorneyData, out string)`	Создаёт отзыв исходящей доверенности **Тип возвращаемого значения**: `EdiPowerOfAttorney` *Параметры:* • `organizationBox` — Ящик организации • `organizationDepartment` — подразделение организации • `powerOfAttorneyData` — Данные МЧД • `sendErrors` — отправлять ошибки
`FindAnyPowerOfAttorneyCards(Guid, Guid)`	Находит любую из карточек доверенности **Тип возвращаемого значения**: `EdiPowerOfAttorney` *Параметры:* • `cardId` — идентификатор карточки, для которой запрашивается МЧД • `powerOfAttorneyCardId` — идентификатор карточки МЧД
`FindAssignedPowerOfAttorneyCards(Guid, Guid)`	Находит назначенные карточки доверенности **Тип возвращаемого значения**: `EdiPowerOfAttorney` *Параметры:* • `cardId` — идентификатор карточки, для которой запрашивается МЧД • `powerOfAttorneyCardId` — идентификатор карточки МЧД

PowerOfAttorneyData — класс

Содержит данные машиночитаемой доверенности.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public class PowerOfAttorneyData
```

Свойства

Имя	Описание
`RegistrationNumber`	Задает или возвращает регистрационный номер **Тип**: `Guid`
`IssuerInn`	Задает или возвращает ИНН доверителя **Тип**: `string`
`RepresentativeInn`	Задает или возвращает ИНН доверенного лица **Тип**: `string`
`IsDefault`	Задает или возвращает признак использования по умолчанию **Тип**: `bool`
`PowerOfAttorneyStatus`	Задает или возвращает статус МЧД **Тип**: `PowerOfAttorneyStatus`
`ValidationStatus`	Задает или возвращает статус проверки доверенности **Тип**: `ValidationStatus`

Имя	Описание
`ValidationErrors`	Выявленные при проверке ошибки. **Тип**: `string`

PowerOfAttorneyDataReader — класс

Содержит методы для чтения машиночитаемой доверенности

- **Пространство имен:** `DocsVision.Edi.Runtime.BackOffice`
- **Сборка:** `DocsVision.Edi.Runtime.Backoffice.dll`

Синтаксис

```
public class PowerOfAttorneyDataReader : DocumentDataReader
```

Методы

Имя	Описание
`PrepareMessageData(Guid)`	Создаёт экземпляр электронного сообщения электронного обмена. **Тип возвращаемого значения**: `MessageData`. *Параметры:* • `cardId` — идентификатор карточки, для которой формируется сообщение электронного обмена для отправки оператору ЭДО. • `documentType` — тип документа. • `documentVersion` — версия документа. В стандартной конфигурации модуля метод вызывается при отправке оператору ЭДО сообщения электронного обмена с новым документом.

Имя	Описание
`ReadAdditionalAttributes(MessageData, Document)`	Читает дополнительные атрибуты *Параметры:* • `messageData` — электронное сообщение, полученное от контрагента • `document` — документ

Имя	Описание
`PrepareFileData(Guid, Guid, Guid, bool, string)`	Создаёт экземпляр файла сообщения электронного обмена. **Тип возвращаемого значения**: `MessageFile`. *Параметры:* • `cardId` — идентификатор карточки, содержащий файл. • `fileId` — идентификатор файла, передаваемого в сообщении. • `signatureId` — идентификатор подписи, которой подписан файл. • `isMain` — файл является основным. • `tempFolder` — временный каталог, для выгрузки файла. В стандартной конфигурации модуля метод вызывается при отправке оператору ЭДО сообщения электронного обмена с новым документом. В большинстве случаев полностью реализовывать данный метод не требуется, т.к. принцип подготовки файла электронного сообщения стандартный. Чтобы использовать стандартный механизм формирования, достаточно вернуть из метода значение `null`.

Имя	Описание
`GetFileDigest(Guid, Guid)`	Предоставляет описание (дайджест) файла карточки. **Тип возвращаемого значения**: `System.String`. *Параметры:* • `cardId` — идентификатор карточки. • `fileId` — идентификатор файла, для которого возвращается описание. В стандартной конфигурации модуля метод вызывается при открытии формы отправки электронного документа.
`GetFileContent(Guid, Guid)`	Предоставляет содержимое файла карточки. **Тип возвращаемого значения**: `System.String`. *Параметры:* • `cardId` — идентификатор карточки. • `fileId` — идентификатор файла, для которого возвращается содержимое. В стандартной конфигурации модуля метод вызывается при открытии формы отправки электронного документа.

Имя	Описание
`GetReplyFileDigest(Guid, Guid, MessageFileType)`	Предоставляет дайджест файла ответа **Тип возвращаемого значения**: `System.String`. *Параметры:* • `cardId` — идентификатор карточки. • `fileId` — идентификатор файла, для которого возвращается содержимое. • `fileType` — тип файла, для которого возвращается содержимое.

PowerOfAttorneyService — класс

Сервис машиночитаемой доверенности

- **Пространство имен:** `DocsVision.Edi.Runtime.Diadoc`
- **Сборка:** `DocsVision.Edi.Runtime.Diadoc.dll`

Синтаксис

```
public class PowerOfAttorneyService : IPowerOfAttorneyService
```

Методы

Имя	Описание
`Initialize(Dictionary<string, string>)`	Создаёт экземпляр электронного сообщения электронного обмена. *Параметры:* • `settings` — коллекция настроек оператора ЭДО из *Справочника настроек операторов ЮЗДО*. Каждая запись содержит ключ и значение, например: `Login` и Docsvision@diadoc.ru.

Имя	Описание
`RegisterPowerOfAttorney(MessageData)`	Регистрирует доверенность *Параметры:* • `messageData` — сообщение типа `MessageData`, содержащее запрос на регистрацию МЧД.
`RevokePowerOfAttorney(MessageData)`	Отвязывает МЧД от сотрудника в Диадок, в реестре ФНС доверенность не отзывается *Параметры:* • `messageData` — сообщение типа `MessageData`, содержащее запрос на отзыв МЧД.
`CheckPowerOfAttorneyStatus(MessageData)`	Проверяет статус доверенности *Параметры:* • `messageData` — сообщение типа `MessageData`, содержащее запрос на проверку статуса МЧД.

PowerOfAttorneyService — класс

Сервис машиночитаемой доверенности

- **Пространство имен:** `DocsVision.Edi.Runtime`
- **Сборка:** `DocsVision.Edi.Runtime.dll`

Синтаксис

