ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»

(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ДОКУМЕНТООБОРОТА ДЛЯ АДМИНИСТРАЦИИ СЕЛЬСКОГО ПОСЕЛЕНИЯ С.ВЕЛИКОМИХАЙЛОВКА

Выпускная квалификационная работа
обучающегося по направлению подготовки 02.03.02 Фундаментальная информатика и информационные технологии
заочной формы обучения, группы 07001360
Вознюк Дмитрия Константиновича

Научный руководитель
к.т.н., доцент
Чашин Ю.Г.

БЕЛГОРОД 2017

ОГЛАВЛЕНИЕ

ВЕДЕНИЕ

В настоящей дипломной работе рассматривается деятельность администрации сельского поселения с. Великомихайловка.

Предметом дипломной работы является система автоматизации документоооборота администрации сельского поселения с.Великомихайловка.

Объектом дипломной работы является система, средствами которой осуществляется разработка прикладного решения автоматизирующего деятельность администрации.

При решении поставленных задач в процессе работы использовались методы:

- аналитический метод;
- статистический метод;
- сравнительный метод;
- методы сбора и обработки данных.

Практическая значимость проекта заключается в современности и актуальности используемых технологий, благодаря которым решались поставленные задачи, в оптимизации множества процессов и новизне приводимой информации. Основные положения данного дипломного проекта доведены до возможных рекомендаций администрации пожелавшим вывести свой бизнес в интернет.

Разработка и проектирование автоматизированной информационной системы начинается с создания концептуальной модели использования системы. Прежде всего должна быть определена целесообразность создания системы, ее конкретные функции и подлежащие автоматизации задачи. Должна быть выполнена оценка не только целей, но и возможностей создания системы. Далее проводится анализ требований к

автоматизированной информационной системе, детальное проектирование, взаимосвязь этапов, программирование и тестирование, минимизация потерь при переходе от одного уровня представления информации к другому, интеграция в существующую систему, внедрение и поддержка.

Существует три класса методологий проектирования автоматизированной информационной системы:

- концептуальное моделирование предметной области;
- выявление требований и спецификация информационной системы через ее макетирование;
- системная архитектура программных средств, поддерживаемая инструментальными средствами CASE-технологии (CASE — Computer Aided Software Engineering — технология создания и сопровождения ПО различных систем).

Современные методологии проектирования систем должны обеспечивать описание объектов автоматизации, описание функциональных возможностей автоматизированной системы, спецификацию проекта, гарантирующую достижение заданных характеристик системы, детальный план создания системы с оценкой сроков разработки, описание реализации конкретной системы.

Спецификация — точное, полное, ясно сформулированное описание требований для данной задачи.

В основе создания и использования автоматизированной информационной системы лежит понятие жизненного цикла.

Жизненный цикл является моделью создания и использования автоматизированной информационной системы, которая отражает различные состояния системы с момента возникновения в данном комплексе средств до момента его полного выхода из употребления.

Для автоматизированной системы условно выделяют следующие основные этапы их жизненного цикла:

1) анализ — определение того, что должна делать система;

2) проектирование — определение того, как система будет функционировать: прежде всего спецификация подсистем, функциональных компонентов и способов их взаимодействия в системе;

3) разработка — создание функциональных компонентов и отдельных подсистем, соединение подсистем в единое целое;

4) тестирование — проверку функционального и параметрического соответствия системы показателям, определенным на этапе анализа;

5) внедрение — установку и ввод системы в действие;

6) сопровождение — обеспечение штатного процесса эксплуатации системы на предприятии заказчика.

Цель работы – разработка автотизированной системы документооборота, для администрации сельского поселения с. Великомихайловка, в конечном итоге позволяющей снизить издержки администрации на данный процесс.

Задачи работы:

1) Описать предметную область;

2) Рассмотреть особенности использования системы «1С: Предприятие»;

3) Разработать автоматизированную систему документооборота администрации;

Выпускная квалификационная работасодержит 104 страницы, 77 рисунка и приложений на 27 страницах.

ГЛАВА 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Организационная структура администрации

Под организационной структурой администрации понимаются состав, соподчиненность, взаимодействие и распределение работ по подразделениям и органам управления, между которыми устанавливаются определенные отношения по поводу реализации властных полномочий, потоков команд и информации (см. рис.1.1).

Рис. 1.1. Организационная структура администрации сельского поселения с. Великомихайловка

1.1.1 Выбор комплекса задач автоматизации и характеристика существующих бизнес процессов

Под документооборотом понимается движение документов на предприятии с момента их создания или получения до завершающего исполнения или отправки.

Как правило, документы различаются по типам носителей информации (см. рис. 1.2).

Рис. 1.2. Типы документов и их взаимодействие

Для любой организации можно выделить три основных потока документов (таблица 1.1):

1) Входящие. Входящий документ – документ, поступивший в учреждение. Большинство входящих документов должны порождать соответствующие исходящие, в установленные сроки. Сроки могут быть установлены нормативными актами, предписывающими то или иное время ответа на соответствующий входящий документ, или могут быть указаны непосредственно во входящем документе.

2) Исходящие. Исходящий документ – официальный документ, отправляемый из учреждения. Большинство исходящих документов являются ответом организации на входящие документы. Некоторая часть исходящих документов готовится на основе внутренних документов организации.

Небольшое число исходящих документов может требовать поступления входящих документов (например, запросы в другие организации).

3) Внутренние. Внутренний документ – официальный документ, не выходящий за пределы подготовившей его организации. Данные документы используются для организации работы учреждения (организации), так как они обеспечивают целенаправленное решение управленческих задач в пределах одной организации.

Таблица 1.1

Традиционная работа с документами в организации без использования автоматизированной системы документооборота

Форма документа	Документопоток		
	входящей информации	внутренних документов	исходящей информации
Электронные документы	Сообщения электронной почты, факсимильная информация	Сообщения в корпоративной сети, факсы	Ответы и письма по электронной почте, факсимильная информация
Бумажные документы	Письма, договоры и контракты, законодательные акты, нормативные документы, периодические издания, Книги, реклама	Приказы, инструкции, отчеты, служебные записки, командировочные документы, бухгалтерские документы	Письма, договоры и контракты, пресс – релизы

От правильной постановки документооборота в организации зависит оперативность, экономичность, надежность функционирования аппарата управления, организация и культура труда персонала управления.

Документооборот в организациях имеет три основные системы:

1. централизованную;

2. децентрализованную;

3. смешанную.

Выбор системы документооборота зависит от характера деятельности, функций, организационной структуры организации.

При централизованной системе все операции, связанные с документационным обеспечением управления сосредотачиваются в одном месте, в единой для всей организации службе или у секретаря (прием документов, их регистрация, контроль исполнения, хранение дел и др.), что позволяет значительно повысить качествообработки. При децентрализованной системе все видыработ с документами производятся непосредственно в структурных подразделениях организации, то есть документооборот каждое функциональное подразделение ведет самостоятельно

При смешанной системе документооборота одни операции (прием, отправка корреспонденции) осуществляются в одной из служб ДОУ, а другие (регистрация, оформление, составление документов, формирование дел) производятся в других, функциональных подразделениях.

Порядок движения документов в организации включает в себя организацию документооборота, включая технологию личной работы исполнителей, создание информационно–поисковых систем по документам организации, контроль их исполнения. Процесс движения документов от момента их создания (получения) до момента передачи в архив, можно представить в виде схем, приведенных нарис.1.3.

Рис.1.3. Жизненный цикл документа

Цели автоматизации документооборота организации.Безбумажный документооборот предусматривает применение компьютера в управленческой деятельности для хранения, поиска и отображения информации, позволяя свести к минимуму или исключить полностью применение бумажных носителей.

Необходимость автоматизации документооборота продиктована проблемами, возникающими в работе компаний всех организационно-правовых форм, которые по сути своей идентичны:

- потеря документа, либо его долгий поиск;
- поиск документов в рамках организации затруднен из-за децентрализованной регистрации;

- задержки прохождения, исполнения документов (сложные схемы согласования документов и человеческий фактор);
- фактическая бесконтрольность исполнения (отсутствие информации об исполнении и исполнителе);
- проблемы контроля исполнения документа (отсутствие или плохая организация упреждающего контроля);
- неисполнение документов или задержка их исполнения; избыточность документооборота (в том числе дублирование документов);
- большой штат сотрудников, работающих с документами (рутинные технологические операции).
- цели автоматизации документооборота организации, вне зависимости от организационно-правовых форм, довольно схожи и заключаются в следующем:
- повышение качества и оперативности управления, и как следствие этого обеспечение конкурентоспособности администрации на рынке;
- объединение в единый документооборотенный цикл всех структурных подразделений организации, включая территориально-удаленные;
- обеспечение оперативного, и в то же время разграниченного доступа к информационным (документационным) ресурсам организации;
- снижение трудовых и временных затрат и накладных расходов, и как следствие, получение экономического эффекта;
- заложение основы постепенного перехода к электронному документообороту на предприятии, работа на перспективу.

Достижение поставленных целей возможно посредством решения задач автоматизации документооборота организации, которые условно можно систематизировать по следующим областям:

Подготовка и оформление документов.

- повышение качества и оперативности подготовки документов;

- унификация процесса работы с документами на всех этапах его существования.

Организация документооборота и исполнения документов.

- исключение дублирования ввода информации о документе на различных этапах работы с ним (процесс регистрации);
- исключение возможности потери документа (создание документальной базы организации);
- упорядочение документооборота организации (упрощение схем прохождения документов – маршрутизация);
- повышение оперативности и качества работы исполнителей с документами;
- сокращение сроков исполнения, прохождения документов;
- своевременное информирование сотрудников и руководства о поступивших и создаваемых документах (исключение дублирования работы над одним и тем же документом).

Организация контроля исполнения документов.

- объединение документационных потоков всех подразделений организации, в том числе территориально-удаленных;
- оперативное получение информации о состоянии исполнения и месте нахождения любого документа;
- обеспечение отслеживания этапов прохождением документов в подразделениях организации с момента их получения/создания до завершения работы с ними (исполнения).

Организация хранения документов, поисковая система.

- обеспечение централизованного хранения текстов документов, подготовленных в электронном виде, их графических образов и материалов к ним;
- обеспечение возможности оперативного поиска и организации логического связывания документов, относящихся к одному вопросу;

- обеспечение оперативного поиска и подборки документов (материалов) по тематическому набору реквизитов.

Таким образом, суммируя все выше сказанное можно сделать вывод, что автоматизация документационного обеспечения управления организации, вне зависимости от организационно-правовых форм, ввиду комплексного подхода к решению проблем документооборота, повышает оперативность управления, эффективность работы ее сотрудников, следовательно, приводит к повышению конкурентоспособности на рынке.

История развития информационных систем в управлении документами.

Огромное увеличение объемов информации и большие изменения спроса информации стали предъявлять новые требования к организации информационно-документационного обслуживания в организации. Стали меняться требования к службам, занимающимся информационно-документационным обеспечением. За рубежом их стали называть службой управления информационно-документационными ресурсами, в нашей стране – службой документационного обеспечения управления.

1.1.2Определение места проектируемой задачи в комплексе задач и ее описание

Отразим действующие процессы документооборота в организации с использованием средств BPwin. Для отражениея этих процессов используем модель AS-IS.

Процесс: Регистрация документа.

Вход: Входящий документ.

Выход: Зарегистрированный входящий документ.

Алгоритм: Зарегистрировать поступивший входящий документ, присвоив ему уникальный номер, в соответствии с Инструкцией по делопроизводству и внутренним Порядком документооборота.

Процесс: Обработка документа.

Вход: Зарегистрированный входящий документ.

Выход: Исполненный входящий документ, исходящий ответный документ.

Алгоритм: Исполнить поступивший документ и, если требуется, подготовить и отправить ответный исходящий документ.

Подпроцессы:

Вынесение резолюций.

Постановка на контроль.

Исполнение документа.

Снятие с контроля.

Процесс: Списание в дело.

Вход: Исполненный входящий документ.

Выход: Списанный документ.

Алгоритм: Подшить обработанный документ в дело с соответствующей номенклатурой.

Процесс: Формирование отчетов.

Вход: Списанный документ, исходящий ответный документ.

Выход: Реестр зарегистрированных входящих/ исходящих документов, Реестр документов, переданных на исполнение, Сводка об исполнении документов.

Алгоритм: Ручное формирование необходимых отчетов.

Подпроцессы:

Поиск форм отчетов.

Ручное заполнение форм отчетов.

Печать отчетов.

Эта модель показывает, как функционирует документооборот в организации, когда процессы не автоматизированы.

После внедрения проектируемой информационной системы, организация документооборота будет представлена в виде модели TO-BE.

Процесс: Автоматизированное рабочее место «Делопроизводство».

Вход: Входящие документы.

Выход: Автоматизированные отчеты.

Подпроцессы:

АРМ «Делопроизводство: работа с документами».

АРМ «Делопроизводство: отчетность».

Процесс: АРМ «Делопроизводство: работа с документами».

Вход: Входящие документы.

Выход: Исполненные документы, исходящие ответные документы.

Алгоритм: Автоматизированный учет документов и контроль за их исполнением.

Подпроцессы:

Регистрация в АРМ «Делопроизводство».

Рассмотрение документа.

Исполнение документа.

Процесс: АРМ «Делопроизводство: отчетность».

Вход: Исполненные документы, исходящие ответные документы.

Выход: Автоматизированные отчеты.

Алгоритм: Автоматизирование формирование отчетов и вывод их на бумажные носители.

Подпроцессы:

Автоматическая загрузка данных в шаблоны отчетов.

Вывод отчетов.

1.1.4 Требования к системе

С началом перехода к единому информационному пространству, организацией межведомственного электронного документооборота необходимость унификации СЭД, обеспечения их совместимости с общегосударственными системами обмена документами, электронного взаимодействия и архивного хранения выходят на первый план. Частично на

решение вопросов взаимодействия систем СЭД направлен ГОСТ Р 53898-2010. «Системы электронного документооборота. Взаимодействие систем управления документами. Требования к электронному сообщению».

«Требования к информационным системам электронного документооборота» предназначены для федеральных органов исполнительной власти, но в соответствии со ст. 11 Федерального закона № 149-ФЗ от 27.07.2006 распространяются и на иные государственные органы и органы местного самоуправления. Коммерческие организации имеют право организовывать СЭД по собственному усмотрению, но, учитывая роль государства в нашей стране, обычно все крупные и средние коммерческие организации ориентируются на правила, установленные государством для удобства взаимодействия с государственными органами.

Эти требования носят рамочный характер и поэтому в 2013 г. по заказу Федерального архивного агентства Всероссийским научно-исследовательским институтом документоведения и архивного дела (ВНИИДАД) были разработаны «Архивоведческие и документоведческие функциональные требования к информационным системам, обеспечивающим электронный документооборот в процессе внутренней деятельности федеральных органов исполнительной власти».

«Требования к информационным системам электронного документооборота…» определяют минимальный набор функций, который должен присутствовать в СЭД, а также требования к организации использования СЭД в учреждении.

Одно из главных требований к СЭД – это ее масштабируемость как по числу подключенных рабочих мест, так и по количеству содержащихся в СЭД документов. Надо учитывать, что современные системы управления документами используются практически всеми сотрудниками организации, работающими с документами, причем общая тенденция – это использование как стационарных рабочих мест, так и доступа к документам с мобильных устройств, удаленный доступ к системе. По количеству хранящихся в СЭД

документов следует иметь в виду, что так как в системе хранятся не только окончательные оформленные и подписанные документы, но и промежуточные рабочие версии, то количество файлов, проектов документов и документов, поступающих в СЭД, за год в несколько раз превышает суммарное количество документов, регистрируемых службой ДОУ (входящих, исходящих и внутренних). Требования предусматривают, что СЭД должна обеспечивать хранение всех документов за период не менее 5 лет, но на практике приходится ориентироваться на сроки не менее 10-15 лет, так как это тот период, в течение которого документы в электронной форме продолжают активно использоваться, тем более что п. 20 пп. е) тех же Требований предусматривает возможность хранения документов в сроки до ста лет.

Важный параметр СЭД – ее быстродействие. Если аппаратно-программный комплекс (сервер СЭД) недостаточно производительный для данного количества одновременно работающих в системе пользователей и (или) для данного объема базы данных (количества документов в системе), то сотрудникам придется ждать открытия карточки документа или самого документа, следовательно, производительность сотрудников падает. Поэтому в Требованиях заложены временные параметры, которым должна соответствовать производительность СЭД:

- время получения доступа к СЭД - не более трех секунд;
- время получения доступа к карточке, создаваемой при регистрации документа и содержащей данные, описывающие контекст, содержание, структуру документа, действия, совершенные с документом в ходе подготовки, рассмотрения, исполнения и хранения, а также идентификационные данные (метаданные) - не более пяти секунд.

В любой системе может случиться сбой как программный, так и аппаратный. Но сбой СЭД приводит к невозможности работы с документами всех сотрудников организации, поэтому Требования устанавливают жесткие рамки для времени простоя при сбоях и перезагрузке СЭД - не более 30

минут. Также СЭД должна обеспечивать автоматическое уведомление пользователей о сбое в работе системы. В первую очередь обычно настраивают автоматическое уведомление через SMS и по электронной почте администратора и технолога СЭД.

Еще одна распространенная ситуация – по какой-то причине документ поврежден или случайно стерт пользователем. Требования предусматривают, что в этом случае в течение 30 минут электронный документ должен быть восстановлен из резервной копии. В организации, в соответствии с Требованиями, должно быть не менее одной резервной копии электронных документов, хранящихся в СЭД. Однако на практике для обеспечения сохранности создают не менее двух резервных копий, желательно на различных носителях. Это минимизирует риски потери электронных документов.

Коэффициент надежности СЭД должен составлять не менее 0,98.

Еще один показатель – это уровень защищенности СЭД от несанкционированного доступа. Для государственных учреждений, работающих с документами ограниченного доступа, это должна быть сертификация не ниже класса 1Г. Однако в виду высокой стоимости создания и эксплуатации защищенных СЭД, с документами ограниченного доступа обычно стараются работать в традиционном режиме, на бумаге, так как они, как правило, составляют небольшую долю документов организации. В противном случае обычно для работы с такими документами устанавливают специально выделенные компьютеры или даже отдельную защищенную сеть, не имеющую соединения с открытой компьютерной сетью и Интернет. Однако и в этом случае предусматривается работа с документами уровня ДСП, но никак не с документами, содержащими государственную тайну.

Основная часть «Требований к информационным системам электронного документооборота…» - это описание того, как в СЭД должны быть построены процессы документационного обеспечения управления.

Подчеркивается, что СЭД должна обеспечивать работу со всеми видами и категориями документов и проектами документов организации.

СЭД, используемые государственными учреждениями, должны обеспечивать взаимодействие с системами межведомственного электронного документооборота (МЭДО), межведомственного электронного взаимодействия (СМЭВ), другими информационными системами.

Работа СЭД должна соответствовать положениям ГОСТ Р ИСО 15489-1-2007 «Система стандартов по информации, библиотечному и издательскому делу. Управление документами. Общие требования», в том числе в области обеспечения аутентичности, целостности и достоверности электронного документа, а также Правилам делопроизводства в федеральных органах исполнительной власти, утвержденным Постановлением Правительства РФ от 15.06.2009 № 477 (пп. 9 и 11 Требований).

СЭД должна обеспечивать все основные делопроизводственные процессы:

Сохранение документа или сведений о документе (проекте документа) в СЭД (его регистрация или, в терминах Требований – ввод документа в систему):

- доведение документа до исполнителя (пользователя СЭД);
- согласование документа;
- подписание документа;
- передачу (отправку) документа;
- «хранение и учет документов, в соответствии с инструкцией по делопроизводству в ФОИВ, а также контроль исполнительской дисциплины, подготовку справочных материалов и списание документов в архив», то есть контроль исполнения, информационно-справочную работу, текущее хранение и учет, включая подготовку документов для передачи в государственный архив или на депозитарное хранение.

Особенность автоматизированной системы делопроизводства – это наличие функции протоколирования всех действий пользователей и

системных событий. Другими словами, все, что происходит в СЭД – создается или регистрируется документ, просто просматривается файл, вносится правка – вся эта информация сохраняется в специальных служебных файлах, что позволяет всегда сказать, кто и когда смотрел или правил документ (карточку документа). Отдельно в Требованиях прописана обязательность фиксации даты и времени ввода документа в систему. Эта информация фиксируется как в регистрационной карточке (метаданных по документу), так и в контрольной информации (протоколе действий в СЭД).

В соответствии с п. 17 Требований протоколированию подлежит информация обо всех действиях, совершенных с документами или наборами документов, проектами документов, регистрационной карточкой (метаданными). Это сведения:

- о пользователе СЭД ФОИВ, выполнившим действие;
- о дате и времени совершения действия;
- о вводе в СЭД документов, проектов документов;
- о перемещении раздела (подраздела) в классификационной схеме;
- об изменениях в указаниях по срокам хранения и последующих действиях с документами;
- о действиях, выполненных администратором СЭД ФОИВ в ходе экспертизы ценности документа, проводимой в соответствии с Федеральным законом от 22.10.2004 № 125-ФЗ "Об архивном деле в Российской Федерации";
- о наложении и снятии запрета на уничтожение раздела (подраздела) классификационной схемы;
- о любом изменении или уничтожении метаданных пользователем СЭД;
- об изменениях прав доступа к документам;
- о передаче документов;
- об уничтожении документов;
- о печати документа или метаданных.

Другими словами, СЭД должна позволять в любой момент получить информацию, кто и когда открывал, просматривал, редактировал документ или регистрационную карточку к нему, а также, с какими документами работал тот или иной сотрудник.

Требования Минкомсвязи РФ делят делопроизводственные процессы, поддерживаемые СЭД, на следующие группы:

а) обработка входящих и исходящих документов на бумажном носителе, созданных или поступивших в организацию и включенных в СЭД ФОВ путем регистрации, сканирования и создания электронного образа документов (включая документы, полученные посредством почтовой связи, электросвязи и фельдъегерской связи);

б) обработка электронных документов, полученных или переданных по системе межведомственного электронного документооборота;

в) обработка электронных документов, полученных или переданных с использованием системы межведомственного электронного взаимодействия;

г) обработка электронных документов, полученных или переданных по электронной почте;

д) обработка внутренних документов в СЭД.

В организациях, не являющихся государственными органами, пункты б) и в) отсутствуют, документы поступают только или по традиционным каналам связи, или по электронной почте.

В случае поступления документа на бумаге ввод документа в СЭД включает его регистрацию, сканирование и создание электронного образа документа.

В случае поступления документа в электронной форме ввод документа в СЭД – это его загрузка в СЭД, регистрация с запретом внесения изменений в поступивший документ.

В организации может быть утвержден и включен в инструкцию по делопроизводству перечень документов, для которых запрещено создание их электронных образов, например, документы с грифом ДСП, с пометкой

«личное», конфиденциальные документы и т. п. В случае поступления такого документа он регистрируется в СЭД, но его электронный образ не создается.

Для проектов электронных документов на каждом этапе их создания, согласования и подписания осуществляется фиксация содержимого документа путем создания версий документов и их прикрепления к карточке документа.

СЭД должна поддерживать прикрепление к регистрационной карточке любых форматов файлов. Это важно, так как СЭД обычно используется много лет и за это время могут появиться новые версии программ и, соответственно, форматы файлов, которые должны будут также поддерживаться СЭД. СЭД должна позволять вводить в систему и регистрировать файлы документов даже в том случае, когда приложение, в котором был создан документ, на данном рабочем месте отсутствует (не установлено). При этом некоторые, наиболее распространенные форматы, СЭД должна уметь отображать обязательно. Это pdf, rtf, doc, tiff.

СЭД должна позволять размещать документы в иерархической схеме, состоящей из разделов и подразделов, в соответствии с которой организуется систематизация и организация хранения документов в СЭД (классификационная схема). Следует иметь в виду, что физически документы размещаются на сервере, системе хранения, в порядке, определяемом внутренней конфигурацией и принципами хранения файлов в данной СЭД, а классификационная схема – это просто поле в регистрационной карточке, позволяющее быстро находить документы по классификационным признакам.

В основу классификационной схемы обычно кладут номенклатуру дел организации.

В регистрационной карточке СЭД должны быть определены те поля, которые обязательны для заполнения. При вводе документа СЭД должна запрашивать у пользователя заполнение обязательных полей (метаданных) (п. 13 Требований).

В ходе работы с документом в СЭД могут вводиться не только резолюции, но и комментарии и поручения по документу. Для подписания (а при необходимости и согласования) документа СЭД предусматривает возможность подключения средств электронной подписи в соответствии с Федеральным законом «Об электронной подписи».

При отправке документов традиционными способами (на бумаге) СЭД обеспечивает надпечатку конвертов и печать списков рассылки.

Сроки хранения документов, включенных в соответствующие разделы (подразделы), устанавливаются в соответствии с Перечнем типовых управленческих архивных документов, образующихся в процессе деятельности государственных органов, органов местного самоуправления и организаций, с указанием сроков хранения, утвержденным Приказом Министерства культуры РФ от 25.08.2010 № 558.5

В соответствии с установленными сроками хранения СЭД должна обеспечивать следующие действия:

1) хранить документ постоянно;

2) провести экспертизу ценности документов;

3) по завершении календарного года создавать документы по установленной форме: акт о выделении к уничтожению документов (разделов) с истекшими сроками хранения и описи на документы постоянного и долговременного (свыше 10 лет) срока хранения;

4) выделять документы к уничтожению (удалять из системы) с сохранением в СЭД информации о выделении документов к уничтожению;

5) передавать документы на хранение в другое хранилище (автоматизированную систему), в том числе экспортировать годовые разделы документов постоянного срока хранения для передачи на хранение в государственные архивы и экспортировать годовые разделы документов по личному составу для передачи в архивы документов по личному составу.

На практике передача на государственное хранение требует обеспечения совместимости СЭД по формату экспорта годового раздела с

программным комплексом «Архивный фонд», используемым в государственных и муниципальных архивах.

В Требованиях содержится положение об обеспечении сроков хранения с длительностью не менее чем до ста лет. Однако в настоящее время такие технологии находятся в стадии проработки, и автору не известна ни одна СЭД, которая могла бы обеспечить сама по себе такие большие сроки хранения юридически значимых документов в электронной форме.

Рассмотренные Требования Минкомсвязи РФ дополняют разработанные ВНИИДАД «Архивоведческие и документоведческие функциональные требования к информационным системам, обеспечивающим электронный документооборот в процессе внутренней деятельности федеральных органов исполнительной власти». Они важны как для работников делопроизводственных служб, так и для сотрудников IT-подразделений, обеспечивающих внедрение или настройку систем электронного делопроизводства и документооборота.

В целом рассмотренные «Требования к информационным системам электронного документооборота федеральных органов исполнительной власти, учитывающие в том числе необходимость обработки посредством данных систем служебной информации ограниченного распространения» можно и нужно использовать не только на стадии выбора, внедрения и первичной настройки СЭД, но и для анализа уже функционирующих СЭД для определения соответствия СЭД, используемой в конкретной организации, современным требованиям.

Обзор аналогов

Существуют различные виды классификации систем электронного документооборота, однако самая показательная классификация представляет собой разделение СЭДов по титульному функционалу. Любая СЭД позиционируется своим разработчиком как предназначенная преимущественно для выполнения определенного типа задач. Это не значит, что в ней не предусмотрены какие-либо дополнительные технологии,

присущие большинству «собратьев». Просто набор инструментов каждой системы электронного документооборота имеет свои сильные и слабые стороны. В классификации по титульному функционалу как раз и учитываются сильные стороны СЭДов.

Итак, выглядит она следующим образом:

- СЭДы, предназначенные для создания и работы с электронной документацией, а также цифровыми аналогами бумажных документов.
- Системы для учета, автоматизирующие регистрацию событий и документов на протяжении всего их жизненного цикла (электронные картотеки).
- СЭДы, основной задачей которых является автоматизация работы с большими хранилищами корпоративной информации.
- Системы, управляющие электронными архивами с документацией.
- СЭДы, функционал которых специализируется на извлечении нужной информации из архивов и других электронных источников.
- Системы, управляющие корпоративными процессами, обработкой документов и деятельностью сотрудников организации, которые привлекаются к работе с деловой документацией.
- Информационные СЭДы, управляющие устройствами для хранения данных.

Существует также краткая классификация СЭДов по стране происхождения, используемая только в России. Согласно ей, системы электронного документооборота в нашей стране подразделяются на отечественные, импортные и российские, разработанные на зарубежной платформе Lotus/Domino. В нашем обзоре фигурируют примеры только двух зарубежных СЭД – EMC Documentum и Lotus Domino.Doc. Все остальные СЭД имеют «гражданство» России.

Оценку систем электронного документооборота мы проводили, ориентируясь на сравнение пяти параметров, которые являются

определяющими при выборе СЭДа для автоматизации документооборота на предприятии. Шкала – десятибалльная.

Таблица 1.2

Лучшие системы электронного документооборота

Место	Программа/сервис	Цена	Простота освоения	Функциональность	Учет российского законодательства	Техническая поддержка	Общая оценка
1	Дело	8	8	10	10	10	9,2
2-4	1С:Архив	9	7	10	10	9	9
2-4	CompanyMedia	9	9	9	10	8	9
4-6	EMC Documentum	–	9	10	8	8	8,8
4-6	Логика	9	10	9	10	6	8,8
4-6	ЕВФРАТ	9	10	9	10	6	8,8
7	DIRECTUM	10	8	8	10	7	8,6
8	Lotus Domino.Doc	–	8	8	9	9	8,5
9	OPTIMA-WorkFlow	10	8	8	9	7	8.4
10	LanDocs	8	7	7	10	8	8

Примечание: итоговая оценка системы в таблице не претендует на стопроцентную объективность и отражает мнение автора, основанное на глубоком анализе данной темы. Этот обзор систем электронного документооборота сделан только среди качественных программных продуктов, которые смело можно выбрать для использования в различных видах бизнеса.

Дело. Эта система электронного документооборота - является признанным лидером в своем сегменте на территории всего постсоветского пространства. В ней успешно осуществляют документооборот и делопроизводство как крупнейшие холдинги и корпорации, так и предприятия малого бизнеса. По отношению к этой системе уместно применить тавтологию: «ДЕЛО» знает свое дело. Действительно, данный

софт идеально подходит для глубокой автоматизации делопроизводства и документооборота.

Функции, организации, выбрать, российские, возможности, россии, системы

Плюсы. Возможность отслеживания всех этапов перемещения любого электронного документа.

Простота и удобство в создании проектов документации.

Общая отлаженность и функциональность системы.

Минусы. К «натянутым» минусам можно отнести несколько архаичный интерфейс и определенную сложность в освоении.

Цена лицензии на использование системы «ДЕЛО» в рамках одного рабочего места (СУБД – Oracle) зависит от планируемого общего количества этих мест и варьируется в пределах от 11 000 рублей (201-500 р/м) до 13 400 рублей (1-5 р/м). Если в организации применяется СУБД Microsoft SQL Server, то лицензия за одно рабочее место обойдется от 7200 до 9500 рублей соответственно.

Логика. Программа «Логика СЭД» до 2012 года называлась «Босс-Референт» и была одной из популярнейших российских систем электронного документооборота, серьезным конкурентом СЭД «Дело». Смена названия никак не отразилась на качестве этого программного продукта, и он по-прежнему является надежной и функциональной системой для управления делопроизводством на предприятиях любого типа и размера, функции, организации, выбрать, российские, возможности, россии, системы

Плюсы. Возможность построения сложных многоуровневых маршрутов согласования документации.Высокая степень защиты данных пользователя.Относительная легкость освоения основных компонентов системы.Гибко настраиваемые и модифицируемые процессы обработки документации.

Минусы. Неудобное разграничение прав доступа при серьезной текучке кадров.Не самая добросовестная работа технической поддержки.Архаичный интерфейс.

Стоимость одной лицензии на использование данной программы зависит от количества сотрудников, подключенных к ней. Если их число не превышает 49 человек, то цена будет 5 900 за каждое рабочее место; от 50 до 199 подключенных работников – 5 200 рублей; а если рабочих мест более 200, то цена одной лицензии будет минимальной – 4900 рублей.

ЕВФРАТ. Эта система электронного документооборота разработана в полном соответствии с требованиями стандарта качества ISO 9000 и российских ГОСТов в области делопроизводства. От своих «коллег по цеху» «ЕВФРАТ» отличается наличием множества собственных уникальных программных разработок, которые нельзя встретить в конкурирующих СЭДах. Чисто технологически данная система является одной из самых «продвинутых» на современном российском рынке СЭДов, функции, организации, выбрать, российские, возможности, россии, системы

Плюсы. В комплект поставки этой системы входит встроенная СУБД «Ника», что автоматически освобождает организацию-пользователя от приобретения дополнительного программного обеспечения.Дружелюбный интерфейс, обладающий приятным запоминающимся дизайном.Встроенный механизм ролей для управления правами доступа.

Минусы. Довольно медленная скорость работы, особенно на слабых компьютерах.Периодические сбои в работе и нерасторопная техподдержка.

Стандартная лицензия с установкой системы на собственный сервер организации стоит от 5200 до 7300 рублей на одно рабочее место, больше пользователей – ниже цена. Однако имеется также вариант размещения серверной компоненты на оборудовании компании-производителя. В этом случае применяется система абонентской платы – четыре тарифа, стоимостью от 10 000 до 95 000 рублей в месяц.

1С:Архив. Это одна из лучших и уж точно самая универсальная программа для управления документооборотом предприятия. «1С:Архив» обеспечивает надежное и, что самое главное, централизованное хранение деловой документации различного формата, с обеспечением доступа к ней уполномоченного персонала, который может производить редактирование файлов.

Функции, организации, выбрать, российские, возможности, россии, системы

Плюсы. Отлаженный алгоритм быстрого поиска необходимых данных.

Возможность хранения документов любых типов – от текстовых и графических, до аудио- и видеофайлов.

Широчайшие возможности масштабирования, позволяющие успешно применять данный софт и на крупных, и на маленьких предприятиях.

Главный плюс «1С:Архив», выделяющий эту программу на фоне конкурентов, – это оптимальное сочетание цены продукта и возможностей его функционала.

Поддержка интеграции с внешними приложениями.

Минусы. Потребление большого количества системных ресурсов.

Повышенная сложность освоения даже для опытных пользователей.

Цена «сборки» этой программы варьируется от 12 000 до 57 000 рублей, причем первую сумму придется отдать исключительно за апгрейд предыдущей версии «1С:Архив».

DIRECTUM. Простая и функциональная СЭД DIRECTUM станет отличным решением для предприятий малого и среднего бизнеса. Благодаря данной системе электронный документооборот можно успешно совмещать с традиционным бумажным, чтобы впоследствии «безболезненно» полностью перевести организацию на работу в DIRECTUM. Продвинутая технология Workflow обеспечивает эффективную автоматизацию процессов делопроизводства, функции, организации, выбрать, российские, возможности, россии, системы

Плюсы. Наличие специализированных программных инструментов, максимально упрощающих поиск и идентификацию документации.

Широкие возможности для самостоятельной модификации системы под конкретные задачи.

Расширенные возможности интеграции с другими программами.

Минусы. Ориентированность системы на руководящий состав предприятия – простым делопроизводителям работать в ней труднее.

Несколько запутанная и непрозрачная ценовая политика.

Лицензии на использование данной СЭД приобретаются как по отдельности, так и в рамках пакетных предложений. Самая дешевая базовая клиентская лицензия стоит 7 800 рублей. Стоимость же пакетов лицензий начинается от 148 200 рублей (базовый на 20 сотрудников) и доходит до 2 010 000 рублей (на 200 работников).

OPTIMA-WorkFlow. Эта система электронного документооборота является крепким середняком в своем сегменте. Она находится в постоянном развитии и пока что не может на равных конкурировать с признанными мастодонтами рынка. Однако, OPTIMA-WorkFlow имеет ряд «козырей в рукаве» – уникальных технологий, которые выделяют ее на общем фон, функции, организации, выбрать, российские, возможности, россии, системы

Плюсы. Реализация функции серийного ввода одинаковых видов документов и их регистрационной информации.

Сканирование программой-антивирусом.

Индексирование обычных и зашифрованных документов по выбору.

Мультиязычный интерфейс пользователя.

Минусы. Недостатки типовых функциональных возможностей, которые перманентно устраняются выходящими обновлениями.

Типовые решения на базе данной платформы стоят от 55 000 до 75 000 рублей.

EMC Documentum. Данная платформа для автоматизации процессов документооборота разработана мировым лидером IT-индустрии, компанией

EMC. Мощный функционал вкупе с гибкой настройкой отдельных инструментов делают EMC Documentum лучшей иностранной СЭД из доступных на российском рынке, функции, организации, выбрать, российские, возможности, россии, системы

Плюсы. Высокая степень удобства управления проектно-конструкторскими документами.

Внедрение механизма check-in/check-out, позволяющего управлять распределением прав доступа.

Наличие функции, позволяющей реализовывать процедуру согласования на нескольких уровнях.

Поддержка устройств сканирования и распознавания документов.

Минусы. Работа только с браузером Internet Explorer.

Отсутствие в открытом доступе технической и практической документации по системе.

Частые «тормоза» и лаги при высокой загруженности системы.

Фиксированных цен на использование данной платформы нет. Стоимость ее внедрения обговаривается индивидуально с каждым заказчиком.

LanDocs. Разработанная в 1997 году отечественной компанией «ЛАНИТ» эта платформа для автоматизации ведения документооборота по сей день остается востребованной многими предприятиями и учреждениями. LanDocs позволяет выстроить комфортную среду делопроизводства и документооборота, предоставляя всем категориям пользователей необходимый набор инструментов для управления ею, функции, организации, выбрать, российские, возможности, россии, системы

Плюсы. Наличие интегрированных инструментов криптографической защиты.

Возможность включения в процесс документооборота сотрудников из удаленных филиалов.

Наличие функции пакетного сканирования бумажной документации.

Минусы. Довольно часто возникающие проблемы с производительностью.

Слабые возможности по расширению функционала.

Сложность в освоении системы с нуля.

Цена серверной лицензии варьируется от 30 000 до 216 000 рублей. Пользовательские же лицензии можно приобрести по цене от 5 600 до 8 400 рублей.

CompanyMedia – это целый набор программных решений, предназначенных для автоматизации бизнес-процессов, документооборота и делопроизводства. От конкурентов данная СЭД отличается непревзойденной гибкостью настроек и наличием независимых модулей, которые можно устанавливать по отдельности.

Плюсы. Возможность успешной работы системы на предприятиях со сложной корпоративной структурой и территориальным устройством.

Беспрецедентная надежность системы, которая позволяет ей активно функционировать круглые сутки, 365 дней в году.

Автоматизированная поддержка работы с несколькими типами рабочих мест.

Разграничение прав доступа, согласно иерархической структуре организации.

Минусы. В веб-интерфейсе ограничена поддержка некоторых браузеров, в частности FireFox.

Ориентированность системы преимущественно на средний и крупный бизнес.

Конечная цена данного софта складывается из множества составляющих, включая обучение персонала, установку дополнительных модулей и передачу прав пользования системой. Верхний предел итоговой суммы – 99 000 рублей. Самый дешевый тематический модуль обойдется в 4000 рублей.

Lotus Domino.Doc. Данная СЭД является приложением к известной платформе Notes/Domino, обладающей высокоуровневой системой обеспечения безопасности данных. Также Lotus Domino.Doc имеет продвинутый электронный архив, который позволяет реализовать объемное хранилище корпоративной документации, функции, организации, выбрать, российские, возможности, россии, системы

Плюсы. Наличие уникальных систем репликации, предназначенных для комплексного решения задач по управлению потоками документов в распределенной среде.

Возможность поиска документов по отдельным частям хранилища информации.

Каталоги LDAP масштабируются вплоть до уровня корпораций.

Минусы.Недостатки данной системы в основном вытекают из ее западного происхождения – имеются определенные вопросы к интерфейсу и реализации некоторых функций.

Стоимость пользовательских и серверных лицензий рассчитывается индивидуально на сайте платформы.

Подводя некую черту под обзором, хотелось бы отметить важный нюанс: внедрение системы электронного документооборота в ваш бизнес вовсе не гарантирует мгновенное повышение его доходности, а, в отдельных случаях, может и навредить. Если ваша организация успешно функционирует и приносит прибыль, работая без СЭДов, то тысячу раз подумайте, прежде чем решиться на их внедрение. Не нужно гнаться за «прогрессивной модой» и устанавливать данные системы без серьезной потребности в них.

1.2 Обоснование проектных решений по программному обеспечению

Приложение «1С: Предприятие» предназначено для решения обширного перечня задач управления и автоматизации учета, которые стоят перед современными стремительно развивающимися администрациими.

«1С: Предприятие» - совокупность прикладных решений, построенных на целостной технологической платформе и по нераздельным принципам. Задачи управления и учета могут основательно отличаться, ведь они формируются на основе рода деятельности администрации, специфики оказываемых услуг или продукции, отрасли, структуры и размера компании, необходимого уровня автоматизации.

Довольно трудно представить себе одну утилиту массового использования, которая бы удовлетворяла потребности большинства. Каждому руководителю необходимо решение, совпадающее со спецификой его детища и, вместе с этим, он понимает, что существуют преимущества в использовании проверенного массового продукта. Сочетание данных потребностей обеспечивает «1С: Предприятие».

Настоящий продукт поставляется с шаблонными конфигурациями, которые реализуют общие схемы учета и используются в большинстве организаций. Утилита может быть адаптирована к различным особенностям учета на определенном предприятии.

Система в своем составе содержит конфигуратор, который позволяет:

- настроить продукт на разные виды учета;
- организовать произвольно структурированные документы, всевозможные справочники;
- реализовать любые методологии учета;
- настроить внешний вид форм ввода информации;
- настроить поведение и алгоритмы работы утилиты, в разных ситуациях при помощи объектно-ориентированного языка;

- представлять наглядно информацию в виде диаграмм;
- быстро изменять конфигурацию путем применения «конструкторов»;
- различным образом оформлять документы и отчеты с использованием разнообразных рамок, шрифтов, рисунков, цветов.

Очень важное преимущество «1С: Предприятие» - открытость системы. Руководитель администрации может быть полностью уверен, что его сотрудники с легкостью поймут принцип работы программного обеспечения.

В комплекте системы поставляются средства, нужные для доработки прикладных решений и внесения изменений в них произвольной сложности. «Работа с файлами» - один из таких инструментов, он предназначен для редактирования и просмотра файлов следующих форматов:

Программные продукты системы «1С: Предприятие» поставляются с типовыми конфигурациями. Типовые конфигурации реализуют наиболее общие схемы учета и могут использоваться в большинстве организаций.

Для отражения специфики конкретного учреждения типовую конфигурацию можно изменить.

«1С:Предприятие» имеет режим запуска «Конфигуратор», который обеспечивает:

- настройку системы на различные виды учета,
- реализацию любой методологии учета,
- организацию любых справочников и документов произвольной структуры,
- настройку внешнего вида форм ввода информации,
- настройку поведения и алгоритмов работы системы в различных ситуациях с помощью встроенного языка,
- широкие оформительские возможности создания печатных форм документов и отчетов с использованием различных шрифтов, рамок, цветов, рисунков,

- возможность наглядного представления информации в виде диаграмм,
- быстрое изменение конфигурации с помощью визуальных средств разработки.

В случаях когда система учета, имеет уникальный характер типовая конфигурация, входящая в состав программных продуктов системы «1С:Предприятие», может быть взята как образец для создания уникальной конфигурации, полностью ориентированной на особенности вашей организации.

Конфигуратор, входящий в состав программных продуктов системы «1С:Предприятие», позволяет не только изменять элементы типовой конфигурации, но и создать собственную конфигурацию «с нуля». Такая разработка может быть выполнена силами сотрудников организации, в которой установлена система или специалистами администрации франчайзинговой сети фирмы «1С»,

Создание оригинальных конфигураций позволяет решать с помощью «1С:Предприятия» самые разнообразные задачи по автоматизации экономической деятельности.

«1С:Предприятие» как предметно-ориентированная среда разработки имеет определенные преимущества. Поскольку круг задач более точно очерчен, то и набор средств и технологий можно подобрать с большей определенностью. В задачу платформы входит предоставление разработчику интегрированного набора инструментов, необходимых для быстрой разработки, распространения и поддержки прикладного решения для автоматизации бизнеса. При этом отдельные «детали» могут уступать по функциональности универсальным средствам разработки и специализированным средствам управления жизненным циклом, используемым разработчиками. Эффект достигается благодаря общему набору средств и их тесной интеграции.

«1С: Предприятие» является системой программ для автоматизации различных областей экономической деятельности. В конкретный программный продукт, входящий в систему программ 1С: Предприятие, включаются те функции и возможности, которые отвечают назначению этого продукта.

«1С:Предприятие» — прикладная программа, служащая для комплексной автоматизации всевозможных видов учета, финансово-экономического анализа на предприятии любой сферы деятельности и любой организационной структуры. Программа обладает способностью адаптирования к особенностям некоторой конкретной области деятельности путём задействования той или иной конфигурации. Она может работать в двух режимах — пользовательском и конфигурирования. В режиме конфигурирования она обладает широким спектром механизмов для разработки необходимой конфигурации с нуля, либо подстройки некой стандартной конфигурации, под нужды конкретного администрации.

«1С:Предприятие» включает технологическую платформу, и пользовательский режим работы на ней. Технологическая платформа предоставляет объекты, данных и метаданных, и механизмы управления объектами. Совокупность объектов, данных и метаданных, также связей между ними, задаваемых программистом, представляет собой конфигурацию. При автоматизации какой-либо деятельности разрабатывается своя конфигурация объектов и связей между ними задаваемых программно, которая и представляет собой законченное прикладное решение. Конфигурация создаётся в специальном режиме работы программного продукта под названием «Конфигуратор», который позволяет при разработке незамедлительно проверять её работоспособность в режиме «1С:Предприятие», осуществляя отладку. Пользователи работают исключительно в режиме «1С:Предприятие», в котором получают доступ ко всем функциям, сообразно правам каждого конкретного пользователя, реализованным в данном прикладном решении (конфигурации). Платформа

позволяет создавать свою собственную конфигурацию "с нуля", если по каким-либо причинам использование типовой конфигурации представляется нецелесообразным.

Главной особенностью системы «1С: Предприятие» является возможность построения отчетов, с помощью системы компоновки данных. Она предназначена для создания отчетов на основе их декларативного описания. Данная система позволяет реализовать следующие возможности:

- Возможность создания различных вариантов отчета;
- Возможность задания различных вариантов пользовательских настроек;
- Использование автоматически генерируемых форм просмотра и настройки отчета;
- Разбиение исполнения отчета на этапы;
- Программное влияние на процесс выполнения отчета;
- Настройка структуры отчета;
- Совмещение в отчете нескольких таблиц.

Таким образом, будет создано новое программное средство управления предприятием, которое будет отличаться от всех существующих версий следующим:

- Удобный и понятный интерфейс;
- Простота освоения и использования;
- Соответствие всем правилам оформления форм и отчетов;
- Отсутствие дополнительных затрат на приобретение и освоение типовых конфигураций;
- Постоянное обновление;
- Необходимое своевременное редактирование (создание, изменение и удаление).

Основной причиной выбора платформы «1С: Предприятие 8» в качестве среды разработки было то, что у 80% бюджетных организаций преимущественно установлена платформа 1С,а высокая простота и скорость

разработки прикладных решений в данной платформе послужит большим плюсом. Также очень существенно то, что выбранная платформа позволит разработчикам, сопровождающим программистам и пользователям легко изменять, дорабатывать, обновлять программу. Существуют 2 роли при работе с программой с 1С – это программист и пользователь, соответственно функционирование системы «1С: Предприятие» будет делиться на два процесса:

- конфигурирование - описание модели предметной области средствами системы для программиста;
- исполнение - обработку данных предметной области для пользователя.

Результатом конфигурирования будет являться конфигурация, которая представляет собой модель предметной области.

На этапе конфигурирования система оперирует такими понятиями как «Документы», «Отчеты», «Справочники», «Реквизиты» и другие. Совокупность этих понятий и определяет концепцию системы. На уровне системы определены сами понятия и стандартные операции по их обработке. Средства конфигурирования позволяют описать структуры информации, входящей в эти объекты и алгоритмы, описывающие специфику их обработки, для отражения различных особенностей учета.

В процессе конфигурирования формируется структура информационной базы, алгоритмы обработки, формы диалогов и выходных документов. Информационная структура проектируется на уровне предусмотренных в системе типов, обрабатываемых объектов предметной области - константы, справочники, документы, регистры, перечисления, журналы расчетов, бухгалтерские счета, операции, проводки и другие.

При работе пользователя в режиме исполнения конфигурации? обработка информации выполняется как штатными средствами системы, и с использованием алгоритмов, созданных на этапе конфигурирования.

Для реализации новой конфигурации были использованы следующие базовые объекты метаданных:

- Константы;
- Справочники;
- Документы;
- Регистры накопления;
- Отчеты.

ГЛАВА 2.ПРОЕКТНАЯ ЧАСТЬ

2.1 Информационное обеспечение задачи

2.1.1 Информационная модель и её описание

Информационная модель представляет собой схему движения входных, промежуточных и результативных потоков и функций предметной области. Кроме того, она объясняет, на основе каких входных документов и какой нормативно-справочной информации происходит выполнение функций по обработке данных и формирование конкретных выходных документов.

В качестве информационной модели будем использовать схему данных (ГОСТ 19.701-90). Схемы данных отображают путь данных при решении задач и определяют этапы обработки, а также различные применяемые носители данных. Схема данных состоит из следующих элементов:

- символов данных (символы данных могут также указывать вид носителя данных);
- символов линий, указывающих потоки данных между процессами и носителями данных;
- символов процесса, который следует выполнить над данными (символы процесса могут также указывать функции, выполняемые вычислительной машиной);
- специальных символов, используемых для облегчения написания и чтения схемы[1].

Весь цикл обработки информации можно разбить на два этапа:

1. Прием, обработка и ввод первичной входящей информации (паспортные данные, реквизиты организаций).
2. Формирование документов.

Графическое представление информационной модели на рис. 2.2.

Рис. 2.2. Схема потоков данных после внедрения системы электронного документооборота

2.1.2Характеристика нормативно-справочной, входной и оперативной информации

В качестве оперативной информации используется информация:

- Корресподенты (Справочник корреспондентов);
- Контактные лица(Сотрудники и справочник должностей);
- Реквизиты(Справочник видов документов);
- Входящие документы;
- Исходящие документы;
- Внутренние документы;
- Процессы;
- Задачи.

2.1.3 Характеристика результатной информации

В ходе деятельности разработанной системы документооборота формируются следующие выходные отчеты по Документам:

- Внутренние документы;
- Входящие документы;
- Договоры;
- Документооборот по организациям;
- Журнал передачи;
- Корреспонденты;
- Сводка по видам документов;
- Справка об объеме документооборота;
- Файлы.

По Задачам и процессам:

- Процессы;
- Задачи;
- Поручения;

- Согласования;
- Утверждения;
- Справка об исполнительской дисциплине;
- Шаблоны процессов.

2.2 Программное обеспечение задачи

Сценарии диалога, формирующийся на основе дерева функций, приведен на рис. 2.5.

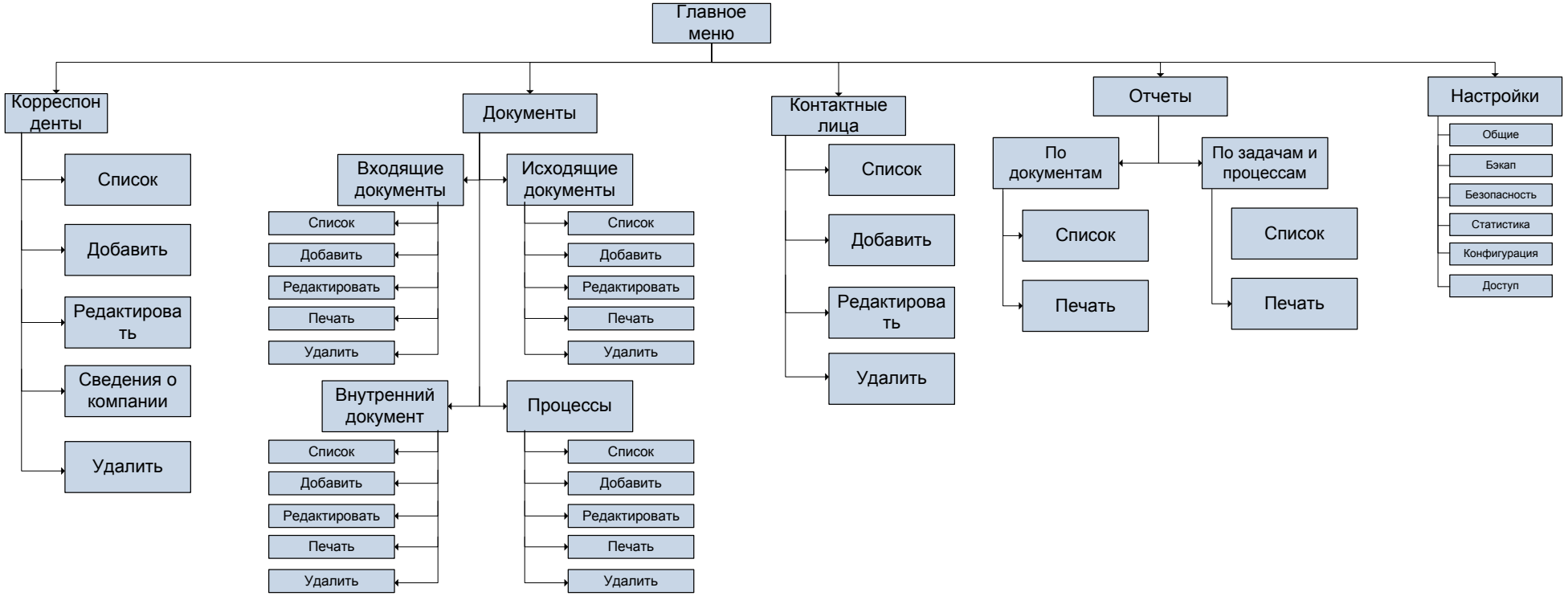


Рис. 2.5. Сценарий диалога для пользователя

2.2.2 Характеристика базы данных

Модель базы данных показана на рис. 2.6.