```
public class PowerOfAttorneyService
```

Методы

Имя	Описание
`PowerOfAttorneyService(OperatorsOperator)`	Получает МЧД *Параметры:* • `ediOperator` — оператор ЮЗДО
`PowerOfAttorneyService(ObjectContext, OperatorsOperator, bool)`	Получает МЧД *Параметры:* • `objectContext` — контекст объектов. • `ediOperator` — оператор ЮЗДО • `useExtension` — использовать расширение
`ProcessPowerOfAttorneyToSend(EdiPowerOfAttorney, ILogWriter)`	Обрабатывает карточку МЧД *Параметры:* • `ediPowerOfAttorney` — машиночитаемая доверенность • `logWriter` — запись в журнале
`CheckPowerOfAttorneyStatus(EdiPowerOfAttorney, ILogWriter)`	Проверяет статус МЧД *Параметры:* • `ediPowerOfAttorney` — машиночитаемая доверенность • `logWriter` — запись в журнале

PowerOfAttorneyStatus — перечисление

Представляет статус доверенности.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public enum PowerOfAttorneyStatus
```

Члены

Имя	Описание
`Creating = 0`	Подготовка
`Registering = 2`	Регистрация
`Registered = 3`	Зарегистрирована
`AssignedToEmployee = 4`	Передана сотруднику
`Error = 5`	Ошибка
`Recalled = 6`	Отозвана

PowerOfAttorneyUpdater — класс

Класс обновления машиночитаемой доверенности

- **Пространство имен:** `DocsVision.Edi.Runtime.BackOffice`
- **Сборка:** `DocsVision.Edi.Runtime.Backoffice.dll`

Синтаксис

```
public class PowerOfAttorneyUpdater : IDocumentUpdater
```

Методы

Имя	Описание
`PowerOfAttorneyService(OperatorsOperator)`	Получает МЧД *Параметры:* • `ediOperator` — оператор ЮЗДО
`PowerOfAttorneyService(ObjectContext, OperatorsOperator, bool)`	Получает МЧД *Параметры:* • `objectContext` — контекст объектов. • `ediOperator` — оператор ЮЗДО • `useExtension` — использовать расширение

Имя	Описание
`Initialize(ObjectContext)`	*Параметры:* • `objectContext` — контекст объектов.
`AddSignatureToDocument(Guid, Guid, MessageFile, string)`	*Параметры:* • `cardId` — идентификатор карточки. • `fileId` — идентификатор файла. • `messageFile` — подписываемый файл. • `partnerName` — название контрагента.
`CreateNewSignatures(Guid, X509Certificate2, bool)`	*Параметры:* • `cardId` — идентификатор карточки. • `certificate` — сертификат подписи. • `signAdditionalFiles` — подписывать дополнительные файлы: `true` — подписывать; `false` — не подписывать.
`CheckCertificate(Guid, X509Certificate2)`	*Параметры:* • `cardId` — идентификатор карточки. • `certificate` — сертификат подписи.
`ChangeDocumentState(Guid, Guid)`	*Параметры:* • `cardId` — идентификатор карточки. • `newStateId` — идентификатор нового состояния.
`Guid GetNewStateId(MessageFileType)`	*Параметры:* • `messageFileType` — Тип файла сообщения
`UpdateDocumentDataFromFile(Guid, Guid)`	*Параметры:* • `cardId` — идентификатор карточки. • `fileId` — идентификатор подписываемого файла.

Имя	Описание
`AddLinkToMessageCard(Guid, Guid)`	*Параметры:* • `messageCardId` — Идентификатор карточки сообщения. • `documentCardId` — Идентификатор карточки документа.
`UpdatePowerOfAttorneyStatus(Guid, PowerOfAttorneyData)`	*Параметры:* • `cardId` — идентификатор карточки. • `powerOfAttorneyData` — Данные МЧД.
`ChangePowerOfAttorneyState(Guid, Guid, MessageFileType)`	*Параметры:* • `cardId` — идентификатор карточки. • `powerOfAttorneyCardId` — идентификатор карточки МЧД. • `messageFileType` — Тип файла сообщения.

IImportService — интерфейс

Интерфейс `IImportService` определяет методы, которые должен реализовывать компонент импорта настроек ЭДО.

- **Пространство имен:** `DocsVision.Edi.Interfaces`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public interface IImportService
```

Методы

Имя	Описание
`ImportPartners`	Предоставляет список контрагентов и их ящиков, зарегистрированных у организации в системе оператора ЭДО. **Тип возвращаемого значения**: `ItemCollection<Partner>`. В стандартной конфигурации модуля метод используется при вызове в *Справочнике настроек операторов ЮЗДО* команды загрузки данных контрагентов.
`ImportUnits`	Предоставляет список подразделений организации и их ящиков ящиками, зарегистрированных у организации в системе оператора ЭДО. **Тип возвращаемого значения**: `ItemCollection<Unit>`. В стандартной конфигурации модуля метод используется при вызове в *Справочнике настроек операторов ЮЗДО* команды загрузки данных организации.
`Initialize(Dictionary<string, string>)`	Создаёт экземпляр электронного сообщения электронного обмена. *Параметры:* • `settings` — коллекция настроек оператора ЭДО из *Справочника настроек операторов ЮЗДО*. Каждая запись содержит ключ и значение, например: `Login` и `Docsvision@diadoc.ru`.

Примеры

См. пример реализации интерфейса в разделе Разработка компонента импорта

настроек организации из системы оператора ЭДО.

ISignatureService — интерфейс

Интерфейс `ISignatureService` определяет методы, которые должен реализовывать компонент сервиса подписей.

- **Пространство имен:** `DocsVision.Edi.Interfaces`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public interface ISignatureService
```

Свойства

Имя	Описание
`CreateCommentSignature(String, X509Certificate2)`	Создаёт в Docsvision подпись для текстового сообщения. Метод должен вернуть идентификатор созданной подписи. **Тип возвращаемого значения**: `Guid`. *Параметры:* • `commentText` — подписываемое текстовое сообщение. • `certificate` — сертификат подписи. В стандартной конфигурации модуля метод вызывается, например, при подписании комментария к отказу в подписании полученного документа.