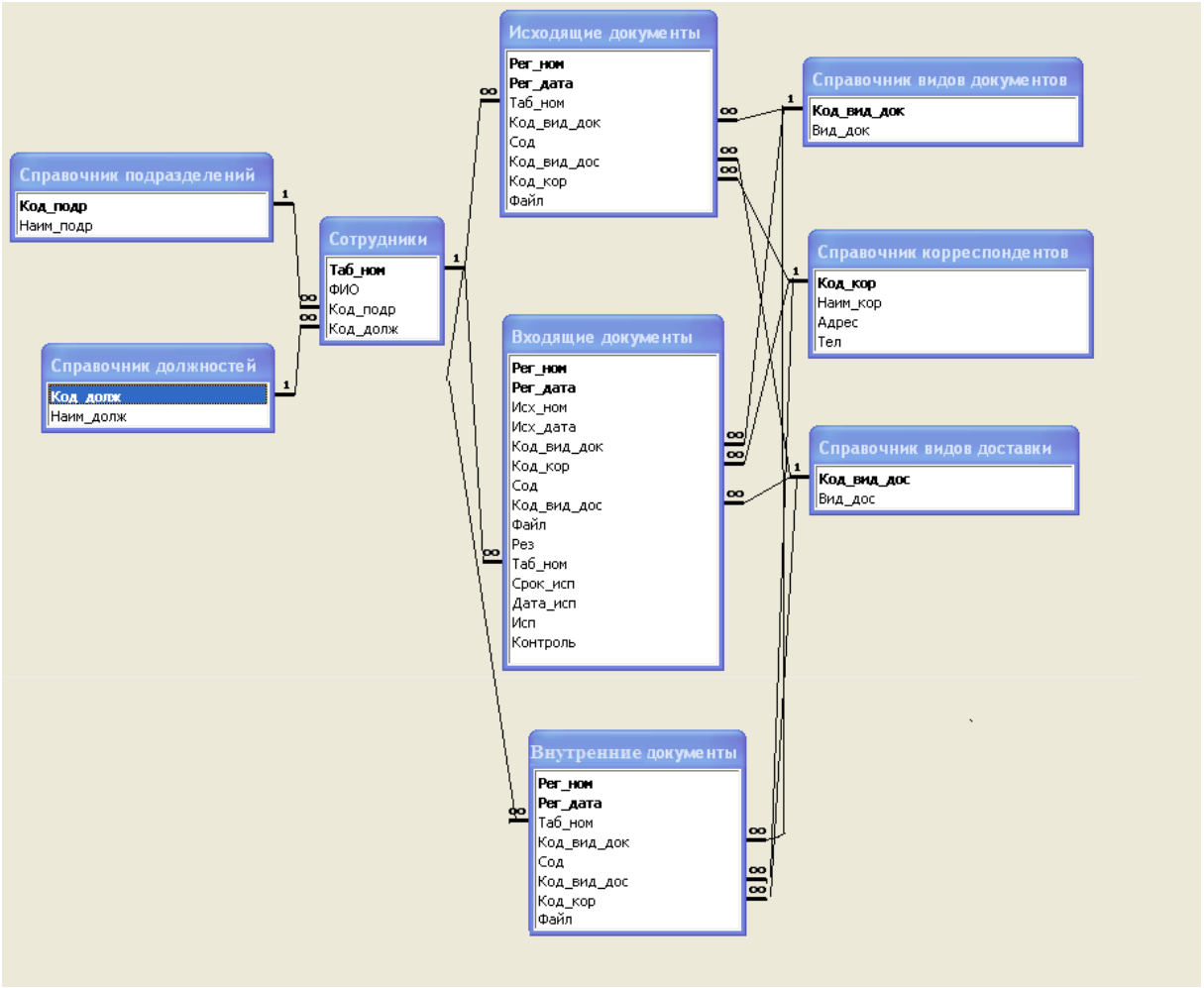


Рис. 2.6.ER-модель разрабатываемой базы данных

2.2.3 Структурная схема пакета (дерево вызова программных модулей)

Для разработки системы автоматизации администрации используется система приложение. Эта система имеет модульную форму организации конфигурации. В глобальных модулях хранятся переменные, процедуры и функции доступные из любых других модулей. Можно также выделить другие группы модулей:

- справочники
- документы

- отчеты
- обработки
- регистры

Следовательно, структуру программы можно описать следующими основными блоками представленными на рис. 2.7.

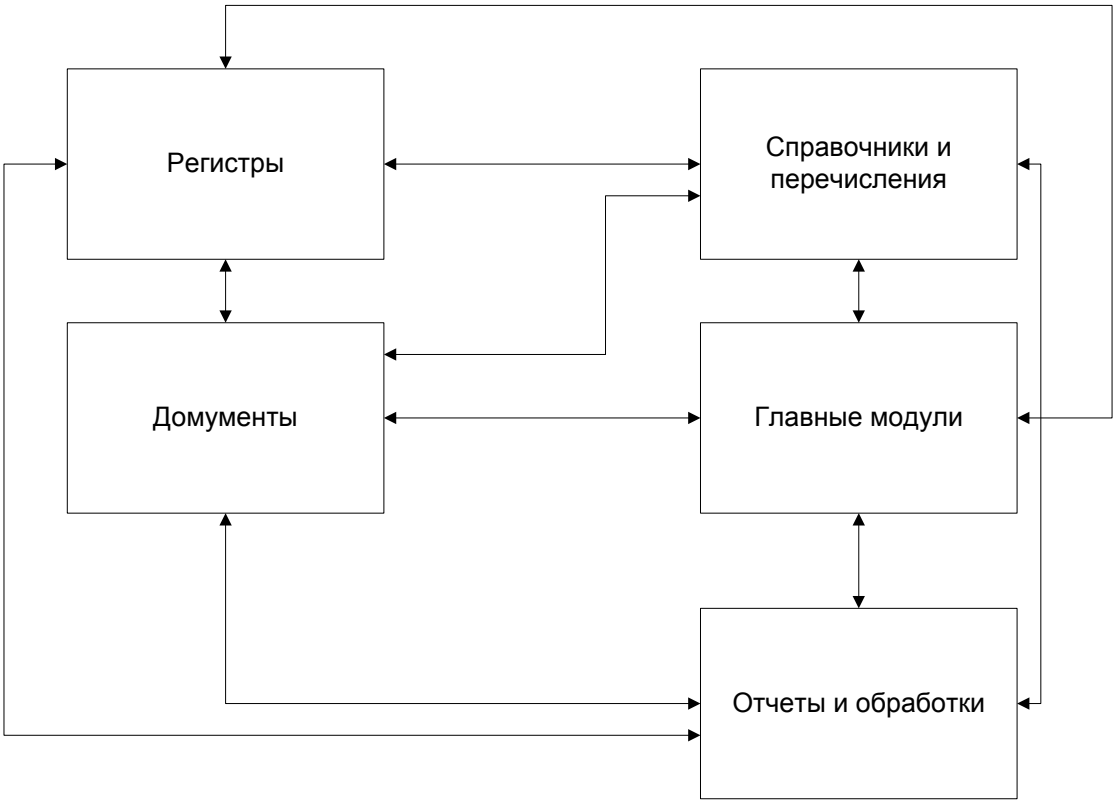


Рис. 2.7.Блок-схема основных модулей программы

Работа с программой начинается с вывода главного окна программы и активизации системы меню. Работа программы осуществляется по диалоговому и событийному режиму, при этом под диалогом понимается предоставление пользователю нескольких альтернатив и обработка его выбора. В диалоговую систему входят главное меню с соответствующими всплывающими подменю, также диалоговые окна. Под событиями понимаются процессы, активизируемые пользователем, а также программные события – получение определенным полем фокуса редактирование или потеря фокуса ввода. На основании данных событий активизируются процедуры контроля допустимости данных.

Описание программных модулей представлено в таблице 2.3.

Таблица 2.3

Описание программных модулей

№ п/п	Наименование модуля	Функции модуля
1)	Документы	Совокупность алгоритмов, выполняющих электронный документооборот– основную функцию системы
2)	Регистры	Совокупность алгоритмов, выполняющих функции заполнения данными
3)	Отчеты и обработки	Позволяет получить отчеты
4)	Справочники и перечисления	Данный модуль позволят осуществлять управление работой системы через управление содержимым справочников
5)	Главные модули	Обеспечивает работу системы

2.2.4 Описание программных модулей

Работа с программой начинается с выбора пользователя и вывода информационного окна и активизации системы меню.

Работа программы осуществляется по диалоговому и событийному режиму, при этом по диалогом понимается предоставление пользователю нескольких альтернатив и обработка его выбора. В диалоговую систему входят главное меню, с соответствующими всплывающими подменю, а также диалоговые окна. Под событиями понимаются процессы активизируемые пользователем, а также программные события – получение определенным полем фокуса редактирование или потеря фокуса ввода. На основании данных событий активизируются процедуры контроля допустимости данных.

Программа состоит из следующих основных модулей.

Глобальный модуль - конфигурация среды окружения, формирование основного экрана программы, создание системы главного меню и соответствующих подменю, активизация меню.

Процедуры формирования отчетов – обеспечение выдачи установленных форм документов на основании критериев, определяемых пользователем и информационной базы.

Модуль справочников и модуль документов - обеспечение ввода информации с первичных документов в базы данных, контроль за допустимостью значений и обеспечение ввода данных путем выбора из списка.

Все модули в программе связаны между собой по данным, которые анализируются на входе и вырабатываются на выходе. Данные в модули поступают через диалог с пользователем, параметры и документы информационной базы.

Для ведения информационной базы могут быть выполнены операции просмотра и печати документов, их редактирование, ведение нормативно-справочных документов, а также создание архивов и восстановление документов базы данных. Операции осуществляются путем выбора соответствующих пунктов в главном и подчиненных меню.

Данные через диалог могут быть получены прямым и косвенным способом. Прямой способ реализуется путем их ввода по шаблону или по запросу конкретных значений. Косвенный способ – путем меню.

Параметры/входные документы – входные данные, полученные в виде конкретных значений, переданных в оперативной памяти смежным модулям/функциям.

Контрольный пример реализации проекта и его описание:

1. Основные разделы панели навигации.

1.1. Рабочий стол

На рабочем столе располагается ярлык желтого цвета 1С Предприятие. При двойном щелчке мыши по нему появляется окно идентификации пользователя, где в поле «Пользователь» необходимо внести свою Фамилию и

инициалы (например, как показано на рис. 2.8 Елистратова Л.Е.) и ввести персональный пароль. Далее нажать кнопку «Ок».

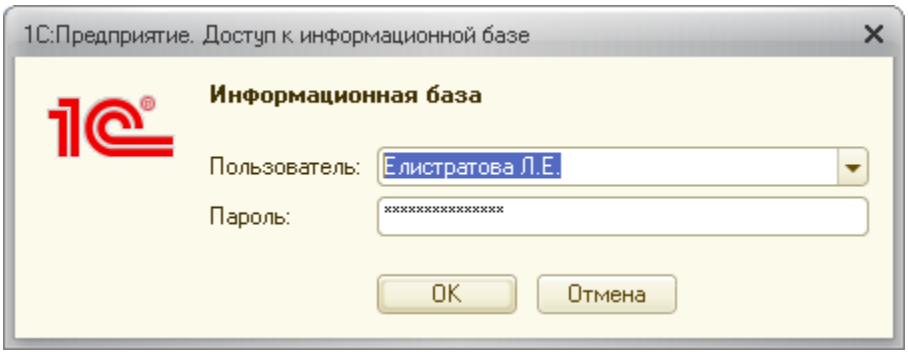


Рис. 2.8. Вход в программу

Вы вошли в программу. Первое, что видит каждый пользователь, осуществивший вход в программу - это раздел меню Рабочий стол (см. рис.2.9).

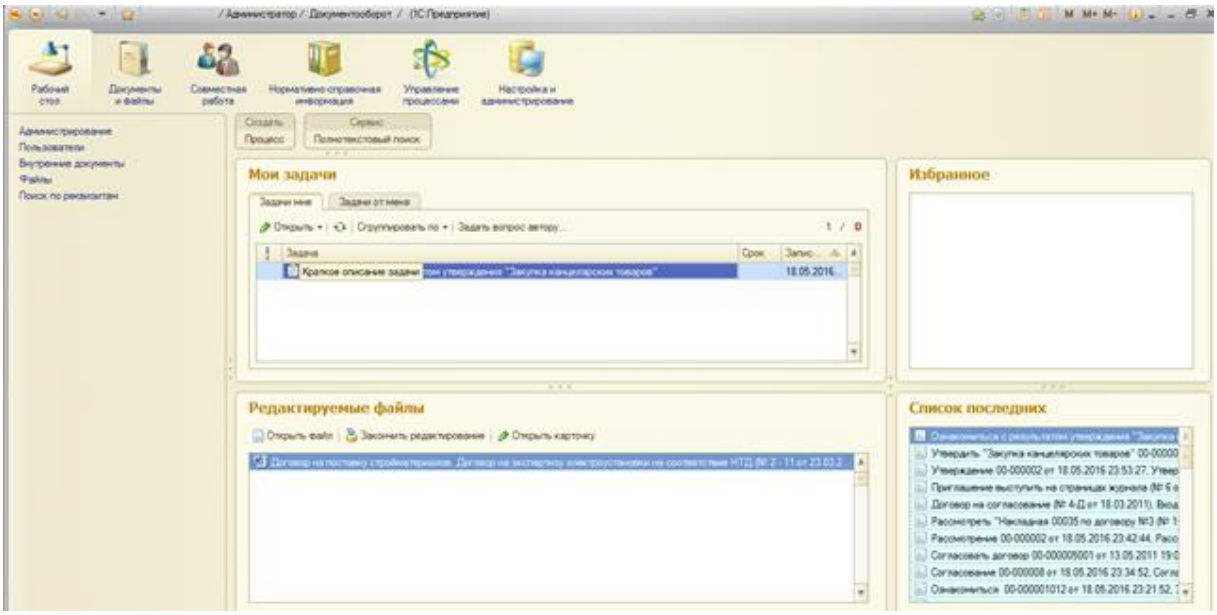


Рис. 2.9. Меню Рабочий стол

Рабочий стол является основным рабочим экраном сотрудников. Он состоит из трех блоков:

- «Мои задачи»;
- «Редактируемые файлы» часть экрана, где отображаются файлы, находящиеся на редактировании данного пользователя;
- «Избранное», куда можно самостоятельно поместить ссылки на любой раздел или любую информацию в системе.

Ссылки создаются путем одновременного нажатия сочетания клавиш «Ctrl+D» непосредственно из формы данных (формы документа или элемента справочника).

Основным рабочим блоком каждого сотрудника является блок «Мои задачи». Рассмотрим его подробнее.

Мои задачи.Часть экрана, на которой отображаются все задачи данного пользователя, сформированные другими пользователями системы с какой-либо целью. Цели формирования задач другими пользователями могут быть самые различные: согласовать документ, ознакомиться с документом, рассмотреть документ и так далее. Все поступившие задачи располагаются в хронологическом порядке сверху вниз.

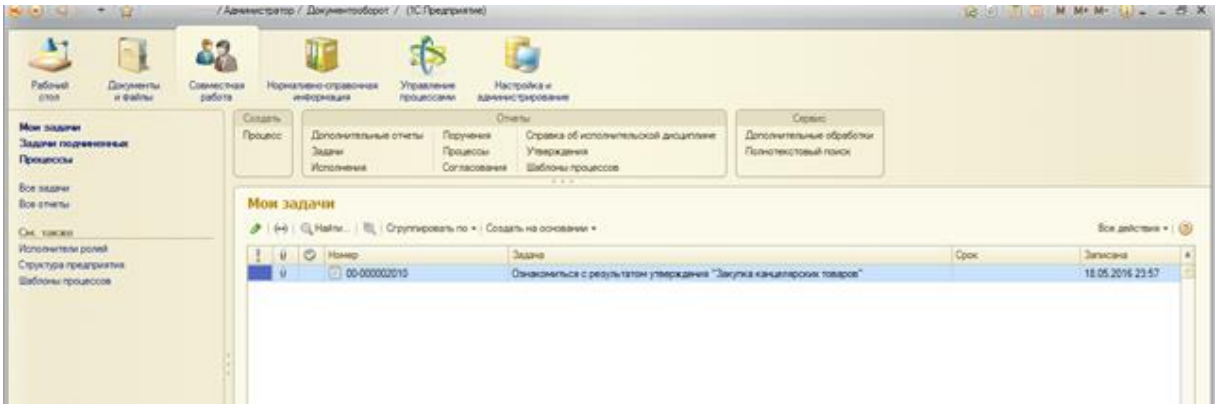


Рис. 2.10. Мои задачи

Для удобства просмотра задач существует возможность группировки по важности, точкам маршрута, автору, предмету и бизнес-процессу. Для группировки необходимо выполнить команду «Сгруппировать по...» и указать вид группировки. Наиболее рекомендуемый и удобный вид группировки - точка маршрута. При таком виде группировке поступившие задачи располагаются структурировано в виде групп. Для снятия группировки следует выбрать «Без группировки».

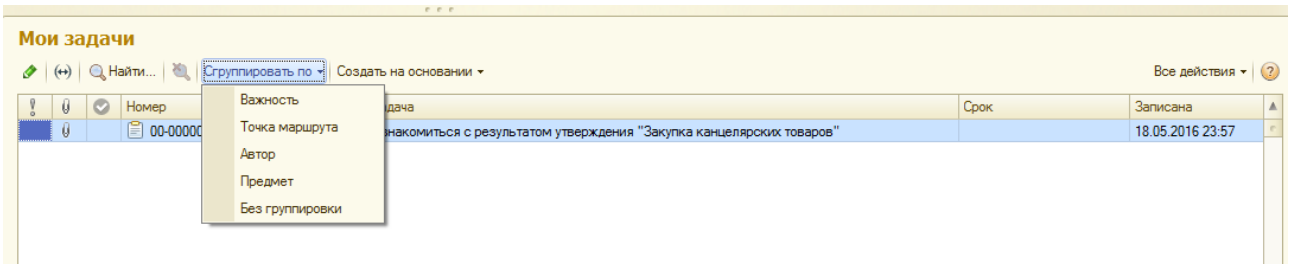


Рис. 2.11. Мои задачи

Для просмотра поступившей задачи открываем ее двойным щелчком левой клавиши мыши. Появляется карточка задачи, где можно просмотреть все уточняющие данные по поступившей задаче (автор, исполнитель, дата формирования задачи, срок исполнения задачи, важность).

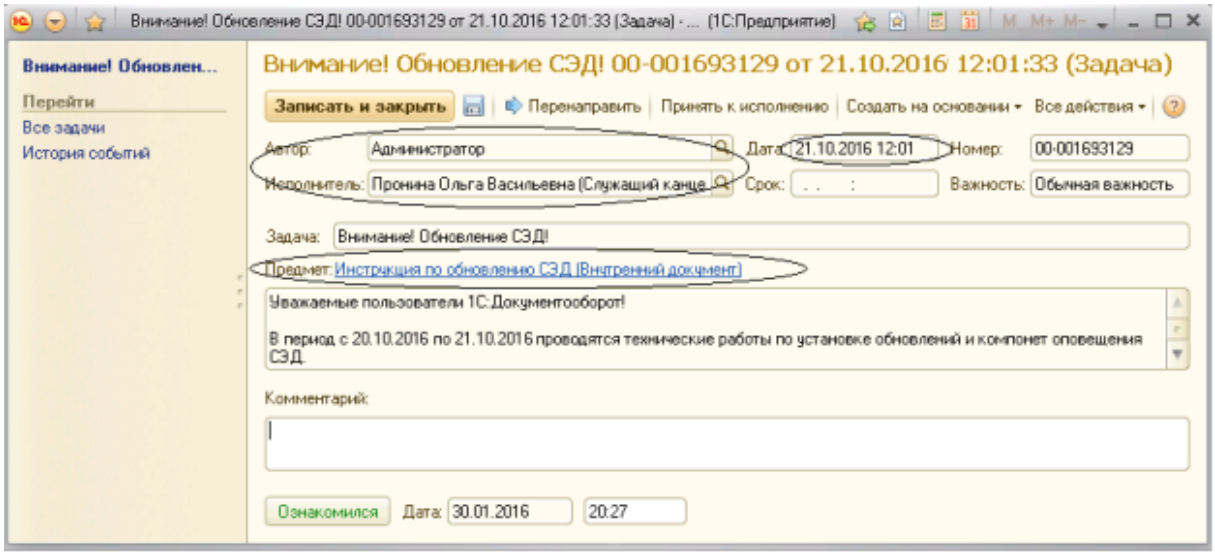


Рис. 2.12. Поступившая задача

Просмотр прикрепленного документа осуществляется нажатием на синюю ссылку с названием документа (см. рис. 2.13).

Рис. 2.13. Просмотр прикрепленного документа

С помощью команды Принять к исполнению можно отметить задачу как принятую к исполнению.

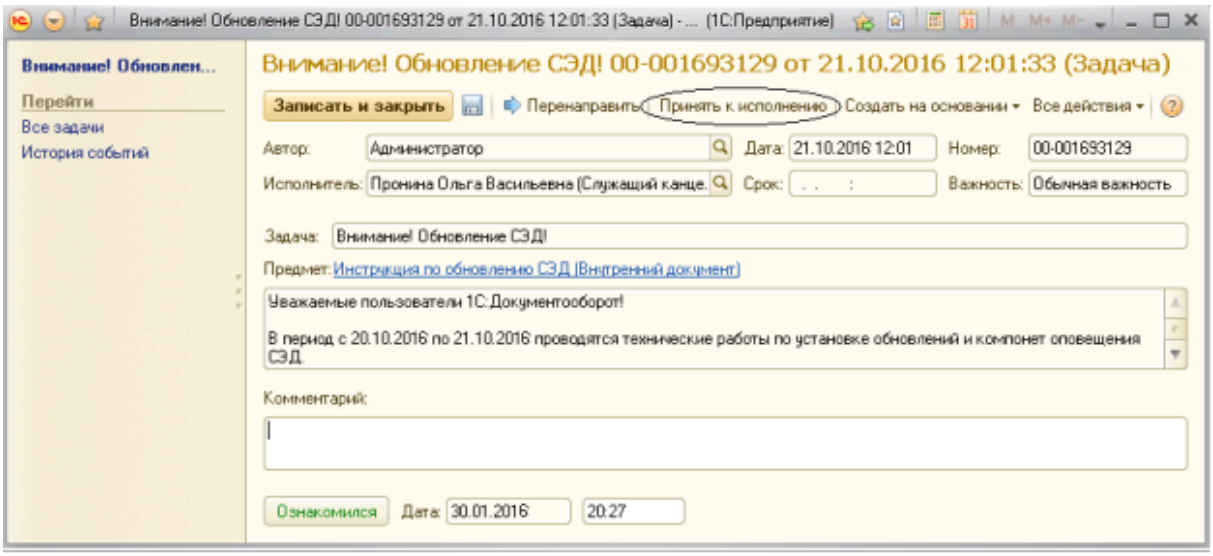


Рис.2.14. Принятие к исполнению

Принятая задача отображается обычным, а не жирным шрифтом. Если задача имела ролевую адресацию (то есть была назначена не лично Вам, а должности, исполнителем которой Вы являетесь), то при выполнении команды «Принять к исполнению» адресация задачи будет изменена и она будет адресована лично Вам, а не роли. Это удобно чтобы использовать для «захвата» задач в том случае, если исполнителями роли являются сразу несколько сотрудников. Отменить принятие к исполнению можно при помощи соответствующей команды в меню «Все действия» в карточке задачи.

Пункт меню «Перенаправить» (см. рис. 2.15) в карточке задачи используется только для переадресации задачи другому пользователю, либо роли/должности в случае ошибочной адресации задачи текущему пользователю.

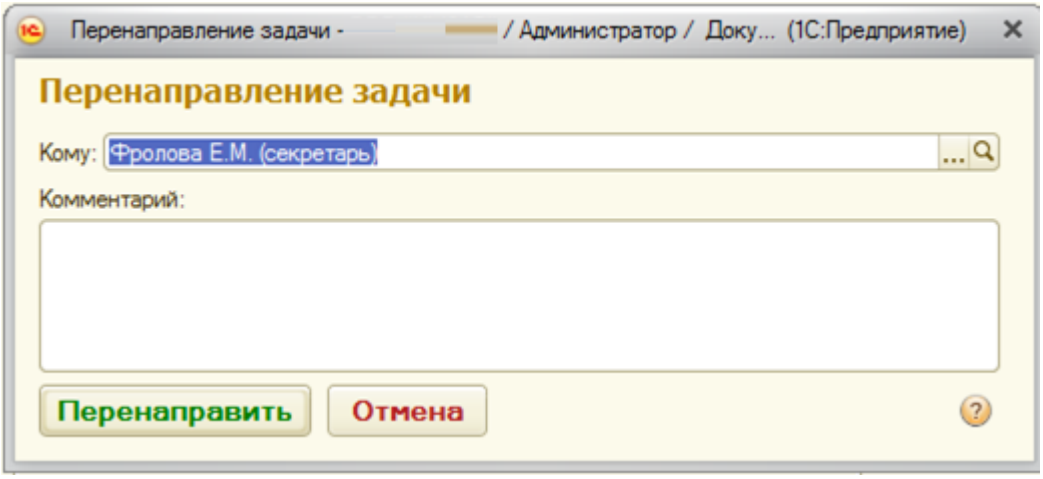


Рис.2.15. Пункт меню «Перенаправить»

При перенаправлении задачи нужно заполнить поле Исполнителю (если выбираем ФИО), либо Исполнителям роли (если выбираем должность). Так же можно ввести необходимый комментарий в поле Комментарий.

Пункт меню «Создать на основании» используется для создания новой задачи кому-либо на основании поступившей Вам.

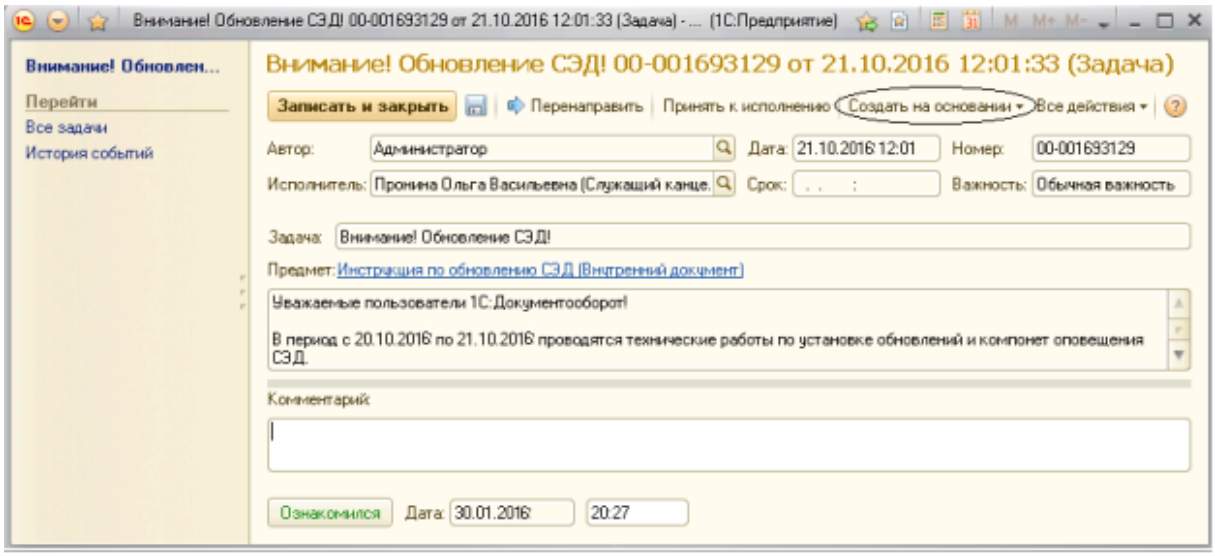


Рис. 2.16. Пункт меню «Создать на основании»

Данный пункт меню удобно использовать руководителям подразделений для быстрой переадресации поступивших документов на исполнение ответственным лицам (отписывать на исполнение), на ознакомление в нужные подразделения и так далее.

Важно! При отправке документов на исполнение ответственным лицам нужно использовать только команду «Создать на основании» и далее выбрать Исполнение. Это важно,так как при получении задачи на исполнение на свой рабочий стол пользователь-адресат увидит наименование задачи «Исполнить...» и поймет, что данный документ был отписан ему именно на исполнение. Адресат в личный доступ документа добавляется автоматически.

При использовании же команды «Перенаправить» пользователь получит перенаправленную задачу, в таком же виде и с таким же наименованием, как она поступила руководителю.

1.2. Документы и файлы

Данный раздел предназначен для работы с документами и файлами:

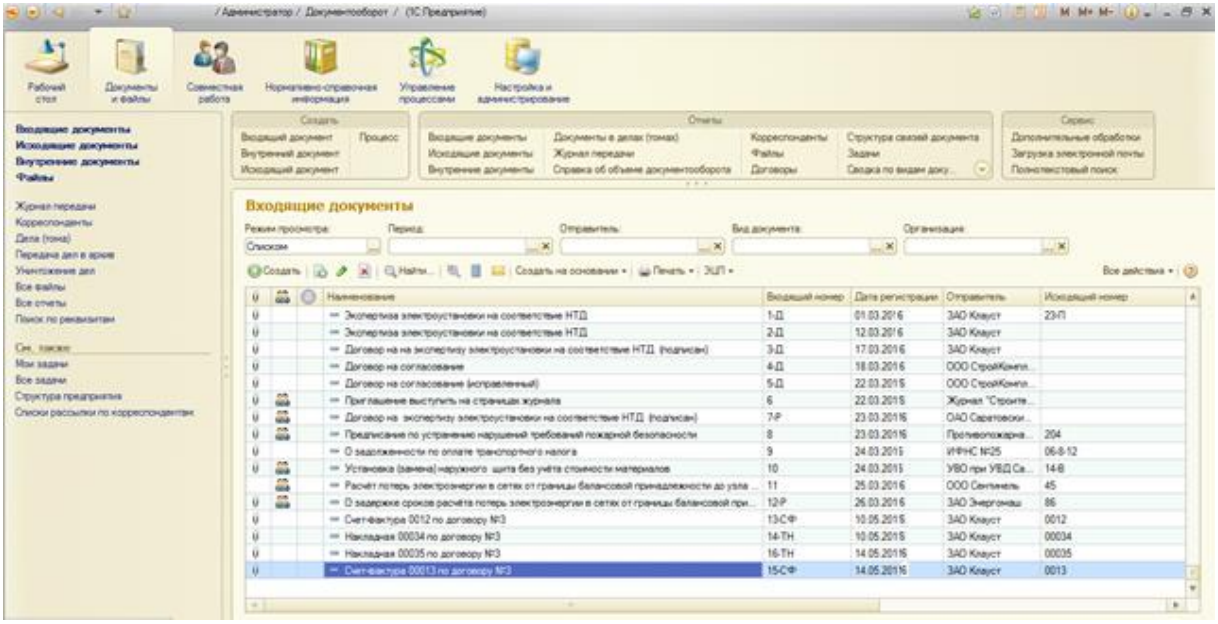


Рис. 2.17. Документы и файлы

Пользователь имеет возможность отобразить список входящих, исходящих и внутренних документов, просмотреть карточку регистрации документа и прикрепленные к карточке файлы, зарегистрировать новый документ, внести изменения в карточку и/или прикрепленный файл.

Отображение структуры внутренних документов возможно настроить при помощи командной кнопки «Вид» на панели инструментов.

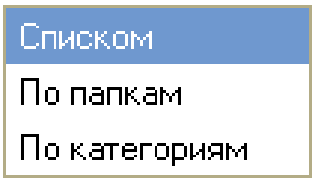


Рис. 2.18. Командная кнопка «Вид»

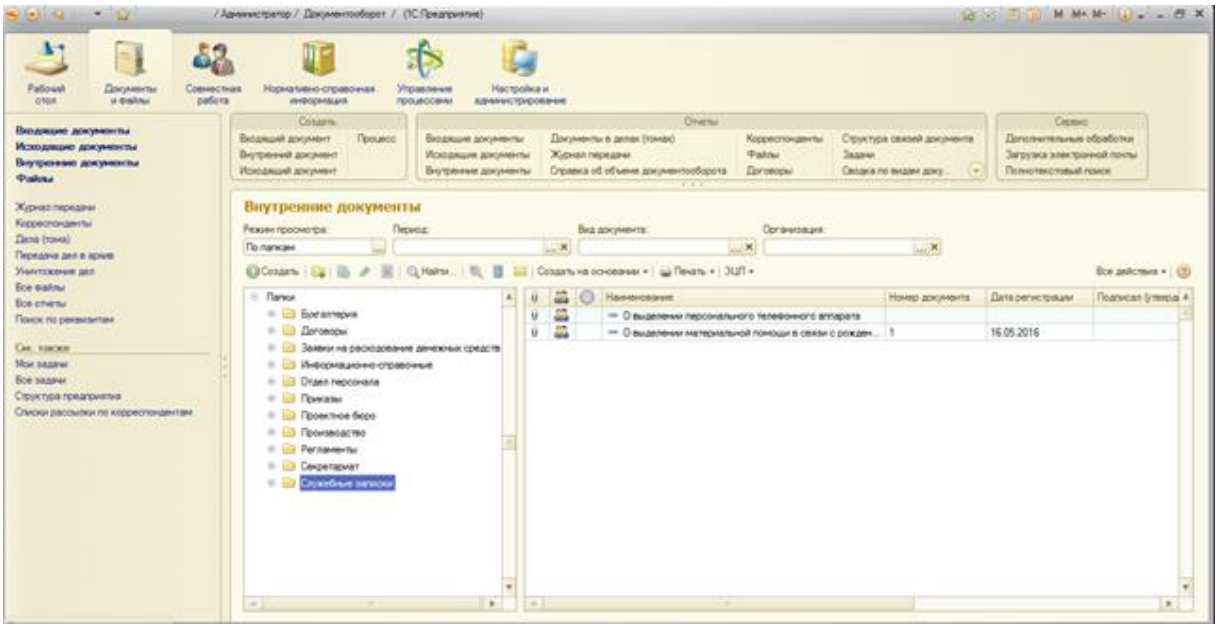


Рис.2.19. Отображение структуры внутренних документов

Список документов можно представить в удобном для пользователя виде, установив группировку для представления списка выбором из контекстного меню, вызывается при нажатии правой кнопки мыши:

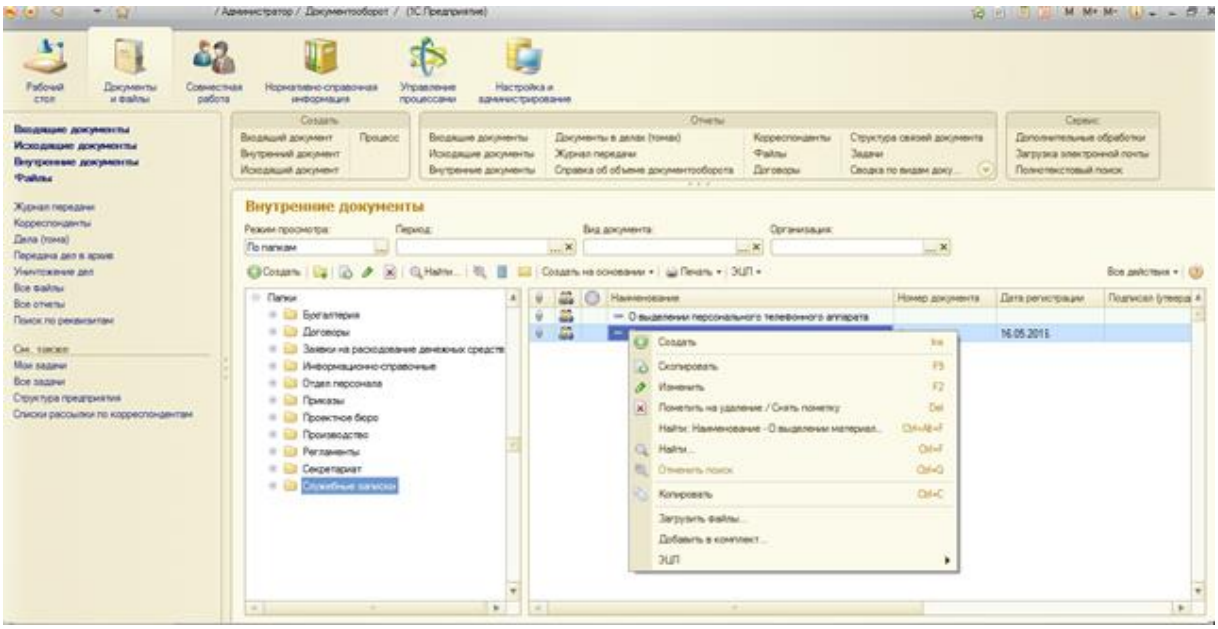


Рис. 2.20. Список документов

Окно ввода информации о новом внутреннем документе отображается на экране при нажатии кнопки «Создать».

Пользователю необходимо заполнить поля «Наименование», «Вид документа», «Корреспондент» и другие.

В случае если необходимо к данному внутреннему документу прикрепить файл с содержимым документа (заранее созданные или отсканированные документы), это осуществляется по кнопке «Добавить» на вкладке «Файлы». Прикрепляемые файлы выбираются в соответствующем диалоговом окне:

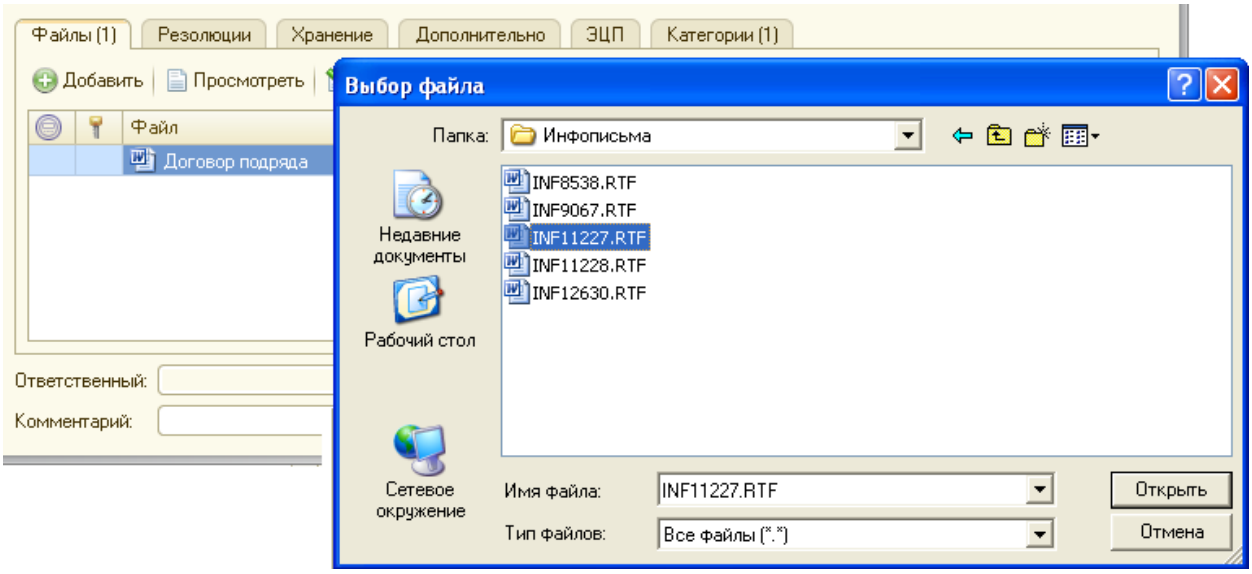


Рис. 2.21. Прикрепляемые файлы

После заполнения всех требуемых реквизитов необходимо сохранить введенный документ, нажав кнопку «Записать и закрыть». После сохранения созданный внутренний документ отобразится в общем списке внутренних документов.

Подраздел «Файлы» предназначен для работы с обычными файлами, рождающимися в процессе ежедневной деятельности организации, например, как проектные материалы, черновики, результаты обсуждений и другие. Программа позволяет работать с файлами любых типов - офисные документы, изображения, тексты, аудио- и видео-файлы, архивы, файлы систем проектирования и другие.

Структура папок подраздела «Файлы» в целом соответствует организационной структуре администрации. Каждый отдел, подразделение хранит файлы в своей папке.

Отдельно выделена общая для всех пользователей папка «Шаблоны файлов», в которой хранятся шаблоны стандартных документов, используемых в организации:

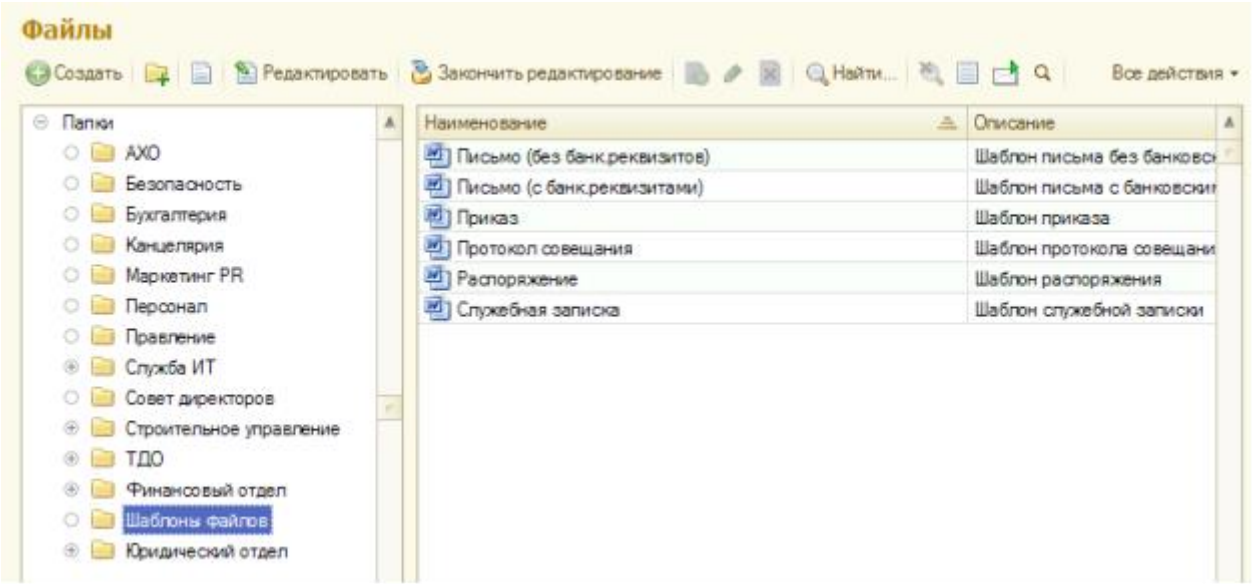


Рис. 2.22. Шаблоны файлов

Чтобы перейти с созданию файла необходимо на панели инструментов нажать кнопку «Создать». Далее необходимо выбрать способ добавления файла из предложенного (см. рис.2.23).

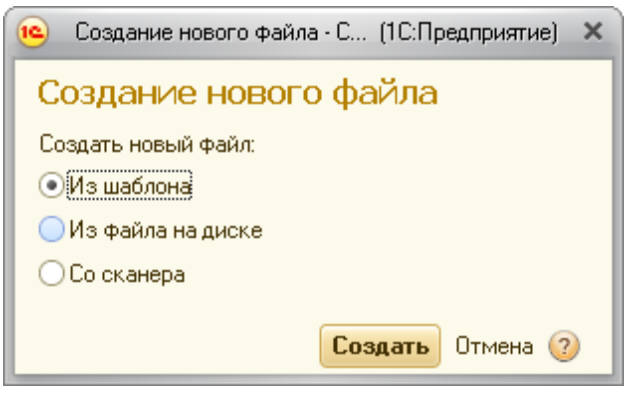


Рис. 2.23. Создание файлов

Загрузить файлы в информационную базу можно такими способами:

- перенести файлы и каталоги мышкой (Drag&Drop);
- импортировать каталоги или файлы;

– создать новый файл на основании другого файла, уже помещенного в информационную базу;

– создать новый файл путем сканирования бумажного документа.

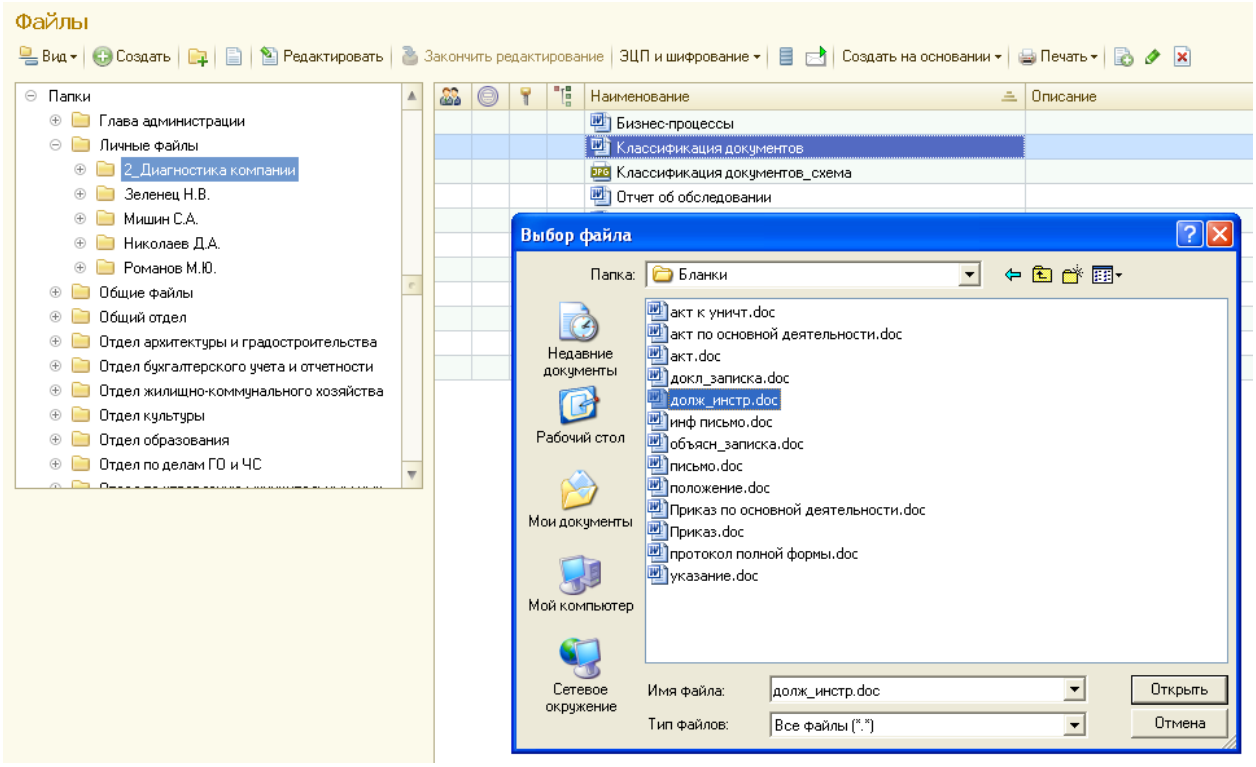


Рис. 2.24. Загрузка файлов в информационную базу

С файлами, хранящимися в информационной базе, возможны следующие действия:

– просмотреть,

– редактировать,

– закончить редактирование с сохранением измененного файла в информационной базе и снятием пометки занятости,

– занять файл с его открытием или без открытия,

– открыть каталог файла - открытие каталога на локальном компьютере с сохранением в нем файла ИБ, открытого для просмотра или редактирования,

– отменить редактирование - снятие с файла пометки занятости без сохранения изменений,

- сохранить изменения - сохранение изменений без снятия с файла пометки занятости,
- сохранить как - сохранение файла в указанном каталоге на локальном компьютере или переносном устройстве.

Программа обеспечивает коллективный доступ сотрудников к файлам как для просмотра, так и для редактирования с использованием разграничения прав доступа и механизма версионирования файлов. Конфликты при одновременном редактировании документов исключаются благодаря механизму блокировки файлов.

Для каждого файла, хранящегося в информационной базе, всегда можно установить авторство и дату создания версии. Каждая версия может сопровождаться кратким описанием внесенных изменений.

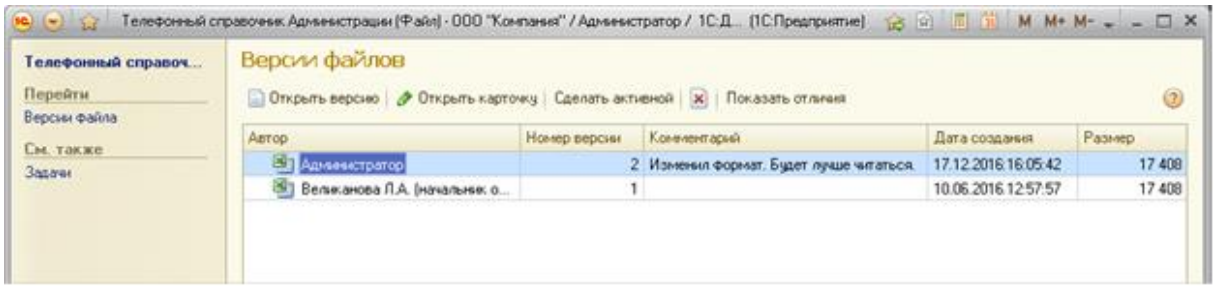


Рис. 2.25. Авторство и дата создания файлов

При редактировании файлов их предыдущие версии автоматически сохраняются в программе. Количество хранимых версий не ограничено. При необходимости можно вернуться к любой версии файла.

Непосредственно из карточки файла можно посмотреть список версий, удалить ненужные версии, сменить активную/текущую версию. Для файлов популярных форматов (doc, rtf, html, txt, odt) поддерживается сравнение версий.

Файлы, занятые текущим пользователем, отображаются в списке файлов информационной базы зеленым цветом, занятые другими пользователями - серым цветом.

При попытке отредактировать файл, занятый другим пользователем, будет выдано сообщение, как показано на рис.2.26.

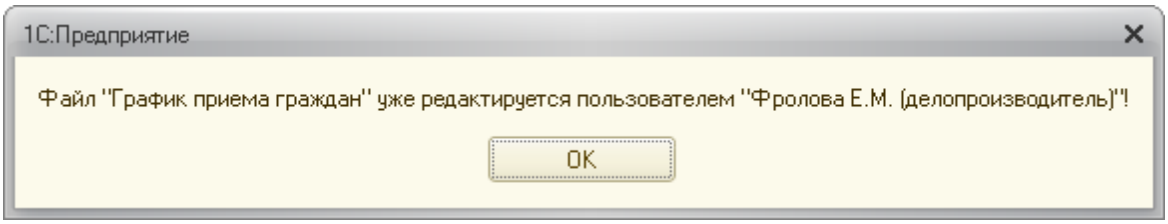


Рис. 2.26. Сообщение при редактирование другим пользователем

Список файлов, занятых пользователем, отображается у него на экране «Рабочий стол» (раздел «Редактируемые файлы») и сразу виден при запуске программы. Пользователю следует освобождать файлы, с которыми он завершил работу (кнопка «Закончить редактирование»).