Имя	Описание
`CreateFileSignature(MessageFile, X509Certificate2)`	Создаёт в Docsvision подпись для файла сообщения. Метод должен вернуть идентификатор созданной подписи. **Тип возвращаемого значения**: `Guid`. *Параметры:* • `messageFile` — подписываемый файл. • `certificate` — сертификат подписи. В стандартной конфигурации модуля метод вызывается, например, при подписании файла сообщения с запросом уточнения для полученного документа.
`Initialize(ObjectContext)`	Инициализирует экземпляр компонента сервиса подписей. *Параметры:* • `objectContext` — контекст объектов.

LoaderService — класс

Сервис загрузки входящих и исходящих документов. Не рекомендуется использовать напрямую доступен, предпочтительнее вызывать методы через `BoxesService`.

Вызывает экземпляры `ILoaderService` для соответствующего оператора обмена.

- **Пространство имен:** `DocsVision.Edi.Runtime`
- **Сборка:** `DocsVision.Edi.Runtime.dll`

Синтаксис

```
public class LoaderService
```

Методы

Имя	Описание
`public ItemCollection<MessageData> LoadIncomingDocuments(string, DateTime, DateTime, string)`	Загружает входящие документы в указанном промежутке с … по …, по ID ящика и ID последнего документа
`public ItemCollection<MessageData> LoadOutgoingDocuments(string, DateTime, DateTime, string)`	Загружает исходящие документы в указанном промежутке с … по …, по ID ящика и ID последнего документа

OutgoingMessagesService — класс

Сервис отвечает за отправку отправка исходящих документов, ответов и прочих задач. Может быть использован в собственных скриптах и приложениях для отправки документов и прочего.

- **Пространство имен:** `DocsVision.Edi.Runtime`
- **Сборка:** `DocsVision.Edi.Runtime.dll`

Синтаксис

```
public class OutgoingMessagesService
```

Методы

Имя	Описание
`public IEnumerable<EdiMessage> CreateOutgoingMessageCards(OperatorsBox, OperatorsDepartment, IEnumerable<RecipientFiles>, out string)`	Создаёт карточки исходящих сообщений *Параметры:* • `organizationBox` — ящик организации • `organizationDepartment` — подразделение организации • `recipientFiles` — коллекция файлов получателя • `sendErrors` — отправлять ошибки

Имя	Описание
`public IEnumerable<EdiMessage> CreateOutgoingMessageCards(OperatorsBox, OperatorsDepartment, string, string, IEnumerable<RecipientFiles>, out string)`	Создаёт карточки исходящего сообщения *Параметры:* • `organizationBox` — ящик организации • `organizationDepartment` — подразделение организации • `documentType` — тип документа • `documentVersion` — версия документа • `recipientFiles` — коллекция файлов получателя • `sendErrors` — отправлять ошибки
`public IEnumerable<EdiMessage> CreateOutgoingRejectionReplies(Guid, string, Guid)`	Создаёт исходящий ответный отказ в подписании *Параметры:* • `cardId` — ID карточки документа • `rejectionText` — текст комментария с отказом • `rejectionSignatureId` — ID отклонённой подписи
`public IEnumerable<EdiMessage> CreateOutgoingRejectionReplies(Guid, string, X509Certificate2)`	Создаёт исходящий ответный отказ в подписании *Параметры:* • `cardId` — ID карточки документа • `rejectionText` — текст комментария с отказом • `certificate` — сертификат

Имя	Описание
`public IEnumerable<EdiMessage> CreateOutgoingSignatureReplies(Guid, ItemCollection<DocumentFileData>)`	Создаёт исходящую ответную подпись *Параметры:* • `cardId` — ID карточки • `signedFiles` — подписанные файлы
`public IEnumerable<EdiMessage> CreateOutgoingInvoiceReplies(Guid, X509Certificate2, string)`	Создаёт ответные исходящие счёт-фактуры *Параметры:* • `cardId` — ID карточки документа • `replyData` — ответные данные • `certificate` — сертификат
`public IEnumerable<EdiMessage> CreateRevocationRequests(IEnumerable<EdiMessage>, string, X509Certificate2, out string)`	Создаёт исходящие запросы на аннулирование *Параметры:* • `messageCards` — карточки сообщений • `revocationText` — текст отказа • `certificate` — сертификат • `sendErrors` — отправлять ошибки
`public IEnumerable<EdiMessage> CreateOutgoingRevRejectionReplies(Guid, string, X509Certificate2)`	Создаёт исходящие ответы на аннулирование подписи *Параметры:* • `cardId` — ID карточки документа • `rejectionText` — текст комментария с отказом • `certificate` — сертификат
`public IEnumerable<EdiMessage> CreateOutgoingRevSignatureReplies(Guid, X509Certificate2)`	Создаёт исходящие ответы на аннулирование подписи *Параметры:* • `cardId` — ID карточки

Имя	Описание
`public IEnumerable<EdiMessage> CreateOutgoingReceipts(Guid, X509Certificate2)`	Отправляет исходящие извещения *Параметры:* • `cardId` — карточка • `certificate` — сертификат
`public IEnumerable<EdiMessage> CreateCorrectionRequests(IEnumerable<EdiMessage>, string, X509Certificate2, out string sendErrors)`	Отправляет исходящие запросы на исправление *Параметры:* • `messageCards` — карточки сообщений • `comment` — комментарий • `certificate` — сертификат • `sendErrors` — отправлять ошибки
`public IEnumerable<EdiMessage> FindSentMessageCards(Guid)`	Находит отправленные карточки сообщений по ID карточки
`public IEnumerable<EdiMessage> FindReceivedMessageCards(Guid)`	Находит полученные карточки сообщений по ID карточки
`public IEnumerable<EdiMessage> FindIncomingMessageCards(Guid)`	Находит входящие карточки сообщений по ID карточки
`public IEnumerable<EdiMessage> FindCorrectionMessageCards(Guid)`	Находит карточки сообщений для исправления по ID карточки
`public IEnumerable<EdiMessage> FindRevocationMessageCards(Guid)`	Находит карточки сообщений для аннулирования по ID карточки
`public IEnumerable<EdiMessage> FindReceiptMessageCards(Guid)`	Находит карточки сообщений, по которым требуется отправить извещения по ID карточки

FileContentType — перечисление

Определяет возможные типы содержимого в файле карточки *Документ*.