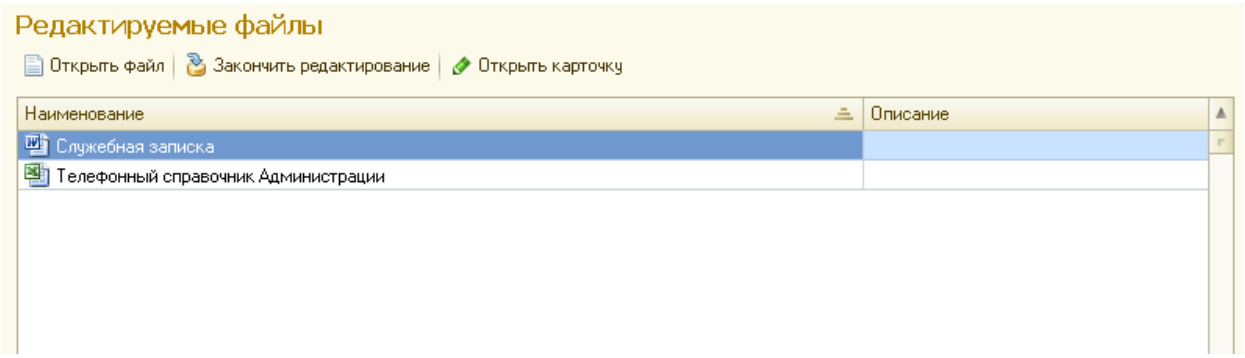


Рис. 2.27. Рекомендуемые файлы

Напоминание о занятых файлах автоматически выводится на экран при завершении работы с программой:

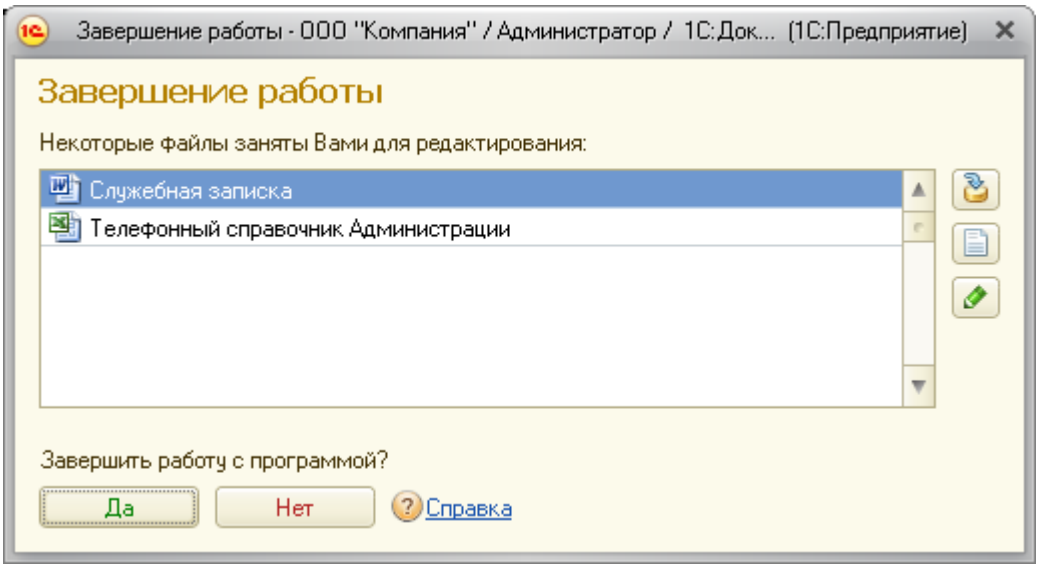


Рис. 2.28. Завершение работы

При согласовании и подготовке документа в формате MS Word несколькими пользователями удобным является механизм рецензирования, при котором все корректировки и примечания сохраняются в общем файле. Редактирование возможно для файлов, которые не заняты для редактирования.

После открытия файла в Microsoft Word необходимо перевести редактирование файла в режим «Рецензирование» Для этого в Microsoft Word следует нажать в меню «Рецензирование» и включить режим «Исправления в измененном документе»:

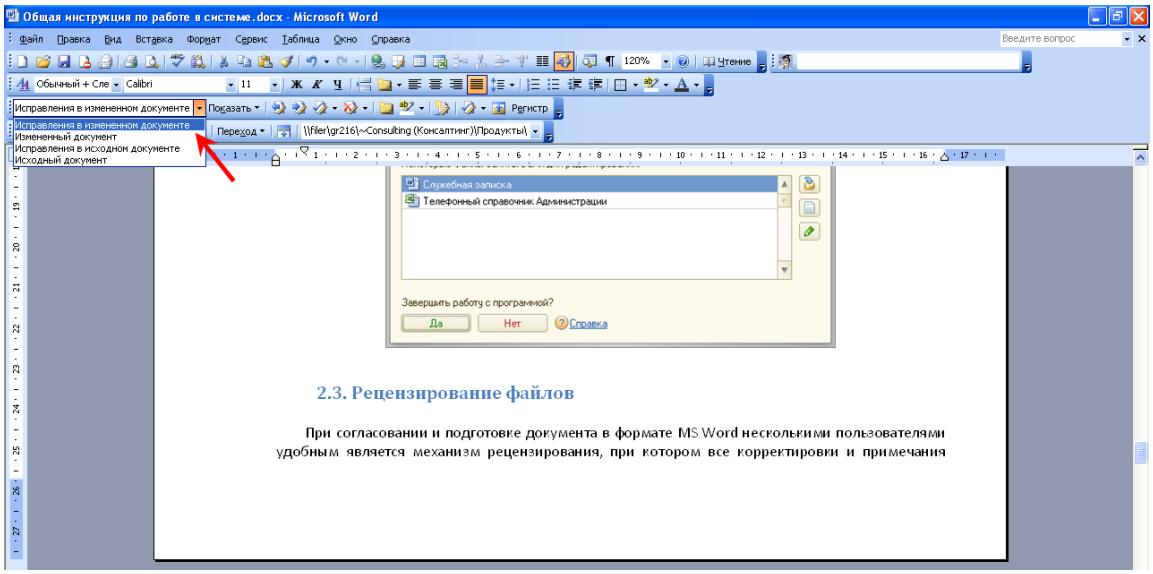


Рис. 2.29. Microsoft Word

Для внесения своих исправлений используется раздел меню «Примечания». Для этого необходимо выделить текст, с которым пользователь не согласен или хочет добавить свой комментарий, и далее нажать кнопку «Добавить примечание».

Примечания разных пользователей будут выделены разным цветом.

Пример добавления своего варианта текста в режиме примечаний представлен на рис.2.30.

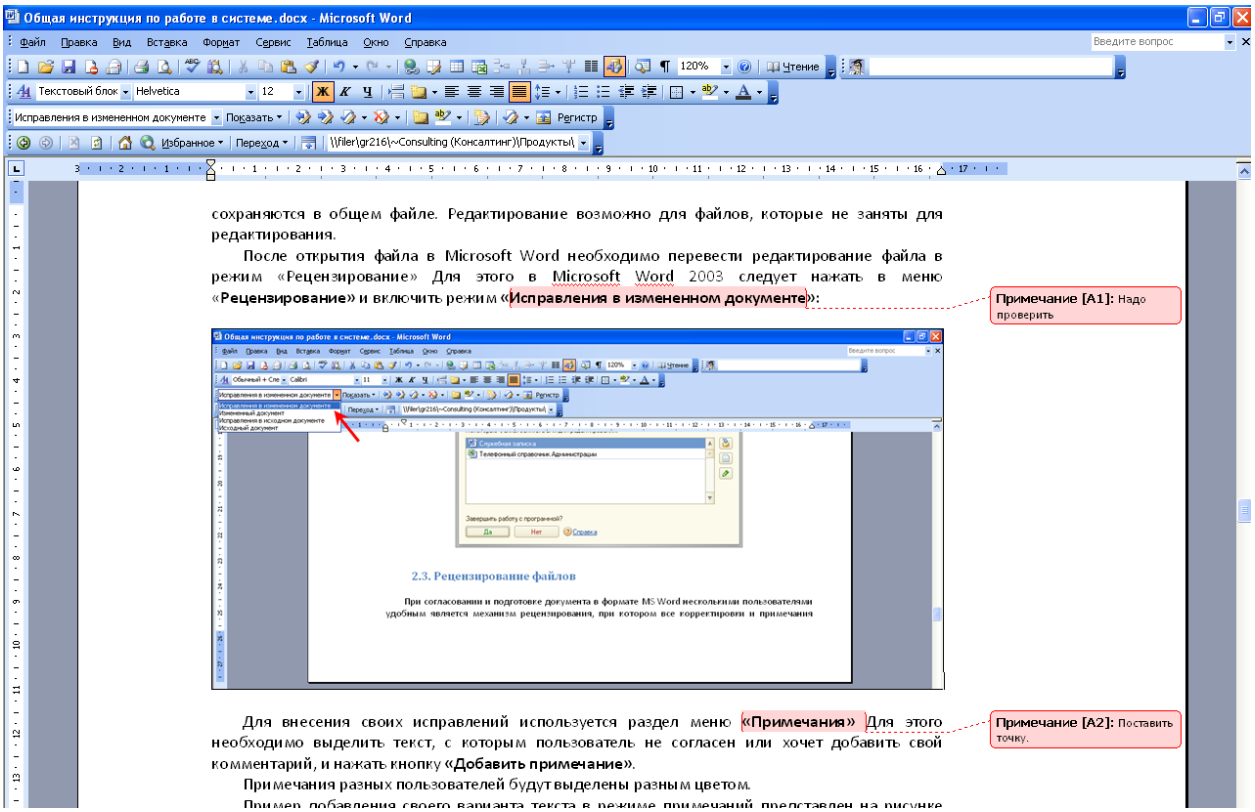


Рис. 2.30. Пример добавления своего варианта текста в режиме примечаний

После окончания редактирования в программе Word документ необходимо сохранить и вернуться в систему, для закрытия редактирования. Для этого следует поставить курсор на редактируемый документ, нажать кнопку «Закончить редактирование» и выполнить инструкцию системы по сохранению версии, добавив, если нужно, произвольный комментарий:

2.4. Поиск по содержимому файлов

На каждый файл в программе автоматически заводится учетно-регистрационная карточка, которая обеспечивает его быстрый поиск. Кроме того, в программе предусмотрен полнотекстовый поиск не только по всем полям учетно-регистрационной карточки, но и по содержимому файлов популярных форматов.

Полнотекстовый поиск производится по всем данным (документы, файлы, задачи, бизнес- процессы, еженедельные отчеты и так далее) с учетом русской, английской и украинской морфологии. Поддерживается поиск похожих слов.

В стандартном варианте в программе реализована возможность использования семи бизнес- процессов.

3.1 Поручение. Поручением пользуются руководители для быстрейшего оформления разных поручений для исполнителя. Оно может быть оформлено на основании любыхдокументов

3.2 Исполнение. Исполнение похоже на поручение. Руководители используют его так же быстрого оформления исполнения, используется в отличии от поручения для нескольких исполнителей с таким же сроком.

3.3 Ознакомление. Ознакомлением пользуются все службы персонала, рекламы, службы которые ознакамливают персонал, а так же распоряжениями и другими документами, которые предназначены для обязательного ознакомления.

3.4 Согласование. Согласованием пользуется любой сотрудник для обхода документами всех участников согласования по заранее составленному списку, после всего етого документ возвращается к автору процесса.

3.5 Рассмотрение. Этот процесс используют для передачи входящего или внутреннего документа лицу на котором ответственность на рассмотрение. При создании процесса автор может вписывать проект резолюции, чтобы помочь лицу которое ответственное.

3.6 Утверждение. Утверждение это процесс который используют для автоматизации процесса утверждения исходящих документов управляющим перед отправкой. В начале руководителю поступает задача «Утвердить», которая содержит ссылку на исходящий документ, как только руководитель рассмотрит документ автору поступает задача «Ознакомиться с результатами утверждения».

3.7 Регистрация. Регистрацию используют если нужно готовый документ зарегистрировать в канцелярии в журнале. Сам документ (служебная записка и т.д)) готовится и согласовывается исполняющим лицом.

3.8 Запуск задачи «Поручение»

Данный бизнес- процесс используется для выдачи поручения одному сотруднику.

Процесс «Поручение» может быть подготовлен и запущен на основании любого документа введенного в систему. Для этого необходимо выбрать нужный документ, выделив его, и далее нажать кнопку «Создать на основании», и выбрать «Поручение».

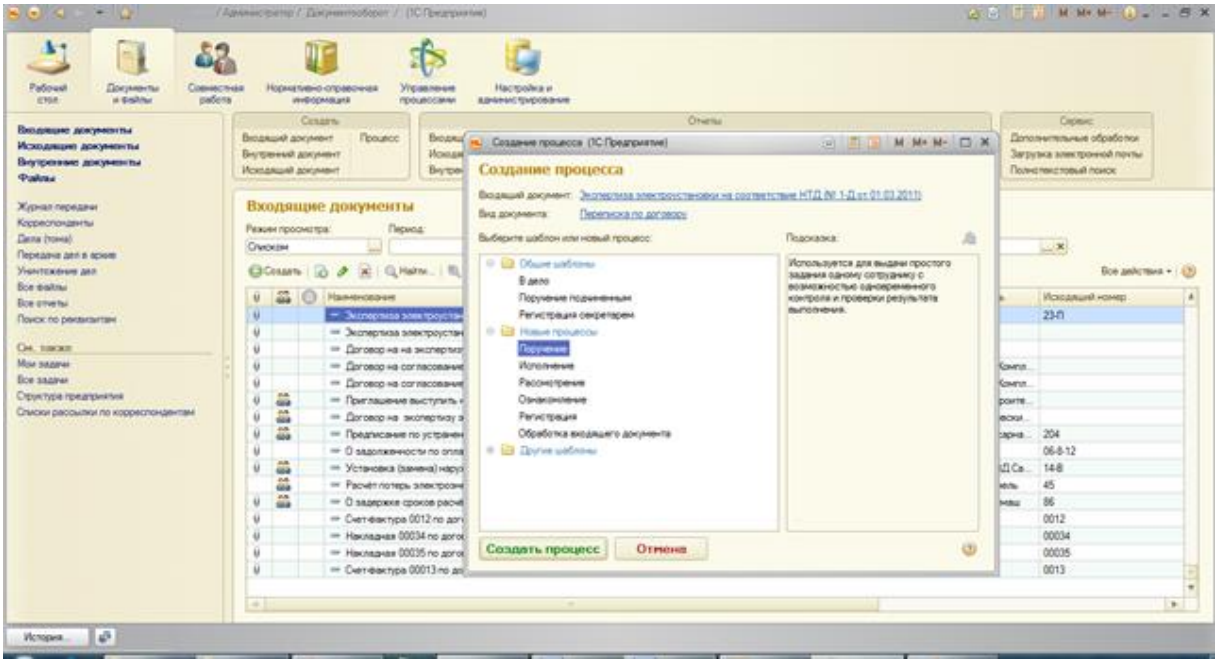


Рис. 2.33. Запуск задачи «Поручение»

Либо зайти в раздел Задачи и бизнес-процессы и нажать кнопку Создать:

Далее откроется карточка нового бизнес-процесса. Нужно заполнить основные поля и нажать кнопку Стартовать и закрыть. Бизнес-процесс запущен.

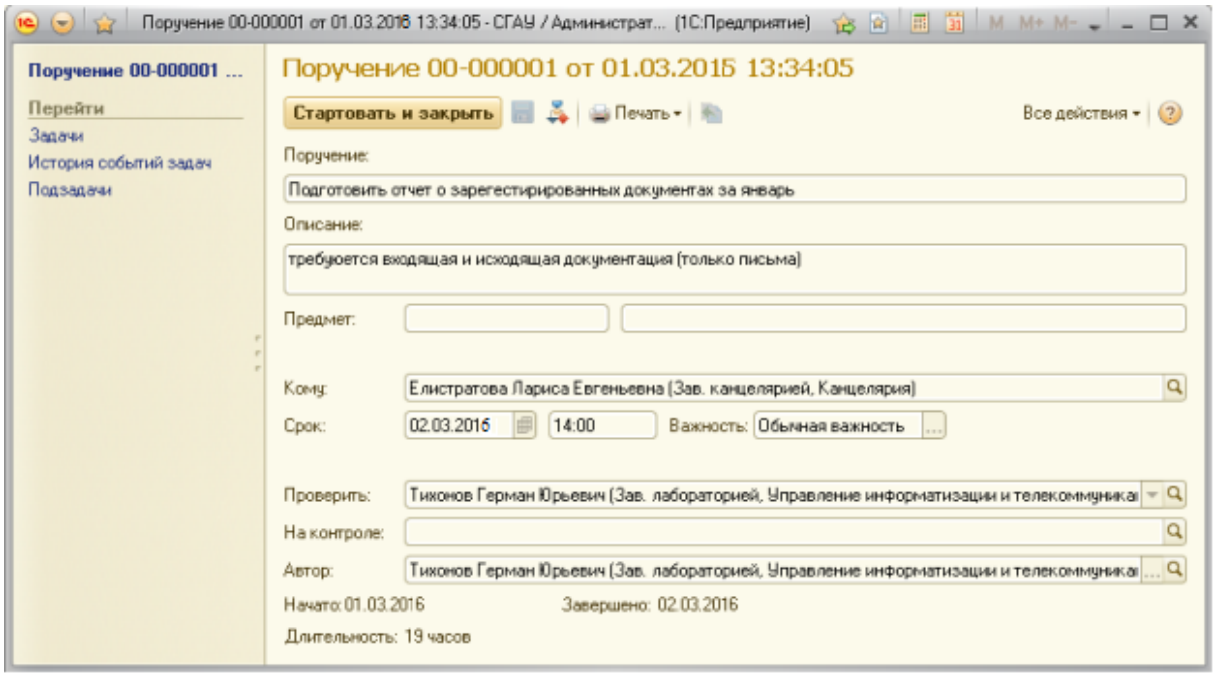


Рис. 2.34. Карточка нового бизнес-процесса

3.1.1 Завершение задачи «Поручение».

После выполнения поручения исполнитель открывает соответствующую задачу на рабочем столе и фиксирует ее исполнение нажатием кнопки Выполнено:

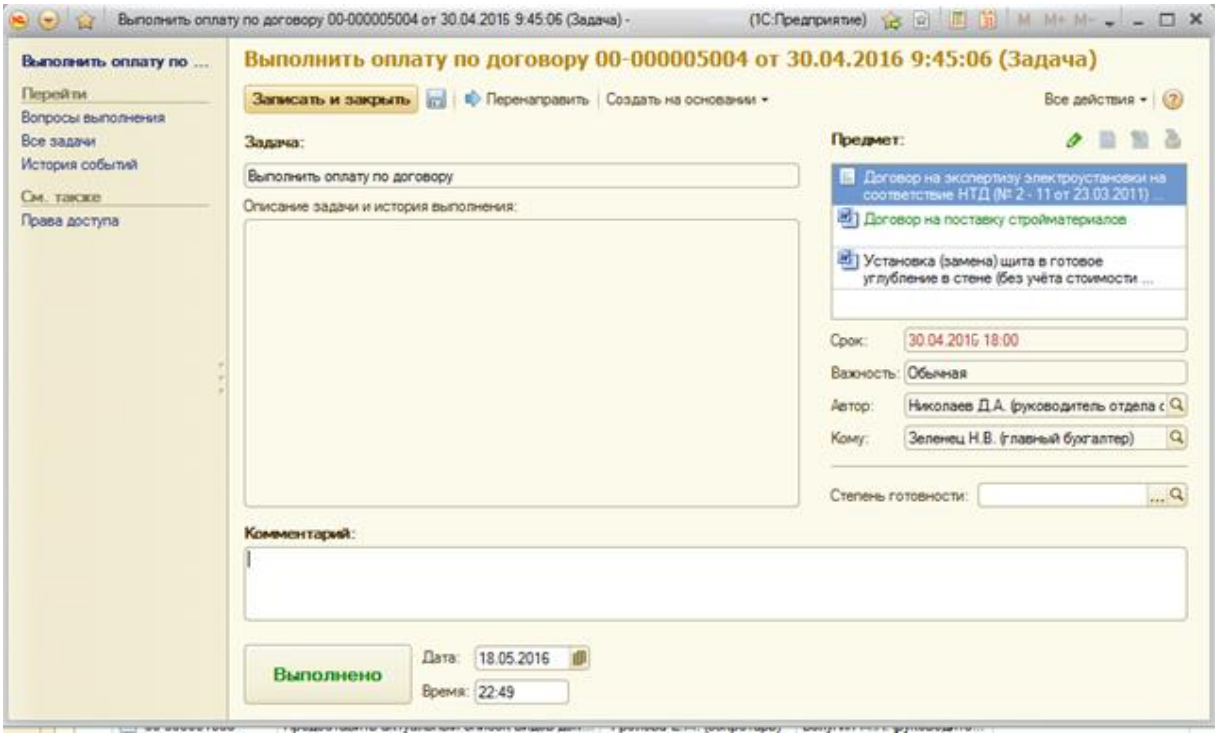


Рис. 2.36. Завершение задачи «Поручение»

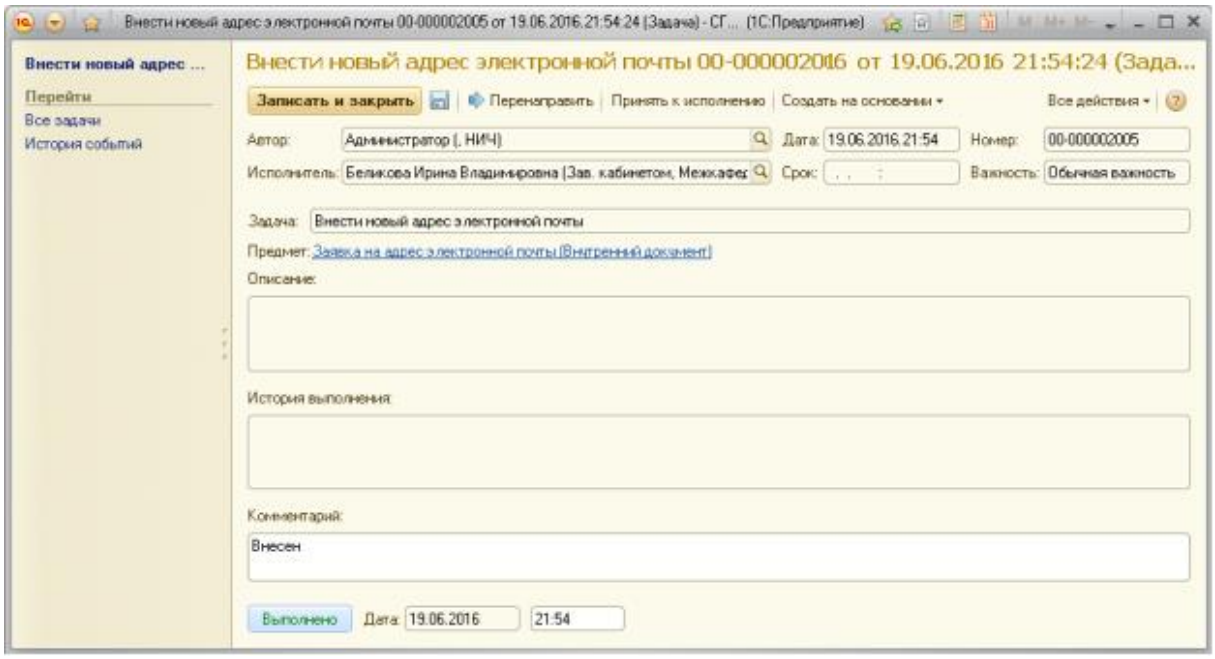


Рис. 2.37. Завершение задачи «Поручение»

При необходимости в поле «Комментарий» внести соответствующие комментарии.

3.1.2 Задача «Контроль поручения»

Пользователь, указанный в поле «Контролер», получит задачу одновременно с исполнителем. В обязанности Контролера входит слежение за сроками выполнения поручения. Для завершения задачи контроля, необходимо нажать соответствующую кнопку.

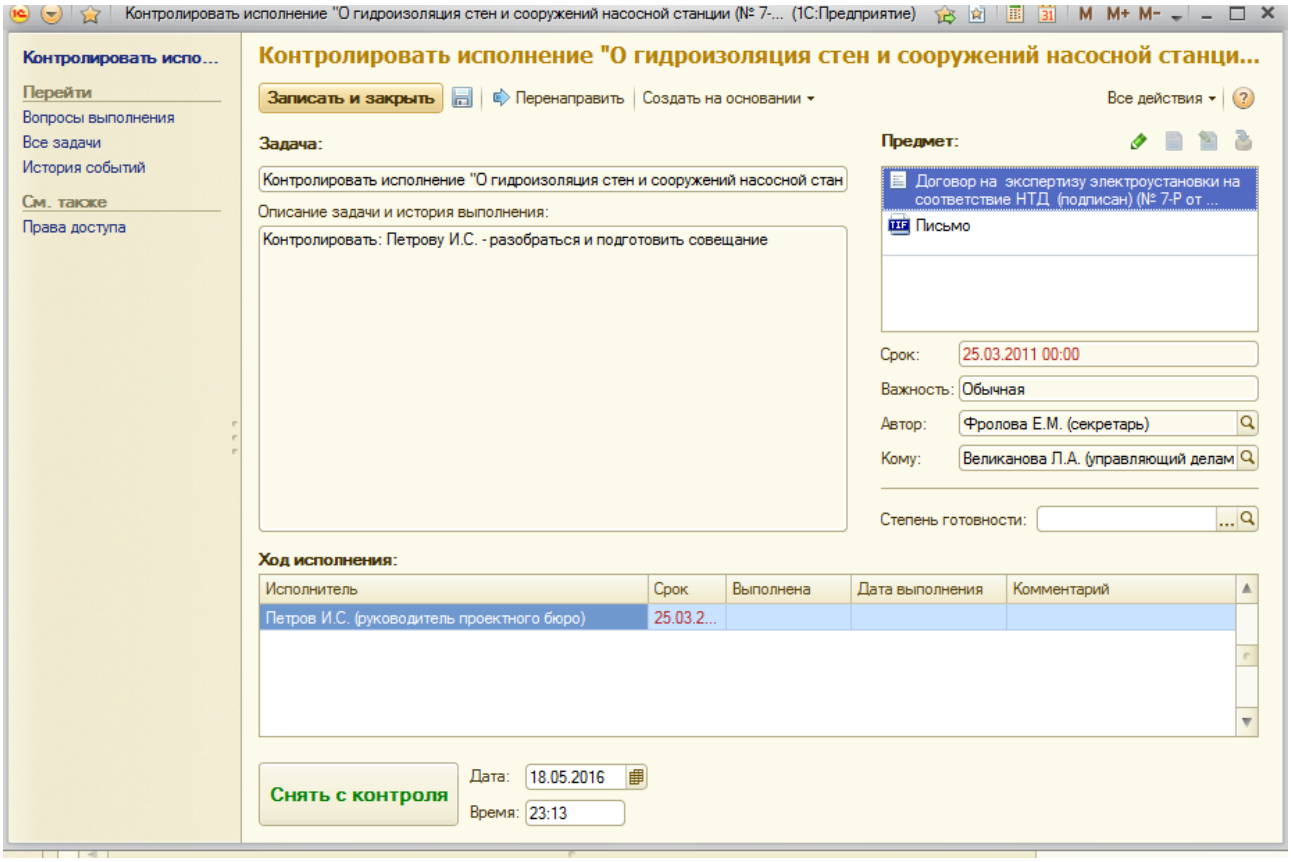


Рис. 2.38. Задача «Контроль поручения»

После выполнения задачи исполнителем, она поступает на проверку проверяющему (по умолчанию - автор бизнес-процесса). Проверяющий имеет возможность вернуть исполнителю задачу на доработку, и тогда эта же задача поступит адресату заново.

Бизнес-процесс «Поручение» считается завершенным, после того как исполнитель выполнил свою задачу и проверяющий подтвердил выполнение, задача контролера при этом завершается автоматически.

3.2 Запуск задачи «Исполнение»

Этот бизнес-процесс используется для автоматизации исполнения распоряжений руководителя одним или несколькими исполнителями с одновременным контролем. Его можно создавать непосредственно, из карточки документа, при помощи меню «Создать на основании».

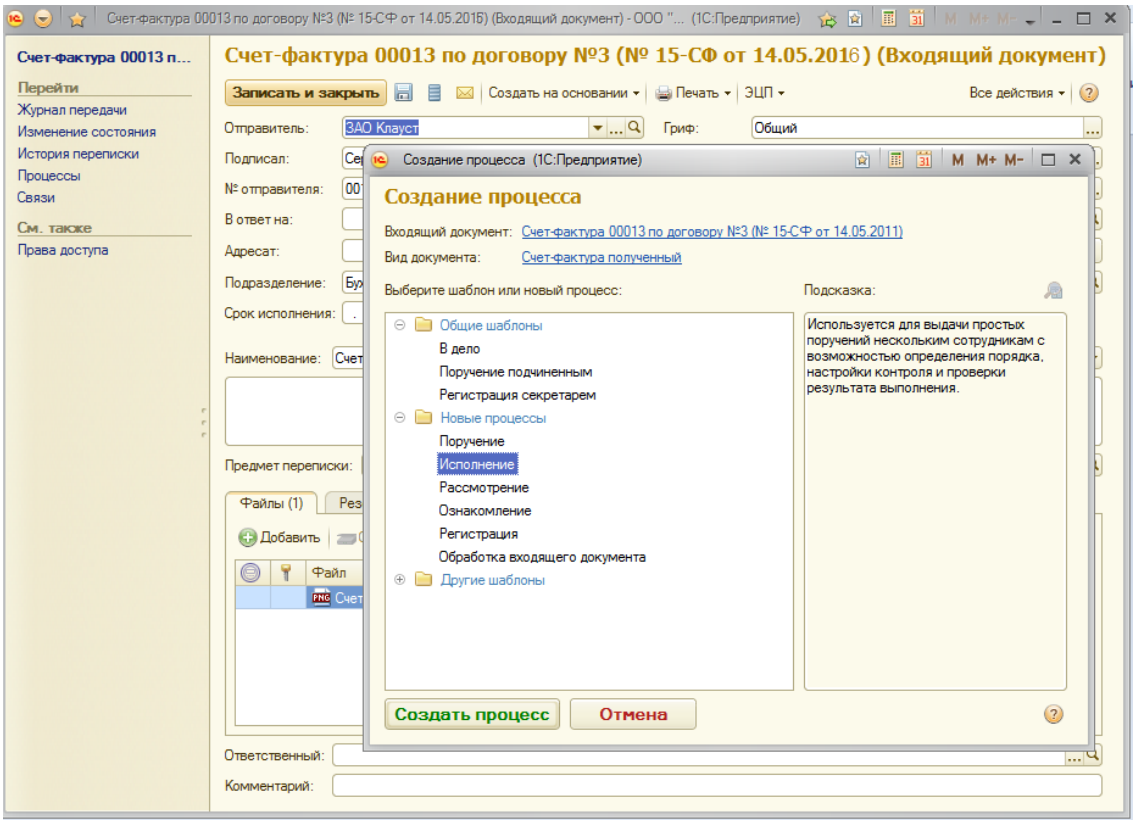


Рис. 2.39. Запуск задачи «Исполнение»

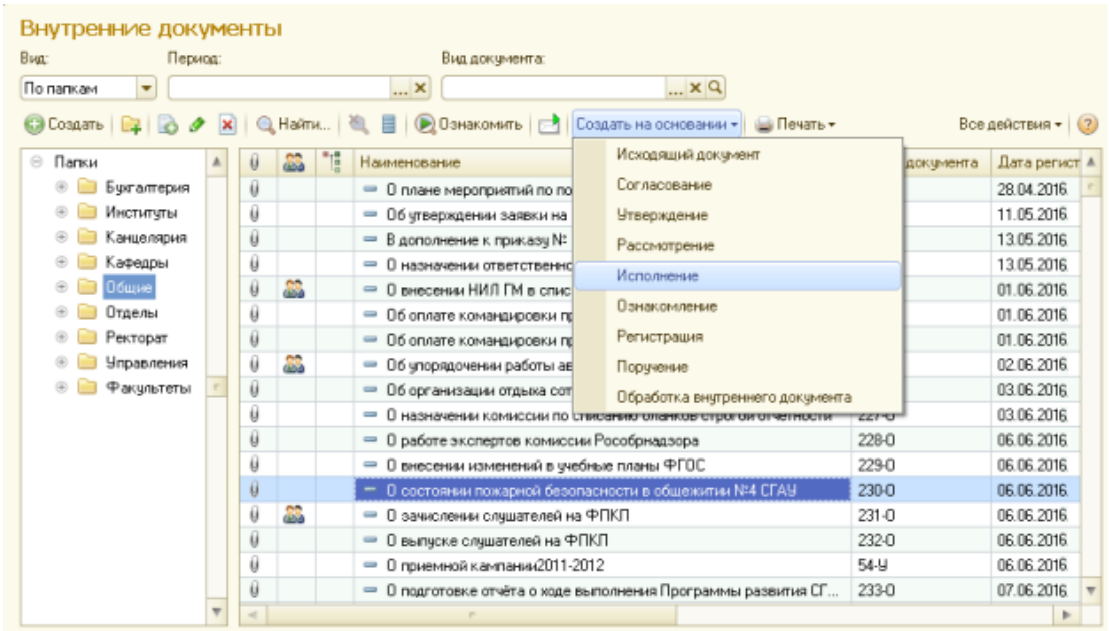


Рис. 2.40. Запуск задачи «Исполнение»

В открывшейся карточке заполняем необходимые поля.

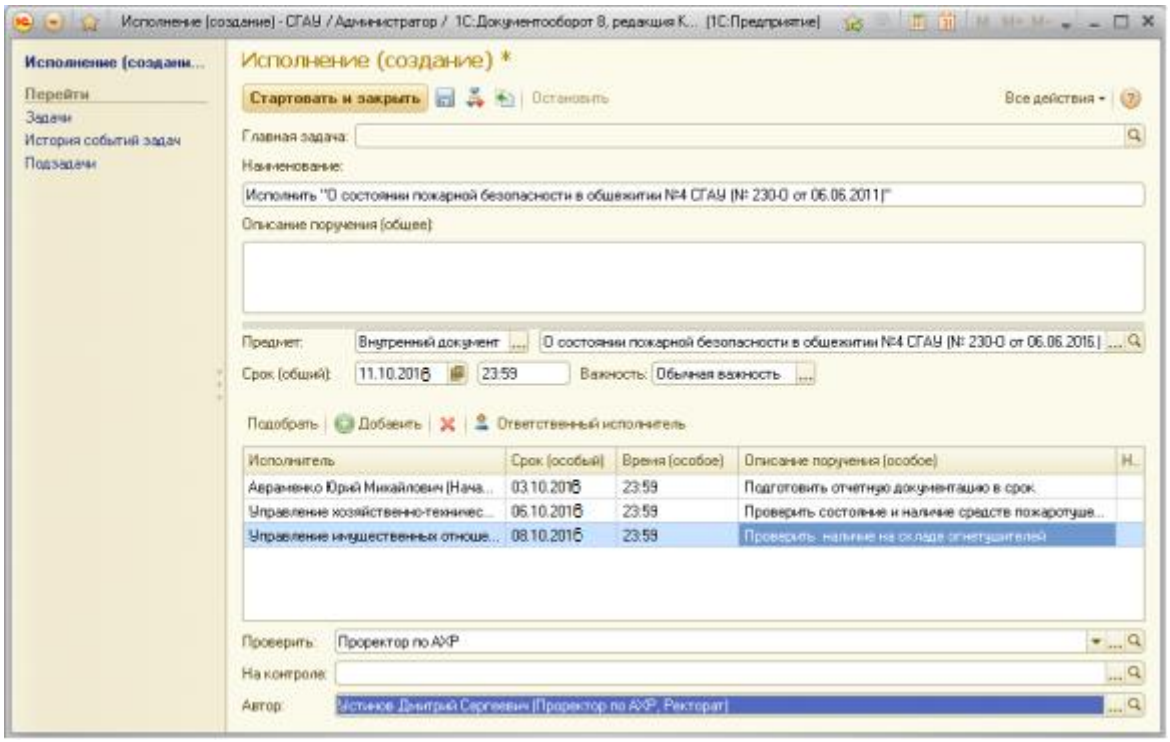


Рис. 2.41. Создание на основании

В поле Описание поручения вносится описание поручения общего для всех адресатов бизнес-процесса.

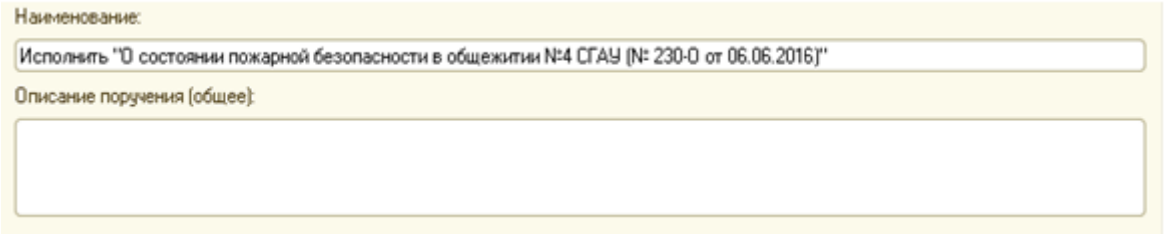


Рис. 2.42. Описание поручения

При подборе исполнителей можно для каждого из них указать индивидуальный срок и описание поручения.

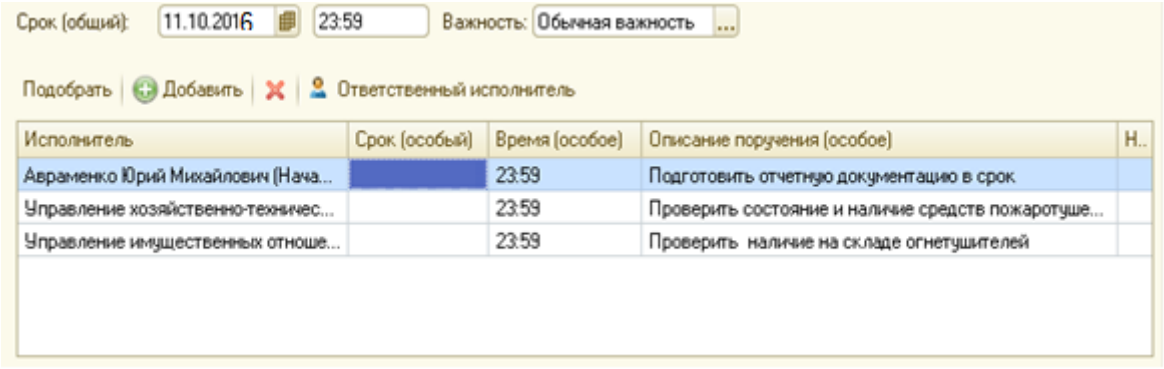


Рис. 2.43. Описание поручения

Пользователь, указанный в поле На контроле, получит задачу Контролировать одновременно с исполнителями. В обязанности контролера обычно входит слежение за сроками исполнения.

Пользователь, указанный в поле Проверить, получит задачу после того, как все исполнители выполнили свои задачи. В обязанности проверяющего входит проверка результата исполнения задач.

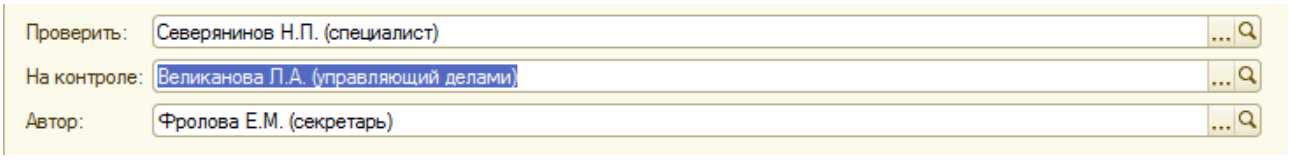


Рис. 2.44. Проверка результата исполнения задач

Запускается бизнес-процесс нажатием на кнопку « Стартовать и закрыть».

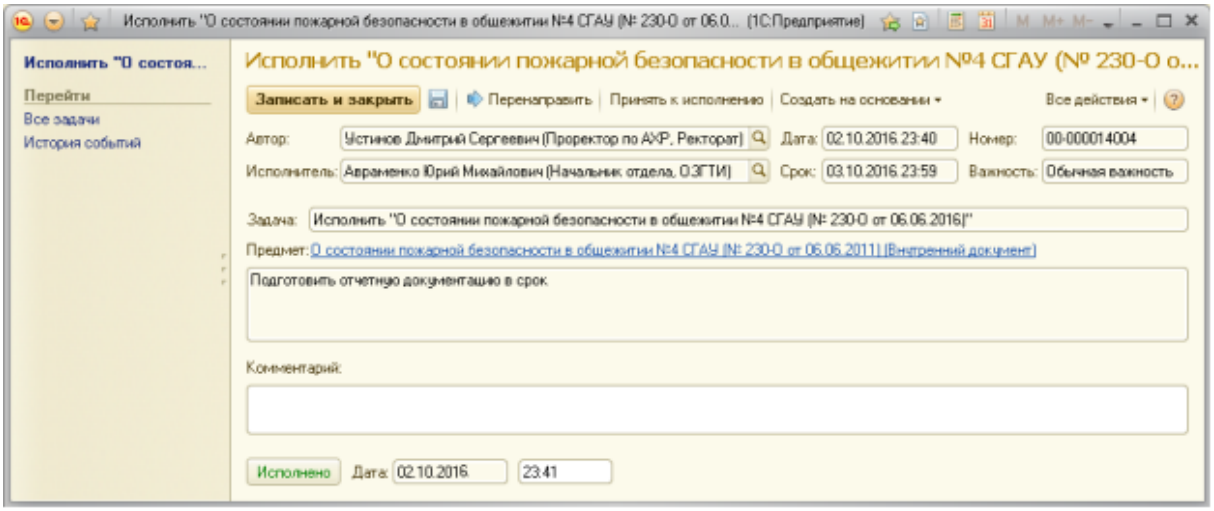


Рис. 2.45.Завершение задачи «Исполнение»

3.2.1 Завершение задачи «Исполнение»

После выполнения поручения исполнитель открывает соответствующую задачу на рабочем столе, и фиксирует ее исполнение нажатием кнопки «Исполнено»:

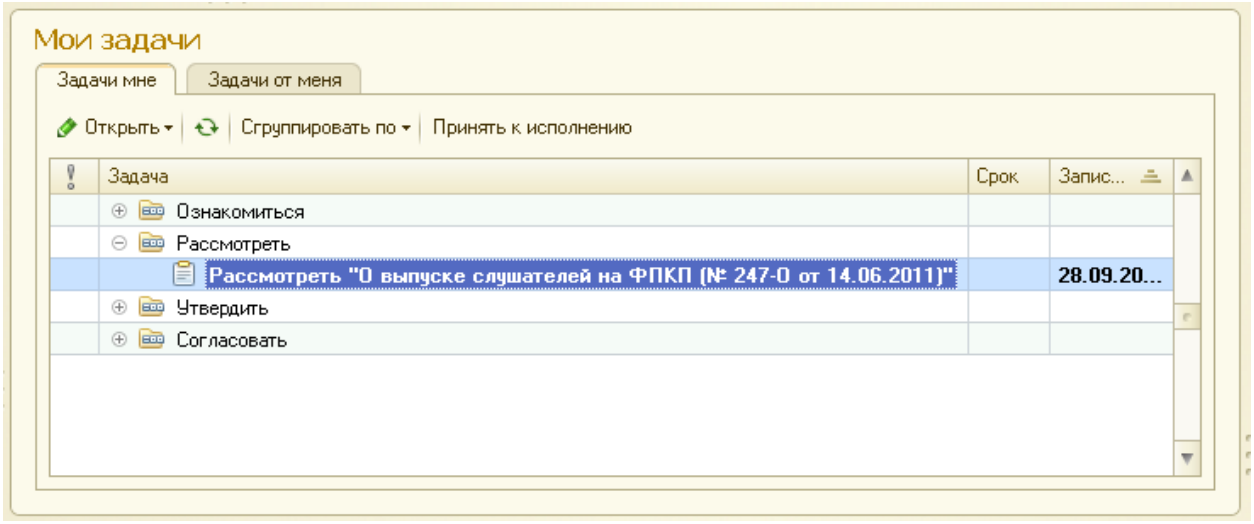


Рис. 2.46. Задача «Контроль исполнения»

Результаты могут быть отражены в поле «Комментарий».

3.2.2 Задача «Контроль исполнения»

При работе со списком задач на рабочем столе пользователю следует обратить внимание на возможности опции Сгруппировать по на рабочем столе.

В данном примере опция включена в режим «Точка маршрута»:

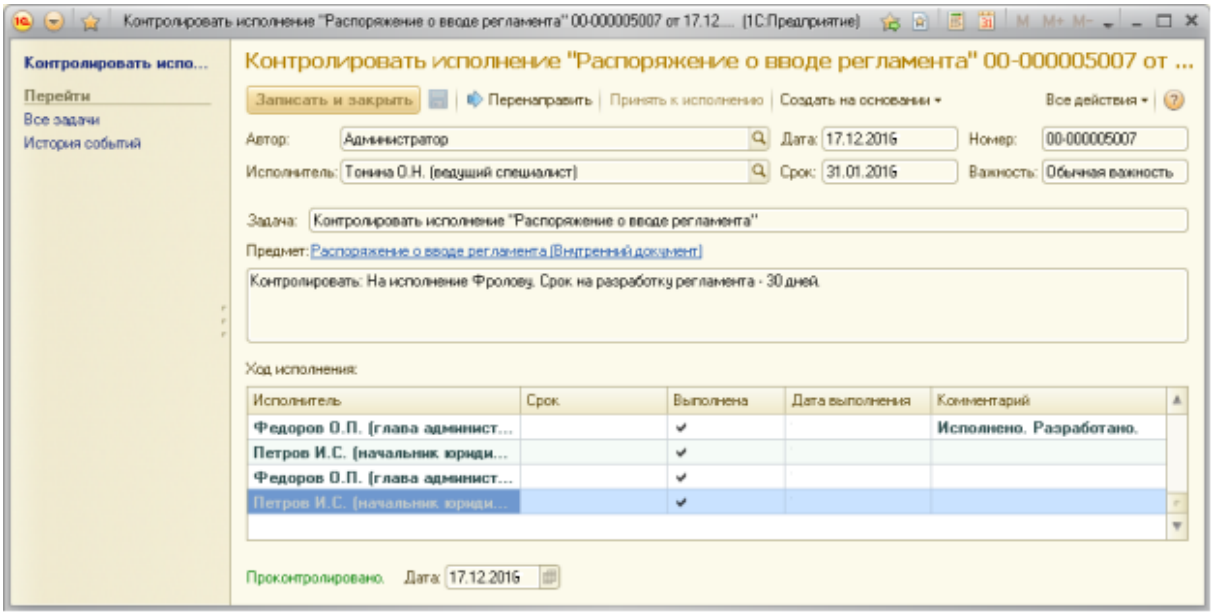


Рис. 2.47. «Контролер»

Задача «Контролировать» становится выполненной после того, как все исполнители данной задачи сделают отметку о выполнении поручения.

«Контролер» может, открыв задачу, следить за ходом ее выполнения:

Бизнес-процесс «Исполнение» считается завершенным, после того как исполнитель выполнил свою задачу и проверяющий подтвердил выполнение, задача контролера при этом завершается автоматически.

3.3 Запуск задачи «Ознакомление»

Процесс «Ознакомление», может быть подготовлен и запущен на основании любого документа, введенного в систему. Для этого необходимо выбрать нужный документ, в его карточке выбрать пункт меню «Создать на основании» и далее - выбрать «Ознакомление»:

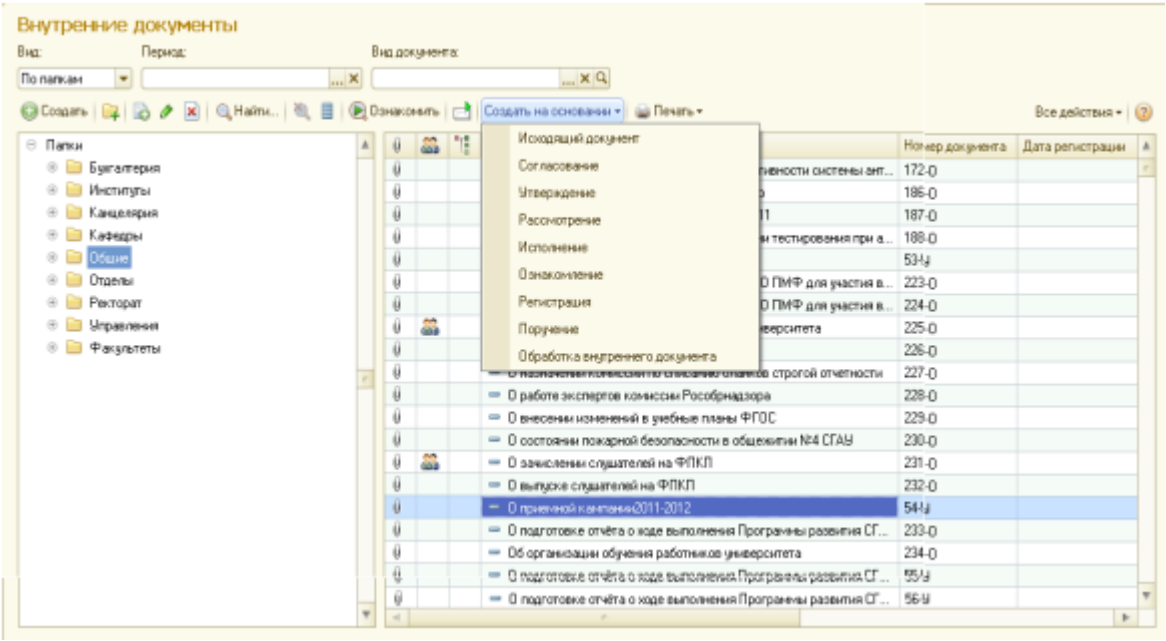


Рис.2.48. Запуск задачи «Ознакомление»

По аналогии процесс Ознакомление можно создать, осуществив переход в пункт меню Задачи и бизнес-процессы.