- **Пространство имен:** `DocsVision.Edi.Entities`
- **Сборка:** `DocsVision.Edi.Interfaces.dll`

Синтаксис

```
public enum FileContentType
```

Члены

Имя члена	Описание
`NonFormalized`	Неформализованный документ.
`SellerInvoice`	УПД продавца.

EdiMessageEventType — перечисление

Определяет возможные типы событий *журнала обмена* в *Карточки обмена сообщениями*.

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public enum EdiMessageEventType
```

Члены

Имя члена	Описание
`Information`	Информация.
`Error`	Ошибка.
`Reply`	Ответ.
`FromDocsvision`	Из Docsvision.
`FromPartner`	От контрагента.
`FromOperator`	От оператора обмена.

EdiMessageFileState — перечисление

Определяет возможные типы содержимого в `MessageFile`.

- **Пространство имен:** `DocsVision.Edi.ObjectModel`

- **Сборка:** DocsVision.Edi.ObjectModel.dll

Синтаксис

```
public enum EdiMessageFileState
```

Члены

Имя члена	Описание
Creating	Создаётся.
SignatureRequired	Получен на подпись от контрагента.
SentToPartner	Отправлен на подпись контрагенту.
PartnerRejected	Получен отказ от контрагента.
PartnerSigned	Получена подпись от контрагента.
Error	Ошибка.
ReplySignature	Ответная подпись.
ReplyRejection	Ответный отказ от подписи.
RevocationRequested	Запрошен отзыв.
RevocationSent	Отзыв отправлен.
Revoked	Отозвано.
Received	Получено.

EdiMessageFileContentType — перечисление

Определяет возможные типы содержимого в файле сообщения электронного обмена.

- **Пространство имен:** DocsVision.Edi.ObjectModel
- **Сборка:** DocsVision.Edi.ObjectModel.dll

Синтаксис

```
public enum EdiMessageFileContentType
```

Члены

Имя члена	Описание
`NonFormalized`	Неформализованный документ.
`SellerInvoice`	УПД продавца.

EdiMessageFileType — перечисление

Определяет возможные типы файлов в *Карточки обмена сообщениями*.

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public enum EdiMessageFileType
```

Члены

Имя члена	Описание
`Main`	Основной.
`Additional`	Дополнительный.

EdiMessageReplyFileType — перечисление

Определяет возможные типы файлов в *Карточки обмена сообщениями*.

- **Пространство имен:** `DocsVision.Edi.ObjectModel`
- **Сборка:** `DocsVision.Edi.ObjectModel.dll`

Синтаксис

```
public enum EdiMessageReplyFileType
```

Члены

Имя члена	Описание
`InvoiceConfirmation`	Подтверждение получения УПД.

Имя члена	Описание
`InvoiceReceipt`	Извещение о получении УПД.
`InvoiceReply`	Ответный титул УПД.
`SignatureRejection`	Отказ от подписи.
`InvoiceCorrectionRequest`	Запрос на уточнение УПД.

Методы расширения для Web-клиента

Для работы с модулем в web-интерфейсе Web-клиент приносит расширения:

- Клиентское расширение, устанавливается в каталог `Docsvision\WebClient\Site\Content\Modules\EdiWebExtension\` и собирается в папку `\edi-module\WebClient\Edi.WebExtension\dist`
- Серверное расширение и сборки EDI устанавливаются в каталог `Docsvision\WebClient\Site\Extensions\DocsVision.Edi.ServerExtension\`.

Расширения `EdiWebServerExtension` и `EdiWebClientExtension` добавляют в систему следующие методы:

Скрипты	Операция	Используется в разметках	Когда использовать
`approveRevocation`	Подтвердить аннулирование	Исходящий УПД (просмотр), Входящий УПД (просмотр)	Для подтверждения аннулирования. При этом нужно будет выбрать сертификат квалифицированной подписи для формирования ответной подписи. Для неформализованных документов

Скрипты	Операция	Используется в разметках	Когда использовать
`rejectRevocation`	Отказать в аннулировании	Исходящий УПД (просмотр), Входящий УПД (просмотр)	Для отказа в аннулировании. При этом нужно будет указать причину отказа. Для неформализованных документов
`printInvoiceUniversalTransferDocument` Расширение из папки в модулях `CustomPrintInvoice`.	Печать	Исходящий УПД (просмотр), Входящий УПД (просмотр)	Выводит на печать печатную форму формализованного документа
`signAndSendOutgoing`	Подписать и отправить	Исходящий УПД (просмотр)	При нажатии будет вызвана функция подписания документа с отправкой контрагенту. Для неформализованных исходящих
`signIncoming`	Подписать	-	Для неформализованных входящих
`signAndSendInvoice`	Подписать	Входящий УПД (просмотр)	При нажатии будет вызвана функция подписания документа с отправкой контрагенту

Скрипты	Операция	Используется в разметках	Когда использовать
`rejectAndSendInvoice`	Отказать	Входящий УПД (просмотр)	При нажатии будет вызвана функция отказа в подписании с указанием причины отказа. Для неформализованных входящих
`receipt`	Отправить извещение	Входящий УПД (просмотр)	Отправляет извещение-подтверждение о получении оператора ЭДО
`revoke`	Аннулировать	Входящий УПД (просмотр)	При нажатии будет вызвана функция аннулирования документа вне зависимости от его типа

Руководство разработчика Модуля интеграции с реестром МЧД

Описание взаимодействия с API модуля

Модуль Интеграция с реестром МЧД состоит из следующих компонентов:

- Серверное расширение Web-клиента.
- Серверное расширение для модуля Базовые объекты.
- Библиотеки API модуля.