Далее на панели навигации выбрать пункт Ознакомления:

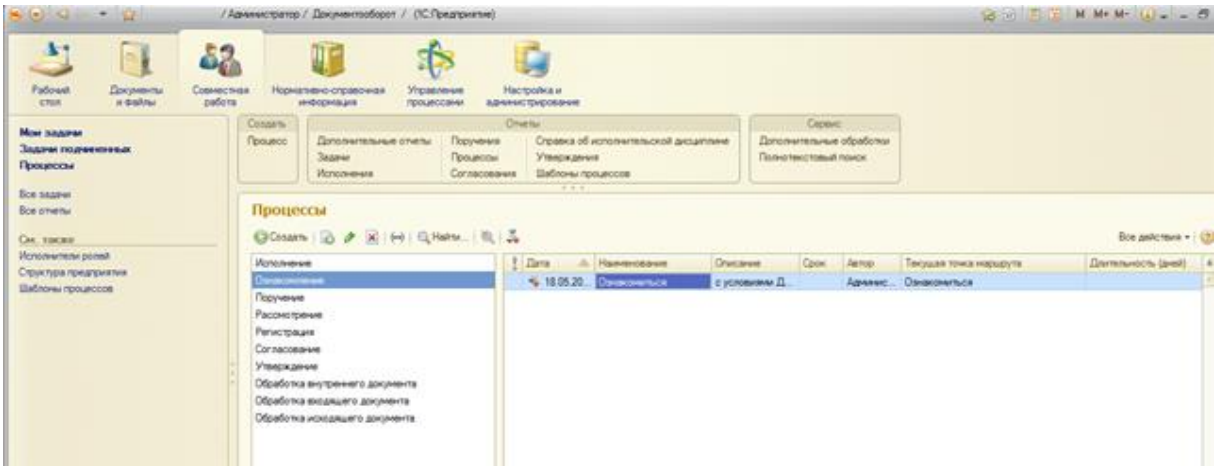


Рис. 2.49. Пункт Ознакомления

Форму представления списка можно изменить, выбрав пункт Изменить форму из контекстного меню, оно вызывается нажатием правой кнопки мыши:

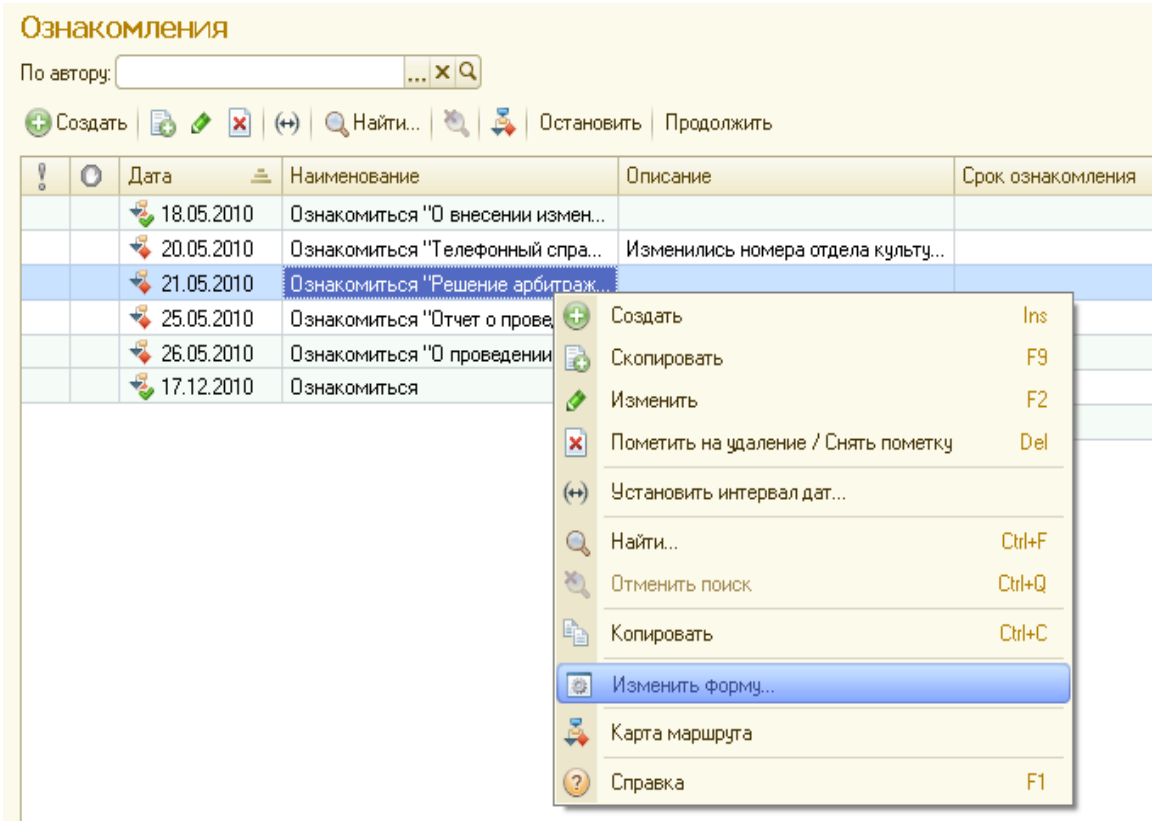


Рис. 2.50. Форму представления списка

Окно ввода информации о новом процессе отображается на экране при нажатии кнопки Создать:

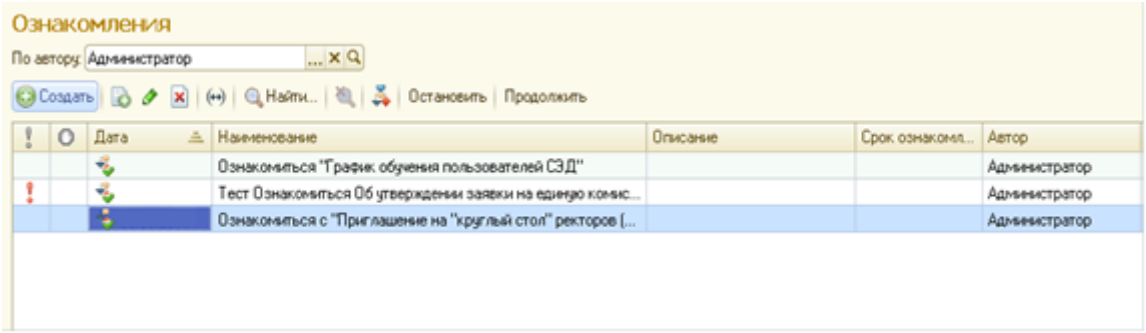


Рис. 2.51. Окно ввода информации о новом процессе

Далее нужно заполнить необходимые поля карточки задачи, указать список адресатов на ознакомление и приложить предмет/документ, относительно которого создается ознакомление:

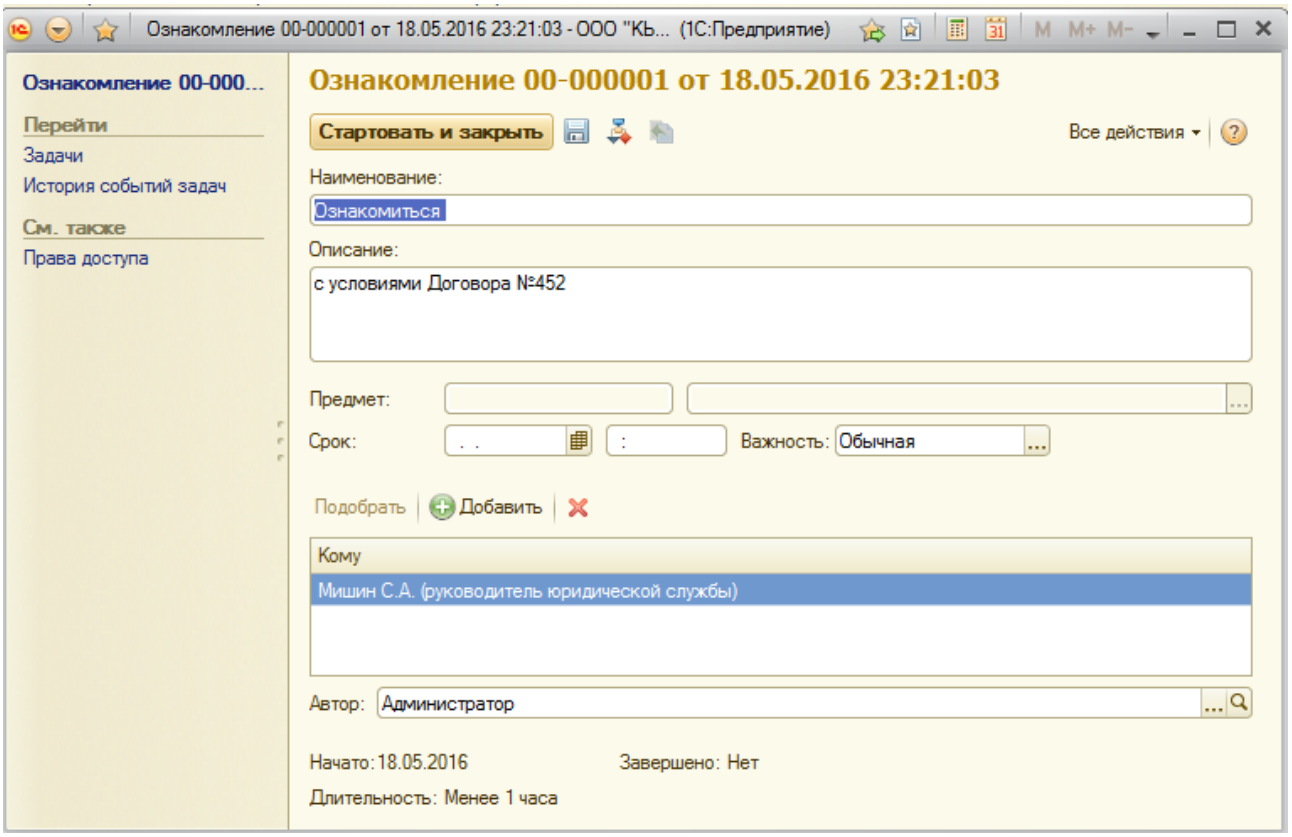


Рис. 2.52. Заполнение необходимых полей карточки задачи

Процесс запускается при нажатии кнопки Стартовать и закрыть.

После запуска процесса Ознакомление, пользователи - получившие задачу Ознакомиться могут перейти по ссылке Предмет и просмотреть прикрепленный внутренний документ:

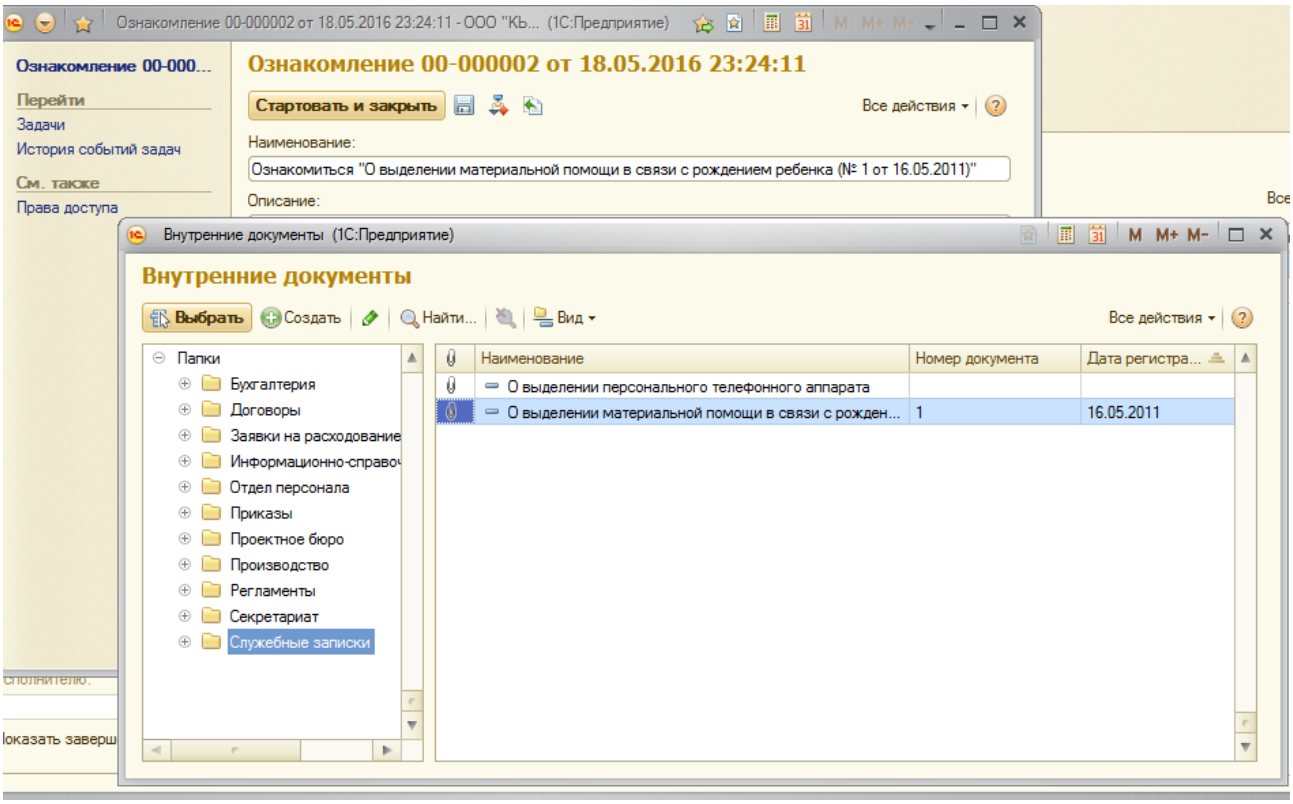


Рис. 2.53. Просмотр прикрепленного внутреннего документа

3.3.1 Завершение задачи «Ознакомление»

После запуска процесса «Ознакомление», адресаты, которым была направлена данная задача (Ознакомление) могут увидеть ее на рабочем столе программы:

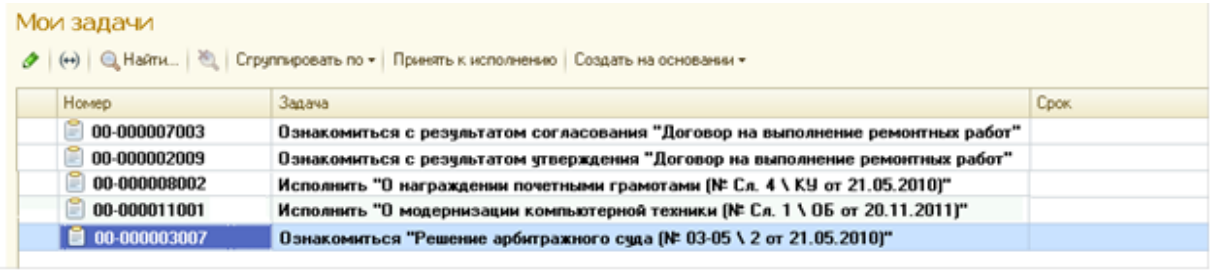
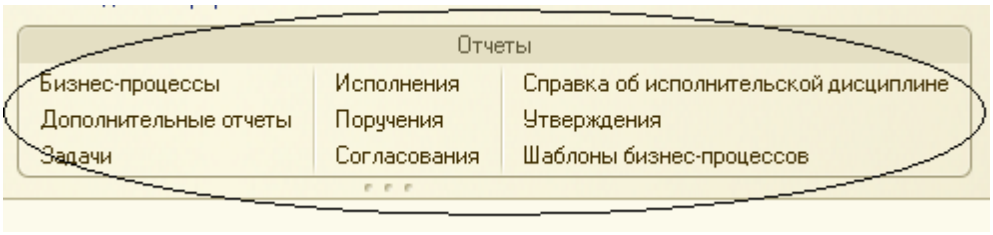


Рис. 2.54. Завершение задачи «Ознакомление»

При открытии карточки задачи сотрудник - исполнитель может просмотреть прикрепленный к ней файл, нажав на ссылку Предмет, внести свои комментарии. Задача завершается нажатием на кнопку - Ознакомлен.

3.3.2 Контроль исполнения процесса

Ответственный сотрудник может контролировать процесс Ознакомления с помощью отчетов, представленных в программе:



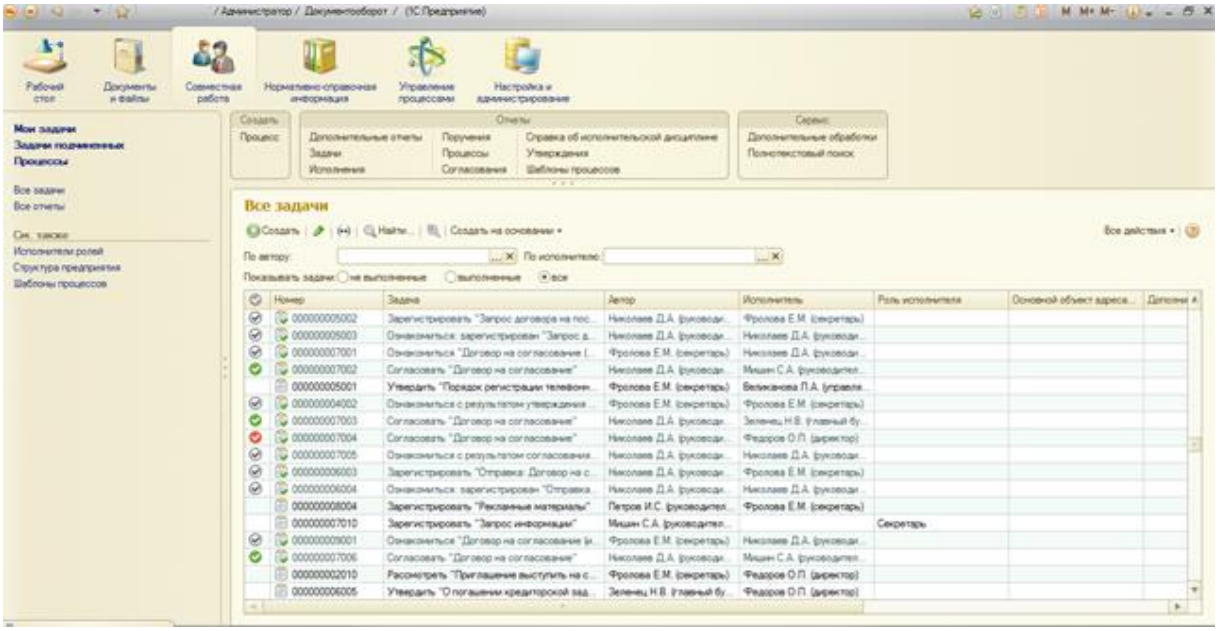


Рис. 2.55. Контроль исполнения процесса

Пример отчета Текущие бизнес-процессы:

Пример отчета Список текущих задач:

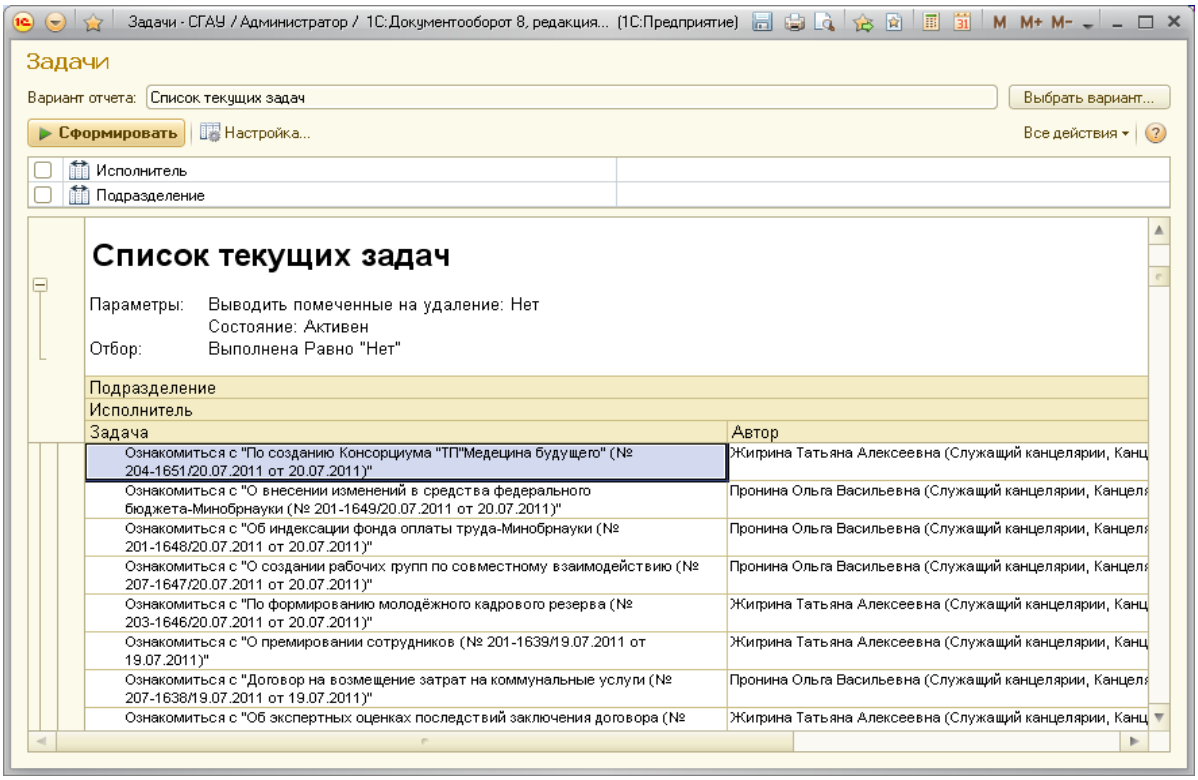


Рис. 2.56. Пример отчета Список текущих задач

Пример отчета Задачи, не принятые к исполнению:

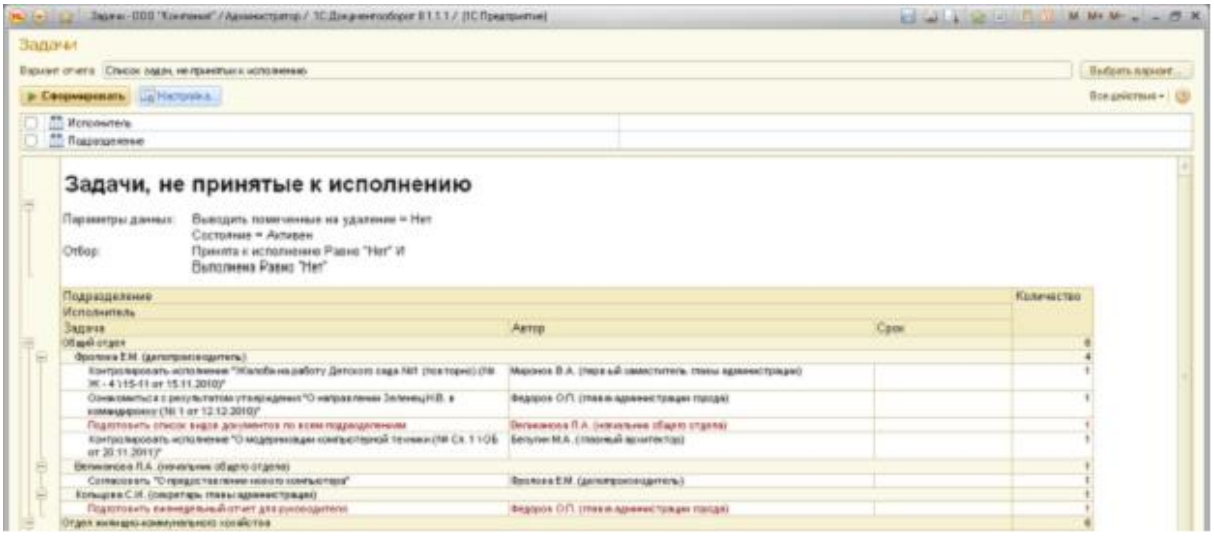


Рис.2.57. Пример отчета Задачи, не принятые к исполнению

3.4 Запуск задачи «Согласование»

Данный бизнес-процесс используется для согласования исходящих, внутренних документов и файлов с ответственными лицами.

Бизнес-процесс обходит всех участников согласования (рецензентов) по заранее составленному списку, после чего возвращается к автору (инициатору

бизнес-процесса). В случае отрицательного результата согласования автор имеет возможность повторить согласование.

При создании задачи Согласование выбираем нужный документ и нажимаем на кнопку панели задач Создать на основании

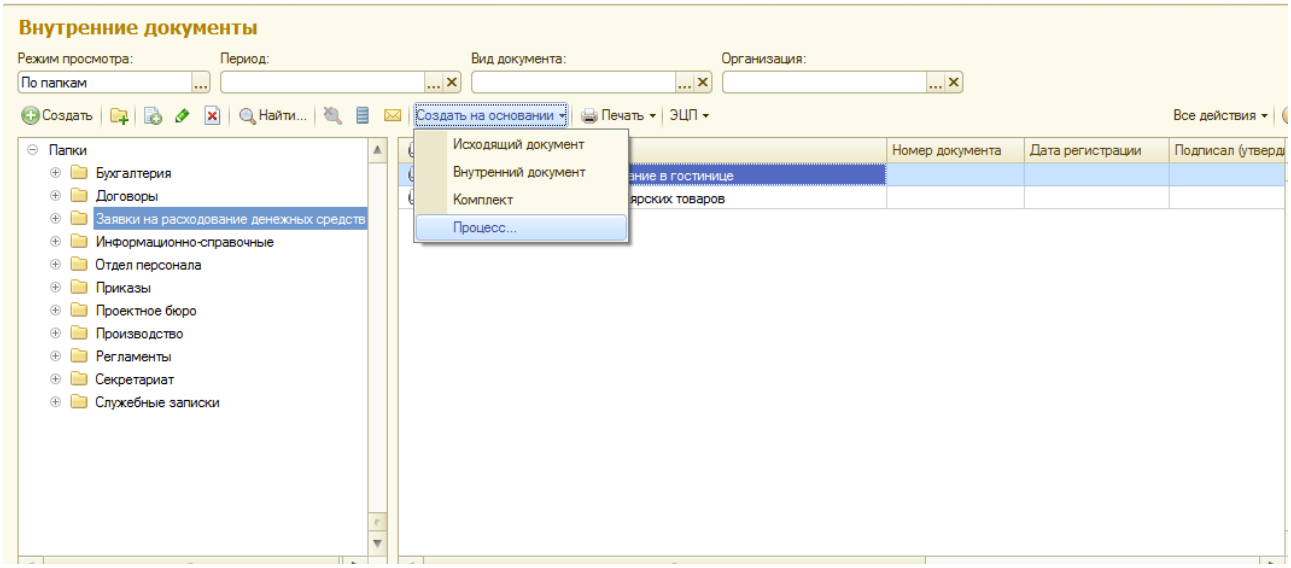


Рис. 2.58. Запуск задачи «Согласование»

В открывшейся карточке заполняем необходимые поля.

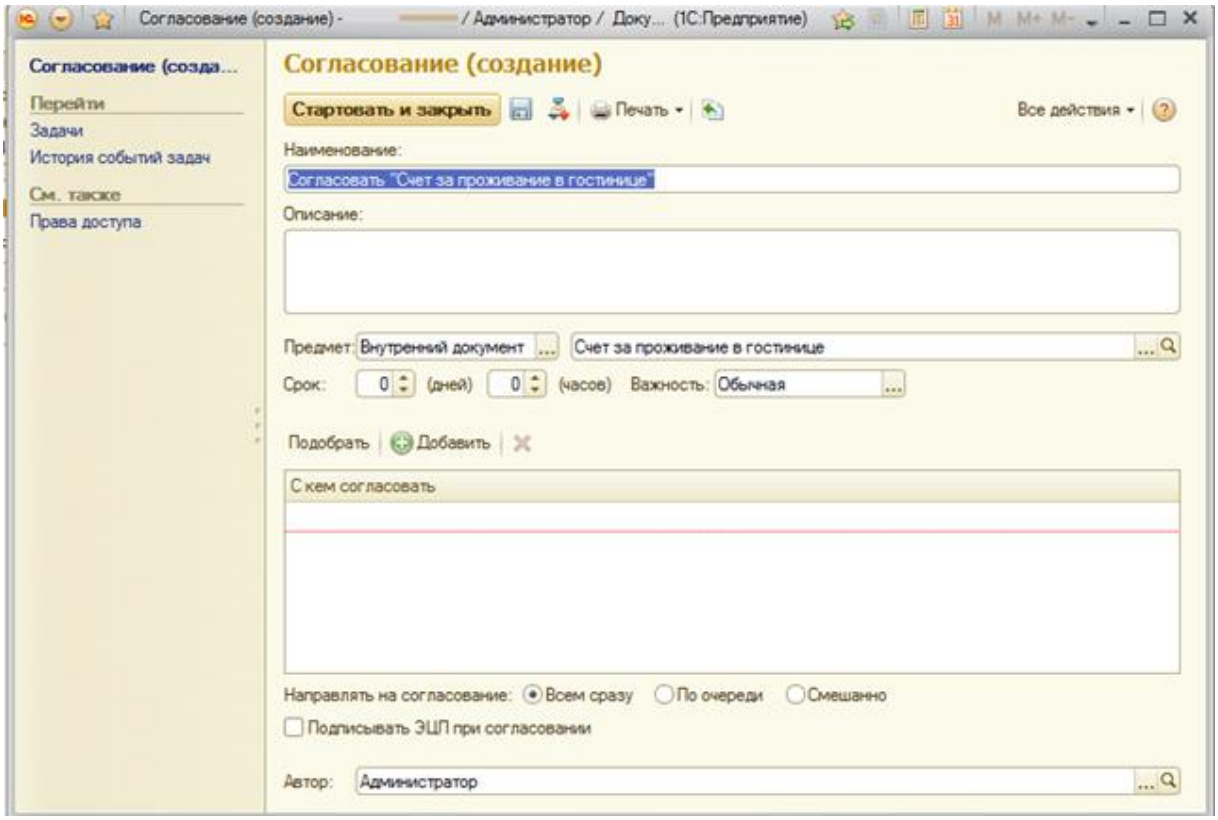


Рис. 2.59. Заполнение полей карточки

При заполнении поля «С кем согласовать» можно указывать как непосредственно исполнителя (его Ф.И.О.), так и роль (должность). Роль указывается в случае добавления руководителя в должности.

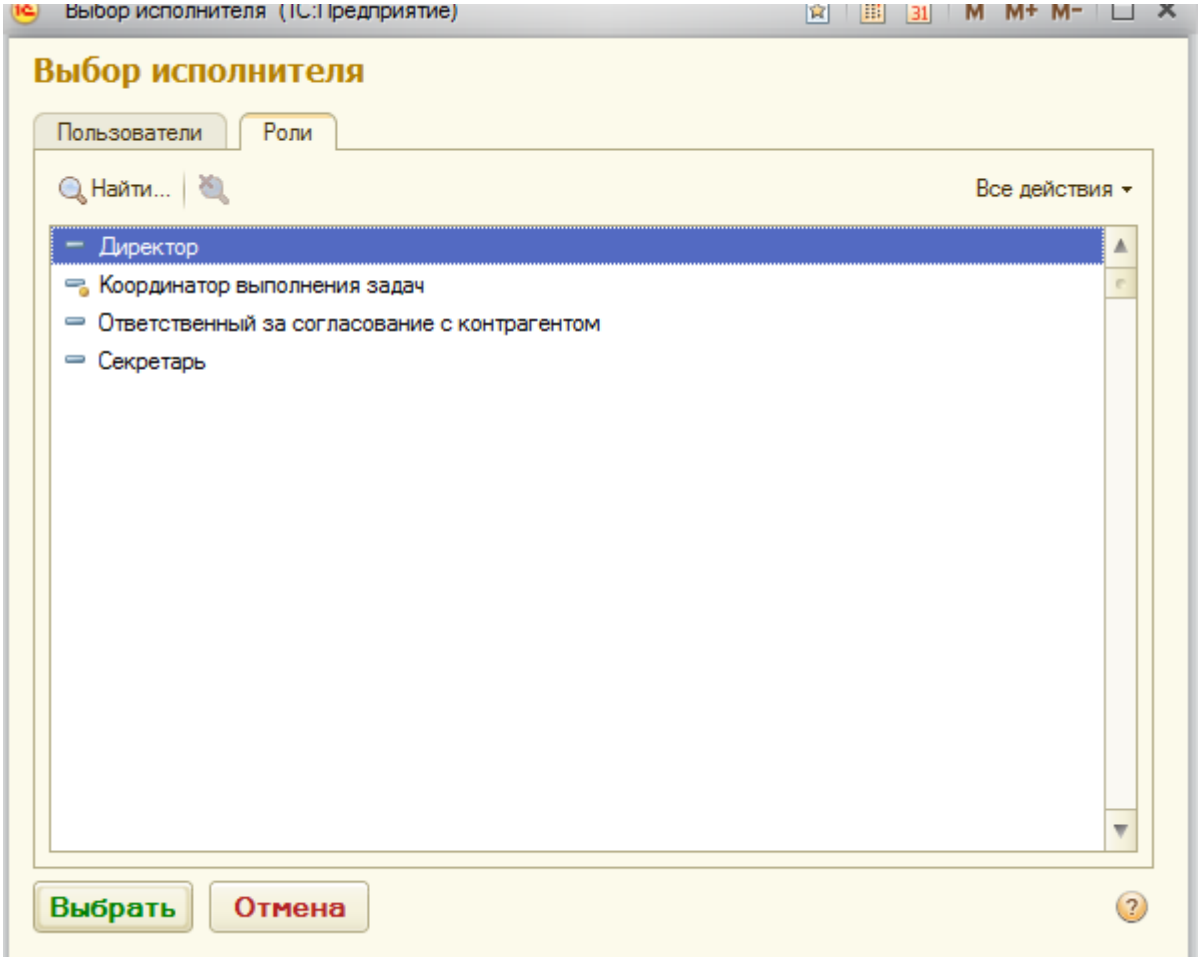


Рис.2.60. Заполнение полей карточки

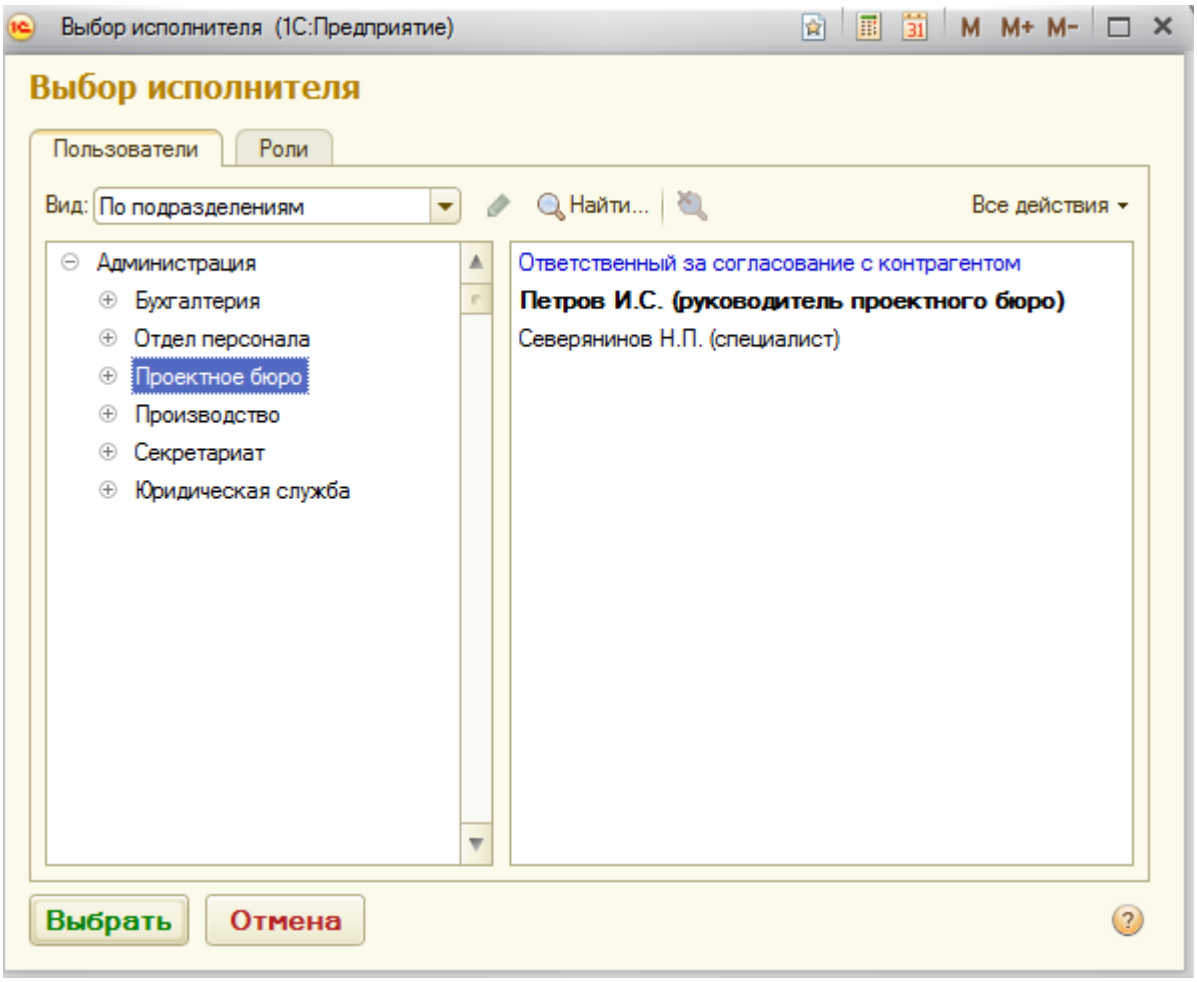


Рис. 2.61. Заполнение полей карточки

Предусмотрено три варианта обхода согласующих - «все вместе», «по очереди» и «смешанно». Если установлен тип согласования «все вместе», то задачи всем адресатам формируется одновременно. При этом бизнес-процесс возвращается к автору после завершения задач всеми согласующими. Если установлен тип согласования «по очереди», то задача следующему адресату формируется только после завершения задачи предыдущему. При этом бизнес-процесс сразу возвращается к автору, как только один из согласующих принял отрицательное решение. Если установлен тип согласования «смешанно», то для каждого исполнителя можно указать порядок формирования задачи - вместе с предыдущим либо после предыдущего. В результате тип согласования «смешанно» позволяет сочетать этапы одновременного и последовательного согласования.

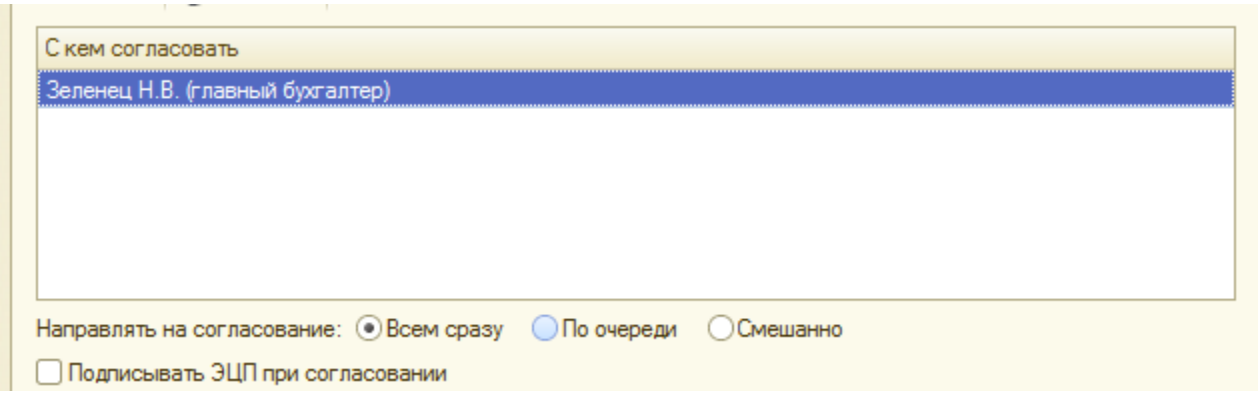


Рис. 2.62. Согласование

Для запуска бизнес-процесса нажимаем Стартовать и закрыть.

3.4.1. Завершение задачи «Согласование»

При поступлении адресату данной задачи есть три варианта ее завершения:

Согласовано, Согласовано с замечаниями, Не согласовано. При выборе Согласовано с замечаниями или Не согласовано обязательно нужно внести комментарий в соответствующее поле.

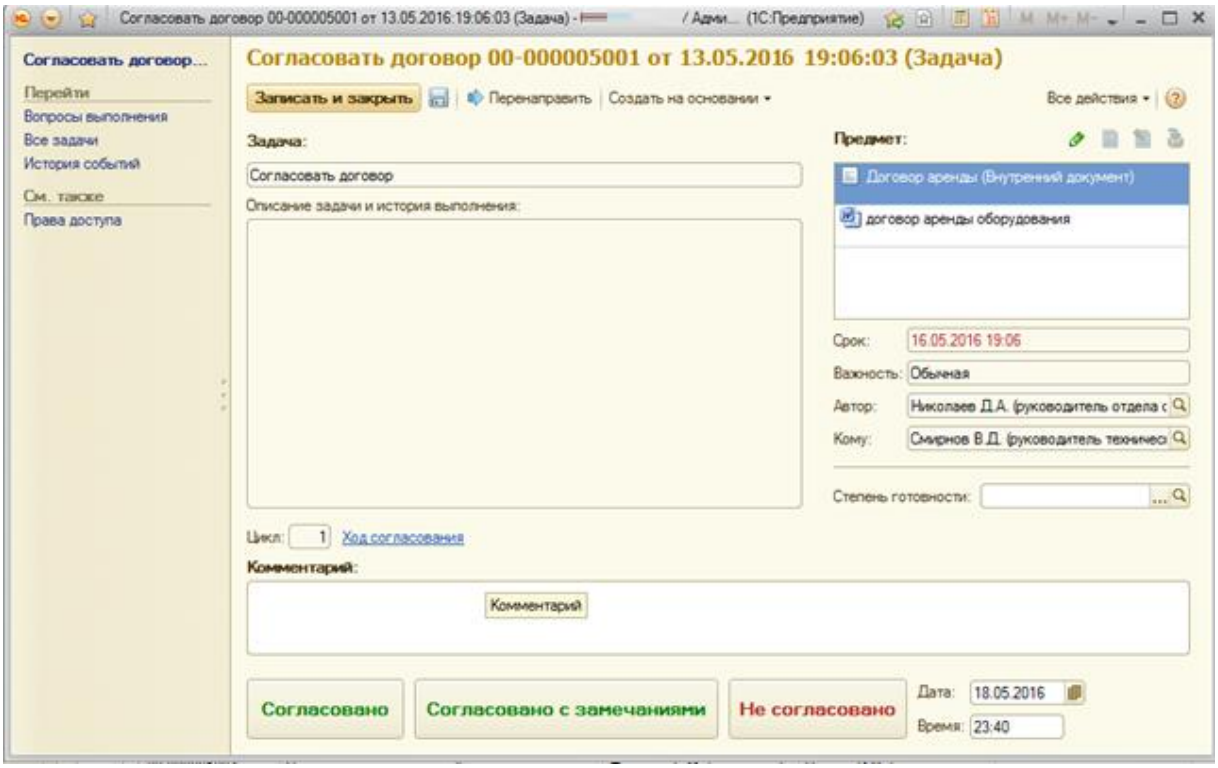


Рис. 2.63. Завершение задачи «Согласование»

После завершения всех задач автору бизнес процесса поступает задача Ознакомиться с результатом согласования. Если открыть карточку данной задачи, то можно ознакомиться с результатами согласования, сроками, внесенными комментариями.

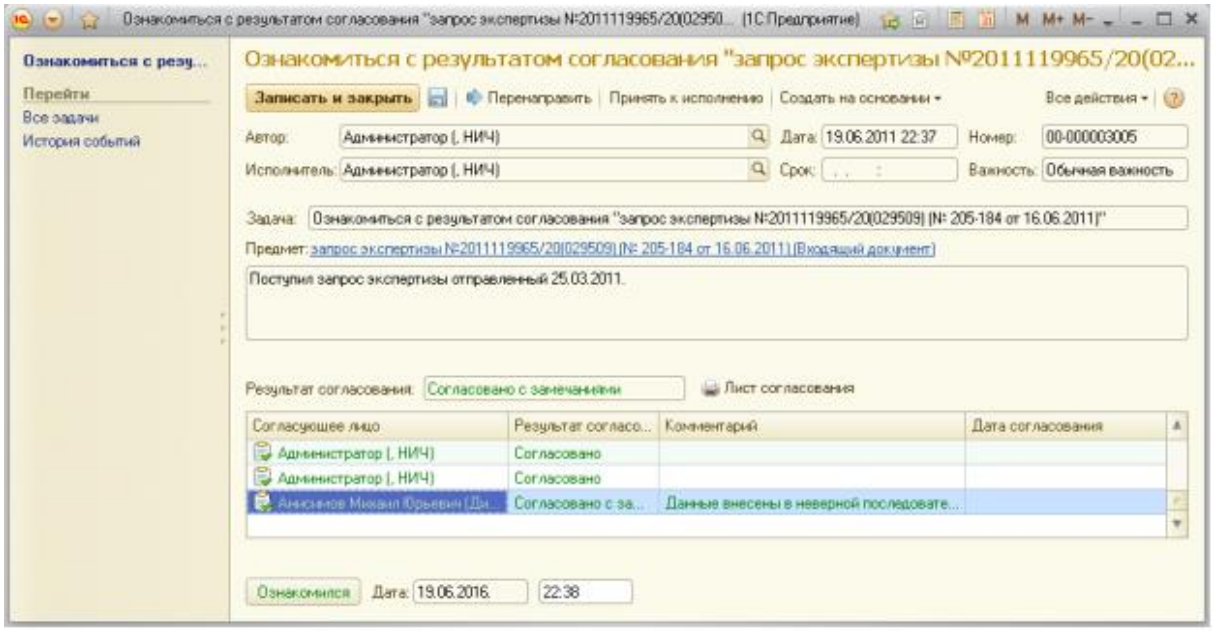


Рис. 2.64. Ознакомление с результатом согласования

Бизнес-процесс считается завершенным после того, как автор завершит задачу Ознакомиться с результатом согласования.

3.5 Запуск задачи «Рассмотрение»

Этот бизнес-процесс используется для автоматизации процесса рассмотрения документов руководителем. Его можно создавать непосредственно из карточки документа, при помощи меню «Создать на основании».

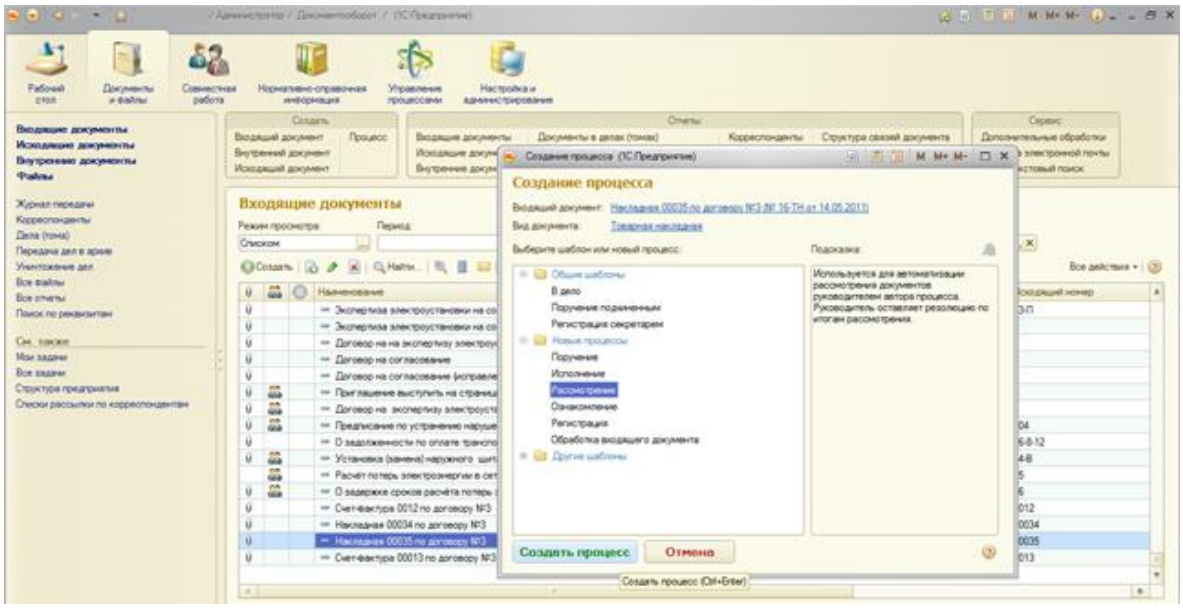


Рис. 2.65. Запуск задачи «Рассмотрение»

При выборе рассматривающего можно указать не только конкретного пользователя, но и роль/должность. Роль указывается в случае добавления руководителя в должности: директор, секретарь, начальник управления, а для всех остальных пользователей ставим Ф.И.О.

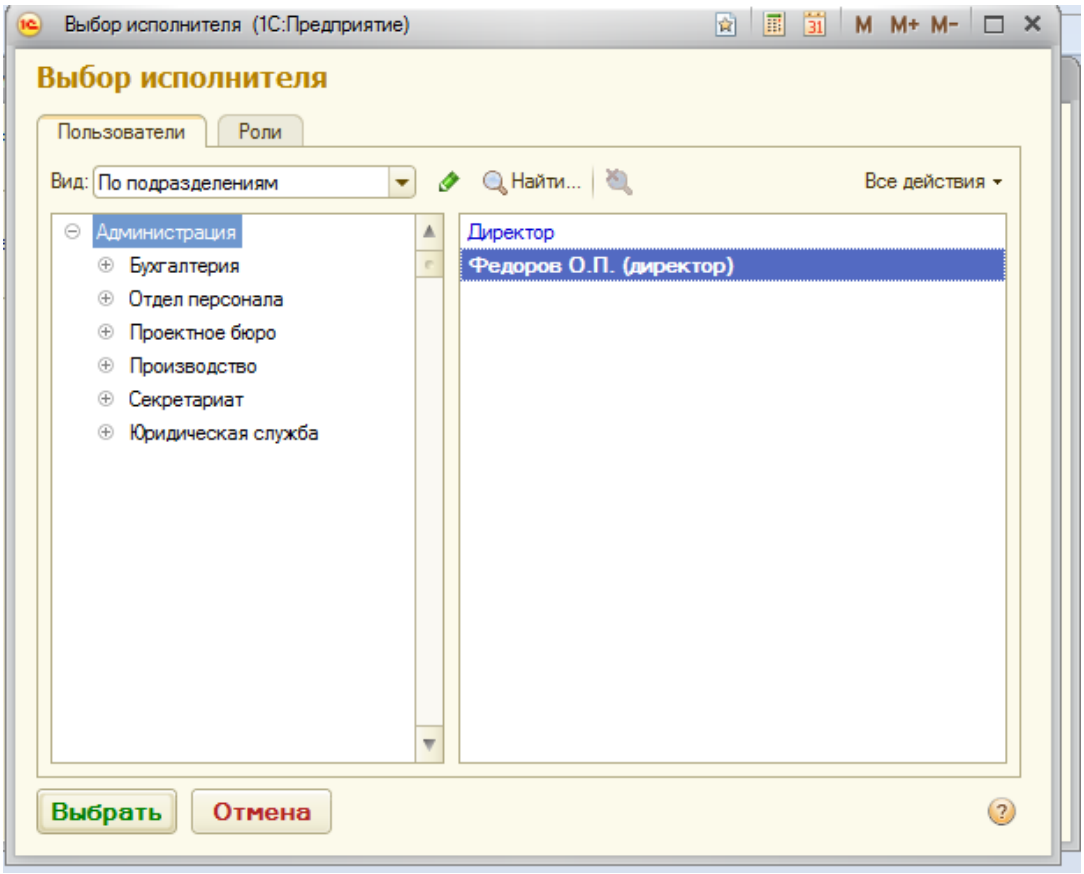


Рис. 2.66. Выбор рассматривающего

Запускается бизнес-процесс нажатием на кнопку Стартовать и закрыть.

3.5.1 Завершение задачи «Рассмотрение»

Сначала руководителю поступает задача Рассмотреть, содержащая ссылку на документ. На закладке Что рассмотреть показываются данные бизнес-процесса.

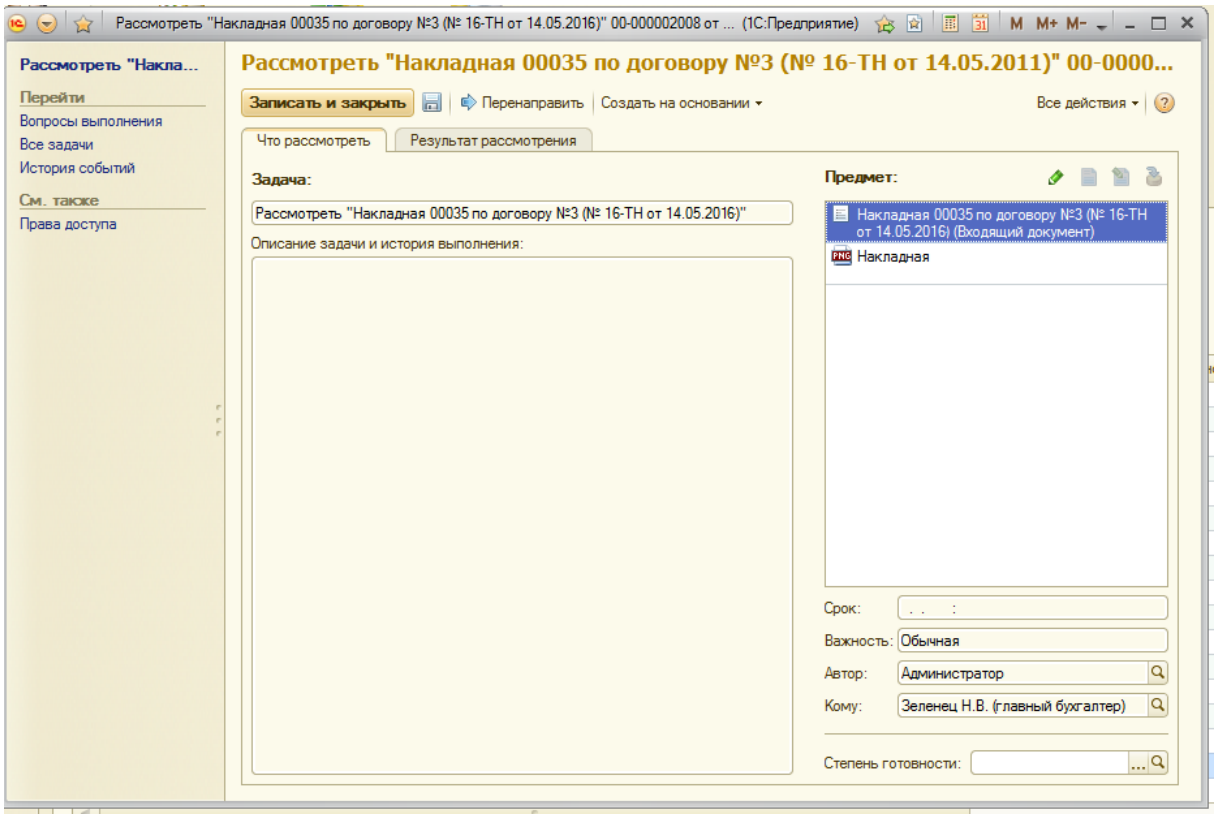


Рис. 2.67. Завершение задачи «Рассмотрение»

Возможны три варианта рассмотрения:

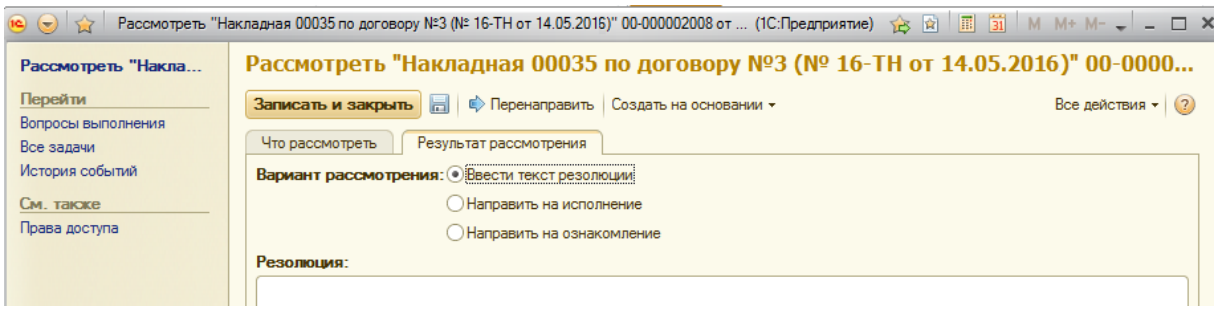


Рис.2.68. Три варианта рассмотрения

– Ввести текст резолюции - в поле Резолюция следует ввести текст резолюции (поле уже может содержать проект резолюции, определенный при создании бизнес-процесса);

– Направить на исполнение - формируется бизнес-процесс «Исполнение», открывается карточка бизнес-процесса «Исполнение».

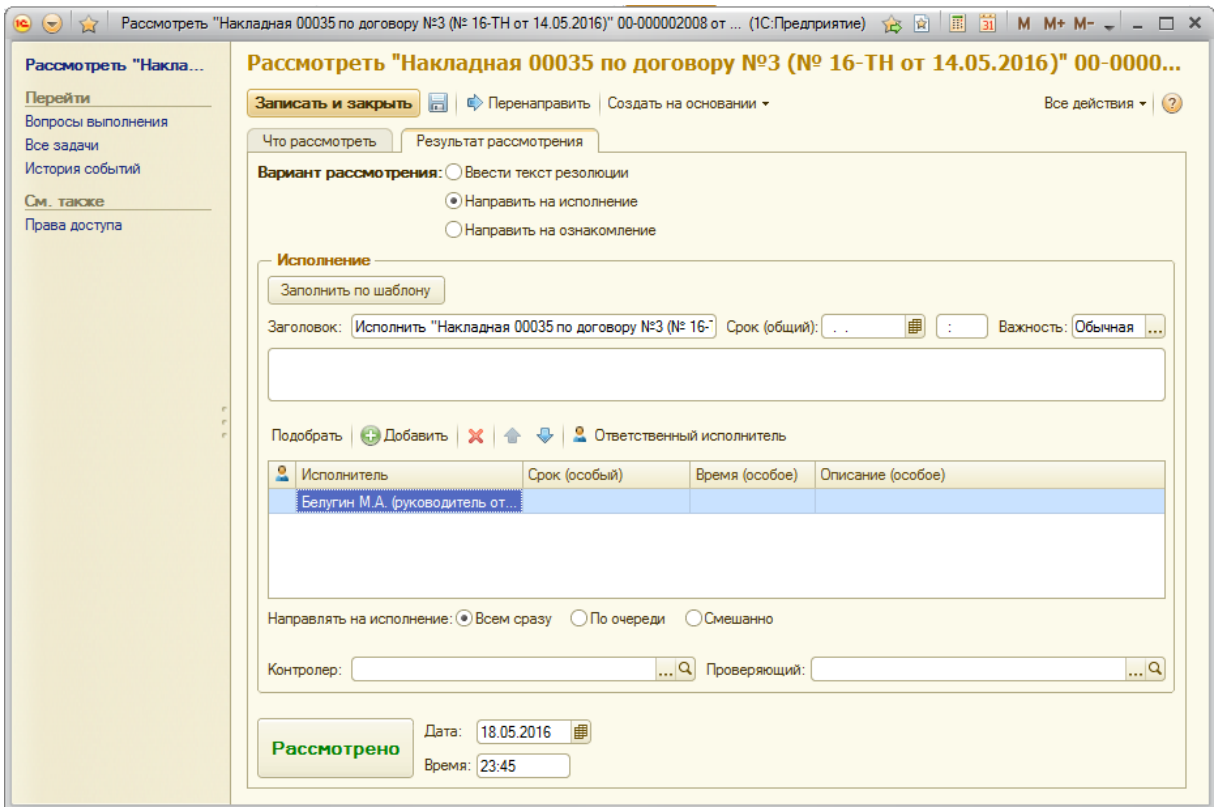


Рис. 2.69. Вариант исполнения

В поле Заголовок указывается основная цель исполнения (формируется автоматически), в поле Срок - дата исполнения, в поле Описание - подробное описание выполняемого действия, в списке исполнителей - указывается исполнитель или несколько исполнителей, в поле Контролер - пользователь, контролирующий исполнение (он получит задачу Контролировать одновременно с исполнителями), в поле Проверяющий - пользователь, проверяющий исполнение (он получит задачу после того, как все исполнители выполнили свои задачи).

Направить на ознакомление - формируется бизнес-процесс ознакомление

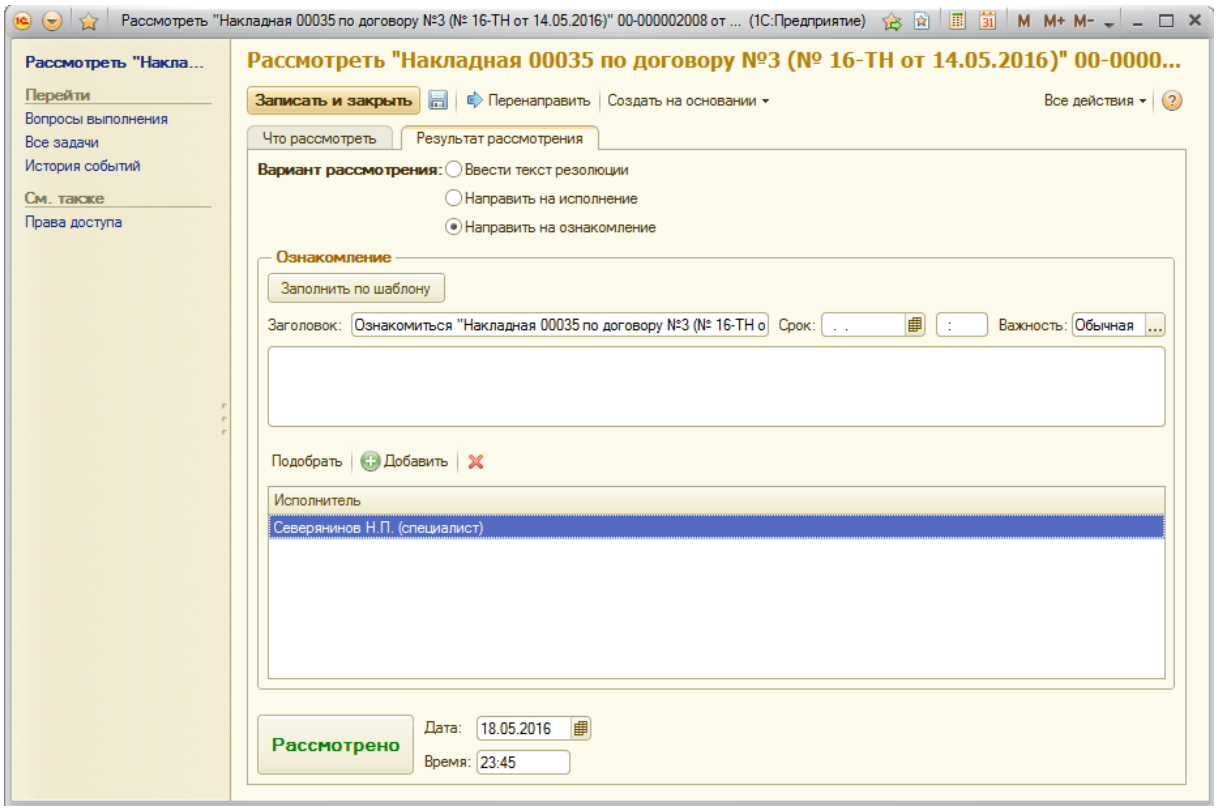


Рис. 2.70. Направление на ознакомление

В поле Заголовок указывается основная цель ознакомления(формируется автоматически), в поле Срок - дата исполнения, в поле Описание - подробное описание выполняемого действия, в списке исполнителей - указывается исполнитель или несколько исполнителей.

После заполнения информации о рассмотрении следует нажать кнопку - Рассмотрено.

3.6 Запуск задачи «Утверждение»

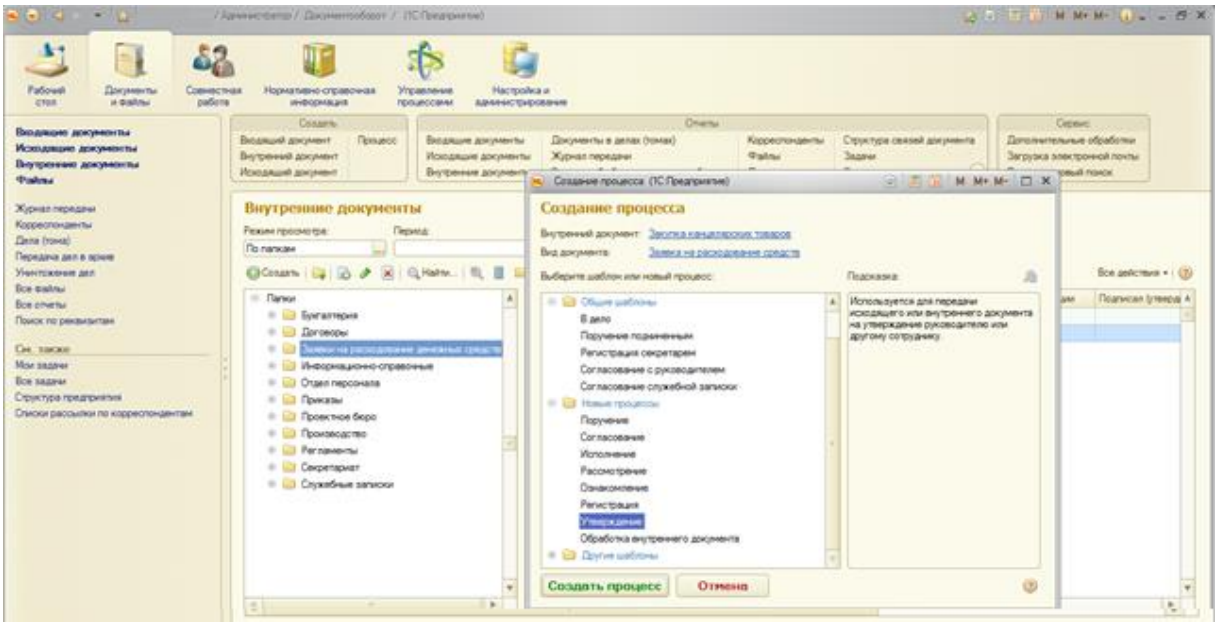


Рис. 2.71. Запуск задачи «Утверждение»

Этот бизнес-процесс используется для передачи исходящего или внутреннего документа на утверждение руководителю. Его можно создавать непосредственно из карточки документа, при помощи меню Создать на основании.

В открывшейся карточке заполняем необходимые поля.

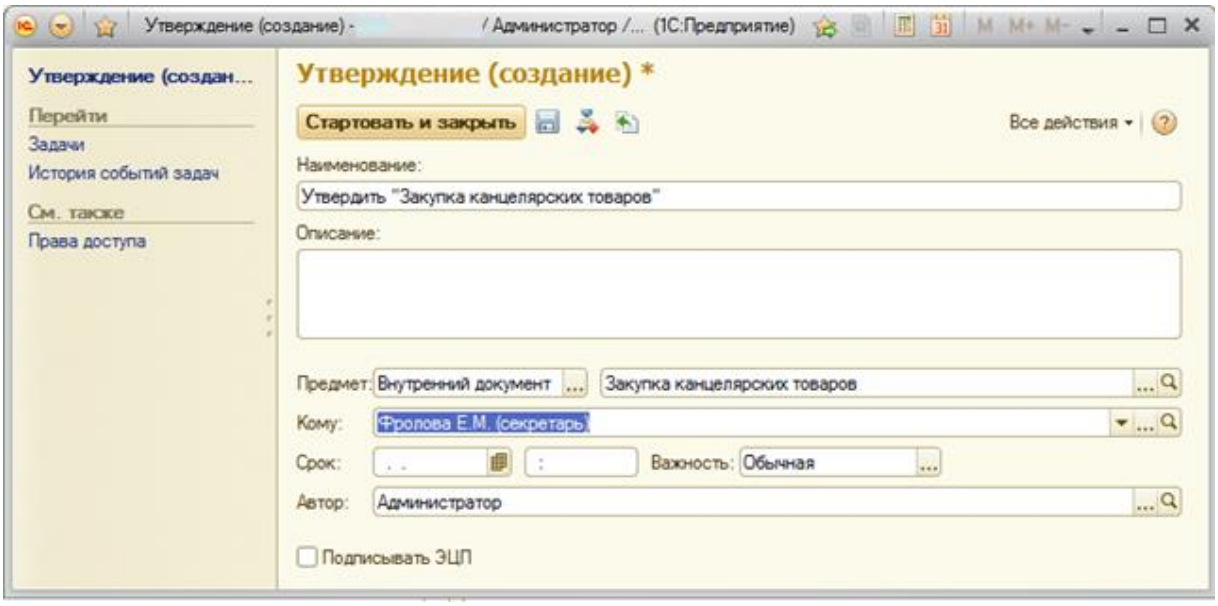


Рис. 2.72. Заполнение необходимых полей карточки

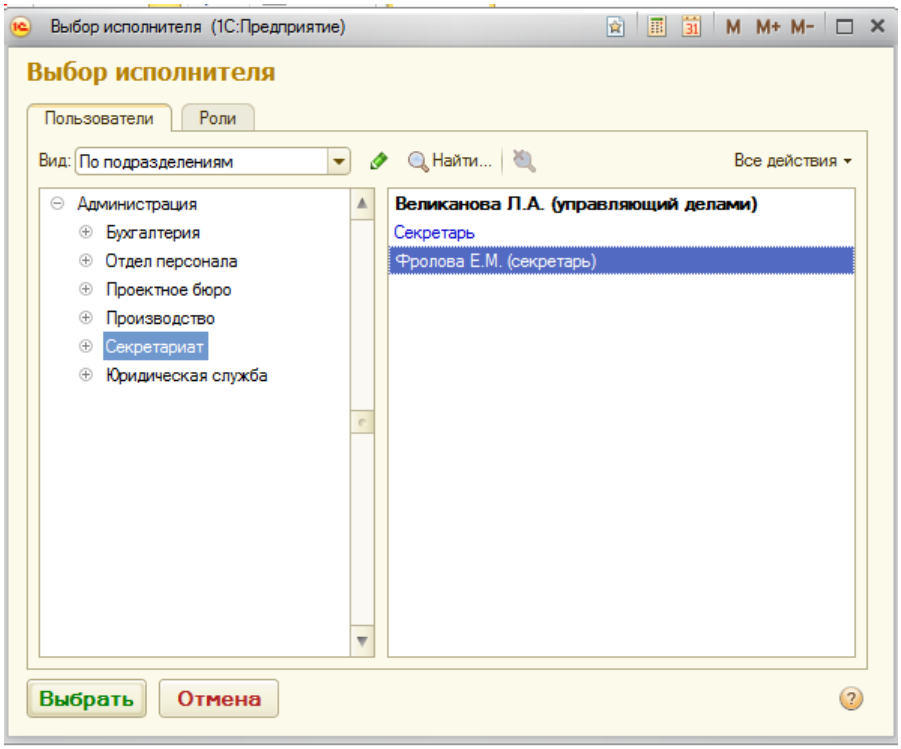


Рис. 2.73. Выбор исполнителя

При выборе рассматривающего можно указать не только конкретного пользователя, но и роль/должность. Роль указывается в случае добавления руководителя в должности: директор, секретарь, начальник управления, а для всех остальных пользователей ставим Ф.И.О.

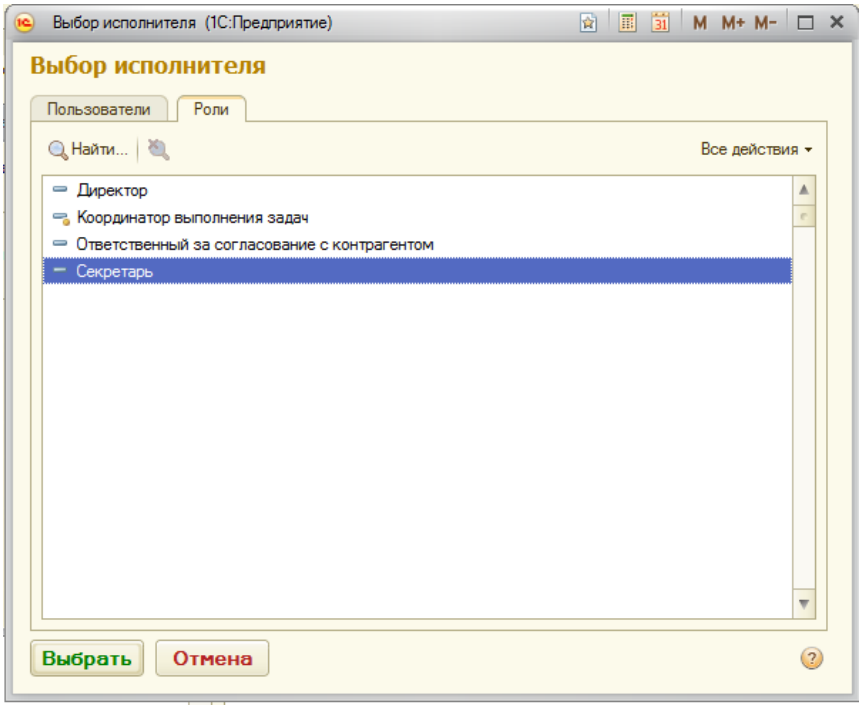


Рис 2.73. Выбор рассматривающего

Запускается бизнес-процесс нажатием на кнопку Стартовать и закрыть.

3.6.1 Завершение задачи «Утверждение»

После запуска процесса Утверждение, адресат, которому была направлена данная задача (утверждение) сможет увидеть ее на рабочем столе программы:

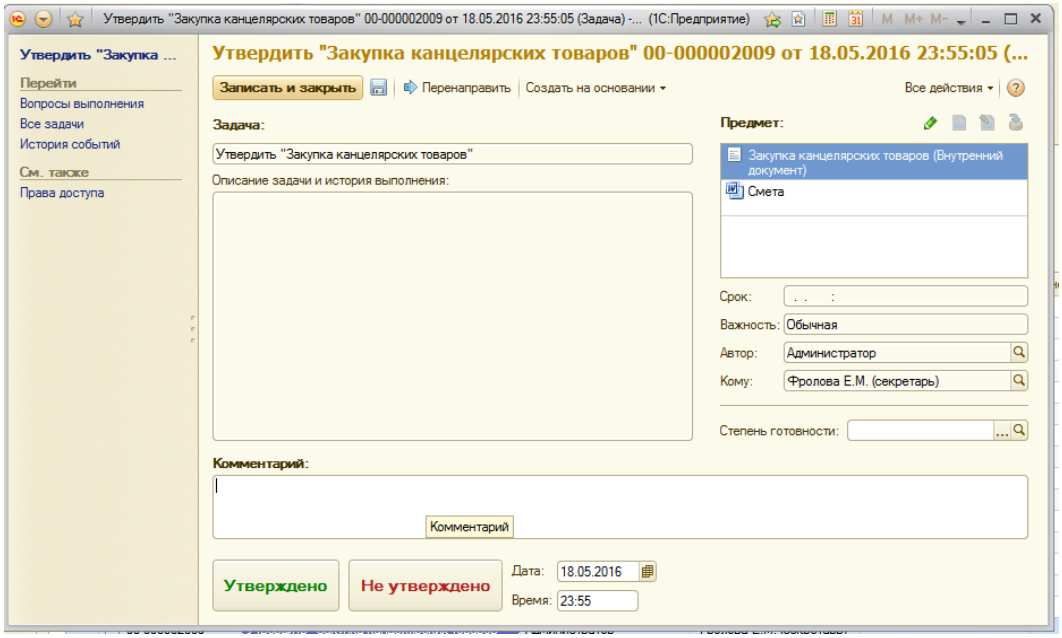


Рис. 2.74. Завершение задачи «Утверждение»

Возможны два варианта завершения данной задачи Утверждено, либо Не Утверждено. В поле Комментарий можно внести необходимые комментарии.

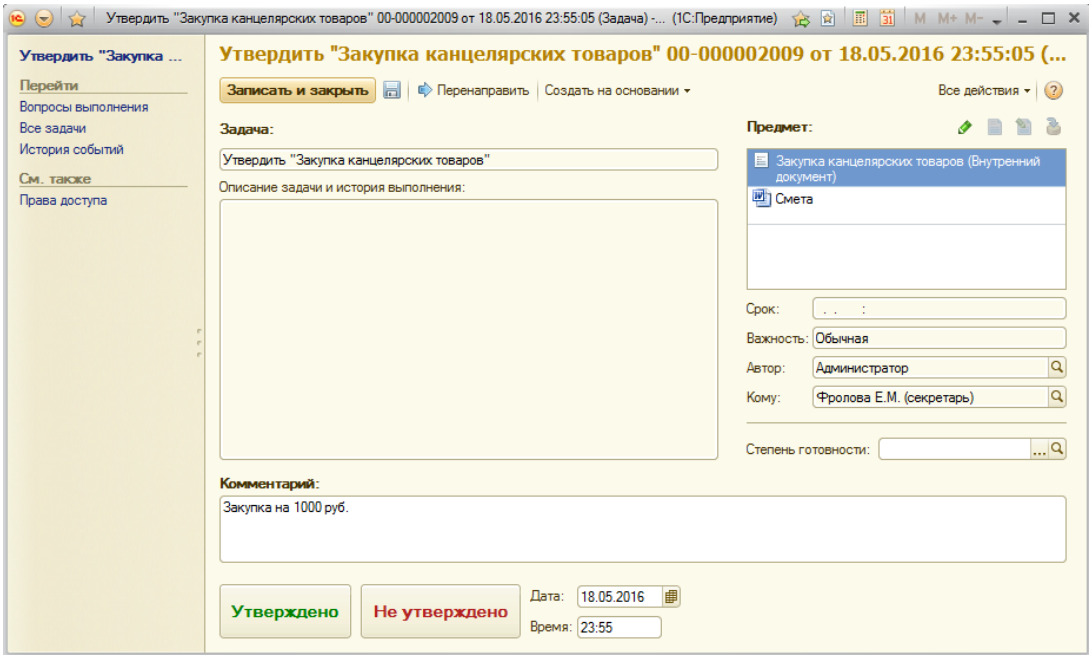


Рис. 2.75. Комментарии

После завершения всех задач автору бизнес процесса поступает задача Ознакомиться с результатом утверждения. Если открыть карточку данной задачи, можно ознакомиться с результатами утверждения, сроками, внесенными комментариями.

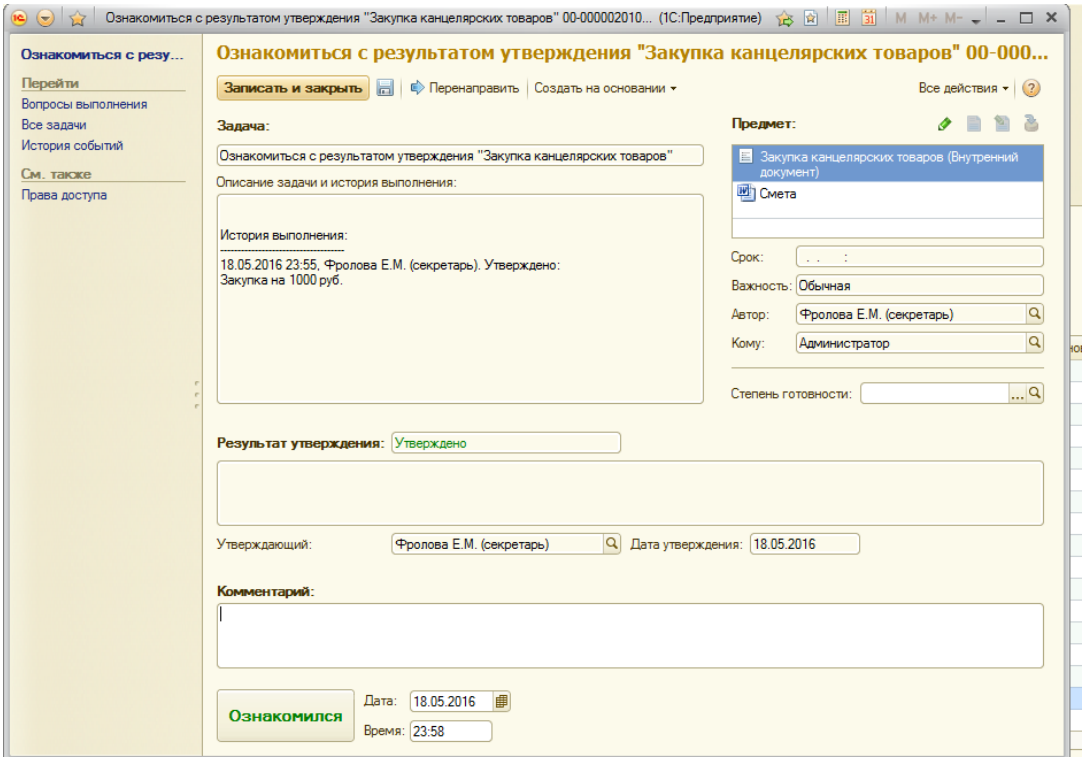


Рис. 2.76. Ознакомление с результатом утверждения

3.5.1 Раздел меню «Все задачи»

Этот раздел меню используется для просмотра информации о текущих и завершенных бизнес-процессах.

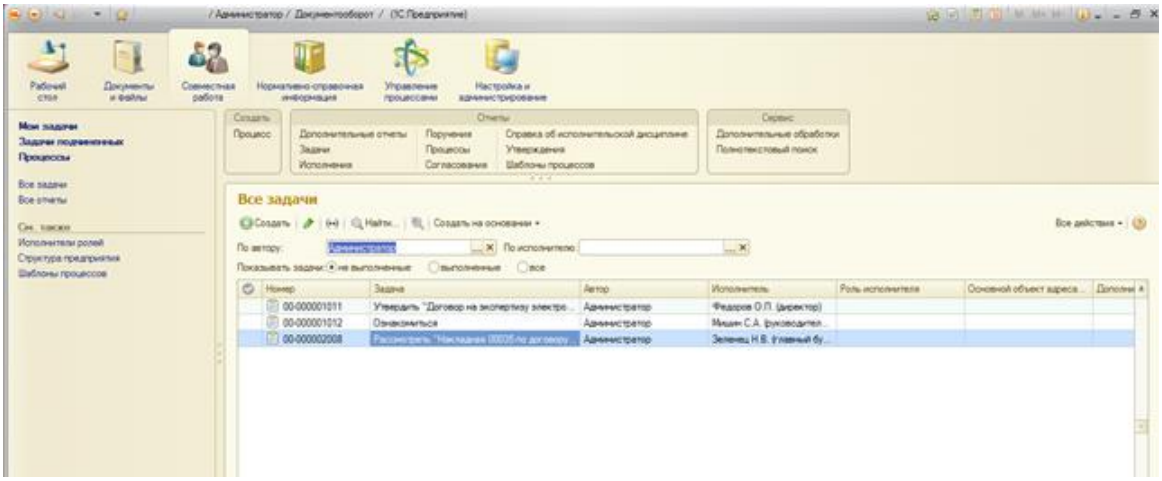


Рис. 2.77. Раздел меню «Все задачи»

Для удобства просмотра можно установить отбор по автору, по исполнителю, просматривать только выполненные, невыполненные или все задачи,также указать интервал дат по дате записи задачи. Например, чтоб просмотреть список всех задач, которые завершил/выполнил пользователь за весь период работы в программе, нужно в поле По исполнителю добавить фамилию нужного пользователя, а в поле Показывать задачи отметить вариант Выполненные. Далее автоматически сформируется список выполненных задач выбранного пользователя:

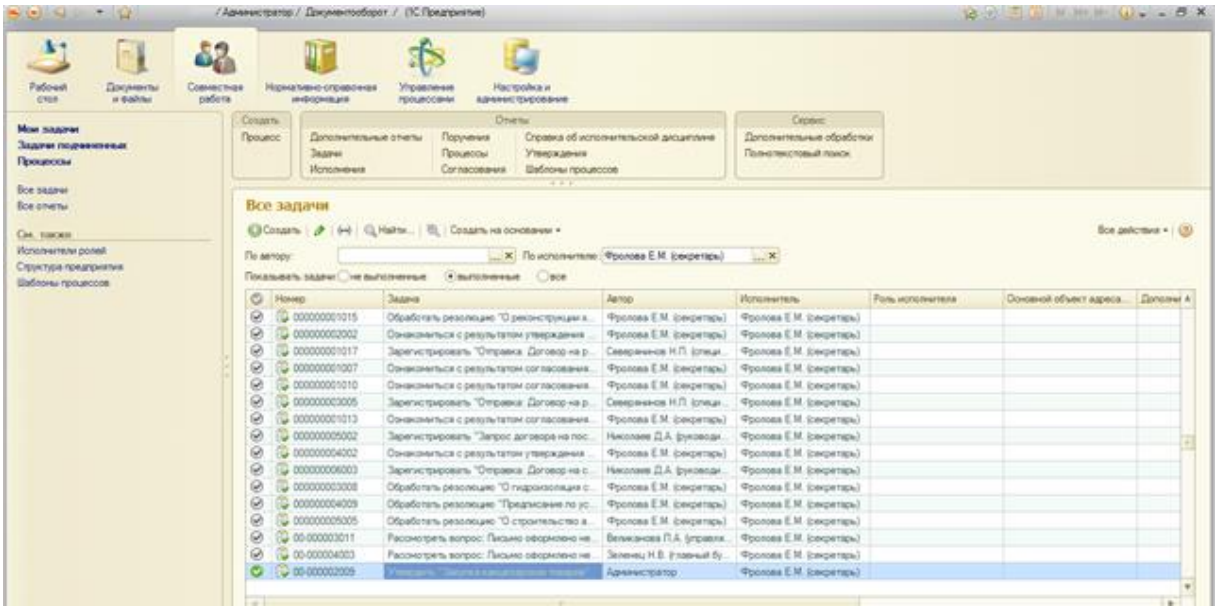


Рис. 2.78. Список выполненных задач выбранного пользователя

Очищаются поля фильтра нажатием на соответствующую кнопку:

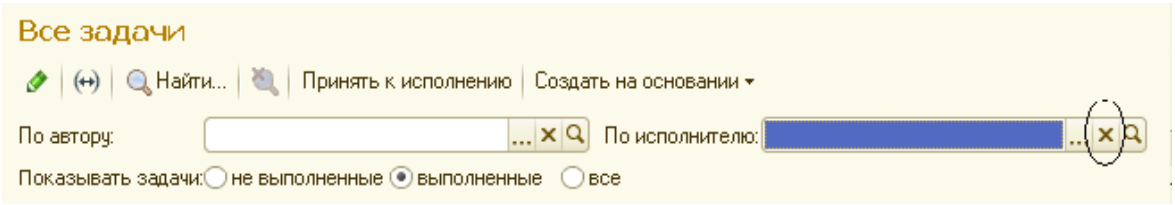


Рис. 2.79. Очистка полей фильтра

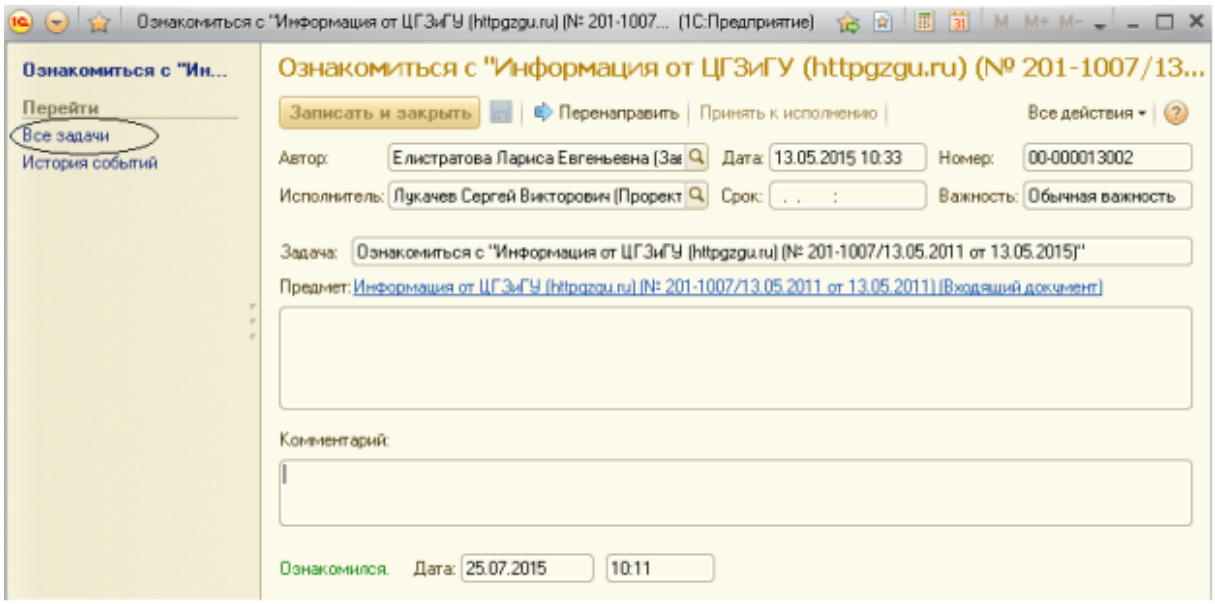


Рис. 2.80. Все задачи в карточке задачи

С помощью команды Все задачи в карточке задачи можно посмотреть список всех других задач по данному бизнес-процессу.

ГЛАВА 3.АПРОБАЦИЯ

При реализации проекта по внедрению автоматизированной информационной системы могут возникнуть сложности, не позволяющие гарантировать успешность функционирования системы. Главное правило внедрения автоматизированной информационной системы заключается в следующем: система должна внедряться повсеместно, в каждом подразделении, где осуществляется работа по редакции и хранению информации. В противном случае эффективность от внедрения системы будет минимальной. Основное препятствие для внедрения автоматизированной информационной системы заключается в том, что многие сотрудники сопротивляются применению нововведений. Настороженное отношение персонала к инновациям можно объяснить тем, что сотрудники не готовы обучаться работе с новым комплексом, вероятно, они не обладают достаточным уровнем знаний. В итоге эта проблема осложняет процессы по внедрению автоматизированной системы. Особенно актуальна данная трудность для тех предприятий, где непосредственные решения по перемещению и обновлению персонала принимаются только руководителем. Одновременно использование автоматизированных информационных систем вызывает ряд вопросов, связанных с техническим, экономическим и организационным аспектом внедрения. Организационные факторы связаны с отсутствием мотивации у сотрудников, что мешает приступить к работе с новой системой, требует повышения уровня технических знаний для грамотного анализа потребностей клиента. Экономические проблемы внедрения автоматизированной информационной системы заключаются в том, что в систему необходимо вложить много средств, инвестиции не могут окупиться сразу. Технические проблемы внедрения автоматизированной информационной системы — это необходимость создания или совершенствования инфраструктуры с целью

обеспечения интеграции новой системы с уже внедренными технологическими решениями. Попробуем разобраться с обозначенными проблемами глубже. Ключевыми при организации внедрения автоматизированной информационной системы являются организационные вопросы. Слабая проработка данных вопросов влечет за собой не результативность работы системы. Организация автоматизированного рабочего места, по мнению некоторых руководителей учреждения, не главная задача при внедрении автоматизированной информационной системы, однако именно эти «мелочи» становятся решающими в деле повышения эффективности процессов организации. Недостаточный интерес руководства в проекте может послужить причиной того, что внедрение может затянуться на очень продолжительный период. С целью «административного ресурса» поддержки автоматизированной информационной системы необходимо показать все преимущества, которые даёт новая система, и обучать руководство практической работе с ней.

Пристального внимания заслуживает организационный аспект, а именно разработка и утверждение плана внедрения автоматизированной информационной системы, выбор руководителя проекта и формирование рабочей группы. Определение конечных целей проекта — основное решение организационных трудностей. После того, как цели установлены, на первый план выходит формирование рабочей группы проекта. Сотрудники организации и специалисты ИТ-службы должны тесно сотрудничать для реализации своих целей и для достижения успеха. Кроме того, следует четко распределить прямые обязанности и сферы ответственности среди работников всех отделов, привлеченных к внедрению. Еще несколько трудностей могут возникнуть в процессе внедрения автоматизированной информационной системы. Среди них недостаточная проработанность внутрикорпоративных документов, регламентирующих процессы работы с автоматизированной информационной системы в организации и слабое внимание со стороны руководства к вопросам обучения конечных пользователей. Подготовка нормативных правил по процессам автоматизированной информационной системы должна вестись

параллельно с опытной эксплуатацией и предполагает разработку подробных инструкций по внесению данных в систему для сотрудников, соответствующих приказов и положений. Вопросам обучения сотрудников также должно уделяться самое пристальное внимание при внедрении проекта информатизации. При этом обучение должно осуществляться на постоянной основе и на всех этапах внедрения. Необходимо четкое понимание со стороны Правительства Ивановкой области, что создание автоматизированной информационной системы достаточно длительный проект, и ее развитие и поддержка будет требовать постоянного вложения средств. На обучении персонала часто экономят, считая, что достаточно обучить одного человека, а он обучит всех остальных. По мнению специалистов, достаточно сложно обучить даже 50 % сотрудников без отрыва от производства, однако выходом из сложившейся ситуации будет организация дистанционного обучения непосредственно на рабочих местах. Некоторые проблемы, возникающие при внедрении системы, достаточно хорошо изучены, формализованы и имеют эффективные методологии решения. Заблаговременное изучение этих проблем и подготовка к ним значительно облегчают процесс внедрения и повышают эффективность дальнейшего использования системы. Далее приведены основные проблемы и задачи, возникающие в большинстве случаев при внедрении информационных систем и рекомендации по их решению. Основные проблемы и задачи, требующие особого внимания при их решении: Отсутствие постановки задачи менеджмента в организации; Необходимость в частичной или полной реорганизации структуры организации; Сопротивление сотрудников организации; Временное увеличение нагрузки на сотрудников во время внедрения информационной системы; Необходимость в формировании квалифицированной группы внедрения и сопровождения системы, выбор сильного руководителя группы.

Теперь опишем эти пункты подробнее: Отсутствие постановки задачи менеджмента в организации. Этот пункт является наиболее значимым и сложным. На первый взгляд, его тема перекликается с содержанием второго

пункта, посвященного реорганизации структуры организации. Однако, на самом деле, он является более глобальным и включает в себя не только методологии управления, но также философские и психологические аспекты. Дело в том, что большинство руководителей управляют только исходя из своего опыта, своей интуиции, своего видения и весьма неструктурированных данных о его состоянии и динамике.

Как правило, если руководителя попросить описать в каком-либо виде структуру деятельности своей организации или набор положений, исходя из которых, он принимает управленческие решения, дело достаточно быстро заходит в тупик. Грамотная постановка задач менеджмента является важнейшим фактором, влияющим, как и на успех деятельности организации в целом, так и на успех проекта автоматизации. К сожалению, на настоящий момент в России до конца не сложился национальный подход к менеджменту, и в данный момент российское управление представляет собой гремучую смесь из теории западного менеджмента (которая во многом не является адекватной существующей ситуации) и советско-российского опыта, который, хотя и во многом гармонирует с общими жизненными принципами, но уже не отвечает жестким требованиям рыночной конкуренции.

Поэтому, первое, что необходимо сделать для того, чтобы проект внедрения информационной системы оказался удачным — максимально формализовать все те контуры управления, которые собственно планируется автоматизировать.

В большинстве случаев, для осуществления этого не обойтись без привлечения профессиональных консультантов, но по опыту, затраты на консультантов просто не сопоставимы с убытками от проваленного проекта автоматизации. Необходимость, в частичной реорганизация структуры и деятельности организации при внедрении информационной системы управления предприятием. Прежде чем приступать к внедрению информационной системы, обычно необходимо произвести частичную реорганизацию его структуры и технологий. Поэтому, одним из важнейших

этапов проекта внедрения, является полное и достоверное обследование организации во всех аспектах его деятельности. На основе заключения, полученного в результате обследования, строится вся дальнейшая схема построения информационной системы. Несомненно, можно автоматизировать все, по принципу «как есть», однако, этого не следует делать по ряду причин. Дело в том, что в результате обследования обычно фиксируется большое количество мест возникновения необоснованных дополнительных затрат, а также противоречий в организационной структуре, устранение которых позволило бы уменьшить производственные и логистические издержки, а также существенно сократить время исполнения различных этапов основных процессов. Под термином реорганизация мы даже не имеем в виду реинжиниринг в его классическом западном понимании, с полной перестройкой всей внутрихозяйственной и коммерческой деятельности. Реорганизация может быть проведена в ряде локальных точек, где она объективно необходима, что не повлечет за собой ощутимый спад активности текущей деятельности.

Сопротивление сотрудников организации. – При внедрении информационной системы в большинстве случаев возникает активное сопротивление сотрудников на местах, которое вполне способно сорвать, или существенно затянуть проект внедрения. Временное увеличение нагрузки на сотрудников, при внедрении системы управления предприятием. – На некоторых этапах проекта внедрения временно возрастает нагрузка на сотрудников организации. Этосвязано с тем, что помимо выполнения обычных рабочих обязанностей, сотрудникам необходимо осваивать новые знания и технологии [5]. Формирование квалифицированной группы внедрения и сопровождения системы, руководителя группы. Внедрение большинства крупных систем автоматизации производится по следующей технологии: на предприятии формируется небольшая рабочая группа или назначается ответственный сотрудник за использование, которая проходит максимально полное обучение работе с системой, затем на эту группу ложится значительная часть работы по внедрению системы и дальнейшему ее сопровождению.

Несмотря на проблемы внедрения, упомянутые выше, эффект от автоматизированной информационной системы измеряется повышением качества работы организации. Следование, этим рекомендациям позволит снизить риски, возникающие при внедрении информационных систем и увеличить эффективность деятельности организации.

ЗАКЛЮЧЕНИЕ

Дипломный проект ставил целью разработать автоматизированную систему документооборота для администрации сельского поселения с.Великомихайловка.

Созданная система автоматизирует следующие функции:

- ведение различной справочной информации;
- ведение оперативного учета;
- вывод информации о документации.

Данный дипломный проект состоит из трёх частей.

В аналитической части выполнен комплекс работ, которые направлены на обоснование необходимости автоматизации: определена сущность задачи, описаны основные свойства системы, дано описание всем существующим бизнес—процессам, рассмотрены вопросы, которые связаны с анализом существующих разработок в этой области. В результате проведенной работы выбрана стратегия автоматизации по участкам. Как наилучший вариант принято решение о самостоятельной разработке информационной системы, так как компания располагает квалифицированными сотрудниками для разработки и внедрения данной информационной системы, кроме того, только в данном случае гарантируется соответствие программного продукта бизнес—процессам, происходящим на предприятии.

Проектная часть посвящена рассмотрению этапов жизненного цикла проекта. Также обосновываются проектные решения по информационному, техническому, программному обеспечению.

Далее дана характеристика информационной архитектуры разрабатываемого проекта, построена информационная модель задачи, проведено моделирование «сущность—связь» (ER—модель), описана структура полей таблиц базы данных, проанализированы все информационные

потоки входной, оперативной, нормативно - справочной и результатной информации.

В процессе реализации проектных решений по программному обеспечению построены: дерево диалога (сценарий работы с системой), структурная схема пакета и ряд других компонент проекта, подробно раскрывающих сущность машинной реализации задачи.

Третья часть дипломного проекта посвящена апробации автоматизированной информационной системы.

Разработанная информационная система подлежит интеграции с уже существующими на предприятии программными комплексами, кроме того, несмотря на то, что разрабатывалась для конкретнойадминистрации, может быть использована и на других, аналогичных предприятиях за счет своей универсальности с минимальными изменениями.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. ГОСТ Р ИСО/МЭК 12207/99. Государственный стандарт РФ. Информационная технология. Процессы жизненного цикла информационных систем. Издание официальное. - М., 1999

2. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. ГОСТ 19.701-90 (ИСО 5807-85) / Государственный комитет СССР по управлению качеством продукции и стандартам, 01.01.1992.

3. Автоматизированные информационные системы, базы и банки данных. Вводный курс: Учебное пособие, М.: Гелиос АРВ, 2007. - 368 с., ил

4. Астелс, Дэвид; Миллер Гранвилл; Новак, Мирослав, Практическое руководство по экстремальному программированию, Пер. с англ. - М.: Издательский дом "Вильямс", 2008. - 320 с.: ил. - Парал. тит. англ

5. Баженова И. Ю. , Основы проектирования приложений баз данных, Издательства: Бином. Лаборатория знаний, Интернет-университет информационных технологий, 2008 г., , 328 стр.

6. Вендров А.М., CASE-технологии. Современные методы и средства проектирования информационных систем - М.: Финансы и статистика, 2007 г, 456 стр.

7. Вигерс Карл, Разработка требований к программному обеспечению, Пер, с англ. - М.:Издательско-торговый дом "Русская Редакция", 2008. -576с.: ил

8. Гашков С. Б., Э. А. Применко, М. А. Черепнев Криптографические методы защиты информации, М, Издательство: Академия, 2010 г., 304 стр.

9. Гвоздева Т. В., Б. А. Баллод, Проектирование информационных систем, М, Издательство: Феникс, 2009 г., 512 стр.

10. Голицына О. Л., И. И. Попов, Н. В. Максимов, Т. Л. Партыка, Информационные технологии, М, Издательство Инфра-М, 2009 г., 608 стр.

11. Дейв Крейн, Эрик Паскарелло, Даррен Джеймс. AJAX в действии: Учебник – М.: Вильямс, 2007. 450 – 490 с.

12. Дэвид Флэнаган. JavaScript. Подробное руководство: Учебник – М.: Символ Плюс, 2008. 243 – 249 с.

13. Емельянова Н. З., Партыка Т. Л., И. И. Попов, Проектирование информационных систем, М, Издательство: Форум, 2009 г., 432 стр.

14. Емельянова Н. З., Т. Л. Партыка, И. И. Попов, М, Издательство Форум, 2007 г., , 416 стр.

15. Илюшечкин В. М. , Основы использования и проектирования баз данных, М, Издательство Юрайт, 2010 г., 224 стр.

16. Котляров В. П., Т. В. Коликова, Основы тестирования программного обеспечения, Издательства: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2009 г., 288 стр.

17. Кристиан Дари, Богдан Бринзаре, Филип Черчез-Тоза, Михай Бусика. AJAX и PHP. Разработка динамических веб-приложений: Учебник – М.: Символ Плюс, 2007, 289 стр.

18. Кузин А. В., С. В. Левонисова, Базы данных, М, Издательство: Академия, 2008 г., 320 стр.

19. Кузнецов С. Д., Основы баз данных, М, Издательства: Бином. Лаборатория знаний, Интернет-университет информационных технологий, 2007 г., 488 стр.

20. Молчанов А. Ю., Системное программное обеспечение, М, Издательство: Питер, 2010 г., 400 стр.

21. Незнанов А. А., Программирование и алгоритмизация, М, Издательство: Академия, 2010 г., 304 стр.

22. Пирогов В. Ю., Информационные системы и базы данных. М, Организация и проектирование, Издательство: БХВ-Петербург, 2009 г.528 стр.

23. Предметно-ориентированные экономические информационные системы, М, Издательство: Финансы и статистика, 2007 г., 224 стр.

24. Реляционные базы данных: практические приемы оптимальных решений. – СПб.: БХВ-Петербург, 2009 – 400с.:ил;

25. Симионов Ю.Ф., Боромотов В.В. Информационный менеджмент. — Ростов н.Д: Феникс, 2008, 250с., ил.;

26. Чипига А. Ф., Информационная безопасность автоматизированных систем, М, Издательство: Гелиос АРВ, 2010 г., 336 стр.