Серверное расширение `M4dRegistry.ServerExtension` представляет собой расширение, которое получает от Web-клиента запросы на регистрацию действий производимых с доверенностью: регистрация, проверка статуса, отзыв.

Расширение для Службы фоновых операций состоит из двух библиотек:

- `M4dRegistry.WorkerService` — содержит реализацию фабрики заданий и сообщений для сервиса,
- `M4dRegistry.WorkerServiceExtension` — обрабатывает сообщения выполняя методы менеджера для работы с внешним сервисом.

Библиотека объектной модели содержит описание классов и интерфейсов используемых в решении.

Объектная модель модуля

ILogTransferService — интерфейс

Журнал обмена сообщениями `LogTransfer` — это новый тип карточек. Для работы с карточками этого типа предусмотрен сервис `ILogTransferService`.

- **Пространство имён:** `DocsVision.M4dRegistry.ObjectModel.Services`
- **Сборка:** `DocsVision.M4dRegistry.ObjectModel.dll`

Синтаксис

```
public interface ILogTransferService
```

Методы

Имя	Описание
`CreateLogTransferCard`	Создает карточку журнала обмена сообщениями.
`FindCardByM4dId`	Находит карточку журнала обмена сообщениями по ID доверенности.
`FindCardsByInn`	Находит все карточки журнала обмена сообщениями.
`CreateLogTransferLogItem(Guid, Guid, M4dRegistryAction)`	Создает запись в журнале карточке журнала обмена сообщениями.
`CreateLogTransferLogItem(Guid, Guid, M4dRegistryAction, M4dRegistryOperationStatus)`	Создает запись в журнале карточке журнала обмена сообщениями.
`FindLogTransferLogItem(Guid, M4dRegistryAction)`	Ищет записи в карточке журнала обмена сообщениями.
`FindLogTransferLogItem(Guid, M4dRegistryOperationStatus)`	Ищет записи в карточке журнала обмена сообщениями.
`FindLogTransferLogItem(Guid, M4dRegistryAction, M4dRegistryOperationStatus)`	Ищет записи в карточке журнала обмена сообщениями.
`GetAllLogTransferRows(Guid)`	Возвращает все строки из журнала карточки журнала обмена сообщениями.

CreateLogTransferCard — метод (Guid, Guid, string)

Создает карточку журнала обмена сообщениями.

- **Пространство имён:** `DocsVision.M4dRegistry.ObjectModel.Services`
- **Сборка:** `DocsVision.M4dRegistry.ObjectModel.dll`

Синтаксис

```
Guid CreateLogTransferCard(Guid attorneyId, Guid documentId, string inn)
```

Параметры

Guid

Тип: `attorneyId`

ID доверенности (СКД).

Guid

Тип: `documentId`

ID документа.

string

Тип: `inn`

ИНН организации, выдавшей доверенность.

Возвращаемое значение

Тип: `Guid`

ID созданной карточки.

Примечания

Параметры `attorneyId` и `documentId` взаимоисключающие, если указан один то другой должен быть `Guid.Empty`.

FindCardByM4dId — метод (Guid)

Находит карточку журнала обмена сообщениями по ID доверенности.

- **Пространство имён:** `DocsVision.M4dRegistry.ObjectModel.Services`
- **Сборка:** `DocsVision.M4dRegistry.ObjectModel.dll`

Синтаксис

```
Guid FindCardByM4dId(Guid m4dId)
```

Параметры

Guid

Тип: `m4dId`

ID доверенности (СКД).

Возвращаемое значение

Тип: `Guid`

Карточка журнала обмена сообщениями.

FindCardsByInn — метод (string)

Находит все карточки журнала обмена сообщениями.

- **Пространство имён:** `DocsVision.M4dRegistry.ObjectModel.Services`
- **Сборка:** `DocsVision.M4dRegistry.ObjectModel.dll`

Синтаксис

```
List<RowData> FindCardsByInn(string inn)
```

Параметры

string

Тип: `inn`

ИНН организации.

Возвращаемое значение

Тип: `List<RowData>`

Список карточек журнала обмена сообщениями

CreateLogTransferLogItem — метод (Guid, Guid, M4dRegistryAction)

Создает запись в журнале карточке журнала обмена сообщениями.

- **Пространство имён:** `DocsVision.M4dRegistry.ObjectModel.Services`
- **Сборка:** `DocsVision.M4dRegistry.ObjectModel.dll`

Синтаксис

```
Guid CreateLogTransferLogItem(Guid cardId, Guid employeeId, M4dRegistryAction action)
```

Параметры

Guid

Тип: `cardId`

ID карточки журнала обмена сообщениями.

Guid

Тип: `employeeId`

ID сотрудника, выполнившего запрос.

M4dRegistryAction

Тип: `action`

Действие, которое обрабатывает запрос.

Возвращаемое значение

Тип: `Guid`

ID созданной записи

CreateLogTransferLogItem — метод (Guid, Guid, M4dRegistryAction, M4dRegistryOperationStatus)

Создает запись в журнале карточке журнала обмена сообщениями.

- **Пространство имён:** `DocsVision.M4dRegistry.ObjectModel.Services`
- **Сборка:** `DocsVision.M4dRegistry.ObjectModel.dll`

Синтаксис

```
Guid cardId, Guid employeeId, M4dRegistryAction action, M4dRegistryOperationStatus status
```

Параметры

Guid

Тип: `cardId`

ID карточки журнала обмена сообщениями.

Guid

Тип: `employeeId`

ID сотрудника, выполнившего запрос.

M4dRegistryAction

Тип: `action`

Действие, которое обрабатывает запрос.

M4dRegistryOperationStatus

Тип: `status`

Результат выполнения действия.

Возвращаемое значение

Тип: `Guid`

ID созданной записи.

FindLogTransferLogItem — метод (Guid, M4dRegistryAction)

Ищет записи в карточке журнала обмена сообщениями.

- **Пространство имён:** `DocsVision.M4dRegistry.ObjectModel.Services`
- **Сборка:** `DocsVision.M4dRegistry.ObjectModel.dll`

Синтаксис

```
List<RowData> FindLogTransferLogItem(Guid cardId, M4dRegistryAction action)
```

Параметры

Guid

Тип: `cardId`

ID карточки, где выполняется поиск

M4dRegistryAction

Тип: `action`

Действие, которое ищем.