ПРИЛОЖЕНИЕ

Листинг программных модулей

```
////////////////////////////////////////////////////////////////////////////////
/
// МОДУЛЬ СОДЕРЖИТ ПРОЦЕДУРЫ И ФУНКЦИИ РАБОТЫ С ВНУТРЕННИМИ, ВХОДЯЩИМИ И ИСХОДЯЩИМИ ДОКУМЕНТАМИ
//
// Формирует наименование документа из заголовка
Функция НаименованиеДокумента(Документ) Экспорт
Заголовок = СокрЛП(Документ.Заголовок);
ДлинаНаименования =
Документ.Метаданные().ДлинаНаименования;
Если
ЗначениеЗаполнено(Документ.РегистрационныйНомер)
Тогда
Постфикс =
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСТР("ru = ' (№ %1 от %2)'"),
СокрЛП(Документ.РегистрационныйНомер),
Формат(Документ.ДатаРегистрации, "ДЛФ=D"));
Иначе
Постфикс = "";
КонецЕсли;
Если СтрДлина(Заголовок + Постфикс) >
ДлинаНаименования Тогда
Заголовок = Лев(Заголовок, ДлинаНаименования -
СтрДлина(Постфикс));
ДлинаЗаголовка = СтрДлина(Заголовок);
ПозицияПробела = ДлинаЗаголовка;
Пока ПозицияПробела > 0 Цикл
Если Сред(Заголовок, ПозицияПробела, 1) = " " Тогда
Прервать;
КонецЕсли;
ПозицияПробела = ПозицияПробела - 1;
КонецЦикла;
Если ПозицияПробела > 1 Тогда
Заголовок = Лев(Заголовок, ПозицияПробела - 1);
КонецЕсли;
КонецЕсли;
Возврат Заголовок + Постфикс;
КонецФункции
// Формирует наименование номенклатуры дел
Функция
НаименованиеНоменклатурыДел(НоменклатураДел)
Экспорт
Заголовок = НоменклатураДел.ПолноеНаименование;
Заголовок = СтрЗаменить(Заголовок, Символы.ПС, " ");
Заголовок = СокрЛП(Заголовок);
ДлинаНаименования =
НоменклатураДел.Метаданные().ДлинаНаименования;
Если СтрДлина(Заголовок) > ДлинаНаименования Тогда
Заголовок = Лев(Заголовок, ДлинаНаименования);
ДлинаЗаголовка = СтрДлина(Заголовок);
ПозицияПробела = ДлинаЗаголовка;
Пока ПозицияПробела > 0 Цикл
Если Сред(Заголовок, ПозицияПробела, 1) = " " Тогда
Прервать;
КонецЕсли;
ПозицияПробела = ПозицияПробела - 1;
КонецЦикла;
Если ПозицияПробела > 1 Тогда
Заголовок = Лев(Заголовок, ПозицияПробела - 1);
КонецЕсли;
КонецЕсли;
Возврат Заголовок;
КонецФункции
// Возвращает количество файлов по документу
Функция КоличествоФайлов(Документ) Экспорт
Если Не ЗначениеЗаполнено(Документ) Тогда
Возврат 0;
КонецЕсли;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|	КОЛИЧЕСТВО(*) КАК Количество
|ИЗ
|	Справочник.Файлы КАК Файлы
|ГДЕ
|	Файлы.ВладелецФайла = &Документ";
Запрос.УстановитьПараметр("Документ", Документ);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат 0;
КонецЕсли;
Выборка = Результат.Выбрать();
Выборка.Следующий();
Возврат Выборка.Количество;
КонецФункции
// Возвращает количество задач по документу
Функция КоличествоЗадач(Документ,
ТолькоНевыполненные = Ложь) Экспорт
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|	КОЛИЧЕСТВО(*) КАК Количество
|ИЗ
|	Задача.ЗадачаИсполнителя КАК
ЗадачаИсполнителя
|ГДЕ
|	ЗадачаИсполнителя.Предмет = &Документ
|	И (НЕ
ЗадачаИсполнителя.ПометкаУдаления)";
Если ТолькоНевыполненные Тогда
Запрос.Текст = Запрос.Текст + " И (НЕ
ЗадачаИсполнителя.Выполнена)";
КонецЕсли;
Запрос.УстановитьПараметр("Документ", Документ);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат 0;
КонецЕсли;
Выборка = Результат.Выбрать();
Выборка.Следующий();
Возврат Выборка.Количество;
КонецФункции
// Возвращает вид документа по умолчанию
Функция ПолучитьВидДокументаПоУмолчанию(Ссылка)
Экспорт
ВидДокумента = Неопределено;
Если ТипЗнч(Ссылка) =
Тип("СправочникСсылка.ВходящиеДокументы") Тогда
ВидДокумента =
ХранилищеОбщихНастроек.Загрузить("НастройкиРаботыСДокументами", "ВидВходящегоДокумента");
ТипМетаданных = "ВидыВходящихДокументов";
ИначеЕсли ТипЗнч(Ссылка) =
Тип("СправочникСсылка.ИсходящиеДокументы") Тогда
ВидДокумента =
ХранилищеОбщихНастроек.Загрузить("НастройкиРаботыСДокументами", "ВидИсходящегоДокумента");
ТипМетаданных = "ВидыИсходящихДокументов";
ИначеЕсли ТипЗнч(Ссылка) =
Тип("СправочникСсылка.ВнутренниеДокументы") Тогда
ВидДокумента =
ХранилищеОбщихНастроек.Загрузить("НастройкиРаботыСДокументами", "ВидВнутреннегоДокумента");
ТипМетаданных = "ВидыВнутреннихДокументов";
КонецЕсли;
Если Не ЗначениеЗаполнено(ВидДокумента) Тогда
Запрос = Новый Запрос;
Запрос.Текст = СтрЗаменить(
"ВЫБРАТЬ РАЗРЕШЕННЫЕ ПЕРВЫЕ 1
|	ТаблицаВидаДокументов.Ссылка КАК Ссылка
|ИЗ
|	&ВидДокумента КАК
ТаблицаВидаДокументов
|ГДЕ
|	(НЕ
ТаблицаВидаДокументов.ПометкаУдаления)
|	И (НЕ ТаблицаВидаДокументов.ЭтоГруппа)
|	И ЛОЖЬ В
|		(ВЫБРАТЬ
|
|		ЛОЖЬ КАК ЗначениеЛожь
|		ИЗ
|
|		(ВЫБРАТЬ ПЕРВЫЕ 2
|
|		ИСТИНА КАК ЗначениеИстина
|		ИЗ
|
|		&ВидДокумента КАК
ТаблицаВидаДокументов
|		ГДЕ
|
|		(НЕ
ТаблицаВидаДокументов.ПометкаУдаления)
|		И (НЕ ТаблицаВидаДокументов.ЭтоГруппа)
|		)
КАК ВыбранныеОбъекты
|	ИМЕЮЩИЕ
|
|		КОЛИЧЕСТВО(ВыбранныеОбъекты.Значение
Истина) = 1)", "&ВидДокумента", "Справочник." +
ТипМетаданных);
Результат = Запрос.Выполнить();
Если Не Результат.Пустой() Тогда
Выборка = Результат.Выбрать();
Выборка.Следующий();
ВидДокумента = Выборка.Ссылка;
КонецЕсли;
КонецЕсли;
Возврат ВидДокумента;
КонецФункции
// Возвращает способ доставки по умолчанию
Функция ПолучитьСпособДоставкиПоУмолчанию(Тип)
Экспорт
СпособДоставки =
Справочники.СпособыДоставки.ПустаяСсылка();
Если Тип = "СпособПолучения" Тогда
СпособДоставки =
ХранилищеОбщихНастроек.Загрузить("НастройкиРаботыСДокументами", "СпособПолучения");
ИначеЕсли Тип = "СпособОтправки" Тогда
СпособДоставки =
ХранилищеОбщихНастроек.Загрузить("НастройкиРаботыСДокументами", "СпособОтправки");
КонецЕсли;
Если Не ЗначениеЗаполнено(СпособДоставки) Тогда
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ ПЕРВЫЕ 1
|	СпособыДоставки.Ссылка КАК Ссылка
|ИЗ
|	Справочник.СпособыДоставки КАК
СпособыДоставки
|ГДЕ
|	(НЕ СпособыДоставки.ПометкаУдаления)
|	И ЛОЖЬ В
|		(ВЫБРАТЬ
|
|		ЛОЖЬ КАК ЗначениеЛожь
|		ИЗ
|
|		(ВЫБРАТЬ ПЕРВЫЕ 2
|
|		ИСТИНА КАК ЗначениеИстина
|		ИЗ
|
|		Справочник.СпособыДоставки КАК
СпособыДоставки
|		ГДЕ
|
|		(НЕ СпособыДоставки.ПометкаУдаления)
|		)
КАК ВыбранныеОбъекты
|	ИМЕЮЩИЕ
|
|		КОЛИЧЕСТВО(ВыбранныеОбъекты.Значение
Истина) = 1)";
Результат = Запрос.Выполнить();
Если Не Результат.Пустой() Тогда
Выборка = Результат.Выбрать();
Выборка.Следующий();
СпособДоставки = Выборка.Ссылка;
КонецЕсли;
КонецЕсли;
Возврат СпособДоставки;
КонецФункции
Функция ПолучитьВалютуПоУмолчанию() Экспорт
Валюта =
ХранилищеОбщихНастроек.Загрузить("НастройкиРаботыСДокументами", "Валюта");
Если ЗначениеЗаполнено(Валюта) Тогда
Возврат Валюта;
КонецЕсли;
Валюта = Константы.ОсновнаяВалюта.Получить();
Возврат Валюта;
КонецФункции
// Получает актуальное состояние документа
Функция ПолучитьСостояниеДокумента(Документ) Экспорт
Если Не
ПолучитьФункциональнуюОпцию("ИспользоватьСостояния Документов") Тогда
Возврат
Перечисления.СостоянияДокументов.ПустаяСсылка();
КонецЕсли;
Если Не ЗначениеЗаполнено(Документ) Тогда
Возврат
Перечисления.СостоянияДокументов.ПустаяСсылка();
КонецЕсли;
УстановитьПривилегированныйРежим(Истина);
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|
|	СостоянияДокументовСрезПоследних.Состояние КАК Состояние
|ИЗ
|
|	РегистрСведений.СостоянияДокументов.Срез
```

```
Последних(, Документ = &Документ) КАК
СостоянияДокументовСрезПоследних";
Запрос.УстановитьПараметр("Документ", Документ);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат
Перечисления.СостоянияДокументов.ПустаяСсылка();
КонецЕсли;
Выборка = Результат.Выбрать();
Выборка.Следующий();
Возврат Выборка.Состояние;
КонецФункции
// Устанавливает состояние документа
Процедура ЗаписатьСостояниеДокумента(Документ,
Период, Состояние, Установил) Экспорт
Если Не
ПолучитьФункциональнуюОпцию("ИспользоватьСостояния
Документов") Тогда
Возврат;
КонецЕсли;
ТекущееСостояние =
ПолучитьСостояниеДокумента(Документ);
Если ТекущееСостояние = Состояние Тогда
Возврат;
КонецЕсли;
УстановитьПривилегированныйРежим(Истина);
МенеджерЗаписи =
РегистрыСведений.СостоянияДокументов.СоздатьМенедже
рЗаписи();
МенеджерЗаписи.Период = Период;
МенеджерЗаписи.Документ = Документ;
МенеджерЗаписи.Состояние = Состояние;
МенеджерЗаписи.Установил = Установил;
МенеджерЗаписи.Записать();
КонецПроцедуры
// Удаляет все состояния документа, установленные
объектом Установил
//
Процедура УдалитьСостояниеДокумента(Документ,
Установил) Экспорт
Если Не
ПолучитьФункциональнуюОпцию("ИспользоватьСостояния
Документов") Тогда
Возврат;
КонецЕсли;
УстановитьПривилегированныйРежим(Истина);
Набор =
РегистрыСведений.СостоянияДокументов.СоздатьНаборЗап
исей();
Набор.Отбор.Документ.Установить(Документ);
Набор.Прочитать();
ЗаписиКУдалению = Новый Массив;
Для каждого Запись Из Набор Цикл
Если Запись.Установил = Установил Тогда
ЗаписиКУдалению.Добавить(Запись);
КонецЕсли;
КонецЦикла;
Для каждого Запись Из ЗаписиКУдалению Цикл
Набор.Удалить(Запись);
КонецЦикла;
Набор.Записать(Истина);
КонецПроцедуры

// Возвращает ключ актуального состояния документа
Функция ПолучитьКлючСостоянияДокумента(Документ)
Экспорт
Если Не ЗначениеЗаполнено(Документ) Тогда
Возврат Неопределено;
КонецЕсли;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	СостоянияДокументовСрезПоследних.Период
КАК Период
|ИЗ
|
	РегистрСведений.СостоянияДокументов.Срез
Последних(, Документ = &Документ) КАК
СостоянияДокументовСрезПоследних";
Запрос.УстановитьПараметр("Документ", Документ);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат Неопределено;
КонецЕсли;
Выборка = Результат.Выбрать();
Выборка.Следующий();
Возврат
РегистрыСведений.СостоянияДокументов.СоздатьКлючЗап
иси( Новый Структура("Период, Документ",
Выборка.Период, Документ) );
КонецФункции
// Возвращает признак использования видов входящих
документов
Функция ИспользоватьВидыВходящихДокументов()
Экспорт
Возврат
ПолучитьФункциональнуюОпцию("ИспользоватьВидыВход
ящихДокументов");
КонецФункции
// Возвращает признак использования видов исходящих
документов
Функция ИспользоватьВидыИсходящихДокументов()
Экспорт
Возврат
ПолучитьФункциональнуюОпцию("ИспользоватьВидыИсхо
дящихДокументов");
КонецФункции
// Возвращает признак использования видов внутренних
документов
Функция ИспользоватьВидыВнутреннихДокументов()
Экспорт
Возврат
ПолучитьФункциональнуюОпцию("ИспользоватьВидыВнут
реннихДокументов");
КонецФункции
// Возвращает признак использования видов документов по
объекту
Функция ИспользоватьВидыДокументов(ВидДокумента)
Экспорт
ИспользоватьВидыДокументов = Ложь;
Если ТипЗнч(ВидДокумента) =
Тип("СправочникСсылка.ВидыВходящихДокументов")
Тогда
ИспользоватьВидыДокументов =
Делопроизводство.ИспользоватьВидыВходящихДокументов
();
ИначеЕсли ТипЗнч(ВидДокумента) =
Тип("СправочникСсылка.ВидыИсходящихДокументов")
Тогда
ИспользоватьВидыДокументов =
Делопроизводство.ИспользоватьВидыИсходящихДокумento
в();
ИначеЕсли ТипЗнч(ВидДокумента) =
Тип("СправочникСсылка.ВидыВнутреннихДокументов")
Тогда
ИспользоватьВидыДокументов =
Делопроизводство.ИспользоватьВидыВнутреннихДокумент
ов();
КонецЕсли;
Возврат ИспользоватьВидыДокументов;
КонецФункции
// Возвращает признак использования номенклатуры дел
Функция ИспользоватьНоменклатуруДел() Экспорт
Возврат
ПолучитьФункциональнуюОпцию("ИспользоватьНоменклат
уруДел");
КонецФункции
// Проверяет уникальность регистрационного номера
Функция РегистрационныйНомерУникален(Объект)
Экспорт
УстановитьПривилегированныйРежим(Истина);
Нумератор =
Нумерация.ПолучитьНумераторДокумента(Объект.ВидДоку
мента);
Если ЗначениеЗаполнено(Нумератор) Тогда //
автонумерация
Периодичность = Нумератор.Периодичность;
Иначе

		// ручная нумерация
Периодичность =
Перечисления.ПериодичностьНумераторов.Год;
КонецЕсли;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	Ссылка
|ИЗ
|	Справочник." +
Объект.Ссылка.Метаданные().Имя + " КАК Справочник ";
Если
Нумератор.НезависимаяНумерацияПоСвязанномуДокумент
у Тогда
Запрос.Текст = Запрос.Текст +
" ЛЕВОЕ СОЕДИНЕНИЕ
РегистрСведений.СвязиДокументов КАК СвязиДокументов
|	ПО СвязиДокументов.Документ =
Справочник.Ссылка И СвязиДокументов.ТипСвязи =
&ТипСвязи ";
Запрос.УстановитьПараметр("ТипСвязи",
Нумератор.ТипСвязи);
КонецЕсли;
Запрос.Текст = Запрос.Текст +
" ГДЕ
|	РегистрационныйНомер =
&РегистрационныйНомер
|	И ДатаРегистрации МЕЖДУ
&НачалоПериодаНумерации И &КонецПериодаНумерации
|	И Ссылка <>&Ссылка ";
Если
ПолучитьФункциональнуюОпцию("ИспользоватьУчетПоОр
ганизациям") И
Нумератор.НезависимаяНумерацияПоОрганизациям Тогда
Запрос.Текст = Запрос.Текст + " И (Организация =
&Организация) ";
Запрос.УстановитьПараметр("Организация",
Объект.Организация);
КонецЕсли;
Если
Нумератор.НезависимаяНумерацияПоСвязанномуДокумент
у Тогда
СвязанныйДокумент =
СвязиДокументов.ПолучитьСвязанныйДокумент(Объект.Сс
ылка, Нумератор.ТипСвязи);
Если ЗначениеЗаполнено(СвязанныйДокумент) Тогда
Запрос.Текст = Запрос.Текст + " И (СвязанныйДокумент =
&СвязанныйДокумент) ";
Запрос.УстановитьПараметр("СвязанныйДокумент",
СвязанныйДокумент);
КонецЕсли;
КонецЕсли;
Если
ИспользоватьВидыДокументов(Объект.ВидДокумента)
Тогда
Если ЗначениеЗаполнено(Нумератор) Тогда // проверка
уникальности в рамках нумератора
Запрос.Текст = Запрос.Текст + " И
(ВидДокумента.Нумератор = &Нумератор) ";
Запрос.УстановитьПараметр("Нумератор", Нумератор);
Иначе
Запрос.Текст = Запрос.Текст + " И (ВидДокумента =
&ВидДокумента) ";
Запрос.УстановитьПараметр("ВидДокумента",
Объект.ВидДокумента);
КонецЕсли;
КонецЕсли;
Запрос.УстановитьПараметр("РегистрационныйНомер",
Объект.РегистрационныйНомер);
Запрос.УстановитьПараметр("НачалоПериодаНумерации",
Нумерация.НачалоПериодаНумерации(Периодичность,
Объект.ДатаРегистрации));
Запрос.УстановитьПараметр("КонецПериодаНумерации",
Нумерация.КонецПериодаНумерации(Периодичность,
Объект.ДатаРегистрации));
Запрос.УстановитьПараметр("Ссылка", Объект.Ссылка);
Результат = Запрос.Выполнить();
Возврат Результат.Пустой();
КонецФункции
// Устанавливает доступность полей карточки документа в
зависимости от состояния
Процедура УстановитьДоступностьПоСостоянию(Форма,
Состояние, ДоступныеПоля = "") Экспорт
ДоступныеПоля = "";
НедоступныеПоля = "";
НеизменяемыеПоля = "";
Если Не
ПолучитьФункциональнуюОпцию("ИспользоватьСостояния
Документов") Тогда
Возврат;
КонецЕсли;
Если РольДоступна("ПолныеПрава") Тогда
Возврат;
КонецЕсли;
Если ТипЗнч(Форма.Объект.Ссылка) =
Тип("СправочникСсылка.ВходящиеДокументы") Тогда
РольДелопроизводитель =
РольДоступна("ДобавлениеИзменениеВходящихДокумento
в");
НеизменяемыеПоля = Новый Структура("
|ОтправленОтвет,
|ОтветПереадресовавшему,
|ПереадресованДокументом, КонтактноеЛицоПереадресата,
Переадресат, Переадресовать,
|Категории,
|ИсторияСогласования,
ВизыСогласованияПечатьЛистСогласования,
|ГруппаЭЦП, ФормаПодписать, ФайлыПодписатьФайл,
ФайлыДобавитьЭЦПИзФайла,
ФайлыСохранитьВместеСЭЦП,
|ФайлыКонтекстноеМенюПодписатьФайл,
ФайлыКонтекстноеМенюДобавитьЭЦПИзФайла,
ФайлыКонтекстноеМенюСохранитьВместеСЭЦП,
|Комментарий");
ИначеЕсли ТипЗнч(Форма.Объект.Ссылка) =
Тип("СправочникСсылка.ИсходящиеДокументы") Тогда
РольДелопроизводитель =
РольДоступна("РегистрацияИсходящихДокументов");
НеизменяемыеПоля = Новый Структура("
|ПолученОтвет,
|Категории,
|ИсторияСогласования,
ВизыСогласованияПечатьЛистСогласования,
|ГруппаЭЦП, ФормаПодписать, ФайлыПодписатьФайл,
ФайлыДобавитьЭЦПИзФайла,
ФайлыСохранитьВместеСЭЦП,
|ФайлыКонтекстноеМенюПодписатьФайл,
ФайлыКонтекстноеМенюДобавитьЭЦПИзФайла,
ФайлыКонтекстноеМенюСохранитьВместеСЭЦП,
|Комментарий");
ИначеЕсли ТипЗнч(Форма.Объект.Ссылка) =
Тип("СправочникСсылка.ВнутренниеДокументы") Тогда
РольДелопроизводитель =
РольДоступна("РегистрацияВнутреннихДокументов");
НеизменяемыеПоля = Новый Структура("
|Категории,
|ИсторияСогласования,
ВизыСогласованияПечатьЛистСогласования,
|ГруппаЭЦП, ФормаПодписать, ФайлыПодписатьФайл,
ФайлыДобавитьЭЦПИзФайла,
ФайлыСохранитьВместеСЭЦП,
|ФайлыКонтекстноеМенюПодписатьФайл,
ФайлыКонтекстноеМенюДобавитьЭЦПИзФайла,
ФайлыКонтекстноеМенюСохранитьВместеСЭЦП,
|Комментарий,
|НеДействуетВСоответствии");
КонецЕсли;
// Рабочие группы
Если ТипЗнч(НеизменяемыеПоля) = Тип("Структура")
Тогда
НеизменяемыеПоля.Вставить("РабочаяГруппаТаблица");
```

```
НеизменяемыеПоля.Вставить("РабочаяГруппаТаблицаУчастник");
НеизменяемыеПоля.Вставить("ПодобратьУчастниковРабочейГруппы");
НеизменяемыеПоля.Вставить("РабочаяГруппаТаблицаДобавить");
НеизменяемыеПоля.Вставить("РабочаяГруппаТаблицаУдалить");
НеизменяемыеПоля.Вставить("РабочаяГруппаТаблицаРабочаяГруппаОбновить");
НеизменяемыеПоля.Вставить("РабочаяГруппаТаблицаВывестиСписок");
НеизменяемыеПоля.Вставить("ПерезаполнитьРабочуюГруппуДокумента");
НеизменяемыеПоля.Вставить("ЗапретитьАвтоматическоеДобавлениеУчастниковРабочейГруппы");
КонецЕсли;

Если Константы.ОграничиватьДоступностьПолейПоСостоянию.Получить() Тогда
Если Состояние = Перечисления.СостоянияДокументов.НаРегистрации Тогда
Если РольДелопроизводитель Тогда
ДоступныеПоля = "";
НедоступныеПоля = "";
Иначе
ДоступныеПоля = Новый Структура("
|Состояние,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
КонецЕсли;
ИначеЕсли (Состояние = Перечисления.СостоянияДокументов.Зарегистрирован)
И (ТипЗнч(Форма.Объект.Ссылка) = Тип("СправочникСсылка.ВходящиеДокументы")) Тогда
Если РольДелопроизводитель Тогда
ДоступныеПоля = Новый Структура("
|Зарегистрировать, РегистрационныйНомер, ДатаРегистрации,
|Резолюции, РезолюцииДобавить, РезолюцииИзменить, РезолюцииУдалить,
|Состояние,
|СрокИсполнения, Дело,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
Иначе
ДоступныеПоля = Новый Структура("
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
КонецЕсли;
ИначеЕсли (Состояние = Перечисления.СостоянияДокументов.Зарегистрирован)
И (ТипЗнч(Форма.Объект.Ссылка) = Тип("СправочникСсылка.ИсходящиеДокументы")) Тогда
Если РольДелопроизводитель Тогда
Если Форма.Объект.Получатели.НайтиСтроки(Новый Структура("Отправлен", Истина)).Количество() > 0 Тогда
ДоступныеПоля = Новый Структура("
|Зарегистрировать, РегистрационныйНомер,
ДатаРегистрации,
|Состояние,
|СрокИсполнения, Дело,
|Отправлен, ДатаОтправки, СпособОтправки, ВходящийНомер, ВходящаяДата,
|Получатели, ПолучателиОтправлен, ПолучателиДатаОтправки, ПолучателиСпособОтправки, ПолучателиВходящийНомер, ПолучателиВходящаяДата,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
Иначе
ДоступныеПоля = Новый Структура("
|Зарегистрировать, РегистрационныйНомер,
ДатаРегистрации,
|Состояние,
|СрокИсполнения, Дело,
|ЗакончитьРедактирование, Занять, Освободить, Редактировать, СохранитьИзменения, ОбновитьИзФайлаНаДиске, ФайлыЗашифровать, ФайлыРасшифровать,
ФайлыКонтекстноеМенюЗашифровать, ФайлыКонтекстноеМенюРасшифровать,
|КонтекстноеМенюФайлыЗакончитьРедактирование, КонтекстноеМенюФайлыЗанять, КонтекстноеМенюФайлыОсвободить, КонтекстноеМенюФайлыРедактировать, КонтекстноеМенюФайлыСохранитьИзменения, КонтекстноеМенюФайлыОбновитьИзФайлаНаДиске,
|Отправлен, ДатаОтправки, СпособОтправки, ВходящийНомер, ВходящаяДата,
|Получатели, ПолучателиОтправлен, ПолучателиДатаОтправки, ПолучателиСпособОтправки, ПолучателиВходящийНомер, ПолучателиВходящаяДата,
|Файлы, СоздатьФайл, ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
КонецЕсли;
Иначе
Доступныеполя = Новый Структура("
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
КонецЕсли;
ИначеЕсли Состояние = Перечисления.СостоянияДокументов.НаИсполнении Тогда
Если РольДелопроизводитель Тогда
ДоступныеПоля = Новый Структура("
|НеДействует, ДатаОкончанияДействия, Бессрочный, ПорядокПродления,
|Зарегистрировать, РегистрационныйНомер, ДатаРегистрации,
|Состояние, Папка,
|СрокИсполнения, СрокДействияСтрока, Дело, Отменен, ОтменендОкументом,
|ЗакончитьРедактирование, Занять, Освободить, Редактировать, СохранитьИзменения, ОбновитьИзФайлаНаДиске, ФайлыЗашифровать, ФайлыРасшифровать,
ФайлыКонтекстноеМенюЗашифровать, ФайлыКонтекстноеМенюРасшифровать,
|КонтекстноеМенюФайлыЗакончитьРедактирование, КонтекстноеМенюФайлыЗанять, КонтекстноеМенюФайлыОсвободить, КонтекстноеМенюФайлыРедактировать, КонтекстноеМенюФайлыСохранитьИзменения, КонтекстноеМенюФайлыОбновитьИзФайлаНаДиске,
|Файлы, СоздатьФайл, ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
Иначе
ДоступныеПоля = Новый Структура("
|СрокДействияСтрока,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
КонецЕсли;
ИначеЕсли Состояние = Перечисления.СостоянияДокументов.НаРассмотрении Тогда
Если РольДелопроизводитель Тогда
ДоступныеПоля = Новый Структура("
|Зарегистрировать, РегистрационныйНомер, ДатаРегистрации,
|Состояние, Папка,
|СрокИсполнения,
|Резолюции, РезолюцииДобавить, РезолюцииИзменить, РезолюцииУдалить,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
Иначе
ДоступныеПоля = Новый Структура("
|Резолюции, РезолюцииДобавить, РезолюцииИзменить, РезолюцииУдалить,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
КонецЕсли;
ИначеЕсли Состояние = Перечисления.СостоянияДокументов.Рассмотрен Тогда
Если РольДелопроизводитель Тогда
ДоступныеПоля = Новый Структура("
|Зарегистрировать, РегистрационныйНомер, ДатаРегистрации,
|Состояние, Папка,
|СрокИсполнения,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
Иначе
ДоступныеПоля = Новый Структура("
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
КонецЕсли;
ИначеЕсли Состояние = Перечисления.СостоянияДокументов.НаИсполнении Тогда
Если РольДелопроизводитель Тогда
ДоступныеПоля = Новый Структура("
|НеДействует, ДатаОкончанияДействия, Бессрочный, ПорядокПродления,
|Зарегистрировать, РегистрационныйНомер, ДатаРегистрации,
|Состояние, Папка,
|СрокИсполнения,
|Резолюции, РезолюцииДобавить, РезолюцииИзменить, РезолюцииУдалить,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
Иначе
ДоступныеПоля = Новый Структура("
|Резолюции, РезолюцииДобавить, РезолюцииИзменить, РезолюцииУдалить,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
КонецЕсли;
ИначеЕсли Состояние = Перечисления.СостоянияДокументов.Исполнен Тогда
Если РольДелопроизводитель Тогда
ДоступныеПоля = Новый Структура("
|Зарегистрировать, РегистрационныйНомер, ДатаРегистрации,
|Состояние, Папка,
|Дело,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
Иначе
ДоступныеПоля = Новый Структура("
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
КонецЕсли;
ИначеЕсли Состояние = Перечисления.СостоянияДокументов.Проект
Или Состояние = Перечисления.СостоянияДокументов.НеСогласован
Или Состояние = Перечисления.СостоянияДокументов.НеУтвержден Тогда
ДоступныеПоля = "";
НедоступныеПоля = Новый Структура("Подписал, Утвердил, Дело");
ИначеЕсли Состояние = Перечисления.СостоянияДокументов.НаСогласовании Тогда
ДоступныеПоля = Новый Структура("
|Состояние, Папка,
|ВизыСогласования, ВизыСогласованияСоздать, ВизыСогласованияИзменить, ВизыСогласованияУдалить, ВизыСогласованияКонтекстноеМенюСоздать, ВизыСогласованияКонтекстноеМенюИзменить, ВизыСогласованияКонтекстноеМенюУдалить, ВизыСогласованияКонтекстноеМенюОбновить,
|ЗакончитьРедактирование, Занять, Освободить, Редактировать, СохранитьИзменения, ОбновитьИзФайлаНаДиске, ФайлыЗашифровать, ФайлыРасшифровать,
ФайлыКонтекстноеМенюЗашифровать, ФайлыКонтекстноеМенюРасшифровать,
|КонтекстноеМенюФайлыЗакончитьРедактирование, КонтекстноеМенюФайлыЗанять, КонтекстноеМенюФайлыОсвободить, КонтекстноеМенюФайлыРедактировать, КонтекстноеМенюФайлыСохранитьИзменения, КонтекстноеМенюФайлыОбновитьИзФайлаНаДиске,
|Файлы, СоздатьФайл, ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
ИначеЕсли Состояние = Перечисления.СостоянияДокументов.НаУтверждении Тогда
Доступныеполя = Новый Структура("
|Состояние, Папка,
|Подписал, Утвердил,
|ЗакончитьРедактирование, Занять, Освободить, Редактировать, СохранитьИзменения, ОбновитьИзФайлаНаДиске, ФайлыЗашифровать, ФайлыРасшифровать,
ФайлыКонтекстноеМенюЗашифровать, ФайлыКонтекстноеМенюРасшифровать,
|КонтекстноеМенюФайлыЗакончитьРедактирование, КонтекстноеМенюФайлыЗанять, КонтекстноеМенюФайлыОсвободить, КонтекстноеМенюФайлыРедактировать, КонтекстноеМенюФайлыСохранитьИзменения, КонтекстноеМенюФайлыОбновитьИзФайлаНаДиске,
|Файлы, СоздатьФайл, ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
ИначеЕсли Состояние = Перечисления.СостоянияДокументов.Согласован
Или Состояние = Перечисления.СостоянияДокументов.Утвержден Тогда
Если РольДелопроизводитель Тогда
ДоступныеПоля = Новый Структура("
|Состояние, Папка,
|Зарегистрировать, НоменклатураДел,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
|КонтекстноеМенюФайлыОткрытьФайл, КонтекстноеМенюФайлыСохранитьКак");
Иначе
ДоступныеПоля = Новый Структура("
|Состояние,
|ОткрытьФайл, СохранитьКак, НастроитьСписок, ВывестиСписок, Изменить,
```

```
|КонтекстноеМенюФайлыОткрытьФайл,
КонтекстноеМенюФайлыСохранитьКак");
КонецЕсли;
КонецЕсли;
Для Каждого Элемент Из Форма.Элементы Цикл
Если ТипЗнч(Элемент) = Тип("ГруппаФормы") И
Элемент.Имя <> "ГруппаСвойства" Тогда
Продолжить;
КонецЕсли;
Если ТипЗнч(Элемент) = Тип("ДекорацияФормы") Тогда
Продолжить;
КонецЕсли;
Если ТипЗнч(Элемент) = Тип("КнопкаФормы") И
Элемент.Родитель.Имя = "ФормаКомандныйПанель" Тогда
Продолжить;
КонецЕсли;
Если ТипЗнч(Элемент) = Тип("КнопкаФормы") И
Элемент.Родитель.Имя =
"КомандныйПанельФормыСоздатьНаОсновании" Тогда
Продолжить;
КонецЕсли;
Если ТипЗнч(Элемент) = Тип("КнопкаФормы") И
Элемент.Родитель.Имя = "ФормаСоздатьНаОсновании"
Тогда
Продолжить;
КонецЕсли;
Если ТипЗнч(Элемент) = Тип("КнопкаФормы") И
Элемент.Родитель.Имя = "КомандныйПанельФормыПечать"
Тогда
Продолжить;
КонецЕсли;
Если ТипЗнч(Элемент) = Тип("КнопкаФормы") И
Элемент.Родитель.Имя = "ФормаПечать" Тогда
Продолжить;
КонецЕсли;
Если НеизменяемыеПоля.Свойство(Элемент.Имя) Тогда
Продолжить;
КонецЕсли;
РодительНеИзменяемоеПоле = Ложь;
ТекущийРодитель = Элемент.Родитель;
Пока ТипЗнч(ТекущийРодитель) <>
Тип("УправляемаяФорма") Цикл
Если НеизменяемыеПоля.Свойство(ТекущийРодитель.Имя)
Тогда
РодительНеИзменяемоеПоле = Истина;
Прервать;
КонецЕсли;
ТекущийРодитель = ТекущийРодитель.Родитель;
КонецЦикла;
Если РодительНеИзменяемоеПоле Тогда
Продолжить;
КонецЕсли;
Если ДоступныеПоля = "" Тогда
Если (НедоступныеПоля <> "") И
НедоступныеПоля.Свойство(Элемент.Имя) Тогда
Доступность = Ложь;
Иначе
Доступность = Истина;
КонецЕсли;
ИначеЕсли ДоступныеПоля.Свойство(Элемент.Имя) Тогда
Доступность = Истина;
Иначе
Доступность = Ложь;
КонецЕсли;
Если ТипЗнч(Элемент) = Тип("КнопкаФормы") Тогда
Элемент.Доступность = Доступность;
Иначе
Элемент.ТолькоПросмотр = Не Доступность;
КонецЕсли;
КонецЦикла;
КонецЕсли;
// доступность поля состояние зависит от настройки
Если Не
Константы.РазрешитьРучноеИзменениеСостоянияДокумент
ов.Получить()
И
Константы.ИспользоватьБизнесПроцессыИЗадачи.Получить
() Тогда
Форма.Элементы.Состояние.ТолькоПросмотр = Истина;
КонецЕсли;
КонецПроцедуры
// Получает контактное лицо корреспондента
Функция КонтактноеЛицоКорреспондента(Корреспондент)
Экспорт
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ ПЕРВЫЕ 1
|	КонтактныеЛица.Ссылка КАК
КонтактноеЛицо
|ИЗ
|	Справочник.КонтактныеЛица КАК
КонтактныеЛица
|ГДЕ
|	КонтактныеЛица.Владелец = &Владелец
|	И (НЕ КонтактныеЛица.ПометкаУдаления)
|	И ЛОЖЬ В
|		(ВЫБРАТЬ
|
|	ЛОЖЬ КАК ЗначениеЛожь
|		ИЗ
|
|	(ВЫБРАТЬ ПЕРВЫЕ 2
|
|	ИСТИНА КАК ЗначениеИстина
|		ИЗ
|
|	Справочник.КонтактныеЛица КАК
КонтактныеЛица
|		ГДЕ
|
|	(НЕ КонтактныеЛица.ПометкаУдаления)
|
|	И КонтактныеЛица.Владелец = &Владелец
|		)
КАК ВыбранныеОбъекты
|		ИМЕЮЩИЕ
|
|	КОЛИЧЕСТВО(ВыбранныеОбъекты.Значение
Истина) = 1)";
Запрос.УстановитьПараметр("Владелец", Корреспондент);
Результат = Запрос.Выполнить();
Если Не Результат.Пустой() Тогда
Выборка = Результат.Выбрать();
Выборка.Следующий();
Возврат Выборка.КонтактноеЛицо;
КонецЕсли;
Возврат Неопределено;
КонецФункции
// Возвращает количество документов переданного вида
Функция
КоличествоДокументовПоВидуДокумента(ВидДокумента)
Экспорт
УстановитьПривилегированныйРежим(Истина);
Если ТипЗнч(ВидДокумента) =
Тип("СправочникСсылка.ВидыВнутреннихДокументов")
Тогда
Тип = "ВнутренниеДокументы";
ИначеЕсли ТипЗнч(ВидДокумента) =
Тип("СправочникСсылка.ВидыВходящихДокументов")
Тогда
Тип = "ВходящиеДокументы";
ИначеЕсли ТипЗнч(ВидДокумента) =
Тип("СправочникСсылка.ВидыИсходящихДокументов")
Тогда
Тип = "ИсходящиеДокументы";
КонецЕсли;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	КОЛИЧЕСТВО(*) КАК Количество
|ИЗ
|	Справочник." + Тип + "
|ГДЕ
|	ВидДокумента = &ВидДокумента
|	И РегистрационныйНомер <> """""";
Запрос.УстановитьПараметр("ВидДокумента",
ВидДокумента);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат 0;
КонецЕсли;
Выборка = Результат.Выбрать();
Выборка.Следующий();
Возврат Выборка.Количество;
КонецФункции
// Возвращает количество документов для переданного
нумератора
Функция КоличествоДокументовПоНумератору(Нумератор)
Экспорт
УстановитьПривилегированныйРежим(Истина);
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	КОЛИЧЕСТВО(*) КАК Количество
|ИЗ
|	Справочник.ВходящиеДокументы КАК
ВходящиеДокументы
|ГДЕ
|
|	ВходящиеДокументы.ВидДокумента.Нумерат
ор = &Нумератор
|	И
ВходящиеДокументы.РегистрационныйНомер <> """"
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	КОЛИЧЕСТВО(*)
|ИЗ
|	Справочник.ИсходящиеДокументы КАК
ИсходящиеДокументы
|ГДЕ
|
|	ИсходящиеДокументы.ВидДокумента.Нумера
тор = &Нумератор
|	И
ИсходящиеДокументы.РегистрационныйНомер <> """"
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	КОЛИЧЕСТВО(*)
|ИЗ
|	Справочник.ВнутренниеДокументы КАК
ВнутренниеДокументы
|ГДЕ
|
|	ВнутренниеДокументы.ВидДокумента.Нумера
тор = &Нумератор
|	И
ВнутренниеДокументы.РегистрационныйНомер <> """"";
Запрос.УстановитьПараметр("Нумератор", Нумератор);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат 0;
КонецЕсли;
Возврат Результат.Выгрузить().Итог("Количество");
КонецФункции
// Возвращает количество документов с пустым видом
Функция
КоличествоДокументовСПустымВидом(ТипДокумента)
Экспорт
УстановитьПривилегированныйРежим(Истина);
Если ТипДокумента = "ВходящийДокумент" Тогда
ТекстЗапроса =
"ВЫБРАТЬ
|	КОЛИЧЕСТВО(*) КАК Количество
|ИЗ
|	Справочник.ВходящиеДокументы КАК
ВходящиеДокументы
|ГДЕ
|	ВходящиеДокументы.ВидДокумента =
ЗНАЧЕНИЕ(Справочник.ВидыВходящихДокументов.Пуста
яСсылка)";
ИначеЕсли ТипДокумента = "ИсходящийДокумент" Тогда
ТекстЗапроса =
"ВЫБРАТЬ
|	КОЛИЧЕСТВО(*) КАК Количество
|ИЗ
|	Справочник.ИсходящиеДокументы КАК
ИсходящиеДокументы
|ГДЕ
|	ИсходящиеДокументы.ВидДокумента =
ЗНАЧЕНИЕ(Справочник.ВидыИсходящихДокументов.Пуст
аяСсылка)";
ИначеЕсли ТипДокумента = "ВнутреннийДокумент" Тогда
ТекстЗапроса =
"ВЫБРАТЬ
|	КОЛИЧЕСТВО(*) КАК Количество
|ИЗ
|	Справочник.ВнутренниеДокументы КАК
ВнутренниеДокументы
|ГДЕ
|	ВнутренниеДокументы.ВидДокумента =
ЗНАЧЕНИЕ(Справочник.ВидыВнутреннихДокументов.Пуст
аяСсылка)";
КонецЕсли;
Запрос = Новый Запрос(ТекстЗапроса);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат 0;
КонецЕсли;
Выборка = Результат.Выбрать();
Выборка.Следующий();
Возврат Выборка.Количество;
КонецФункции
// Инициализирует персональные настройки работы с
документами - для использования на клиенте
Функция
ПолучитьПерсональныеНастройкиРаботыСДокументамиСе
ервер() Экспорт
Настройки = Новый Структура;
ПоказыватьПредупреждениеПриРегистрации =
ХранилищеОбщихНастроек.Загрузить("НастройкиРаботыС
Документами",
"ПоказыватьПредупреждениеПриРегистрации");
Если ПоказыватьПредупреждениеПриРегистрации =
Неопределено Тогда
ПоказыватьПредупреждениеПриРегистрации = Истина;
ХранилищеОбщихНастроек.Сохранить("НастройкиРаботыС
Документами",
"ПоказыватьПредупреждениеПриРегистрации",
ПоказыватьПредупреждениеПриРегистрации);
КонецЕсли;
Настройки.Вставить("ПоказыватьПредупреждениеПриРегис
трации", ПоказыватьПредупреждениеПриРегистрации И
ПолучитьФункциональнуюОпцию("ИспользоватьСостояния
Документов"));
СпособОтраженияПередачиКорреспонденту =
ХранилищеОбщихНастроек.Загрузить("НастройкиРаботыС
Документами",
"СпособОтраженияПередачиКорреспонденту");
Если СпособОтраженияПередачиКорреспонденту =
Неопределено Тогда
СпособОтраженияПередачиКорреспонденту =
Перечисления.СпособыОтраженияПередачиКорреспонденту
.ЗадаватьВопрос;
ХранилищеОбщихНастроек.Сохранить("НастройкиРаботыС
Документами",
"СпособОтраженияПередачиКорреспонденту",
СпособОтраженияПередачиКорреспонденту);
КонецЕсли;
Настройки.Вставить("СпособОтраженияПередачиКорреспон
денту", СпособОтраженияПередачиКорреспонденту);
УстановитьПривилегированныйРежим(Истина);
Настройки.Вставить("ИспользоватьФайлыУВходящихДоку
ментов",
Константы.ИспользоватьФайлыУВходящихДокументов.Пол
учить());
```

```
Настройки.Вставить("ИспользоватьФайлыУИсходящихДокументов",
Константы.ИспользоватьФайлыУИсходящихДокументов.Получить());
Возврат Настройки; // параметры доступны только для чтения
КонецФункции
// Возвращает ключ записи регистра сведений ЖурналПередачиДокументов
Функция ПолучитьКлючЖурналаПередачи(Период, Документ, ТипЭкземпляра, НомерЭкземпляра) Экспорт
Ключ = РегистрыСведений.ЖурналПередачиДокументов.СоздатьКлючЗаписи(
Новый Структура("Период, Документ, ТипЭкземпляра, НомерЭкземпляра",
Период,
Документ,
ТипЭкземпляра,
НомерЭкземпляра));
Возврат Ключ;
КонецФункции
// Возвращает количество держателей документа
Функция КоличествоКомуПереданДокумент(Документ) Экспорт
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	КОЛИЧЕСТВО(*) КАК Количество
|ИЗ
|	РегистрСведений.ЖурналПередачиДокументов КАК ЖурналПередачиДокументов
|ГДЕ
|	ЖурналПередачиДокументов.Документ = &Документ
|	И ЖурналПередачиДокументов.Возвращен = ЛОЖЬ";
Запрос.УстановитьПараметр("Документ", Документ);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат 0;
КонецЕсли;
Выборка = Результат.Выбрать();
Выборка.Следующий();
Возврат Выборка.Количество;
КонецФункции
// Формирует строку информации о держателях документа
Функция СтрокаКомуПереданДокумент(Документ) Экспорт
Если Не ЗначениеЗаполнено(Документ) Тогда
Возврат "";
КонецЕсли;
ОригиналПередан = "";
ОригиналПереданДата = '00010101';
ОригиналПереданМассив = Новый Массив;
ОригиналПереданКоличество = 0;
КопияПередана = "";
КопияПереданаДата = '00010101';
КопияПереданаМассив = Новый Массив;
КопияПереданаКоличество = 0;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|	ЖурналПередачиДокументов.ТипЭкземпляра КАК ТипЭкземпляра,
|	ЖурналПередачиДокументов.Пользователь КАК Пользователь,
|	ЖурналПередачиДокументов.Период КАК ДатаПередачи,
|	ЖурналПередачиДокументов.НомерЭкземпляра КАК НомерЭкземпляра
|ИЗ
|	РегистрСведений.ЖурналПередачиДокументов КАК ЖурналПередачиДокументов
|ГДЕ
|	ЖурналПередачиДокументов.Документ = &Документ
|	И ЖурналПередачиДокументов.Возвращен = ЛОЖЬ";
Запрос.УстановитьПараметр("Документ", Документ);
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
Если Выборка.ТипЭкземпляра = Перечисления.ТипыЭкземпляров.Оригинал Тогда
Если ОригиналПереданМассив.Найти(Выборка.Пользователь) = Неопределено Тогда
ОригиналПередан = ОригиналПередан + Строка(Выборка.Пользователь) + ", ";
ОригиналПереданМассив.Добавить(Выборка.Пользователь);
КонецЕсли;
ОригиналПереданКоличество = ОригиналПереданКоличество + 1;
ОригиналПереданДата = Выборка.ДатаПередачи;
ИначеЕсли Выборка.ТипЭкземпляра = Перечисления.ТипыЭкземпляров.Копия Тогда
Если КопияПереданаМассив.Найти(Выборка.Пользователь) = Неопределено Тогда
КопияПередана = КопияПередана + Строка(Выборка.Пользователь) + ", ";
КопияПереданаМассив.Добавить(Выборка.Пользователь);
КонецЕсли;
КопияПереданаКоличество = КопияПереданаКоличество + 1;
КопияПереданаДата = Выборка.ДатаПередачи;
КонецЕсли;
КонецЦикла;
Если ОригиналПередан <> "" Тогда
ОригиналПередан = Лев(ОригиналПередан, СтрДлина(ОригиналПередан) - 2);
Если ОригиналПереданКоличество = 1 Тогда
ОригиналПередан = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Оригинал находится у пользователя %1 с %2'"),
ОригиналПередан,
Формат(ОригиналПереданДата, "ДФ=dd.ММ.уууу"));
ИначеЕсли ОригиналПереданКоличество > 1 Тогда
Если ОригиналПереданМассив.Количество() = 1 Тогда
ОригиналПередан = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Оригиналы находятся у пользователя %1'"),
ОригиналПередан);
Иначе
ОригиналПередан = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Оригиналы находятся у пользователей %1'"),
ОригиналПередан);
КонецЕсли;
КонецЕсли;
КонецЕсли;
Если КопияПередана <> "" Тогда
КопияПередана = Лев(КопияПередана, СтрДлина(КопияПередана) - 2);
Если КопияПереданаКоличество = 1 Тогда
КопияПередана = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Копия находится у пользователя %1 с %2'"),
КопияПередана,
Формат(КопияПереданаДата, "ДФ=dd.ММ.уууу"));
ИначеЕсли КопияПереданаКоличество > 1 Тогда
Если КопияПереданаМассив.Количество() = 1 Тогда
КопияПередана = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Копии находятся у пользователя %1'"),
КопияПередана);
Иначе
КопияПередана = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Копии находятся у пользователей %1'"),
КопияПередана);
КонецЕсли;
КонецЕсли;
КонецЕсли;
Если ОригиналПередан <> "" И КопияПередана <> "" Тогда
Возврат ОригиналПередан + Символы.ПС + КопияПередана;
Иначе
Возврат ОригиналПередан + КопияПередана;
КонецЕсли;
КонецФункции
// Получает актуальное состояние дела
Функция ПолучитьСостояниеДела(Дело) Экспорт
Если Не ЗначениеЗаполнено(Дело) Тогда
Возврат Перечисления.СостоянияДелХраненияДокументов.ПустаяСсылка();
КонецЕсли;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	СостоянияДелХраненияДокументов.Состояние КАК Состояние
|ИЗ
|	РегистрСведений.СостоянияДелХраненияДокументов.СрезПоследних(, ДелоХраненияДокументов = &Дело) КАК СостоянияДелХраненияДокументов";
Запрос.УстановитьПараметр("Дело", Дело);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат Перечисления.СостоянияДелХраненияДокументов.ПустаяСсылка();
КонецЕсли;
Выборка = Результат.Выбрать();
Выборка.Следующий();
Возврат Выборка.Состояние;
КонецФункции
// Возвращает режим выбора вида документа
Функция ПолучитьРежимВыбораВидаДокумента(ТипДокумента) Экспорт
Если ТипДокумента = "ВходящийДокумент" Тогда
ВидДокумента = "ВидыВходящихДокументов";
ИначеЕсли ТипДокумента = "ИсходящийДокумент" Тогда
ВидДокумента = "ВидыИсходящихДокументов";
ИначеЕсли ТипДокумента = "ВнутреннийДокумент" Тогда
ВидДокумента = "ВидыВнутреннихДокументов";
КонецЕсли;
Запрос = Новый Запрос;
Запрос.Текст = СтрЗаменить(
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|	ЛОЖЬ КАК ЗначениеЛожь
|ГДЕ
|	ВЫБОР
|		КОГДА ЛОЖЬ В
|		(ВЫБРАТЬ ПЕРВЫЕ 1
|			ЛОЖЬ
|		ИЗ
|			&ВидДокумента КАК ТаблицаВидаДокументов
|		ГДЕ
|			ТаблицаВидаДокументов.ЭтоГруппа)
|		ТОГДА ИСТИНА
|		ИНАЧЕ ЛОЖЬ В
|		(ВЫБРАТЬ
|			ЛОЖЬ КАК ЗначениеЛожь
|		ИЗ
|			(ВЫБРАТЬ ПЕРВЫЕ 16
|				ИСТИНА КАК ЗначениеИстина
|			ИЗ
|				&ВидДокумента КАК ТаблицаВидаДокументов
|			) КАК ВыбранныеОбъекты
|		ИМЕЮЩИЕ
|			КОЛИЧЕСТВО(ВыбранныеОбъекты.ЗначениеИстина) > 15)
|	КОНЕЦ", "&ВидДокумента", "Справочник." + ВидДокумента);
Результат = Запрос.Выполнить();
БыстрыйВыборВидаДокумента = Результат.Пустой();
Возврат БыстрыйВыборВидаДокумента;
КонецФункции
// Проверяет проверку возможности отнесения документа в дело
Функция ДелоМожетСодержатьДокумент(ТипПроверки, ЗначениеПроверки, Дело) Экспорт
УстановитьПривилегированныйРежим(Истина);
Если ТипПроверки = "ВидыДокументов" Тогда
Если ЗначениеЗаполнено(ЗначениеПроверки) И Дело.НоменклатураДел.ВидыДокументов.Количество() > 0 Тогда
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ИСТИНА
|ГДЕ
|	&ВидДокумента В ИЕРАРХИИ
|		(ВЫБРАТЬ
|			ВидыДокументов.ВидДокумента
|		ИЗ
|			Справочник.НоменклатураДел.ВидыДокументов КАК ВидыДокументов
|		ГДЕ
|			ВидыДокументов.Ссылка = &НоменклатураДел)";
Запрос.УстановитьПараметр("ВидДокумента", ЗначениеПроверки);
Запрос.УстановитьПараметр("НоменклатураДел", Дело.НоменклатураДел);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат Ложь;
КонецЕсли;
КонецЕсли;
ИначеЕсли ТипПроверки = "Корреспонденты" Тогда
Если ЗначениеЗаполнено(ЗначениеПроверки) И Дело.НоменклатураДел.Корреспонденты.Количество() > 0 Тогда
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ИСТИНА
|ГДЕ
|	&Корреспондент В ИЕРАРХИИ
|		(ВЫБРАТЬ
|			Корреспонденты.Корреспондент
|		ИЗ
```

```
|
	Справочник.НоменклатураДел.Корреспонденты КАК Корреспонденты
|		ГДЕ
|
	Корреспонденты.Ссылка = &НоменклатураДел)";
Запрос.УстановитьПараметр("Корреспондент", ЗначениеПроверки);
Запрос.УстановитьПараметр("НоменклатураДел", Дело.НоменклатураДел);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат Ложь;
КонецЕсли;
КонецЕсли;
ИначеЕсли ТипПроверки = "ВопросыДеятельности" Тогда
Если ЗначениеЗаполнено(ЗначениеПроверки) И Дело.НоменклатураДел.ВопросыДеятельности.Количество() > 0 Тогда
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ИСТИНА
|ГДЕ
|	&ВопросДеятельности В ИЕРАРХИИ
|		(ВЫБРАТЬ
|
	ВопросыДеятельности.ВопросДеятельности
|		ИЗ
|
	Справочник.НоменклатураДел.ВопросыДеятельности КАК ВопросыДеятельности
|		ГДЕ
|
	ВопросыДеятельности.Ссылка = &НоменклатураДел)";
Запрос.УстановитьПараметр("ВопросДеятельности", ЗначениеПроверки);
Запрос.УстановитьПараметр("НоменклатураДел", Дело.НоменклатураДел);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат Ложь;
КонецЕсли;
КонецЕсли;
КонецЕсли;
Возврат Истина;
КонецФункции
// Возвращает Истина, если корреспондент является юридическим лицом
Функция КорреспондентЮрЛицо(Корреспондент) Экспорт
Возврат ЗначениеЗаполнено(Корреспондент)
И ТипЗнч(Корреспондент) = Тип("СправочникСсылка.Корреспонденты")
И (Корреспондент.ЮрФизЛицо = Перечисления.ЮрФизЛицо.ЮрЛицо
Или Корреспондент.ЮрФизЛицо = Перечисления.ЮрФизЛицо.ИндивидуальныйПредприниматель
Или Корреспондент.ЮрФизЛицо = Перечисления.ЮрФизЛицо.ЮрЛицоНеРезидент);
КонецФункции
// Возвращает структуру данных корреспондента
Функция ПолучитьДанныеКорреспондента(Корреспондент) Экспорт
ДанныеКорреспондента = Новый Структура;
ДанныеКорреспондента.Вставить("КорреспондентЮрЛицо", КорреспондентЮрЛицо(Корреспондент));
ДанныеКорреспондента.Вставить("КонтактноеЛицо", КонтактноеЛицоКорреспондента(Корреспондент));
Возврат ДанныеКорреспондента;
КонецФункции
Процедура УдалитьВременныеСохраненныеПоиски(ТекущийПользователь) Экспорт
СписокСохраненныхПоисков = ХранилищеНастроекДанныхФорм.ПолучитьСписок("Обработка.ПоискПоРеквизитам.Форма.ПоискДокументовИФайлов");
Для Каждого СохраненныйПоиск Из СписокСохраненныхПоисков Цикл
КлючПоиска = СохраненныйПоиск.Значение;
Если Найти(КлючПоиска, "_временный_") > 0 Тогда
УстановитьПривилегированныйРежим(Истина);
ПользовательИБ = ПользователиИнформационнойБазы.НайтиПоУникальномуИдентификатору(ТекущийПользователь.ИдентификаторПользователяИБ);
Если ПользовательИБ <> Неопределено Тогда
ХранилищеНастроекДанныхФорм.Удалить("Обработка.ПоискПоРеквизитам.Форма.ПоискДокументовИФайлов", КлючПоиска, ПользовательИБ.Имя);
КонецЕсли;
КонецЕсли;
КонецЦикла;
КонецПроцедуры
Функция ПолучитьТекущуюДату() Экспорт
Возврат ТекущаяДатаСеанса();
КонецФункции
// Получает признак Отправлен
Функция ПолучитьПризнакОтправлен(ИсходящийДокумент, Получатель, Адресат) Экспорт
Отправлен = Ложь;
ПараметрыОтбора = Новый Структура("Получатель", Получатель);
НайденныеСтроки = ИсходящийДокумент.Получатели.НайтиСтроки(ПараметрыОтбора);
Если НайденныеСтроки.Количество() = 1 Тогда
Отправлен =  НайденныеСтроки[0].Отправлен;
Иначе
ПараметрыОтбора = Новый Структура("Получатель, Адресат", Получатель, Адресат);
НайденныеСтроки = ИсходящийДокумент.Получатели.НайтиСтроки(ПараметрыОтбора);
Если НайденныеСтроки.Количество() = 1 Тогда
Отправлен =  НайденныеСтроки[0].Отправлен;
КонецЕсли;
КонецЕсли;
Возврат Отправлен;
КонецФункции
// Заполнить подчиненные документы
Процедура ЗаполнитьПодчиненныеДокументы(СтрокаДерева, ПараметрыДокумент) Экспорт
Ссылка = СтрокаДерева.Ссылка;
Если ТипЗнч(Ссылка) = Тип("СправочникСсылка.ВходящиеДокументы") Тогда
ТекстЗапроса =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|	ИсходящиеДокументы.Ссылка КАК Ссылка,
|	ИсходящиеДокументы.Заголовок КАК Заголовок,
|
	ИсходящиеДокументы.РегистрационныйНомер КАК РегистрационныйНомер,
|	ИсходящиеДокументы.ДатаРегистрации КАК ДатаРегистрации,
|	1 КАК ИндексКартинки,
|	ИсходящиеДокументы.ДатаСоздания КАК ДатаСоздания
|ИЗ
|	Справочник.ИсходящиеДокументы КАК ИсходящиеДокументы
|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.СвязиДокументов КАК СвязьПредметПереписки
|		ПО ИсходящиеДокументы.Ссылка = СвязьПредметПереписки.Документ
|			И (СвязьПредметПереписки.ТипСвязи = ЗНАЧЕНИЕ(Справочник.ТипыСвязей.ПредметПереписки))
|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.СвязиДокументов КАК СвязьВОтветНа
|		ПО ИсходящиеДокументы.Ссылка = СвязьВОтветНа.Документ
|			И (СвязьВОтветНа.ТипСвязи = ЗНАЧЕНИЕ(Справочник.ТипыСвязей.ОтправленВОтветНа))
|ГДЕ
|	СвязьВОтветНа.СвязанныйДокумент = &Ссылка
|	И СвязьПредметПереписки.СвязанныйДокумент = &ПредметПереписки
|	И (НЕ ИсходящиеДокументы.ПометкаУдаления)
|
|УПОРЯДОЧИТЬ ПО
|	ДатаСоздания";
ИначеЕсли ТипЗнч(Ссылка) = Тип("СправочникСсылка.ИсходящиеДокументы") Тогда
ТекстЗапроса =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|	ВходящиеДокументы.Ссылка КАК Ссылка,
|	ВходящиеДокументы.Заголовок КАК Заголовок,
|	ВходящиеДокументы.РегистрационныйНомер КАК РегистрационныйНомер,
|	ВходящиеДокументы.ДатаРегистрации КАК ДатаРегистрации,
|	0 КАК ИндексКартинки,
|	ВходящиеДокументы.ДатаСоздания КАК ДатаСоздания
|ИЗ
|	Справочник.ВходящиеДокументы КАК ВходящиеДокументы
|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.СвязиДокументов КАК СвязьПредметПереписки
|		ПО ВходящиеДокументы.Ссылка = СвязьПредметПереписки.Документ
|			И (СвязьПредметПереписки.ТипСвязи = ЗНАЧЕНИЕ(Справочник.ТипыСвязей.ПредметПереписки))
|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.СвязиДокументов КАК СвязьВОтветНа
|		ПО ВходящиеДокументы.Ссылка = СвязьВОтветНа.Документ
|			И (СвязьВОтветНа.ТипСвязи = ЗНАЧЕНИЕ(Справочник.ТипыСвязей.ПолученВОтветНа))
|ГДЕ
|	СвязьВОтветНа.СвязанныйДокумент = &Ссылка
|	И СвязьПредметПереписки.СвязанныйДокумент = &ПредметПереписки
|	И (НЕ ВходящиеДокументы.ПометкаУдаления)
|
|УПОРЯДОЧИТЬ ПО
|	ДатаСоздания";
КонецЕсли;
Запрос = Новый Запрос;
Запрос.Текст = ТекстЗапроса;
Запрос.УстановитьПараметр("Ссылка", Ссылка);
Запрос.УстановитьПараметр("ПредметПереписки", ПараметрыДокумент);
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
НоваяСтрока = СтрокаДерева.Строки.Добавить();
ЗаполнитьЗначенияСвойств(НоваяСтрока, Выборка);
Если ТипЗнч(Ссылка) = Тип("СправочникСсылка.ВходящиеДокументы")  Тогда
Отправлен = ПолучитьПризнакОтправлен(Выборка.Ссылка, Ссылка.Отправитель, Ссылка.Подписал);
НоваяСтрока.ИндексКартинки = ?(Отправлен, 1, 3);
КонецЕсли;
ЗаполнитьПодчиненныеДокументы(НоваяСтрока, ПараметрыДокумент);
КонецЦикла;
КонецПроцедуры
// Заполнить дерево переписки
Функция ЗаполнитьДерево(Дерево, ПараметрыДокумент) Экспорт
Дерево.Строки.Очистить();
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|	ВходящийДокумент.Ссылка КАК Ссылка,
|	ВходящийДокумент.ДатаРегистрации,
|	ВходящийДокумент.РегистрационныйНомер,
|	ВходящийДокумент.Заголовок,
|	ВходящийДокумент.ДатаСоздания КАК ДатаСоздания
|ИЗ
|	Справочник.ВходящиеДокументы КАК ВходящийДокумент
|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.СвязиДокументов КАК СвязьПредметПереписки
|		ПО ВходящийДокумент.Ссылка = СвязьПредметПереписки.Документ
|			И (СвязьПредметПереписки.ТипСвязи = ЗНАЧЕНИЕ(Справочник.ТипыСвязей.ПредметПереписки))
|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.СвязиДокументов КАК СвязьВОтветНа
|		ПО ВходящийДокумент.Ссылка = СвязьВОтветНа.Документ
|			И (СвязьВОтветНа.ТипСвязи = ЗНАЧЕНИЕ(Справочник.ТипыСвязей.ПолученВОтветНа))
|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.СвязиДокументов КАК СвязьВОтветНаПредметПереписки
|		ПО (СвязьВОтветНа.СвязанныйДокумент = СвязьВОтветНаПредметПереписки.Документ)
|			И (СвязьВОтветНаПредметПереписки.ТипСвязи = ЗНАЧЕНИЕ(Справочник.ТипыСвязей.ПредметПереписки))
|ГДЕ
|	СвязьПредметПереписки.СвязанныйДокумент = &ПредметПереписки
|	И (СвязьВОтветНа.СвязанныйДокумент ЕСТЬ NULL
|			ИЛИ ЕСТЬNULL(СвязьВОтветНаПредметПереписки.СвязанныйДокумент, """") <>&ПредметПереписки)
|	И (НЕ ВходящийДокумент.ПометкаУдаления)
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	ИсходящийДокумент.Ссылка,
|	ИсходящийДокумент.ДатаРегистрации,
|	ИсходящийДокумент.РегистрационныйНомер,
|	ИсходящийДокумент.Заголовок,
|	ИсходящийДокумент.ДатаСоздания
|ИЗ
|	Справочник.ИсходящиеДокументы КАК ИсходящийДокумент
|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.СвязиДокументов КАК СвязьПредметПереписки
|		ПО ИсходящийДокумент.Ссылка = СвязьПредметПереписки.Документ
|			И (СвязьПредметПереписки.ТипСвязи = ЗНАЧЕНИЕ(Справочник.ТипыСвязей.ПредметПереписки))
|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.СвязиДокументов КАК СвязьВОтветНа
|		ПО ИсходящийДокумент.Ссылка = СвязьВОтветНа.Документ
```

```
|                                                 И
(СвязьВОтветНа.ТипСвязи =
ЗНАЧЕНИЕ(Справочник.ТипыСвязей.ОтправленВОтветНа)
)
|                                 ЛЕВОЕ СОЕДИНЕНИЕ
РегистрСведений.СвязиДокументов КАК
СвязьВОтветНаПредметПереписки
|                                 ПО
(СвязьВОтветНа.СвязанныйДокумент =
СвязьВОтветНаПредметПереписки.Документ)
|                                                 И
(СвязьВОтветНаПредметПереписки.ТипСвязи =
ЗНАЧЕНИЕ(Справочник.ТипыСвязей.ПредметПереписки))
|ГДЕ
|               СвязьПредметПереписки.СвязанныйДокумент
= &ПредметПереписки
|               И (СвязьВОтветНа.СвязанныйДокумент ЕСТЬ
NULL
|                                                 ИЛИ
ЕСТЬNULL(СвязьВОтветНаПредметПереписки.Связанный
Документ, """") <>&ПредметПереписки)
|               И (НЕ
ИсходящийДокумент.ПометкаУдаления)
|
|УПОРЯДОЧИТЬ ПО
|               ДатаСоздания";

Запрос.УстановитьПараметр("ПредметПереписки",
ПараметрыДокумент);
ТаблДокументов = Запрос.Выполнить().Выгрузить();
ТаблДокументов.Сортировать("ДатаРегистрации");
Для Каждого Строка Из ТаблДокументов Цикл
НоваяСтрока = Дерево.Строки.Добавить();
ЗаполнитьЗначенияСвойств(НоваяСтрока, Строка);
Если ТипЗнч(НоваяСтрока.Ссылка) =
Тип("СправочникСсылка.ВходящиеДокументы") Тогда
НоваяСтрока.ИндексКартинки = 0;
ИначеЕсли ТипЗнч(НоваяСтрока.Ссылка) =
Тип("СправочникСсылка.ИсходящиеДокументы") Тогда
Если НоваяСтрока.Ссылка.Получатели.Найти(Истина,
"Отправлен") <> Неопределено Тогда
НоваяСтрока.ИндексКартинки = 1;
Иначе
НоваяСтрока.ИндексКартинки = 3;
КонецЕсли;
КонецЕсли;
ЗаполнитьПодчиненныеДокументы(НоваяСтрока,
ПараметрыДокумент);
КонецЦикла;
КонецФункции
// Выводит номенклатуру дел в дерево
Процедура ЗаполнитьДеревоНоменклатурыДел(Дерево, Год,
Организация) Экспорт
// получение дерева разделов номенклатуры дел
Запрос = Новый Запрос;
ТекстЗапроса =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|               РазделыНоменклатурыДел.Индекс КАК
Индекс,
|               РазделыНоменклатурыДел.Наименование,
|               РазделыНоменклатурыДел.ПометкаУдаления,
|               РазделыНоменклатурыДел.Ссылка КАК
Ссылка,
|               РазделыНоменклатурыДел.Родитель
|ИЗ
|               Справочник.РазделыНоменклатурыДел КАК
РазделыНоменклатурыДел ";
Условие = "";
Если
ПолучитьФункциональнуюОпцию("ИспользоватьУчетПоОр
ганизациям") И ЗначениеЗаполнено(Организация) Тогда
Условие = Условие + " (Организация = &Организация) И ";
Запрос.УстановитьПараметр("Организация", Организация);
КонецЕсли;
Если ЗначениеЗаполнено(Год) Тогда
Условие = Условие + " (Год = &Год) И ";
Запрос.УстановитьПараметр("Год", Год);
КонецЕсли;
Если Условие <> "" Тогда
ТекстЗапроса = ТекстЗапроса + " ГДЕ " + Лев(Условие,
СтрДлина(Условие)-2);
КонецЕсли;
ТекстЗапроса = ТекстЗапроса +
" ИТОГИ ПО
|               Ссылка ТОЛЬКО ИЕРАРХИЯ
|АВТОУПОРЯДОЧИВАНИЕ";
Запрос.Текст = ТекстЗапроса;
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
Если Не ЗначениеЗаполнено(Выборка.Ссылка) Тогда
Продолжить;
КонецЕсли;
Родитель = Выборка.Родитель;
Если Родитель.Пустая() Тогда
СтрокаРодитель = Дерево;
Иначе
СтрокаРодитель = Дерево.Строки.Найти(Родитель,
"Ссылка", Истина);
КонецЕсли;
НоваяСтрока = СтрокаРодитель.Строки.Добавить();
НоваяСтрока.Ссылка = Выборка.Ссылка;
НоваяСтрока.Наименование = Выборка.Индекс + " " +
Выборка.Наименование;
НоваяСтрока.ИндексКартинки =
?(Выборка.ПометкаУдаления, 1, 0);
НоваяСтрока.ЭтоГруппа = Истина;
КонецЦикла;
// получение списка элементов номенклатуры дел
Запрос = Новый Запрос;
ТекстЗапроса =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|               НоменклатураДел.Индекс,
|               НоменклатураДел.Наименование,
|               НоменклатураДел.ПометкаУдаления,
|               НоменклатураДел.Ссылка КАК Ссылка,
|               НоменклатураДел.Раздел
|ИЗ
|               Справочник.НоменклатураДел КАК
НоменклатураДел ";
Условие = "";
Если
ПолучитьФункциональнуюОпцию("ИспользоватьУчетПоОр
ганизациям") И ЗначениеЗаполнено(Организация) Тогда
Условие = Условие + " (Организация = &Организация) И ";
Запрос.УстановитьПараметр("Организация", Организация);
КонецЕсли;
Если ЗначениеЗаполнено(Год) Тогда
Условие = Условие + " (Год = &Год) И ";
Запрос.УстановитьПараметр("Год", Год);
КонецЕсли;
Если Условие <> "" Тогда
ТекстЗапроса = ТекстЗапроса + " ГДЕ " + Лев(Условие,
СтрДлина(Условие)-2);
КонецЕсли;
ТекстЗапроса = ТекстЗапроса + " УПОРЯДОЧИТЬ ПО
Индекс";
Запрос.Текст = ТекстЗапроса;
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
Если Выборка.Раздел.Пустая() Тогда
СтрокаРодитель = Дерево;
Иначе
СтрокаРодитель = Дерево.Строки.Найти(Выборка.Раздел,
"Ссылка", Истина);
КонецЕсли;
Если СтрокаРодитель <> Неопределено Тогда
НоваяСтрока = СтрокаРодитель.Строки.Добавить();
НоваяСтрока.Ссылка = Выборка.Ссылка;
НоваяСтрока.Наименование = Выборка.Индекс + " " +
Выборка.Наименование;
НоваяСтрока.ИндексКартинки =
?(Выборка.ПометкаУдаления, 3, 2);
НоваяСтрока.ЭтоГруппа = Ложь;
КонецЕсли;
КонецЦикла;
Если Дерево.Строки.Количество() > 0 Тогда
НоваяСтрока = Дерево.Строки.Добавить();
НоваяСтрока.Ссылка =
Справочники.НоменклатураДел.ПустаяСсылка();
НоваяСтрока.Наименование = НСтр("ru = '<Номенклатура
дел не указана>'");
НоваяСтрока.ИндексКартинки = 2;
НоваяСтрока.ЭтоГруппа = Ложь;
КонецЕсли;
КонецПроцедуры
// Выводит дела (тома) в дерево
Процедура ЗаполнитьДеревоДелТомов(Дерево, Год,
Организация) Экспорт
Запрос = Новый Запрос;
ТекстЗапроса =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|               ДелаХраненияДокументов.Ссылка,
|               ДелаХраненияДокументов.Наименование,
|               ДелаХраненияДокументов.ПометкаУдаления,
|
|               ДелаХраненияДокументов.НоменклатураДел.Р
аздел КАК Раздел
|ИЗ
|               Справочник.ДелаХраненияДокументов КАК
ДелаХраненияДокументов";
Условие = "";
Если
ПолучитьФункциональнуюОпцию("ИспользоватьУчетПоОр
ганизациям") И ЗначениеЗаполнено(Организация) Тогда
Условие = Условие + "
(ДелаХраненияДокументов.Организация = &Организация)
И ";
Запрос.УстановитьПараметр("Организация", Организация);
КонецЕсли;
Если ЗначениеЗаполнено(Год) Тогда
Условие = Условие + "
(ДелаХраненияДокументов.НоменклатураДел.Год = &Год)
И ";
Запрос.УстановитьПараметр("Год", Год);
КонецЕсли;
Если Условие <> "" Тогда
ТекстЗапроса = ТекстЗапроса + " ГДЕ " + Лев(Условие,
СтрДлина(Условие)-2);
КонецЕсли;
ТекстЗапроса = ТекстЗапроса +
" УПОРЯДОЧИТЬ ПО
|
|               ДелаХраненияДокументов.НоменклатураДел.
Индекс,
|               ДелаХраненияДокументов.НомерТома";
Запрос.Текст = ТекстЗапроса;
ТаблицаДела = Запрос.Выполнить().Выгрузить();
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|               РазделыНоменклатурыДел.Индекс,
|               РазделыНоменклатурыДел.Наименование,
|               РазделыНоменклатурыДел.ПометкаУдаления,
|               РазделыНоменклатурыДел.Ссылка КАК
Ссылка,
|               РазделыНоменклатурыДел.Родитель
|ИЗ
|               Справочник.РазделыНоменклатурыДел КАК
РазделыНоменклатурыДел
|ГДЕ
|               РазделыНоменклатурыДел.Ссылка
В(&Разделы)
|ИТОГИ ПО
|               Ссылка ТОЛЬКО ИЕРАРХИЯ
|АВТОУПОРЯДОЧИВАНИЕ";
Запрос.УстановитьПараметр("Разделы",
ТаблицаДела.ВыгрузитьКолонку("Раздел"));
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
Если Не ЗначениеЗаполнено(Выборка.Ссылка) Тогда
Продолжить;
КонецЕсли;
Родитель = Выборка.Родитель;
Если Родитель.Пустая() Тогда
СтрокаРодитель = Дерево;
Иначе
СтрокаРодитель = Дерево.Строки.Найти(Родитель,
"Ссылка", Истина);
КонецЕсли;
НоваяСтрока = СтрокаРодитель.Строки.Добавить();
НоваяСтрока.Ссылка = Выборка.Ссылка;
НоваяСтрока.Наименование = Выборка.Индекс + " " +
Выборка.Наименование;
НоваяСтрока.ИндексКартинки =
?(Выборка.ПометкаУдаления, 1, 0);
НоваяСтрока.ЭтоГруппа = Истина;
КонецЦикла;

Для Каждого Строка Из ТаблицаДела Цикл
Если Строка.Раздел.Пустая() Тогда
СтрокаРодитель = Дерево;
Иначе
СтрокаРодитель = Дерево.Строки.Найти(Строка.Раздел,
"Ссылка", Истина);
КонецЕсли;
Если СтрокаРодитель <> Неопределено Тогда
НоваяСтрока = СтрокаРодитель.Строки.Добавить();
НоваяСтрока.Ссылка = Строка.Ссылка;
НоваяСтрока.Наименование = Строка.Наименование;
НоваяСтрока.ИндексКартинки =
?(Строка.ПометкаУдаления, 3, 2);
НоваяСтрока.ЭтоГруппа = Ложь;
КонецЕсли;
КонецЦикла;
Если Дерево.Строки.Количество() > 0 Тогда
НоваяСтрока = Дерево.Строки.Добавить();
НоваяСтрока.Ссылка =
Справочники.ДелаХраненияДокументов.ПустаяСсылка();
НоваяСтрока.Наименование = НСтр("ru = '<Дело (том) не
указано>'");
НоваяСтрока.ИндексКартинки = 2;
НоваяСтрока.ЭтоГруппа = Ложь;
КонецЕсли;
КонецПроцедуры
// Возвращает признак необходимости указания связи для
регистрации документа
Функция
ДляРегистрацииНеобходимоУказатьСвязанныйДокумент(Ф
орма) Экспорт
Если Не
ПолучитьФункциональнуюОпцию("ИспользоватьСвязиДоку
ментов") Тогда
Возврат Ложь;
КонецЕсли;
Если Не ЗначениеЗаполнено(Форма.Нумератор) Тогда
Возврат Ложь;
КонецЕсли;
ПараметрыНумератора =
ОбщегоНазначения.ПолучитьЗначенияРеквизитов(Форма.Н
умератор,
"НезависимаяНумерацияПоСвязанномуДокументу,
ТипСвязи");
Если Не
ПараметрыНумератора.НезависимаяНумерацияПоСвязанно
муДокументу Тогда
Возврат Ложь;
КонецЕсли;
Если ЗначениеЗаполнено(Форма.Объект.Ссылка) Тогда
ДокументДляНумерации =
СвязиДокументов.ПолучитьСвязанныйДокумент(Форма.Об
ъект.Ссылка, ПараметрыНумератора.ТипСвязи);
Иначе
ДокументДляНумерации = Неопределено;
КонецЕсли;
ДокументНеНайден = Не
ЗначениеЗаполнено(ДокументДляНумерации);
Если ДокументНеНайден И Форма.ТипСвязиНумератора <>
ПараметрыНумератора.ТипСвязи Тогда
Форма.ТипСвязиНумератора =
ПараметрыНумератора.ТипСвязи;
```

```
СтруктураОбъекта = Новый Структура;
СтруктураОбъекта.Вставить("ВидДокумента",
Форма.Объект.ВидДокумента);
НастройкиСвязи =
СвязиДокументов.ПолучитьНастройкиСвязи(СтруктураОбъ
екта);
СтрокиНастроекСвязи =
НастройкиСвязи.НайтиСтроки(Новый
Структура("ТипСвязи", Форма.ТипСвязиНумератора));
Форма.ТипыВидыСвязанныхДокументовДляНумерации.Оч
истить();
Для Каждого СтрокаНастроекСвязи из
СтрокиНастроекСвязи Цикл
Строка =
Форма.ТипыВидыСвязанныхДокументовДляНумерации.Доб
авить();
Строка.Тип = СтрокаНастроекСвязи.ТипССылкаНа;
Строка.Вид = СтрокаНастроекСвязи.ССылкаНа;
КонецЦикла;
КонецЕсли;
Возврат ДокументНеНайден;
КонецФункции
// Проверяет возможность изменить вид документа, если у
него имеются связанные документы
Процедура
ПроверкаСвязейПриИзмененииВидаДокумента(Объект,
Отказ) Экспорт
Если
Делопроизводство.ИспользоватьВидыДокументов(Объект.В
идДокумента) И Не Объект.Ссылка.Пустая() Тогда
СтарыйВидДокумента =
ОбщегоНазначения.ПолучитьЗначениеРеквизита(Объект.Сс
ылка, "ВидДокумента");
// если изменен вид документа
Если Объект.ВидДокумента <> СтарыйВидДокумента Тогда
УстановитьПривилегированныйРежим(Истина);
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	СвязиДокументов.СвязанныйДокумент,
|	СвязиДокументов.ТипСвязи
|ИЗ
|	РегистрСведений.СвязиДокументов КАК
СвязиДокументов
|ГДЕ
|	СвязиДокументов.Документ = &Документ";
Запрос.УстановитьПараметр("Документ", Объект.Ссылка);
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
ДокументСтруктура = Новый Структура;
ДокументСтруктура.Вставить("Ссылка", Объект.Ссылка);
ДокументСтруктура.Вставить("ВидДокумента",
Объект.ВидДокумента);
НастройкаСвязи =
СвязиДокументов.ПолучитьНастройкуСвязи(ДокументСтру
ктура, Выборка.СвязанныйДокумент, Выборка.ТипСвязи);
Если НастройкаСвязи = Неопределено Тогда
ТекстСообщения =
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = 'Для документа указана связь ""%1"", которую
нельзя использовать для документов этого вида'"),
Строка(Выборка.ТипСвязи));
ОбщегоНазначенияКлиентСервер.СообщитьПользователю(
ТекстСообщения,,"Объект.ВидДокумента",,Отказ);
КонецЕсли;
КонецЦикла;
КонецЕсли;
КонецЕсли;
КонецПроцедуры
// Открывает форму выбора дела
Процедура ВыбратьДело(ЭлементДело, Параметры)
Экспорт
ПараметрыФормы = Параметры;
ПараметрыФормы.Вставить("РежимВыбора",
"ИзКарточкиДокумента");
ОткрытьФорму("Справочник.ДелаХраненияДокументов.Фо
рмаВыбора", ПараметрыФормы, ЭлементДело);
КонецПроцедуры
// Открывает форму выбора получателя
Процедура ВыбратьПолучателя(ЭлементПолучатель,
РеквизитПолучатель) Экспорт
ПараметрыФормы = Новый Структура("Получатель",
РеквизитПолучатель);
ОткрытьФорму("ОбщаяФорма.ВыборПолучателя",
ПараметрыФормы, ЭлементПолучатель);
КонецПроцедуры
// Выполняет проверки перед закрытием документа
Процедура ПередЗакрытиемДокумента(Объект, Отказ)
Экспорт
Если Не ЗначениеЗаполнено(Объект.Ссылка) Тогда
Возврат;
КонецЕсли;
ТекущийПользователь =
ФайловыеФункцииКлиентПовтИсп.ПолучитьПерсональные
НастройкиРаботыСФайлами().ТекущийПользователь;
ПараметрыФормы = Новый Структура;
ПараметрыФормы.Вставить("СообщениеВопрос",
НСтр("ru = 'Закрыть карточку?'"));
ПараметрыФормы.Вставить("СообщениеЗаголовок",
НСтр("ru = 'Некоторые файлы заняты вами для
редактирования:'"));
ПараметрыФормы.Вставить("Заголовок",
Строка(Объект.Ссылка));
ПараметрыФормы.Вставить("ВладелецФайла",
Объект.Ссылка);
ПараметрыФормы.Вставить("Редактирует",
ТекущийПользователь);
РаботаСФайламиКлиент.ОткрытьДиалогСписокЗанятыхФай
лов(Отказ, ПараметрыФормы);
КонецПроцедуры
// Открывает форму текущего состояния документа
Процедура
ОткрытьТекущееСостояниеДокумента(Документ, Элемент)
Экспорт
Если Не ЗначениеЗаполнено(Документ) Тогда
Предупреждение(НСтр("ru = 'Для доступа к состоянию
элемент необходимо записать!'"));
Возврат;
КонецЕсли;
Ключ =
Делопроизводство.ПолучитьКлючСостоянияДокумента(Док
умент);
Если Ключ = Неопределено Тогда
Возврат;
КонецЕсли;
ПараметрыФормы = Новый Структура("Ключ", Ключ);
ОткрытьФормуМодально("РегистрСведений.СостоянияДоку
ментов.ФормаЗаписи", ПараметрыФормы, Элемент);
КонецПроцедуры
// Открывает форму предупреждения при регистрации
документа
Функция ПредупредитьПриРегистрации() Экспорт
ПоказыватьПредупреждениеПриРегистрации =
ДелопроизводствоКлиентПовтИсп.ПолучитьПерсональные
НастройкиРаботыСДокументами().ПоказыватьПредупрежде
ниеПриРегистрации;
Если ПоказыватьПредупреждениеПриРегистрации = Истина
Тогда
Результат =
ОткрытьФормуМодально("ОбщаяФорма.Пред
упреждениеПриРегистрации");
Если Результат <> КодВозвратаДиалога.Да Тогда
Возврат Ложь;
КонецЕсли;
КонецЕсли;
Возврат Истина;
КонецФункции
// Открывает форму предупреждения при перерегистрации
документа
Функция ПредупредитьПриПеререгистрации() Экспорт
ТекстВопроса = НСтр("ru = 'Документ будет
перерегистрирован. Продолжить?'");
Ответ = Вопрос(ТекстВопроса,
РежимДиалогаВопрос.ДаНет);
Если Ответ <> КодВозвратаДиалога.Да Тогда
Возврат Ложь;
КонецЕсли;
Возврат Истина;
КонецФункции
// Открывает форму записи журнала передачи документа
Процедура
ОткрытьКарточкуПередачиДокумента(Документ,
ВладелецФормы) Экспорт
Количество =
Делопроизводство.КоличествоКомуПереданДокумент(Доку
мент);
Если Количество = 0 Тогда
Возврат;
ИначеЕсли Количество = 1 Тогда
ПараметрыФормы = Новый
Структура("ПоказатьДержателяДокумента", Документ);
ОткрытьФорму("РегистрСведений.ЖурналПередачиДокуме
нтов.ФормаЗаписи", ПараметрыФормы, ВладелецФормы);
Иначе
ПараметрыФормы = Новый
Структура("ПоказатьДержателейДокумента", Документ);
ОткрытьФорму("РегистрСведений.ЖурналПередачиДокуме
нтов.Форма.ФормаСпискаДокумента", ПараметрыФормы,
ВладелецФормы);
КонецЕсли;
КонецПроцедуры
// Показывает форму вопроса с возможностью сохранения
ответа в персональные настройки
// Параметры:
//	ЭлементВладелец - элемент
формы, который будет владельцем открываемой формы с
вопросом
//	Заголовок вопроса - заголовок
для формы вопроса
//	ТекстВопроса - формулировка
вопроса
//
	КлючПерсональнойНастройкиПоказаВопроса
- ключ персональной настройки, хранящей флаг
необходимости показа формы
//
	ИмяПерсональнойНастройкипоказаВопроса -
имя персональной настройки, хранящей флаг
необходимости показа формы
//	СписокДоступныхВариантов -
список доступных вариантов ответов на вопрос
//	ВариантОтветаПоУмолчанию -
вариант ответа, который будет помечен как вариант ответа
по умолчанию
// Возвращает:
//	Значение типа
КодВозвратаДиалога
Функция ПоказатьРасширеннуюФормуВопроса(
ЭлементВладелец,
ЗаголовокВопроса,
ТекстВопроса,
КлючПерсональнойНастройкиПоказаВопроса,
ИмяПерсональнойНастройкиПоказаВопроса,
СписокДоступныхВариантов,
ВариантОтветаПоУмолчанию = "Да") Экспорт
ПараметрыФормы = Новый Структура;
ПараметрыФормы.Вставить("Заголовок",
ЗаголовокВопроса);
Параметрыформы.Вставить("ТекстВопроса",
ТекстВопроса);
Параметрыформы.Вставить("СписокДоступныхВариантов",
СписокДоступныхВариантов);
Параметрыформы.Вставить("КлючПерсональнойНастройки
", КлючПерсональнойНастройкиПоказаВопроса);
Параметрыформы.Вставить("ИмяПерсональнойНастройки",
ИмяПерсональнойНастройкиПоказаВопроса);
ПараметрыФормы.Вставить("ВариантОтветаПоУмолчанию"
, ВариантОтветаПоУмолчанию);
Ответ =
ОткрытьФормуМодально("ОбщаяФорма.РасширенныйВопр
ос", ПараметрыФормы, ЭлементВладелец);
Если НЕ ЗначениеЗаполнено(Ответ) ИЛИ Ответ = Ложь
Тогда
Возврат Ответ;
КонецЕсли;
ЗначениеДляВозврата = Неопределено;
Выполнить("ЗначениеДляВозврата = КодВозвратаДиалога."
+ Ответ);
Возврат ЗначениеДляВозврата;
КонецФункции
// Предлагает записать текущий документ
// Возвращает:
//	Истина - Документ удачно записан,
//	Ложь - отказ от записи или не удалось записать.
//
Функция ПредложитьЗаписатьДокумент(Форма, Текст)
Экспорт
Режим = РежимДиалогаВопрос.ДаНет;
Ответ = Вопрос(Текст, РежимДиалогаВопрос.ДаНет, 0);
Если Ответ <> КодВозвратаДиалога.Да Тогда
Возврат Ложь;
КонецЕсли;
Результат = Форма.Записать();
Если Результат Тогда
ПоказатьОповещениеПользователя(
"Создание:",
ПолучитьНавигационнуюССылку(Форма.Объект.Ссылка),
Строка(Форма.Объект.Ссылка),
БиблиотекаКартинок.Информация32);
КонецЕсли;
Возврат Результат;
КонецФункции
// Копирует файл из временного хранилища на клиента и
открывает его для просмотра
//
Процедура
ОткрытьФайлИзВременногоХранилища(АдресВоВременно
мХранилище, ИмяФайла) Экспорт
Если Не РасширениеРаботыСФайламиПодключено() Тогда
Предупреждение(НСтр("ru = 'Не удалось подключить
расширение работы с файлами.'"));
Возврат;
КонецЕсли;
ДвоичныеДанные =
ПолучитьИзВременногоХранилища(АдресВоВременномХра
нилище);
Если ДвоичныеДанные = Неопределено Тогда
Предупреждение(НСтр("ru = 'Не удалось получить файл.
Возможно он был удален.'"));
Возврат;
КонецЕсли;
ПолноеИмяВременногоФайла =
ПолучитьПолноеИмяВременногоФайла(ИмяФайла);
Если ПустаяСтрока(ПолноеИмяВременногоФайла) Тогда
Предупреждение(НСтр("ru = 'Не удалось создать временный
файл.'"));
Возврат;
КонецЕсли;
Попытка
ДвоичныеДанные.Записать(ПолноеИмяВременногоФайла);
Исключение
СообщениеОбОшибке = ОписаниеОшибки();
Предупреждение(СтроковыеФункцииКлиентСервер.Подстав
итьПараметрыВСтроку(
НСтр("ru = 'Не удалось записать временный файл.
|%1'"),
СообщениеОбОшибке));
Возврат;
КонецПопытки;
ОткрытьФайлНаДиске(ПолноеИмяВременногоФайла,
ИмяФайла);
КонецПроцедуры
Процедура ОткрытьФайлНаДиске(ПолноеИмяФайла,
ИмяФайла) Экспорт
Попытка
РаботаСФайламиКлиент.ЗапуститьПриложениеПоИмениФа
йла(ПолноеИмяФайла);
Исключение
Предупреждение(СтроковыеФункцииКлиентСервер.Подстав
итьПараметрыВСтроку(
НСтр("ru = 'Не удалось открыть файл %1
```

```
|%2""),
ИмяФайла,
ОписаниеОшибки()));
Возврат;
КонецПопытки;
КонецПроцедуры
Функция РасширениеРаботыСФайламиПодключено()
Если Не ПодключитьРасширениеРаботыСФайлами() Тогда
Предупреждение(НСтр("ru = 'Не подключено расширение
работы с файлами!'"));
Возврат Ложь;
КонецЕсли;
Возврат Истина;;
КонецФункции
Функция
ПолучитьПолноеИмяВременногоФайла(ИмяФайла) Экспорт
#Если ВебКлиент Тогда
Возврат "";
#Иначе
Возврат ПолучитьИмяВременногоКаталога() + "\" +
ИмяФайла;
#КонецЕсли
КонецФункции
#Если Не ВебКлиент Тогда
Функция ПолучитьИмяВременногоКаталога()
ИмяВременногоКаталога =
ПолучитьИмяВременногоФайла("");
СоздатьКаталог(ИмяВременногоКаталога);
Возврат ИмяВременногоКаталога;
КонецФункции
#КонецЕсли
// Открывает меню выбора папки и сохраняет файл из
временного хранилища в указанную папку
// Возвращает ссылку на файл или Неопределено в случае
неуспеха
//
Функция СоздатьФайлИзВременногоХранилища(Форма,
АдресВоВременномХранилище, ИмяФайла) Экспорт
Папка = Неопределено;
Если Не ВыбратьПапку(Папка) Тогда
Возврат Неопределено;
КонецЕсли;
ПолноеИмяФайла =
ПолучитьПолноеИмяВременногоФайла(ИмяФайла);
Попытка
ДвоичныеДанные =
ПолучитьИзВременногоХранилища(АдресВоВременномХра
нилище);
ДвоичныеДанные.Записать(ПолноеИмяФайла);
Файл =
РаботаСФайламиКлиент.СоздатьДокументНаОсновеФайла(
ПолноеИмяФайла,
Папка,
Форма,
Ложь, //НеОткрыватьКарточкуПослеСозданияИзФайла
ИмяФайла);
Исключение
Текст = НСтр("ru = 'Не удалось сохранить файл.'") +
Символы.ПС + ОписаниеОшибки();
Предупреждение(Текст);
Возврат Неопределено;
КонецПопытки;
Возврат Файл;
КонецФункции
// Открывает меню выбора папки и сохраняет файл из
временного хранилища в указанную папку
// Возвращает ссылку на файл или Неопределено в случае
неуспеха
//
Функция СоздатьФайлИзВременногоФайлаНаДиске(Форма,
ПолноеИмяФайла, ИмяФайла) Экспорт
Папка = Неопределено;
Если Не ВыбратьПапку(Папка) Тогда
Возврат Неопределено;
КонецЕсли;
Попытка
Файл =
РаботаСФайламиКлиент.СоздатьДокументНаОсновеФайла(
ПолноеИмяФайла,
Папка,
Форма,
Ложь, //НеОткрыватьКарточкуПослеСозданияИзФайла
ИмяФайла);
Исключение
Текст = НСтр("ru = 'Не удалось сохранить файл.'") +
Символы.ПС + ОписаниеОшибки();
Предупреждение(Текст);
Возврат Неопределено;
КонецПопытки;
Возврат Файл;
КонецФункции
Функция ВыбратьПапку(Папка)
Результат =
ОткрытьФормуМодально("Справочник.ПапкиФайлов.Форм
аВыбора");
Если Результат = Неопределено Или Результат.Пустая()
Тогда
Возврат Ложь;
КонецЕсли;
Папка = Результат;
Возврат Истина;
КонецФункции
Процедура НайтиСтрокуДереваПоСсылке(Ссылка, Дерево,
Идентификатор) Экспорт
Если Идентификатор <> Неопределено Тогда
Возврат;
КонецЕсли;
Для Каждого Строка Из Дерево.ПолучитьЭлементы() Цикл
Если Строка.Ссылка = Ссылка Тогда
Идентификатор = Строка.ПолучитьИдентификатор();
Прервать;
КонецЕсли;
НайтиСтрокуДереваПоСсылке(Ссылка, Строка,
Идентификатор);
КонецЦикла;
КонецПроцедуры
Функция ВвестиСтрокуСЗаголовком(Значение, Заголовок
= "", Надпись = "", Длина = 0) Экспорт
ПараметрыФормы = Новый Структура;
ПараметрыФормы.Вставить("Значение", Значение);
ПараметрыФормы.Вставить("Заголовок", Заголовок);
ПараметрыФормы.Вставить("Надпись", Надпись);
ПараметрыФормы.Вставить("Длина", Длина);
Результат =
ОткрытьФормуМодально("ОбщаяФорма.ВводСтрокиСЗагол
овком", ПараметрыФормы);
Если ТипЗнч(Результат) = Тип("Строка") И
ЗначениеЗаполнено(Результат) Тогда
Значение = СокрЛП(Результат);
Возврат Истина;
КонецЕсли;
Возврат Ложь;
КонецФункции
// Обработчик подписки на событие
ЗаписатьОбщиеРеквизитыДокументов.
//
Процедура ЗаписатьОбщиеРеквизитыДокументов(Источник,
Отказ) Экспорт
Если Источник.ОбменДанными.Загрузка Тогда
Возврат;
КонецЕсли;
УстановитьПривилегированныйРежим(Истина);
МенеджерЗаписи =
РегистрыСведений.ОбщиеРеквизитыДокументов.СоздатьМе
неджерЗаписи();
МенеджерЗаписи.Документ = Источник.Ссылка;
МенеджерЗаписи.Прочитать();
МенеджерЗаписи.Документ
        = Источник.Ссылка;
МенеджерЗаписи.РегистрационныйНомер =
Источник.РегистрационныйНомер;
МенеджерЗаписи.ДатаРегистрации                    =
Источник.ДатаРегистрации;
МенеджерЗаписи.Сумма
        = Источник.Сумма;
МенеджерЗаписи.Валюта
        = Источник.Валюта;
МенеджерЗаписи.СрокИсполнения                     =
Источник.СрокИсполнения;
МенеджерЗаписи.Заголовок
        = Источник.Заголовок;
МенеджерЗаписи.Записать();
КонецПроцедуры
////////////////////////////////////////////////////////////////////////////////
// Программный интерфейс
// Возвращает структуру прав доступа к указанному объекту
для указанного
// пользователя. Если пользователь не указан, то
используется текущий
// пользователь.
Функция ПолучитьПраваПоОбъекту(ОбъектДоступа,
Пользователь = Неопределено) Экспорт
// Проверка на использование ограничения прав доступа
Если Не
ДокументооборотПраваДоступаПовтИсп.ВключеноИспольз
ованиеПравДоступа() Тогда
// Если права доступа не включены, то все разрешено
Права = Новый Структура(
"Добавление, Изменение, Удаление, УправлениеПравами,
Чтение",
Истина, Истина, Истина, Истина, Истина);
Возврат Права;
КонецЕсли;
// Если использование прав доступа включено, то
выполняется получение прав доступа
УстановитьПривилегированныйРежим(Истина);
Если Не ЗначениеЗаполнено(Пользователь) Тогда
Пользователь = ОбщегоНазначения.ТекущийПользователь();
КонецЕсли;
Права = Новый Структура(
"Добавление, Изменение, Удаление, УправлениеПравами,
Чтение",
Ложь, Ложь, Ложь, Ложь, Ложь);
// Роли Полные права все разрешено
Если
ПользователиСерверПовтИсп.ЭтоПолноправныйПользовате
льИБ(Пользователь) Тогда
Права.Чтение = Истина;
Права.Добавление = Истина;
Права.Изменение = Истина;
Права.Удаление = Истина;
Права.УправлениеПравами = Истина;
Возврат Права;
КонецЕсли;
Запрос = Новый Запрос;
Если ТипЗнч(ОбъектДоступа) =
Тип("СправочникСсылка.ВерсииФайлов")
Или ТипЗнч(ОбъектДоступа) =
Тип("СправочникСсылка.Файлы") Тогда
ВладелецФайла = Неопределено;
Если ТипЗнч(ОбъектДоступа) =
Тип("СправочникСсылка.Файлы") Тогда
ВладелецФайла =
ОбщегоНазначения.ПолучитьЗначениеРеквизита(ОбъектДос
тупа, "ВладелецФайла");
КонецЕсли;
Если ТипЗнч(ОбъектДоступа) =
Тип("СправочникСсылка.ВерсииФайлов") Тогда
Файл =
ОбщегоНазначения.ПолучитьЗначениеРеквизита(ОбъектДос
тупа, "Владелец");
ВладелецФайла =
ОбщегоНазначения.ПолучитьЗначениеРеквизита(Файл,
"ВладелецФайла");
КонецЕсли;
Запрос.Текст =
"ВЫБРАТЬ
|        ПраваПоДескрипторамДоступа.Добавление,
|        ПраваПоДескрипторамДоступа.Изменение,
|        ПраваПоДескрипторамДоступа.Удаление,
|
        ПраваПоДескрипторамДоступа.УправлениеПр
авами,
|        ПраваПоДескрипторамДоступа.Чтение
|ИЗ
|
        РегистрСведений.ПраваПоДескрипторамДосту
па КАК ПраваПоДескрипторамДоступа
|            ВНУТРЕННЕЕ СОЕДИНЕНИЕ
РегистрСведений.ДескрипторыДоступаДляФайлов КАК
ДескрипторыДоступаДляФайлов
|            ПО
ПраваПоДескрипторамДоступа.Дескриптор =
ДескрипторыДоступаДляФайлов.Дескриптор
|ГДЕ
|        ПраваПоДескрипторамДоступа.Пользователь
= &Пользователь
|        И
ДескрипторыДоступаДляФайлов.ВладелецФайла =
&ВладелецФайла";
Запрос.УстановитьПараметр("ВладелецФайла",
ВладелецФайла);
Иначе
Запрос.Текст =
"ВЫБРАТЬ
|        ПраваПоДескрипторамДоступа.Добавление,
|        ПраваПоДескрипторамДоступа.Изменение,
|        ПраваПоДескрипторамДоступа.Удаление,
|
        ПраваПоДескрипторамДоступа.УправлениеПр
авами,
|        ПраваПоДескрипторамДоступа.Чтение
|ИЗ
|
        РегистрСведений.ПраваПоДескрипторамДосту
па КАК ПраваПоДескрипторамДоступа
|            ВНУТРЕННЕЕ СОЕДИНЕНИЕ
РегистрСведений.ДескрипторыДоступаДляОбъектов КАК
ДескрипторыДоступаДляОбъектов
|            ПО
ПраваПоДескрипторамДоступа.Дескриптор =
ДескрипторыДоступаДляОбъектов.Дескриптор
|ГДЕ
|        ПраваПоДескрипторамДоступа.Пользователь
= &Пользователь
|        И ДескрипторыДоступаДляОбъектов.Объект =
&ОбъектДоступа";
Запрос.УстановитьПараметр("ОбъектДоступа",
ОбъектДоступа.Ссылка);
КонецЕсли;
Запрос.УстановитьПараметр("Пользователь",
Пользователь);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Если ВыборкаДетальныеЗаписи.Следующий() Тогда
Права.Добавление = ВыборкаДетальныеЗаписи.Добавление;
Права.Изменение = ВыборкаДетальныеЗаписи.Изменение;
Права.Удаление = ВыборкаДетальныеЗаписи.Удаление;
Права.УправлениеПравами =
ВыборкаДетальныеЗаписи.УправлениеПравами;
Права.Чтение = ВыборкаДетальныеЗаписи.Чтение;
КонецЕсли;
РасширитьДоступПравамиБезОграничения(Права,
ОбъектДоступа, Пользователь);
Возврат Права;
КонецФункции
// Обновляет права доступа по указанному дескриптору
Процедура
ОбновитьПраваДоступаПоДескриптору(Дескриптор,
Немедленно = Неопределено) Экспорт
УстановитьПривилегированныйРежим(Истина);
Менеджер =
ОбщегоНазначения.МенеджерОбъектаПоСсылке(Дескрипто
р);
Менеджер.ОбновитьПрава(
Дескриптор,
Неопределено, // Протокол
Немедленно);
КонецПроцедуры
// Расширяет переданное соответствие с правами доступа,
добавляя к нему
```

```
// руководителей тех пользователей, которые уже перечислены в этом соответствии
//
// Если руководители были добавлены, то возвращает Истина
Функция РасширитьПраваПоРуководителям(ПраваДоступа) Экспорт
УстановитьПривилегированныйРежим(Истина);
РуководителиДобавлены = Ложь;
// Расширяем с учетом передачи прав руководителям
Если
Константы.ДобавлятьРуководителямДоступПодчиненных.Получить() Тогда
ВсеРуководители =
РегистрыСведений.ПодчиненностьСотрудников.ПолучитьВсехРуководителей();
Для каждого Запись Из ПраваДоступа Цикл
// Получение всех руководителей пользователя
Руководители = ВсеРуководители.Получить(Запись.Ключ);
Если Руководители <> Неопределено Тогда
Ст = Новый Структура("Чтение, Добавление, Изменение, Удаление, УправлениеПравами");
ЗаполнитьЗначенияСвойств(Ст, Запись.Значение);
// Обход и добавление каждого руководителя
Для каждого Эл Из Руководители Цикл
ДобавитьЗаписьВСоответствиеПрав(ПраваДоступа, Эл, Ст);
РуководителиДобавлены = Истина;
КонецЦикла;
КонецЕсли;
КонецЦикла;
КонецЕсли;
Возврат РуководителиДобавлены;
КонецФункции
// Расширяет переданное соответствие с правами доступа, добавляя к нему
// делегатов тех пользователей, которые уже перечислены в этом соответствии
//
// Если делегаты были добавлены, то возвращает Истина
Функция РасширитьПраваПоДелегатам(ПраваДоступа, ДескрипторДоступа) Экспорт
Возврат
ДокументооборотПраваДоступаПереопределяемый.РасширитьПраваПоДелегатам(ПраваДоступа, ДескрипторДоступа);
КонецФункции
// Добавляет сведения о правах пользователя в соответствие, объединяя их с уже
// существующими сведениями для данного пользователя
Процедура
ДобавитьЗаписьВСоответствиеПрав(ПраваДоступа,
Пользователь, Знач Права) Экспорт
Эл = ПраваДоступа.Получить(Пользователь);
Если Эл <> Неопределено Тогда
Эл.Чтение = Эл.Чтение ИЛИ Права.Чтение;
Эл.Добавление = Эл.Добавление ИЛИ Права.Добавление;
Эл.Изменение = Эл.Изменение ИЛИ Права.Изменение;
Эл.Удаление = Эл.Удаление ИЛИ Права.Удаление;
Эл.УправлениеПравами = Эл.УправлениеПравами ИЛИ
Права.УправлениеПравами;
Иначе
Эл = Новый Структура("Чтение, Добавление, Изменение, Удаление, УправлениеПравами");
ЗаполнитьЗначенияСвойств(Эл, Права);
КонецЕсли;
ПраваДоступа.Вставить(Пользователь, Эл);
КонецПроцедуры
// Добавляет указанные права указанного пользователя или роли к соответсвию прав,
// производит автоматическое разыменование роли до конкретных пользователей
Процедура ДобавитьПользователяВСоответствиеПрав(
ПраваДоступа,
Пользователь,
ОсновнойОбъектАдресации,
ДополнительныйОбъектАдресации,
Знач Права) Экспорт
// Если пользователь не указан, то запись не добавляется
Если Не ЗначениеЗаполнено(Пользователь) Тогда
Возврат;
КонецЕсли;
Если ТипЗнч(Пользователь) =
Тип("СправочникСсылка.Пользователи") Тогда
РеквизитыПользователя =
ОбщегоНазначения.ПолучитьЗначенияРеквизитов(Пользователь,
"ПометкаУдаления, Недействителен");
Если Не РеквизитыПользователя.ПометкаУдаления И Не
РеквизитыПользователя.Недействителен Тогда
ДобавитьЗаписьВСоответствиеПрав(ПраваДоступа,
Пользователь, Права);
КонецЕсли;
ИначеЕсли ТипЗнч(Пользователь) =
Тип("СправочникСсылка.РолиИсполнителей") Тогда
ИсполнителиРоли =
РегистрыСведений.ИсполнителиЗадач.ПолучитьИсполнителейРоли(
Пользователь,
ОсновнойОбъектАдресации,
ДополнительныйОбъектАдресации);
Для каждого Эл Из ИсполнителиРоли Цикл
ДобавитьЗаписьВСоответствиеПрав(ПраваДоступа,
Эл.Исполнитель, Права);
КонецЦикла;
ИначеЕсли ТипЗнч(Пользователь) =
Тип("СправочникСсылка.ГруппыПользователей") Тогда
ПользователиГруппы =
ДокументооборотПраваДоступаПовтИсп.ПолучитьСоставГруппыПользователей(Пользователь);
Для каждого Эл Из ПользователиГруппы Цикл
ДобавитьЗаписьВСоответствиеПрав(ПраваДоступа,
Эл.Пользователь, Права);
КонецЦикла;
Иначе
ВызватьИсключение(НСтр("ru = 'Неизвестный тип пользователя.'"));
КонецЕсли;
КонецПроцедуры
// Определяет дескриптор доступа для указанного объекта доступа
// Пр необходимости создает новый дескриптор, привязывает его к
// объекту и вычисляет права
Функция
ОпределитьДескрипторДоступаОбъекта(ОбъектДоступа,
Немедленно = Ложь) Экспорт
УстановитьПривилегированныйРежим(Истина);
// Получение ссылки на подходящий дескриптор
Дескриптор = ПолучитьДескрипторДоступа(ОбъектДоступа,
Немедленно);
// Сохранение соответствия объекта доступа и его дескриптора
РегистрыСведений.ДескрипторыДоступаДляОбъектов.Сохранить(Дескриптор.Ссылка, ОбъектДоступа.Ссылка);
Возврат Дескриптор;
КонецФункции
// Определяет дескриптор доступа для указанного владельца файла
// При необходимости создает новый дескриптор, привязывает его к дескриптору
// владельца файла и вычисляет права
Функция
ОпределитьДескрипторДоступаФайлаПоВладельцу(ВладелецФайла) Экспорт
УстановитьПривилегированныйРежим(Истина);
// Получение ссылки на подходящий дескриптор
Дескриптор =
ПолучитьДескрипторДоступаФайла(ВладелецФайла);
Возврат Дескриптор;
КонецФункции
// Определяет дескриптор(ы) доступа для указанного набора записей
// При необходимости создает новый дескриптор и вычисляет права
Функция
ОпределитьДескрипторДоступаНабораЗаписей(НаборЗаписей, Немедленно = Ложь) Экспорт
УстановитьПривилегированныйРежим(Истина);
МетаданныеНабораЗаписей = НаборЗаписей.Метаданные();
ИдентификаторОбъектаМетаданных =
ОбщегоНазначения.ИдентификаторОбъектаМетаданных(МетаданныеНабораЗаписей);
Для каждого Запись Из НаборЗаписей Цикл
Дескриптор =
ПолучитьДескрипторДоступаЗаписиНабораЗаписей(
ИдентификаторОбъектаМетаданных,
Запись,
Немедленно);
КонецЦикла;
Возврат Дескриптор;
КонецФункции
// Копирует права указанного объекта в указанные ПраваДоступа
Процедура СкопироватьПраваОбъекта(ОбъектДоступа,
ПраваДоступа) Экспорт
УстановитьПривилегированныйРежим(Истина);
Для Каждого Эл из ПраваДоступа Цикл
ВызватьИсключение НСтр("ru = 'Для копирования прав объекта указан не пустой набор прав.'");