Возвращаемое значение

Тип: `List<RowData>`

Список строк журнала.

FindLogTransferLogItem — метод (Guid, M4dRegistryOperationStatus)

Ищет записи в карточке журнала обмена сообщениями.

- **Пространство имён:** DocsVision.M4dRegistry.ObjectModel.Services
- **Сборка:** DocsVision.M4dRegistry.ObjectModel.dll

Синтаксис

```
List<RowData> FindLogTransferLogItem(Guid cardId, M4dRegistryOperationStatus status)
```

Параметры

Guid

Тип: cardId

ID карточки, где выполняется поиск.

M4dRegistryAction

Тип: status

Статус операции который ищем.

Возвращаемое значение

Тип: List<RowData>

Список строк журнала.

FindLogTransferLogItem — метод (Guid, M4dRegistryAction, M4dRegistryOperationStatus)

Ищет записи в карточке журнала обмена сообщениями.

- **Пространство имён:** DocsVision.M4dRegistry.ObjectModel.Services
- **Сборка:** DocsVision.M4dRegistry.ObjectModel.dll

Синтаксис

```
List<RowData> FindLogTransferLogItem(Guid cardId, M4dRegistryAction action
M4dRegistryOperationStatus status)
```

Параметры

Guid

Тип: cardId

ID карточки? где выполняется поиск

M4dRegistryAction

Тип: `action`

Действие, которое обрабатывает запрос.

M4dRegistryAction

Тип: `status`

Статус операции который ищем.

Возвращаемое значение

Тип: `List<RowData>`

Список строк журнала.

GetAllLogTransferRows — метод (Guid)

Возвращает все строки из журнала карточки журнала обмена сообщениями.

- **Пространство имён:** `DocsVision.M4dRegistry.ObjectModel.Services`
- **Сборка:** `DocsVision.M4dRegistry.ObjectModel.dll`

Синтаксис

```
GetAllLogTransferRows(Guid attorneyId)
```

Параметры

Guid

Тип: `attorneyId`

ID доверенности (СКД).

Guid

Тип: `employeeId`

ID сотрудника, выполнившего запрос.

M4dRegistryAction

Тип: `action`

Действие, которое обрабатывает запрос.

Возвращаемое значение

Тип: `List<RowData>`

Список строк журнала.

IM4dRegistryManager — интерфейс

Предоставляет методы для взаимодействия с сервисом регистрации. Содержит методы инициализации контекста и выполнения событий для регистрации, проверки, отзыва доверенностей.

- **Пространство имён:** `DocsVision.M4dRegistry.ObjectModel.Services`
- **Сборка:** `DocsVision.M4dRegistry.ObjectModel.dll`

Синтаксис

```
public interface IM4dRegistryManager
```

Инициализация менеджера

```
Initialize(ObjectContext objectContext)
```

Методы

Имя	Описание
`Register(M4dRegistryEventArgs, string)`	Обрабатывает событие регистрации доверенности.
`Check(M4dRegistryEventArgs, string, int)`	Обрабатывает событие проверки статуса доверенности.
`ReCall(M4dRegistryEventArgs, string)`	Обрабатывает событие отзыва доверенности.

Примечания

Библиотека менеджера `M4dRegistry.KonturApi` отвечает непосредственно за взаимодействие с внешним сервисом регистрации Контур.Доверенность и реализует интерфейс `IM4dRegistryManager`.

Register — метод (M4dRegistryEventArgs, string)

Обрабатывает событие регистрации.

- **Пространство имён:** DocsVision.M4dRegistry.ObjectModel.Services
- **Сборка:** DocsVision.M4dRegistry.ObjectModel.dll

Синтаксис

```
Task<M4dRegistryCmdResult> Register(M4dRegistryEventArgs eventArgs, string apiKey)
```

Параметры

M4dRegistryEventArgs

Тип: eventArgs

Аргументы для обработки события.

string

Тип: apiKey

Ключ для доступа к сервису.

Возвращаемое значение

Тип: M4dRegistryCmdResult

Объект результата обработки сообщения.

Check — метод (M4dRegistryEventArgs, string, int)

Обрабатывает событие проверки.

- **Пространство имён:** DocsVision.M4dRegistry.ObjectModel.Services
- **Сборка:** DocsVision.M4dRegistry.ObjectModel.dll

Синтаксис

```
Task<M4dRegistryCmdResult> Check(M4dRegistryEventArgs eventArgs, string apiKey, int
pollingInterval = 0)
```

Параметры

M4dRegistryEventArgs

Тип: `eventArgs`

ID доверенности (СКД).

string

Тип: `apiKey`

Ключ для доступа к сервису.

int

Тип: `pollingInterval`

Интервал опроса сервиса.

Возвращаемое значение

Тип: `M4dRegistryCmdResult`

Объект результата обработки сообщения.

ReCall — метод (M4dRegistryEventArgs, string)

Обрабатывает событие отзыва.

- **Пространство имён:** `DocsVision.M4dRegistry.ObjectModel.Services`
- **Сборка:** `DocsVision.M4dRegistry.ObjectModel.dll`

Синтаксис

```
Task<M4dRegistryCmdResult> ReCall(M4dRegistryEventArgs eventArgs, string apiKey)
```

Параметры

M4dRegistryEventArgs

Тип: `eventArgs`

Аргументы для обработки события.

string

Тип: `apiKey`

Ключ для доступа к сервису.

Возвращаемое значение

Тип: `M4dRegistryCmdResult`

Объект результата обработки сообщения.

M4dRegistryCmdResult — класс

Ответ событий.

- **Пространство имён:** `DocsVision.M4dRegistry.ObjectModel.Services`
- **Сборка:** `DocsVision.M4dRegistry.ObjectModel.dll`

Синтаксис

```
public class ApprovalStageService : ContextService, IApprovalStageService
```

Поля

Имя	Описание
`Result`	Результат обработки события, может принимать одно из значений: * `Success` * `Error` * `Unknown`
`Details`	Подробности выполнения операции, обычно заполняется в случае ошибки и содержит исключение, возникшее в результате работы менеджера

Руководство разработчика службы фоновых операций

Объектная модель модуля

Модуль предоставляет собственную объектную модель для реализации компонентов и работы с данными карточек модуля.

В данном разделе перечислены основные члены объектной модели модуля Служба фоновых операций.

IConnectionSupport — интерфейс

Представлет фабрику, поддерживающую управление сроком жизни циклом соединения

- **Пространство имён:** `Docsvision.WorkerService.Interfaces`
- **Сборка:** `DocsVision.WorkerService.ObjectModel.dll`

Синтаксис

```
public interface IConnectionSupport
```

Методы

Имя	Описание
`SupportedTypes`	Получает типы для соединений, которые могут быть получены.
`CheckConnection<T>`	Проверяет, если соединение живое * `T` — Тип для приведения

IInitializeWithServiceProvider — интерфейс

Интерфейс для классов, поддерживающих инициализацию с провайдером сервиса

- **Пространство имён:** `Docsvision.WorkerService.Interfaces`

- **Сборка:** DocsVision.WorkerService.ObjectModel.dll

Синтаксис

```
public class
```

Методы

Имя	Описание
Initialize	serviceProvider — провайдер сервиса

IMessage — интерфейс

Интерфейс сообщений

- **Пространство имён:** Docsvision.WorkerService.Interfaces
- **Сборка:** DocsVision.WorkerService.ObjectModel.dll

Синтаксис

```
public interface IMessage
```

Свойства

Имя	Описание
Id	Идентификатор сообщения Тип: System.Guid
TypeId	Идентификатор типа сообщения Тип: System.Guid
SubTypeId	Дополнительный идентификатор, уточняющий тип сообщения Тип: System.Guid

Имя	Описание
`TargetId`	Идентификатор цели сообщения Тип: `System.Guid`
`SourceId`	Идентификатор источника сообщения Тип: `System.Guid`
`Data`	Дополнительные данные сообщения Тип: `System.Object`

IRoleTemplateProvider — класс

Представляет шаблон поставщика роли

- **Пространство имён:** `Docsvision.WorkerService.Interfaces`
- **Сборка:** `DocsVision.WorkerService.ObjectModel.dll`

Синтаксис

```
public interface IRoleTemplateProvider
```

Методы

Имя	Описание
`SendMessage(Messages, MessagesOutgoingMessage);`	Получает шаблоны роли Тип: `RoleTemplate`

M4DSettings — класс

Класс для работы с МЧД

- **Пространство имён:** `Docsvision.WorkerService.Interfaces`
- **Сборка:** `DocsVision.WorkerService.ObjectModel.dll`

Синтаксис

```
public class M4DSettings
```

Свойства

Имя	Описание
`Id`	Задаёт или получает идентификатор коннектора. Тип: `System.Guid`
`DisplayName`	Задаёт или получает имя коннектора. Тип: `System.String`
`Endpoint`	Задаёт или получает URL запроса коннектора. Тип: `System.String`
`ApiKey`	Задаёт или получает API-ключ (`SecretKey`) запроса коннектора. Тип: `System.String`
`PollingInterval`	Задаёт или получает интервал отправки запросов к коннектору. Тип: `System.Int32`
`AppClientId`	Задаёт или получает `AppClientId`. Тип: `System.String`
`DiskUrl`	Задаёт или получает `DiskUrl`. Тип: `System.String`
`AppSecret`	Задаёт или получает `AppSecret`. Тип: `System.String`

Messages — класс

Класс для хранения коллекций сообщений, подписок и токенов.

- **Пространство имён:** `Docsvision.WorkerService.ObjectModel`
- **Сборка:** `DocsVision.WorkerService.ObjectModel.dll`

Синтаксис

```
public sealed class Messages : ObjectBase
```

Свойства

Имя	Описание
`MainInfo`	Возвращает основную информацию о сообщении
`IncomingMessages`	Входящие сообщения для владельца карточки
`OutgoingMessages`	Исходящие сообщения от владельца карточки
`Subscriptions`	Подписки на сообщения от других объектов от владельца карточки
`ActivateTokens`	Токены активации

MessagesMainInfo — класс

Основная информация о сообщении

- **Пространство имён:** `Docsvision.WorkerService.ObjectModel`
- **Сборка:** `DocsVision.WorkerService.ObjectModel.dll`

Синтаксис

```
public sealed class MessagesMainInfo : ObjectBase
```

Свойства

Имя	Описание
TypeId	Идентификатор типа Тип: System.Guid
ObjectId	Идентификатор объекта Тип: System.Guid

MessagesActivateToken — класс

Токены активации.

- **Пространство имён:** Docsvision.WorkerService.ObjectModel
- **Сборка:** DocsVision.WorkerService.ObjectModel.dll

Синтаксис

```
public class MessagesActivateToken : ObjectBase
```

Свойства

Имя	Описание
Date	Дата и время токена Тип: System.DateTime

MessagesIncomingMessage — класс

Входящие сообщения для владельца карточки.

- **Пространство имён:** Docsvision.WorkerService.ObjectModel
- **Сборка:** DocsVision.WorkerService.ObjectModel.dll

Синтаксис

```
public sealed class MessagesIncomingMessage : ObjectBase
```

Свойства

Имя	Описание
`CreateDate`	Дата создания сообщения Тип: `System.DateTime`
`TypeId`	Идентификатор типа сервиса сообщения Тип: `System.Guid`
`SubTypeId`	Дополнительный идентификатор, уточняющий тип сообщения Тип: `System.Guid`
`Data`	Данные сообщения Тип: `System.String`
`State`	Состояние обработки сообщения Тип: `MessageState`
`Concurrent`	Определяет, может ли сообщение обрабатываться одновременно с другими сообщениями для целевого объекта\службы Тип: `System.Boolean`
`ErrorCount`	Количество ошибок во время обработки сообщения Тип: `System.Int32`
`WaitingLocks`	Блокировки ожидания Тип: `MessagesIncomingMessageWaitingLock`

MessagesIncomingMessage — класс

Ожидание блокировок входящего сообщения

- **Пространство имён:** Docsvision.WorkerService.ObjectModel
- **Сборка:** DocsVision.WorkerService.ObjectModel.dll

Синтаксис

```
public sealed class MessagesIncomingMessageWaitingLock : ObjectBase
```

Свойства

Имя	Описание
ObjectId	Ожидающий блокировки объект Тип: System.Guid
WaitBeginDate	Дата начала ожидания блокировки Тип: System.DateTime

MessagesOutgoingMessage — класс

Исходящее сообщение

- **Пространство имён:** Docsvision.WorkerService.ObjectModel
- **Сборка:** DocsVision.WorkerService.ObjectModel.dll

Синтаксис

```
public sealed class MessagesOutgoingMessage : ObjectBase
```

Свойства

Имя	Описание
TargetId	Идентификатор цели сообщения Тип: System.Guid
TypeId	Идентификатор типа сервиса сообщения Тип: System.Guid

Имя	Описание
SubTypeId	Дополнительный идентификатор, уточняющий тип сообщения Тип: System.Guid
Data	Дата сообщения Тип: System.DateTime
Concurrent	Определяет, может ли сообщение обрабатываться одновременно с другими сообщениями для целевого объекта\службы Тип: System.Boolean
AutoSendToSelf	Автоматически послать сообщение себе без явной подписки Тип: System.Boolean
Delayed	Отправка сообщения может быть отложена Тип: System.Boolean
DelayedTill	Отправка сообщения отложена до указанной даты Тип: System.DateTime
CreateDate	Дата создания сообщения Тип: System.DateTime

MessagesSubscription — класс

Подписки на сообщения

- **Пространство имён:** Docsvision.WorkerService.ObjectModel
- **Сборка:** DocsVision.WorkerService.ObjectModel.dll

Синтаксис

```
public sealed class MessagesSubscription : ObjectBase
```

Свойства

Имя	Описание
`TypeId`	Идентификатор типа сервиса сообщения Тип: `System.Guid`

WorkerServiceServiceFactory — класс

Фабрика для приложения, создаёт экземпляры сервисов `MessagesService`, управляющие операциями с сообщениями и подписками.

- **Пространство имён:** `Docsvision.WorkerService.ObjectModel.Services`
- **Сборка:** `DocsVision.WorkerService.ObjectModel.dll`

Синтаксис

```
public sealed class WorkerServiceServiceFactory : ServiceFactory
```

Методы

Имя	Описание
`GetService(Type)`	Возвращает ссылку на заблокированный объект по его идентификатору. *Параметры:* • `serviceType` — тип сервиса **Возвращаемое значение** Тип: `object` Экземпляр сервиса

RoleTemplate — класс

Представляет шаблон роли

- **Пространство имён:** `Docsvision.WorkerService.Interfaces`
- **Сборка:** `DocsVision.WorkerService.ObjectModel.dll`

Синтаксис

```
public class RoleTemplate
```

Методы

Имя	Описание
`RoleTemplate`	Инициализирует новый экземпляр класса `RoleTemplate`.
`Name`	Получает или задаёт имя роли.

MessageState — перечисление

Состояние сообщения.

- **Пространство имён:** `Docsvision.WorkerService.Interfaces`
- **Сборка:** `DocsVision.WorkerService.ObjectModel.dll`

Синтаксис

```
public enum MessageState
```

Имя члена	Описание
`Unhandled = 0`	*Не обработано* — сообщение ещё не обрабатывалось.
`Handled = 1`	*Обработано* — сообщение было обработано.
`Error = 2`	*Ошибка* — обработка сообщения завершилась с ошибкой.

Имя члена	Описание
`Failed = 3`	Неудачно
`Paused = 4`	*Блокировано* — при обработке сообщения возникла ошибка, но обработка не завершена.

IMessagesService — интерфейс

Сервис для работы с сообщениями и подписками.

- **Пространство имён:** `Docsvision.WorkerService.ObjectModel.Services`
- **Сборка:** `DocsVision.WorkerService.ObjectModel.dll`

Синтаксис

```
public interface IMessagesService
```

Методы

Имя	Описание
`SendMessage(Messages, MessagesOutgoingMessage);`	Отправляет сообщение *Параметры:* • `messages` — сообщения Тип: `Messages` Карточка сообщения • `message` — сообщение Тип: `MessagesOutgoingMessage` Карточка сообщения

Имя	Описание
`CreateSubscription(Messages, MessagesSubscription);`	Создаёт подписку на сообщения *Параметры:* • `messages` — сообщения Тип: `Messages` • `subscription` — подписка Тип: `MessagesSubscription`
`RemoveSubscription(Messages, MessagesSubscription);`	Удаляет подписку на сообщения *Параметры:* • `messages` — сообщения Тип: `Messages` • `subscription` — подписка Тип: `MessagesSubscription_CL`
`CreateMessagesCard(Guid, Guid);`	Создаёт карточку сообщения *Параметры:* • `objectId` — идентификатор объекта Тип: `System.Guid` • `typeId` — идентификатор типа Тип: `System.Guid` **Возвращаемое значение** Тип: `Messages` Карточка сообщения

Имя	Описание
`CreateWaitingLock(MessagesIncomingMessage, MessagesIncomingMessageWaitingLock);`	Создаёт блокировку ожидания *Параметры:* • `messagesIncomingMessage` — входящее сообщение Тип: `MessagesIncomingMessage` • `messageWaitingLock` — блокировка ожидания Тип: `MessagesIncomingMessageWaitingLock`
`RemoveWaitingLock(MessagesIncomingMessage, MessagesIncomingMessageWaitingLock);`	Удаляет блокировку ожидания *Параметры:* • `messagesIncomingMessage` — входящее сообщение Тип: `MessagesIncomingMessage` • `messageWaitingLock` — блокировка ожидания Тип: `MessagesIncomingMessageWaitingLock`
`ResumePausedMessages(Messages);`	Возобновляет приостановленные сообщения *Параметры:* • `messages` — сообщения Тип: `Messages`

Имя	Описание
`SendMessageImmediate(Messages, MessagesOutgoingMessage);`	Отправляет сообщения немедленно *Параметры:* • `messages` — сообщения Тип: `Messages` • `message` — сообщение Тип: `MessagesOutgoingMessage`