КонецЦикла;
ДескрипторДоступа =
РегистрыСведений.ДескрипторыДоступаДляОбъектов.НайтиДескрипторДляОбъекта(ОбъектДоступа);
Если ЗначениеЗаполнено(ДескрипторДоступа) Тогда
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ПраваПоДескрипторамДоступа.Добавление,
|	ПраваПоДескрипторамДоступа.Изменение,
|	ПраваПоДескрипторамДоступа.Удаление,
|
|	ПраваПоДескрипторамДоступа.УправлениеПравами,
|	ПраваПоДескрипторамДоступа.Чтение,
|	ПраваПоДескрипторамДоступа.Пользователь
|ИЗ
|
|	РегистрСведений.ПраваПоДескрипторамДоступа КАК ПраваПоДескрипторамДоступа
|ГДЕ
|	ПраваПоДескрипторамДоступа.Дескриптор = &Дескриптор";
Запрос.УстановитьПараметр("Дескриптор", ДескрипторДоступа);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
ПраваПользователя = Новый Структура("Чтение, Добавление, Изменение, Удаление, УправлениеПравами");
ПраваПользователя.Чтение =
ВыборкаДетальныеЗаписи.Чтение;
ПраваПользователя.Добавление =
ВыборкаДетальныеЗаписи.Изменение; // По умолчанию право добавления не отличается от права изменения
ПраваПользователя.Изменение =
ВыборкаДетальныеЗаписи.Изменение;
ПраваПользователя.Удаление =
ВыборкаДетальныеЗаписи.Изменение; // По умолчанию право удаления не отличается от права изменения
ПраваПользователя.УправлениеПравами =
ВыборкаДетальныеЗаписи.УправлениеПравами;
ПраваДоступа.Вставить(ВыборкаДетальныеЗаписи.Пользователь, ПраваПользователя);
КонецЦикла;
КонецЕсли;
КонецПроцедуры
// Копирует права указанного дескриптора в указанные ПраваДоступа
Процедура
СкопироватьПраваДескриптора(ДескрипторДоступа,
ПраваДоступа) Экспорт
УстановитьПривилегированныйРежим(Истина);
Для Каждого Эл из ПраваДоступа Цикл
ВызватьИсключение НСтр("ru = 'Для копирования прав объекта указан не пустой набор прав.'");
КонецЦикла;
Если ЗначениеЗаполнено(ДескрипторДоступа) Тогда
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ПраваПоДескрипторамДоступа.Добавление,
|	ПраваПоДескрипторамДоступа.Изменение,
|	ПраваПоДескрипторамДоступа.Удаление,
|
|	ПраваПоДескрипторамДоступа.УправлениеПравами,
|	ПраваПоДескрипторамДоступа.Чтение,
|	ПраваПоДескрипторамДоступа.Пользователь
|ИЗ
|
|	РегистрСведений.ПраваПоДескрипторамДоступа КАК ПраваПоДескрипторамДоступа
|ГДЕ
|	ПраваПоДескрипторамДоступа.Дескриптор = &Дескриптор";
Запрос.УстановитьПараметр("Дескриптор", ДескрипторДоступа);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
ПраваПользователя = Новый Структура("Чтение, Добавление, Изменение, Удаление, УправлениеПравами");
ПраваПользователя.Чтение =
ВыборкаДетальныеЗаписи.Чтение;
ПраваПользователя.Добавление =
ВыборкаДетальныеЗаписи.Изменение; // По умолчанию право добавления не отличается от права изменения
ПраваПользователя.Изменение =
ВыборкаДетальныеЗаписи.Изменение;
ПраваПользователя.Удаление =
ВыборкаДетальныеЗаписи.Изменение; // По умолчанию право удаления не отличается от права изменения
ПраваПользователя.УправлениеПравами =
ВыборкаДетальныеЗаписи.УправлениеПравами;
ПраваДоступа.Вставить(ВыборкаДетальныеЗаписи.Пользователь, ПраваПользователя);
КонецЦикла;
КонецЕсли;
КонецПроцедуры
// Умножает структуру прав доступа для указанных объектов для всех пользователей
Функция УмножитьПраваОбъектов(ОбъектыДоступа)
Экспорт
УстановитьПривилегированныйРежим(Истина);
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ПраваПоДескрипторамДоступа.Добавление,
|	ПраваПоДескрипторамДоступа.Изменение,
|	ПраваПоДескрипторамДоступа.Удаление,
|
|	ПраваПоДескрипторамДоступа.УправлениеПравами,
|	ПраваПоДескрипторамДоступа.Чтение,
|	ПраваПоДескрипторамДоступа.Пользователь
|ПОМЕСТИТЬ ПраваВременная
|ИЗ
|
|	РегистрСведений.ПраваПоДескрипторамДоступа КАК ПраваПоДескрипторамДоступа
|		ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрСведений.ДескрипторыДоступаДляОбъектов КАК ДескрипторыДоступаДляОбъектов
|		ПО
|ПраваПоДескрипторамДоступа.Дескриптор = ДескрипторыДоступаДляОбъектов.Дескриптор
|ГДЕ
|	ДескрипторыДоступаДляОбъектов.Объект В(&ОбъектыДоступа)
|
|ИНДЕКСИРОВАТЬ ПО
```

```
|		ПраваПоДескрипторамДоступа.Пользователь
|;
|
|////////////////////////////////////////////////////////////////////////////////
|ВЫБРАТЬ
|		ПраваВременная.Пользователь,
|		КОЛИЧЕСТВО(ПраваВременная.Добавление) КАК ЧислоРазрешенных
|ПОМЕСТИТЬ ПраваДобавлениеВременная
|ИЗ
|		ПраваВременная КАК ПраваВременная
|ГДЕ
|		ПраваВременная.Добавление
|
|СГРУППИРОВАТЬ ПО
|		ПраваВременная.Пользователь
|
|ИМЕЮЩИЕ
|		КОЛИЧЕСТВО(ПраваВременная.Добавление) = &КоличествоЗаписей
|;
|
|////////////////////////////////////////////////////////////////////////////////
|ВЫБРАТЬ
|		ПраваВременная.Пользователь,
|		КОЛИЧЕСТВО(ПраваВременная.Изменение) КАК ЧислоРазрешенных
|ПОМЕСТИТЬ ПраваИзменениеВременная
|ИЗ
|		ПраваВременная КАК ПраваВременная
|ГДЕ
|		ПраваВременная.Изменение
|
|СГРУППИРОВАТЬ ПО
|		ПраваВременная.Пользователь
|
|ИМЕЮЩИЕ
|		КОЛИЧЕСТВО(ПраваВременная.Изменение) = &КоличествоЗаписей
|;
|
|////////////////////////////////////////////////////////////////////////////////
|ВЫБРАТЬ
|		ПраваВременная.Пользователь,
|		КОЛИЧЕСТВО(ПраваВременная.Удаление) КАК ЧислоРазрешенных
|ПОМЕСТИТЬ ПраваУдалениеВременная
|ИЗ
|		ПраваВременная КАК ПраваВременная
|ГДЕ
|		ПраваВременная.Удаление
|
|СГРУППИРОВАТЬ ПО
|		ПраваВременная.Пользователь
|
|ИМЕЮЩИЕ
|		КОЛИЧЕСТВО(ПраваВременная.Удаление) = &КоличествоЗаписей
|;
|
|////////////////////////////////////////////////////////////////////////////////
|ВЫБРАТЬ
|		ПраваВременная.Пользователь,
|
|		КОЛИЧЕСТВО(ПраваВременная.Управление Правами) КАК ЧислоРазрешенных
|ПОМЕСТИТЬ ПраваУправлениеПравамиВременная
|ИЗ
|		ПраваВременная КАК ПраваВременная
|ГДЕ
|		ПраваВременная.УправлениеПравами
|
|СГРУППИРОВАТЬ ПО
|		ПраваВременная.Пользователь
|
|ИМЕЮЩИЕ
|
|		КОЛИЧЕСТВО(ПраваВременная.Управление Правами) = &КоличествоЗаписей
|;
|
|////////////////////////////////////////////////////////////////////////////////
|ВЫБРАТЬ
|		ПраваВременная.Пользователь,
|		КОЛИЧЕСТВО(ПраваВременная.Чтение) КАК ЧислоРазрешенных
|ПОМЕСТИТЬ ПраваЧтениеВременная
|ИЗ
|		ПраваВременная КАК ПраваВременная
|ГДЕ
|		ПраваВременная.Чтение
|
|СГРУППИРОВАТЬ ПО
|		ПраваВременная.Пользователь
|
|ИМЕЮЩИЕ
|		КОЛИЧЕСТВО(ПраваВременная.Чтение) = &КоличествоЗаписей
|;
|
|////////////////////////////////////////////////////////////////////////////////
|ВЫБРАТЬ РАЗЛИЧНЫЕ
|		ПраваВременная.Пользователь,
|		ВЫБОР
|			КОГДА ПраваДобавлениеВременная.Пользователь ЕСТЬ NULL
|				ТОГДА ЛОЖЬ
|			ИНАЧЕ ИСТИНА
|		КОНЕЦ КАК Добавление,
|		ВЫБОР
|			КОГДА ПраваИзменениеВременная.Пользователь ЕСТЬ NULL
|				ТОГДА ЛОЖЬ
|			ИНАЧЕ ИСТИНА
|		КОНЕЦ КАК Изменение,
|		ВЫБОР
|			КОГДА ПраваУправлениеПравамиВременная.Пользователь ЕСТЬ NULL
|				ТОГДА ЛОЖЬ
|			ИНАЧЕ ИСТИНА
|		КОНЕЦ КАК Удаление,
|		ВЫБОР
|			КОГДА ПраваУдалениеВременная.Пользователь ЕСТЬ NULL
|				ТОГДА ЛОЖЬ
|			ИНАЧЕ ИСТИНА
|		КОНЕЦ КАК УправлениеПравами,
|		ВЫБОР
|			КОГДА ПраваЧтениеВременная.Пользователь ЕСТЬ NULL
|				ТОГДА ЛОЖЬ
|			ИНАЧЕ ИСТИНА
|		КОНЕЦ КАК Чтение
|ИЗ
|		ПраваВременная КАК ПраваВременная
|			ЛЕВОЕ СОЕДИНЕНИЕ ПраваДобавлениеВременная КАК ПраваДобавлениеВременная
|			ПО (ПраваДобавлениеВременная.Пользователь = ПраваВременная.Пользователь)
|			ЛЕВОЕ СОЕДИНЕНИЕ ПраваИзменениеВременная КАК ПраваИзменениеВременная
|			ПО (ПраваИзменениеВременная.Пользователь = ПраваВременная.Пользователь)
|			ЛЕВОЕ СОЕДИНЕНИЕ ПраваУдалениеВременная КАК ПраваУдалениеВременная
|			ПО (ПраваУдалениеВременная.Пользователь = ПраваВременная.Пользователь)
|			ЛЕВОЕ СОЕДИНЕНИЕ ПраваУправлениеПравамиВременная КАК ПраваУправлениеПравамиВременная
|			ПО (ПраваУправлениеПравамиВременная.Пользователь = ПраваВременная.Пользователь)
|			ЛЕВОЕ СОЕДИНЕНИЕ ПраваЧтениеВременная КАК ПраваЧтениеВременная
|			ПО (ПраваЧтениеВременная.Пользователь = ПраваВременная.Пользователь)
|ГДЕ
|		(НЕ ПраваДобавлениеВременная.Пользователь ЕСТЬ NULL
|			ИЛИ НЕ ПраваИзменениеВременная.Пользователь ЕСТЬ NULL
|			ИЛИ НЕ ПраваУправлениеПравамиВременная.Пользователь ЕСТЬ NULL
|			ИЛИ НЕ ПраваУдалениеВременная.Пользователь ЕСТЬ NULL
|			ИЛИ НЕ ПраваЧтениеВременная.Пользователь ЕСТЬ NULL )";
Запрос.УстановитьПараметр("ОбъектыДоступа", ОбъектыДоступа);
Запрос.УстановитьПараметр("КоличествоЗаписей", ОбъектыДоступа.Количество());
Результат = Запрос.Выполнить().Выгрузить();
Возврат Результат;
КонецФункции
// Удаляет все данные, связанные с правами доступа
// Используется для полного обновления прав доступа ко всем данным информационной базы
Процедура УдалитьВсеДанныеОПравахДоступа(ЭтоФоновоеЗадание = Ложь) Экспорт
УстановитьПривилегированныйРежим(Истина);
Если Не ЭтоФоновоеЗадание И Константы.ДокументооборотИспользоватьОтложенноеОбновлениеПравДоступа.Получить() Тогда
// Отключение регл. заданий
ЗаданиеОперативноеВключено = ОтключитьОбработкуОперативнойОчереди();
ЗаданиеДолгоеВключено = ОтключитьОбработкуДолгойОчереди();
// Удаление всех прав объектов
Набор = РегистрыСведений.ПраваПоДескрипторамДоступа.СоздатьНаборЗаписей();
Набор.Записать();
// Включение регл. заданий
Если ЗаданиеОперативноеВключено Тогда
ВключитьОбработкуОперативнойОчереди();
КонецЕсли;
Если ЗаданиеДолгоеВключено Тогда
ВключитьОбработкуДолгойОчереди();
КонецЕсли;
Возврат; // Остальные данные будут удалены фоновым заданием
КонецЕсли;
НачатьТранзакцию();
// Очистка очереди
РегистрыСведений.ОчередьОбновленияПравДоступа.Очистить();
// Удаление всех прав объектов
Набор = РегистрыСведений.ПраваПоДескрипторамДоступа.СоздатьНаборЗаписей();
Набор.Записать();
Набор = РегистрыСведений.ДескрипторыДоступаДляОбъектов.СоздатьНаборЗаписей();
Набор.Записать();
Выборка = Справочники.ДескрипторыДоступаОбъектов.Выбрать();
Пока Выборка.Следующий() Цикл
Объект = Выборка.ПолучитьОбъект();
Объект.Удалить();
КонецЦикла;
// Удаление всех прав файлов
НаборЗаписей = РегистрыСведений.ДескрипторыДоступаДляФайлов.СоздатьНаборЗаписей();
НаборЗаписей.Записать();
Выборка = Справочники.ДескрипторыДоступаФайлов.Выбрать();
Пока Выборка.Следующий() Цикл
Объект = Выборка.ПолучитьОбъект();
Объект.Удалить();
КонецЦикла;
// Удаление всех прав регистров
Выборка = Справочники.ДескрипторыДоступаРегистров.Выбрать();
Пока Выборка.Следующий() Цикл
Объект = Выборка.ПолучитьОбъект();
Объект.Удалить();
КонецЦикла;
ЗафиксироватьТранзакцию();
КонецПроцедуры
// Обновляет сведения о правах доступа всех данных информационной базы.
Процедура ОбновитьПраваВсехДанных() Экспорт
УстановитьПривилегированныйРежим(Истина);
ВремяНачала = ТекущаяДата();
// Пересчет прав всех объектов
ТипыВладельцев = Метаданные.ПодпискиНаСобытия.ДокументооборотПраваДоступаПриЗаписиОбъектаДоступа.Источник.Типы();
Для Каждого Тип Из ТипыВладельцев Цикл
ИмяСправочника = Метаданные.НайтиПоТипу(Тип).ПолноеИмя();
ОпределитьПраваПоИмениОбъекта(ИмяСправочника);
КонецЦикла;
// Пересчет прав всех файлов
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗЛИЧНЫЕ
|		Файлы.ВладелецФайла
|ИЗ
|		Справочник.Файлы КАК Файлы";
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
ОпределитьДескрипторДоступаФайлаПоВладельцу(Выборка.ВладелецФайла);
КонецЦикла;
// Пересчет прав всех регистров
ТипыВладельцев = Метаданные.ПодпискиНаСобытия.ДокументооборотПраваДоступаПередЗаписьюНабораЗаписей.Источник.Типы();
Для Каждого Тип Из ТипыВладельцев Цикл
МетаданныеРегистра = Метаданные.НайтиПоТипу(Тип);
ПолноеИмя = МетаданныеРегистра.ПолноеИмя();
ОпределитьПраваПоИмениРегистра(ПолноеИмя)
КонецЦикла;
КонецПроцедуры
// Определяет права доступа для всех объектов указанной таблицы
// Если права для этого объекта уже определены, то ничего не делает
Процедура ОпределитьПраваПоИмениОбъекта(Таблица) Экспорт
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	Данные.Ссылка
|ИЗ
|	[Таблица] КАК Данные";
Запрос.Текст = СтрЗаменить(Запрос.Текст, "[Таблица]", Таблица);
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
ДокументооборотПраваДоступа.ОпределитьДескрипторДоступаОбъекта(Выборка.Ссылка);
КонецЦикла;
КонецПроцедуры
// Определяет права доступа для всех записей указанного регистра
// Если права для записей этого регситра уже определены, то ничего не делает
```

```
Процедура ОпределитьПраваПоИмениРегистра(ПолноеИмя) Экспорт
МенеджерРегистра = ОбщегоНазначения.МенеджерОбъектаПоПолномуИмени(ПолноеИмя);
МетаданныеРегистра = Метаданные.НайтиПоПолномуИмени(ПолноеИмя);
ПодчиненРегистратору = Ложь;
Если Метаданные.РегистрыНакопления.Содержит(МетаданныеРегистра) Тогда
ПодчиненРегистратору = Истина;
КонецЕсли;
Если Метаданные.РегистрыСведений.Содержит(МетаданныеРегистра) Тогда
Если МетаданныеРегистра.РежимЗаписи = Метаданные.СвойстваОбъектов.РежимЗаписиРегистра.ПодчинениеРегистратору Тогда
ПодчиненРегистратору = Истина;
КонецЕсли;
КонецЕсли;
Если ПодчиненРегистратору Тогда
// Пересчет прав всех записей по уникальным регистраторам
Запрос = Новый Запрос;
Запрос.Текст = "ВЫБРАТЬ РАЗЛИЧНЫЕ Регистр.Регистратор";
Запрос.Текст = Запрос.Текст +
"
|ИЗ
|	%1 КАК Регистр";
Запрос.Текст = СтрЗаменить(Запрос.Текст, "%1", ПолноеИмя);
Выборка = Запрос.Выполнить().Выбрать();
// Обход результатов запроса
Пока Выборка.Следующий() Цикл
// Чтение набора записей с отбором по регистратору
НаборЗаписей = МенеджерРегистра.СоздатьНаборЗаписей();
НаборЗаписей.Отбор.Регистратор.Установить(Выборка.Регистратор);
НаборЗаписей.Прочитать();
// Определение дескриптора для набора записей
ДокументооборотПраваДоступа.ОпределитьДескрипторДоступаНабораЗаписей(НаборЗаписей);
КонецЦикла;
Иначе
СведенияОПолях = Новый Структура("ОбъектДоступа1, ОбъектДоступа2, ОбъектДоступа3");
МенеджерРегистра.ЗаполнитьСведенияОПоляхДоступа(СведенияОПолях);
// Пересчет прав всех уникальных записей всех регистров
Запрос = Новый Запрос;
Запрос.Текст = "ВЫБРАТЬ РАЗЛИЧНЫЕ ";
// Добавление к запросу правообразующих полей
Запрос.Текст = Запрос.Текст +
"Регистр." + СведенияОПолях.ОбъектДоступа1;
Если СведенияОПолях.ОбъектДоступа2 <> Неопределено Тогда
Запрос.Текст = Запрос.Текст + ", Регистр." + СведенияОПолях.ОбъектДоступа2;
КонецЕсли;
Если СведенияОПолях.ОбъектДоступа3 <> Неопределено Тогда
Запрос.Текст = Запрос.Текст + ", Регистр." + СведенияОПолях.ОбъектДоступа3;
КонецЕсли;
// Добавление к запросу всех измерений регистра
Для каждого Измерение Из МетаданныеРегистра.Измерения Цикл
Если Измерение.Имя <> СведенияОПолях.ОбъектДоступа1
И Измерение.Имя <> СведенияОПолях.ОбъектДоступа2
И Измерение.Имя <> СведенияОПолях.ОбъектДоступа2 Тогда
Запрос.Текст = Запрос.Текст + ", Регистр." + Измерение.Имя;
КонецЕсли;
КонецЦикла;
Запрос.Текст = Запрос.Текст +
"
|ИЗ
|	%1 КАК Регистр";
Запрос.Текст = СтрЗаменить(Запрос.Текст, "%1", ПолноеИмя);
Выборка = Запрос.Выполнить().Выбрать();
// Обход результатов запроса
Пока Выборка.Следующий() Цикл
// Чтение набора записей
НаборЗаписей = МенеджерРегистра.СоздатьНаборЗаписей();
Для каждого Измерение Из МетаданныеРегистра.Измерения Цикл
НаборЗаписей.Отбор[Измерение.Имя].Установить(Выборка[Измерение.Имя]);
КонецЦикла;
НаборЗаписей.Прочитать();
// Определение дескриптора для набора записей
ДокументооборотПраваДоступа.ОпределитьДескрипторДоступаНабораЗаписей(НаборЗаписей);
КонецЦикла;
КонецЕсли;
КонецПроцедуры
// Возвращает Истина если у указанного пользователя есть указанная роль
// Если пользователь не указан, то используется текущий пользователь
Функция ЕстьРоль(Роль, Объект = Неопределено, Пользователь = Неопределено) Экспорт
Если УправлениеДоступом.ЭтоПолноправныйПользователь(Пользователь) Тогда
Возврат Истина;
КонецЕсли;
// Проверка, что роль назначается пользователю через группу доступа по профилю.
УстановитьПривилегированныйРежим(Истина);
Если Пользователь = Неопределено Тогда
Пользователь = ОбщегоНазначения.ТекущийПользователь();
КонецЕсли;
ОбъектМетаданныхРоль = Метаданные.Роли.Найти(Роль);
ИдентификаторРоли = ОбщегоНазначения.ИдентификаторОбъектаМетаданных(ОбъектМетаданныхРоль);
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДокументооборотПользователиГруппДоступа.ГруппаДоступа
|ИЗ
|	РегистрСведений.ДокументооборотПользователиГруппДоступа КАК ДокументооборотПользователиГруппДоступа
|		ВНУТРЕННЕЕ СОЕДИНЕНИЕ Справочник.ПрофилиГруппДоступа.Роли КАК ПрофилиГруппДоступаРоли
|		ПО ДокументооборотПользователиГруппДоступа.ГруппаДоступа.Профиль = ПрофилиГруппДоступаРоли.Ссылка
|ГДЕ
|	ДокументооборотПользователиГруппДоступа.Пользователь = &Пользователь
|	И ПрофилиГруппДоступаРоли.Роль = &Роль";
Запрос.УстановитьПараметр("Пользователь", Пользователь);
Запрос.УстановитьПараметр("Роль", ИдентификаторРоли);
Результат = Запрос.Выполнить();
ЕстьРоль = Не Результат.Пустой();
// Проверка значений доступа
Если ЕстьРоль И Объект <> Неопределено Тогда
ЕстьРоль = Ложь;
ДескрипторОбъекта = ПолучитьДескрипторДоступа(Объект);
Выборка = Результат.Выбрать();
Пока Выборка.Следующий() Цикл
ЕстьРоль = Справочники.ДескрипторыДоступаОбъектов.ДоступПоЗначениямРазрешен(
ДескрипторОбъекта, Выборка.ГруппаДоступа);
Если ЕстьРоль Тогда
Прервать;
КонецЕсли;
КонецЦикла;
КонецЕсли;
Возврат ЕстьРоль;
КонецФункции
// Вызывается рекурсивно
Процедура ОбновитьПраваОбъектовЗависящихОтГруппыПользователей(ГруппаПользователей, Немедленно = Неопределено) Экспорт
УстановитьПривилегированныйРежим(Истина);
// Рекурсивный вызов обновления для родительской группы пользователей
Если ЗначениеЗаполнено(ГруппаПользователей.Родитель) Тогда
ОбновитьПраваОбъектовЗависящихОтГруппыПользователей(ГруппаПользователей.Родитель, Немедленно);
КонецЕсли;
// Проверка на отложенное обновление прав доступа
Если ДокументооборотПраваДоступаПовтИсп.ОтложенноеОбновлениеПравДоступа()
И Немедленно <> Истина Тогда
// Добавление в очередь
РегистрыСведений.ОчередьОбновленияПравДоступа.Добавить(ГруппаПользователей);
Возврат;
КонецЕсли;
// Группы доступа
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ГруппыДоступаПользователи.Ссылка
|ИЗ
|	Справочник.ГруппыДоступа.Пользователи КАК ГруппыДоступаПользователи
|ГДЕ
|	ГруппыДоступаПользователи.Пользователь = &Пользователь";
Запрос.УстановитьПараметр("Пользователь", ГруппаПользователей.Ссылка);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
ОбновитьПраваЗависящиеОтГруппыДоступа(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// РегистрСведений НастройкиПравОбъектов
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	НастройкиПравОбъектов.Объект
|ИЗ
|	РегистрСведений.НастройкиПравОбъектов КАК НастройкиПравОбъектов
|ГДЕ
|	НастройкиПравОбъектов.Пользователь = &Пользователь";
Запрос.УстановитьПараметр("Пользователь", ГруппаПользователей.Ссылка);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
ОбновитьПраваПапки(ВыборкаДетальныеЗаписи.Объект);
КонецЦикла;
// Табличная часть "РабочаяГруппа" у дескриптора доступа объекта
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектовРабочаяГруппа.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов.РабочаяГруппа КАК ДескрипторыДоступаОбъектовРабочаяГруппа
|ГДЕ
|	ДескрипторыДоступаОбъектовРабочаяГруппа.Участник = &ГруппаПользователей";
Запрос.УстановитьПараметр("ГруппаПользователей", ГруппаПользователей.Ссылка);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПраво(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// Табличная часть "Пользователи" у дескриптора доступа объекта
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектовПользователи.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов.Пользователи КАК ДескрипторыДоступаОбъектовПользователи
|ГДЕ
|	ДескрипторыДоступаОбъектовПользователи.Пользователь = &ГруппаПользователей";
Запрос.УстановитьПараметр("ГруппаПользователей", ГруппаПользователей.Ссылка);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПраво(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// Если изменена группа ВсеПользователи, то выполняется обновление всех дескрипторов
// для элементов справочников, которые являются группами. Т.к. доступ к группам имеют
// все пользователи без ограничений
Если ГруппаПользователей = Справочники.ГруппыПользователей.ВсеПользователи Тогда
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов
|ГДЕ
|	ДескрипторыДоступаОбъектов.Группа = Истина";
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПраво(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
КонецЕсли;
КонецПроцедуры
// Обновляет права, зависящие от указанного подразделения
// - для руководителя подразделения
// - для входящих в подразделение пользователей
// - для родительских и подчиненных подразделений
Процедура ОбновитьПраваПоПодразделению(Подразделение, Немедленно = Неопределено) Экспорт
УстановитьПривилегированныйРежим(Истина);
// Если не установлена передача руководителям доступа подчиненных
```

```
// то тогда при изменении подразделений обновление прав выполнять
// не требуется
Если Константы.ДобавлятьРуководителямДоступПодчиненных.Получить() Тогда
Если ДокументооборотПраваДоступаПовтИсп.ОтложенноеОбновлениеПравДоступа()
И Немедленно <> Истина Тогда
РегистрыСведений.ОчередьОбновленияПравДоступа.Добавить(Подразделение);
Иначе
ОбновитьПраваПоПодчиненнымПодразделениям(Подразделение);
КонецЕсли;
КонецЕсли;
КонецПроцедуры
// Находит все объекты, которые используют указанный дескриптор
// и обновляет права всех других дескрипторов, которые зависят
// от этих объектов
Процедура ОбновитьПраваЗависимыхДескрипторовДоступа(Дескриптор) Экспорт
УстановитьПривилегированныйРежим(Истина);
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаДляОбъектов.Объект
|ИЗ
|
	РегистрСведений.ДескрипторыДоступаДляОбъектов КАК ДескрипторыДоступаДляОбъектов
|ГДЕ
|
	ДескрипторыДоступаДляОбъектов.Дескриптор = &Дескриптор";
Запрос.УстановитьПараметр("Дескриптор", Дескриптор.Ссылка);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
ОбновитьПраваСвязанныхДескрипторовПоОбъекту(ВыборкаДетальныеЗаписи.Объект);
КонецЦикла;
КонецПроцедуры
// Обновляет все дескрипторы прав доступа, в правах которых встречается
// указанный пользователь
Процедура ОбновитьПраваПоПользователю(Пользователь, Немедленно = Ложь) Экспорт
Если Не Немедленно Тогда
РегистрыСведений.ОчередьОбновленияПравДоступа.Добавить(Пользователь, Дата("00010101000000"));
Иначе
УстановитьПривилегированныйРежим(Истина);
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ПраваПоДескрипторамДоступа.Дескриптор
|ИЗ
|
	РегистрСведений.ПраваПоДескрипторамДоступа КАК ПраваПоДескрипторамДоступа
|ГДЕ
|	ПраваПоДескрипторамДоступа.Пользователь = &Пользователь";
Запрос.УстановитьПараметр("Пользователь", Пользователь);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Немедленно = НеДокументооборотПраваДоступаПовтИсп.ОтложенноеОбновлениеПравДоступа();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
ОбновитьПраваДоступаПоДескриптору(ВыборкаДетальныеЗаписи.Дескриптор, Немедленно);
КонецЦикла;
КонецЕсли;
КонецПроцедуры
// Восстанавливает все права пользователя, которые у него были до
// пометки на удаление или установки признака "Недействителен"
Процедура ВосстановитьПраваПользователя(Пользователь, Немедленно = Ложь) Экспорт
// По группе доступа
Запрос = Новый Запрос("ВЫБРАТЬ
|	ГруппыДоступаПользователи.Ссылка КАК ГруппаДоступа
|ИЗ
|	Справочник.ГруппыДоступа.Пользователи КАК ГруппыДоступаПользователи
|ГДЕ
|	ГруппыДоступаПользователи.Пользователь = &Пользователь");
Запрос.УстановитьПараметр("Пользователь", Пользователь);
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
ОбновитьПраваЗависящиеОтГруппыДоступа(Выборка.ГруппаДоступа);
КонецЦикла;
// По группе пользователей
Запрос = Новый Запрос("ВЫБРАТЬ
|	ГруппыПользователейСостав.Ссылка КАК ГруппаПользователей
|ИЗ
|	Справочник.ГруппыПользователей.Состав КАК ГруппыПользователейСостав
|ГДЕ
|	ГруппыПользователейСостав.Пользователь = &Пользователь");
Запрос.УстановитьПараметр("Пользователь", Пользователь);
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
ОбновитьПраваОбъектовЗависящихОтГруппыПользователей(Выборка.ГруппаПользователей);
КонецЦикла;
// По рабочей группе дескриптора
Запрос = Новый Запрос("ВЫБРАТЬ
|
	ДескрипторыДоступаОбъектовРабочаяГруппа.Ссылка КАК Дескриптор
|ИЗ
|
	Справочник.ДескрипторыДоступаОбъектов.РабочаяГруппа КАК ДескрипторыДоступаОбъектовРабочаяГруппа
|ГДЕ
|
	ДескрипторыДоступаОбъектовРабочаяГруппа.Участник = &Пользователь");
Запрос.УстановитьПараметр("Пользователь", Пользователь);
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
ОбновитьПраваДоступаПоДескриптору(Выборка.Дескриптор, Немедленно);
КонецЦикла;
// По пользователям дескриптора
Запрос = Новый Запрос("ВЫБРАТЬ
|
	ДескрипторыДоступаОбъектовПользователи.Ссылка КАК Дескриптор
|ИЗ
|
	Справочник.ДескрипторыДоступаОбъектов.Пользователи КАК ДескрипторыДоступаОбъектовПользователи
|ГДЕ
|
	ДескрипторыДоступаОбъектовПользователи.Пользователь = &Пользователь");
Запрос.УстановитьПараметр("Пользователь", Пользователь);
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
ОбновитьПраваДоступаПоДескриптору(Выборка.Дескриптор, Немедленно);
КонецЦикла;
// По роли исполнителя в дескрипторе
Запрос = Новый Запрос("ВЫБРАТЬ
|
	ДескрипторыДоступаОбъектовПользователи.Ссылка КАК Дескриптор
|ИЗ
|	РегистрСведений.ИсполнителиЗадач КАК ИсполнителиЗадач
|		ВНУТРЕННЕЕ СОЕДИНЕНИЕ Справочник.ДескрипторыДоступаОбъектов.Пользователи КАК ДескрипторыДоступаОбъектовПользователи
|		ПО ИсполнителиЗадач.РольИсполнителя = ДескрипторыДоступаОбъектовПользователи.Пользователь
|			И ИсполнителиЗадач.ОсновнойОбъектАдресации = ДескрипторыДоступаОбъектовПользователи.ОсновнойОбъектАдресации
|			И ИсполнителиЗадач.ДополнительныйОбъектАдресации = ДескрипторыДоступаОбъектовПользователи.ДополнительныйОбъектАдресации
|ГДЕ
|	ИсполнителиЗадач.Исполнитель = &Пользователь
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|
	ДескрипторыДоступаОбъектовРабочаяГруппа.Ссылка
|ИЗ
|	РегистрСведений.ИсполнителиЗадач КАК ИсполнителиЗадач
|		ВНУТРЕННЕЕ СОЕДИНЕНИЕ Справочник.ДескрипторыДоступаОбъектов.РабочаяГруппа КАК ДескрипторыДоступаОбъектовРабочаяГруппа
|		ПО ИсполнителиЗадач.ОсновнойОбъектАдресации = ДескрипторыДоступаОбъектовРабочаяГруппа.ОсновнойОбъектАдресации
|			И ИсполнителиЗадач.ДополнительныйОбъектАдресации = ДескрипторыДоступаОбъектовРабочаяГруппа.ДополнительныйОбъектАдресации
|ГДЕ
|	ИсполнителиЗадач.Исполнитель = &Пользователь");
Запрос.УстановитьПараметр("Пользователь", Пользователь);
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
ОбновитьПраваДоступаПоДескриптору(Выборка.Дескриптор, Немедленно);
КонецЦикла;
// Права папок
Запрос = Новый Запрос("ВЫБРАТЬ РАЗЛИЧНЫЕ
|	НастройкиПравОбъектов.Объект
|ИЗ
|	РегистрСведений.НастройкиПравОбъектов КАК НастройкиПравОбъектов
|ГДЕ
|	НастройкиПравОбъектов.Пользователь = &Пользователь");
Запрос.УстановитьПараметр("Пользователь", Пользователь);
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
ОбновитьПраваПапки(Выборка.Объект);
КонецЦикла;
КонецПроцедуры
// Заполняет ТЧ дескриптора НастройкаПрав по настройкам папки
Процедура ЗаполнитьНастройкиДескриптора(ДескрипторДоступа, Папка) Экспорт
// Собственные права папки
Запрос = Новый Запрос("ВЫБРАТЬ
|	НастройкиПравОбъектов.Пользователь КАК Пользователь,
|	НастройкиПравОбъектов.Право КАК Право,
|	НастройкиПравОбъектов.ПравоЗапрещено КАК ПравоЗапрещено,
|
	НаследованиеНастроекПравОбъектов.Наследовать
|ИЗ
|
	РегистрСведений.НаследованиеНастроекПравОбъектов КАК НаследованиеНастроекПравОбъектов
|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.НастройкиПравОбъектов КАК НастройкиПравОбъектов
|		ПО НастройкиПравОбъектов.Объект = НаследованиеНастроекПравОбъектов.Родитель
|ГДЕ
|	НаследованиеНастроекПравОбъектов.Объект = &Объект
|	И НаследованиеНастроекПравОбъектов.Родитель = &Объект");
Запрос.УстановитьПараметр("Объект", Папка.Ссылка);
Результат = Запрос.Выполнить();
Наследовать = Истина;
Если Не Результат.Пустой() Тогда
ТаблицаПрав = Результат.Выгрузить();
Наследовать = ТаблицаПрав[0].Наследовать;
Если ЗначениеЗаполнено(ТаблицаПрав[0].Право) Тогда
ДескрипторДоступа.НастройкаПрав.Загрузить(ТаблицаПрав);
КонецЕсли;
Иначе
ДескрипторДоступа.НастройкаПрав.Очистить();
КонецЕсли;
// Наследуемые права
Если Наследовать Тогда
Запрос.Текст = "ВЫБРАТЬ
|	НастройкиПравОбъектов.Пользователь,
|	НастройкиПравОбъектов.Право,
|	НастройкиПравОбъектов.ПравоЗапрещено
|ИЗ
|	РегистрСведений.НастройкиПравОбъектов КАК НастройкиПравОбъектов
|		ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрСведений.НаследованиеНастроекПравОбъектов КАК НаследованиеНастроекПравОбъектов
|		ПО НастройкиПравОбъектов.Объект = НаследованиеНастроекПравОбъектов.Родитель
|ГДЕ
|	НаследованиеНастроекПравОбъектов.Объект = &Объект
|	И НаследованиеНастроекПравОбъектов.Родитель <>&Объект
|	И НастройкиПравОбъектов.НаследованиеРазрешено
|
|УПОРЯДОЧИТЬ ПО
|	Пользователь,
|	Право";
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
ЗаполнитьЗначенияСвойств(ДескрипторДоступа.НастройкаПрав.Добавить(), Выборка);
КонецЦикла;
КонецЕсли;
```

```
КонецПроцедуры
// Включает регл. задание обработки оперативной очереди
Процедура ВключитьОбработкуОперативнойОчереди()
Экспорт
Задание =
РегламентныеЗадания.НайтиПредопределенное("Документо
оборотОбновлениеПравДоступаОперативное");
Задание.Использование = Истина;
Задание.Записать();
КонецПроцедуры
// Включает регл. задание обработки долгой очереди
Процедура ВключитьОбработкуДолгойОчереди() Экспорт
Задание =
РегламентныеЗадания.НайтиПредопределенное("Документо
оборотОбновлениеПравДоступаДолгое");
Задание.Использование = Истина;
Задание.Записать();
КонецПроцедуры
// Останавливает регл. задание обработки оперативной
очереди
// Возвращаемое значение:
// Истина, если регл. задание было активно
// Ложь, если регл. задание уже было отключено
Функция
ОтключитьОбработкуОперативнойОчереди(ТолькоФоновое
Задание = Ложь) Экспорт
// Отключение регл. задания
Задание =
РегламентныеЗадания.НайтиПредопределенное("Документо
оборотОбновлениеПравДоступаОперативное");
ЗаданиеАктивно = Задание.Использование;
Если Не ТолькоФоновоеЗадание И ЗаданиеАктивно Тогда
Задание.Использование = Ложь;
Задание.Записать();
КонецЕсли;
// Остановка фонового задания
Попытка
ИДЗадания = Задание.УникальныйИдентификатор;
Свойства =
РегламентныеЗаданияСервер.ПолучитьСвойстваПоследнего
ФоновогоЗаданияВыполненияРегламентногоЗадания(Задани
е);
РегламентныеЗаданияСервер.ОтменитьФоновоеЗадание(Сво
йства.Идентификатор);
Исключение
КонецПопытки;
Возврат ЗаданиеАктивно;
КонецФункции
// Останавливает регл. задание обработки долгой очереди
// Возвращаемое значение:
// Истина, если регл. задание было активно
// Ложь, если регл. задание уже было отключено
Функция
ОтключитьОбработкуДолгойОчереди(ТолькоФоновоеЗадани
е = Ложь) Экспорт
// Отключение регл. задания
Задание =
РегламентныеЗадания.НайтиПредопределенное("Документо
оборотОбновлениеПравДоступаДолгое");
ЗаданиеАктивно = Задание.Использование;
Если ЗаданиеАктивно Тогда
Если Не ТолькоФоновоеЗадание Тогда
Задание.Использование = Ложь;
Задание.Записать();
КонецЕсли;
// Остановка фонового задания
Попытка
ИДЗадания = Задание.УникальныйИдентификатор;
Свойства =
РегламентныеЗаданияСервер.ПолучитьСвойстваПоследнего
ФоновогоЗаданияВыполненияРегламентногоЗадания(Задани
е);
РегламентныеЗаданияСервер.ОтменитьФоновоеЗадание(Сво
йства.Идентификатор);
Исключение
КонецПопытки;
КонецЕсли;
Возврат ЗаданиеАктивно;
КонецФункции
// Добавляет к переданной стуктуре прав права, не
ограниченные на уровне записей
Процедура
РасширитьДоступПравамиБезОграничения(Права,
ОбъектДоступа, Пользователь = Неопределено) Экспорт
УстановитьПривилегированныйРежим(Истина);
Если Не ЗначениеЗаполнено(Пользователь) Тогда
Пользователь = ОбщегоНазначения.ТекущийПользователь();
КонецЕсли;
// Проверка прав, не ограниченных на уровне записей
Запрос = Новый Запрос;
Запрос.Текст = "ВЫБРАТЬ
|
        МАКСИМУМ(ПраваГруппДоступаНаТаблицы
.ЧтениеБезОграничения) КАК Чтение,
|
        МАКСИМУМ(ПраваГруппДоступаНаТаблицы
.ДобавлениеБезОграничения) КАК Добавление,
|
        МАКСИМУМ(ПраваГруппДоступаНаТаблицы
.ИзменениеБезОграничения) КАК Изменение,
|
        МАКСИМУМ(ПраваГруппДоступаНаТаблицы
.УдалениеБезОграничения) КАК Удаление,
|
        МАКСИМУМ(ПраваГруппДоступаНаТаблицы
.ПраваБезОграничения) КАК УправлениеПравами
|ИЗ
|
        РегистрСведений.ПраваГруппДоступаНаТабли
цы КАК ПраваГруппДоступаНаТаблицы
|        ВНУТРЕННЕЕ СОЕДИНЕНИЕ
Справочник.ГруппыДоступа.Пользователи КАК
ГруппыДоступаПользователи
|        ПО
ПраваГруппДоступаНаТаблицы.ГруппаДоступа =
ГруппыДоступаПользователи.Ссылка
|ГДЕ
|
        ПраваГруппДоступаНаТаблицы.Таблица.Значе
ниеПустойСсылки = &ПустаяСсылка
|        И ГруппыДоступаПользователи.Пользователь
= &Пользователь
|
|ИМЕЮЩИЕ
|
        (МАКСИМУМ(ПраваГруппДоступаНаТаблиц
ы.ЧтениеБезОграничения) = ИСТИНА
|        ИЛИ
МАКСИМУМ(ПраваГруппДоступаНаТаблицы.ДобавлениеБ
езОграничения) = ИСТИНА
|        ИЛИ
МАКСИМУМ(ПраваГруппДоступаНаТаблицы.ИзменениеБ
езОграничения) = ИСТИНА
|        ИЛИ
МАКСИМУМ(ПраваГруппДоступаНаТаблицы.УдалениеБез
Ограничения) = ИСТИНА
|        ИЛИ
МАКСИМУМ(ПраваГруппДоступаНаТаблицы.ПраваБезОгр
аничения) = ИСТИНА)";
Запрос.УстановитьПараметр("ПустаяСсылка",
Новый(ТипЗнч(ОбъектДоступа.Ссылка)));
Запрос.УстановитьПараметр("Пользователь",
Пользователь);
Результат = Запрос.Выполнить();
Если Не Результат.Пустой() Тогда
ВыборкаПравБезОграничений = Результат.Выбрать();
ВыборкаПравБезОграничений.Следующий();
Права.Чтение = Права.Чтение или
ВыборкаПравБезОграничений.Чтение;
Права.Изменение = Права.Изменение или
ВыборкаПравБезОграничений.Изменение;
Права.Удаление = Права.Удаление или
ВыборкаПравБезОграничений.Удаление;
Права.УправлениеПравами = Права.УправлениеПравами
или ВыборкаПравБезОграничений.УправлениеПравами;
// Добавление без ограничений проверяется вместе с
изменением, т.к. тут может
// не быть ограничений РЛС даже если они подразумеваются
Права.Добавление = Права.Добавление или
ВыборкаПравБезОграничений.Добавление и
ВыборкаПравБезОграничений.Изменение;
КонецЕсли;
КонецПроцедуры
////////////////////////////////////////////////////////////////////////////////
// Обработчики подписок и регламентных заданий
// Обработчик подписки
ДокументооборотПраваДоступаПередЗаписьюОбъектаДост
упа
Процедура
ДокументооборотПраваДоступаПередЗаписьюОбъектаДост
упа(Источник, Отказ) Экспорт
Если Не
ДокументооборотПраваДоступаПовтИсп.ВключеноИспольз
ованиеПравДоступа() Тогда
Возврат;
КонецЕсли;
// Сохранение признака ЭтоНовый для использования в
обработчике события ПриЗаписи
Источник.ДополнительныеСвойства.Вставить("ЭтоНовый",
Источник.ЭтоНовый());
// Сохранение старого дескриптора доступа для
обнаружения того, что дескриптор изменен
Если Не Источник.ЭтоНовый() Тогда
СтарыйДескриптор =
РегистрыСведений.ДескрипторыДоступаДляОбъектов.Найт
иДескрипторДляОбъекта(Источник.Ссылка);
Источник.ДополнительныеСвойства.Вставить("СтарыйДеск
риптор", СтарыйДескриптор);
КонецЕсли;
КонецПроцедуры
// Обработчик подписки
ДокументооборотПраваДоступаПередЗаписьюДокумента
Процедура
ДокументооборотПраваДоступаПередЗаписьюДокумента(И
сточник, Отказ, РежимЗаписи, РежимПроведения) Экспорт
Если Не
ДокументооборотПраваДоступаПовтИсп.ВключеноИспольз
ованиеПравДоступа() Тогда
Возврат;
КонецЕсли;
// Сохранение признака ЭтоНовый для использования в
обработчике события ПриЗаписи
Источник.ДополнительныеСвойства.Вставить("ЭтоНовый",
Источник.ЭтоНовый());
КонецПроцедуры
// Обработчик подписки
ДокументооборотПраваДоступаПриЗаписиОбъектаДоступа
Процедура
ДокументооборотПраваДоступаПриЗаписиОбъектаДоступа(
Источник, Отказ) Экспорт
Если Не
ДокументооборотПраваДоступаПовтИсп.ВключеноИспольз
ованиеПравДоступа() Тогда
Возврат;
КонецЕсли;
НеобходимаПроверкаПравПоДескриптору = Не
ПривилегированныйРежим() И Не
РольДоступна("ПолныеПрава");
Дескриптор = ОпределитьДескрипторДоступаОбъекта(
Источник,
НеобходимаПроверкаПравПоДескриптору); // Немедленно
// Проверка права добавления
Если НеобходимаПроверкаПравПоДескриптору Тогда
Права =
РегистрыСведений.ПраваПоДескрипторамДоступа.Получит
ьПраваПоДескриптору(Дескриптор);
РасширитьДоступПравамиБезОграничения(Права,
Источник.Ссылка);
НетПрав = Ложь;
Если Источник.ДополнительныеСвойства.ЭтоНовый Тогда
// Для новых объектов проверяется право добавления
Если Не Права.Добавление Тогда
НетПрав = Истина;
КонецЕсли;
Иначе
// Для существующих объектов проверяется право
изменения
Если Не Права.Изменение Тогда
НетПрав = Истина;
КонецЕсли;
КонецЕсли;
Если НетПрав Тогда
ВызватьИсключение НСтр("ru = 'Недостаточно прав для
выполнения операции.
|Обратитесь к администратору.'");
КонецЕсли;
КонецЕсли;
// Проверка на изменение дескриптора
Если
Источник.ДополнительныеСвойства.Свойство("СтарыйДеск
риптор") Тогда
// Если дескриптор поменялся, то обновляются права всех
данных,
// которые зависят от этого объекта
Если
Источник.ДополнительныеСвойства.СтарыйДескриптор <>
Дескриптор Тогда
ОбновитьПраваЗависимыхДанных(Источник.Ссылка);
КонецЕсли;
КонецЕсли;
КонецПроцедуры
// Обработчик подписки
ДокументооборотПраваДоступаПередЗаписьюФайла
Процедура
ДокументооборотПраваДоступаПередЗаписьюФайла(Источ
ник, Отказ) Экспорт
Если Не
ДокументооборотПраваДоступаПовтИсп.ВключеноИспольз
ованиеПравДоступа() Тогда
Возврат;
КонецЕсли;
Дескриптор = ПолучитьДескрипторДоступаФайла(
Источник.ВладелецФайла,
Истина);
// Проверка права добавления
Если Не ПривилегированныйРежим() И Не
РольДоступна("ПолныеПрава") Тогда
Если Источник.ЭтоНовый() Тогда
Права =
РегистрыСведений.ПраваПоДескрипторамДоступа.Получит
ьПраваПоДескриптору(Дескриптор);
РасширитьДоступПравамиБезОграничения(Права,
Источник.Ссылка);
Если Не Права.Добавление Тогда
ВызватьИсключение НСтр("ru = 'Недостаточно прав для
выполнения операции.
|Обратитесь к администратору.'");
КонецЕсли;
КонецЕсли;
КонецЕсли;
КонецПроцедуры
// Обработчик подписки
ДокументооборотПраваДоступаПередЗаписьюНабораЗаписе
й
Процедура
ДокументооборотПраваДоступаПередЗаписьюНабораЗаписе
й(Источник, Отказ, Замещение) Экспорт
Если Не
ДокументооборотПраваДоступаПовтИсп.ВключеноИспольз
ованиеПравДоступа() Тогда
Возврат;
КонецЕсли;
ОпределитьДескрипторДоступаНабораЗаписей(
Источник,
Истина); // Немедленно
КонецПроцедуры
// Обработчик подписки
ДокументооборотПередЗаписьюПравообразующихОбъектов
Процедура
ДокументооборотПередЗаписьюПравообразующихОбъектов
(Источник, Отказ) Экспорт
```

```
УстановитьПривилегированныйРежим(Истина);
Если Не
ДокументооборотПраваДоступаПовтИсп.ВключеноИспользованиеПравДоступа() Тогда
Возврат;
КонецЕсли;
Если
ДокументооборотПраваДоступаПереопределяемый.ДокументооборотПередЗаписьюПравообразующихОбъектов(Источник, Отказ) Тогда
Возврат;
КонецЕсли;
// Проверка на то, что изменились правообразующие реквизиты
Если ТипЗнч(Источник) =
Тип("СправочникОбъект.ГруппыПользователей") Тогда
СтарыйСостав = Источник.Ссылка.Состав;
Если Источник.ЭтоНовый() Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
ИначеЕсли Источник.Родитель <>
Источник.Ссылка.Родитель Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
Источник.ДополнительныеСвойства.Вставить("СтарыйРодитель", Источник.Ссылка.Родитель);
ИначеЕсли Источник.Состав.Количество() <>
СтарыйСостав.Количество() Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
Иначе
// Поэлементное сравнение
ЗначениеИзменено = Ложь;
Для Сч = 0 По СтарыйСостав.Количество() - 1 Цикл
Если СтарыйСостав[Сч].Пользователь <>
Источник.Состав[Сч].Пользователь Тогда
ЗначениеИзменено = Истина;
Прервать;
КонецЕсли;
КонецЦикла;
Если ЗначениеИзменено Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
КонецЕсли;
КонецЕсли;
ИначеЕсли ТипЗнч(Источник) =
Тип("СправочникОбъект.ГруппыДоступа") Тогда
Если Не Источник.ЭтоГруппа Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
Если Не Источник.ЭтоНовый() Тогда
ЗначенияДоступа =
ПолучитьЗначенияДоступаГруппыДоступа(Источник.Ссылка);
Источник.ДополнительныеСвойства.Вставить("СтарыеЗначенияДоступа", ЗначенияДоступа);
Источник.ДополнительныеСвойства.Вставить("СтарыйПрофиль", Источник.Ссылка.Профиль);
Источник.ДополнительныеСвойства.Вставить("СтарыйСостав", Источник.Ссылка.Пользователи);
КонецЕсли;
КонецЕсли;
ИначеЕсли ТипЗнч(Источник) =
Тип("СправочникОбъект.Пользователи") Тогда
Источник.ДополнительныеСвойства.Вставить("ЭтоНовый",
Источник.ЭтоНовый());
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
Источник.ДополнительныеСвойства.Вставить("ИзмененаПометкаУдаления",
Источник.ПометкаУдаления <>
Источник.Ссылка.ПометкаУдаления);
Источник.ДополнительныеСвойства.Вставить("ИзмененПризнакДействительности",
Источник.Недействителен <>
Источник.Ссылка.Недействителен);
ИначеЕсли ТипЗнч(Источник) =
Тип("СправочникОбъект.ПапкиФайлов")
ИЛИ ТипЗнч(Источник) =
Тип("СправочникОбъект.ПапкиВнутреннихДокументов")
Тогда
Если Источник.Родитель <> Источник.Ссылка.Родитель
Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
КонецЕсли;
ИначеЕсли ТипЗнч(Источник) =
Тип("СправочникОбъект.СтруктураАдминистрации") Тогда
Если Источник.Родитель <> Источник.Ссылка.Родитель
ИЛИ Источник.Руководитель <>
Источник.Ссылка.Руководитель Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
КонецЕсли;
ИначеЕсли ТипЗнч(Источник) =
Тип("СправочникОбъект.ПрофилиГруппДоступа") Тогда
СтарыеРоли = Источник.Ссылка.Роли;
СтарыеВидыДоступа = Источник.Ссылка.ВидыДоступа;
СтарыеЗначенияДоступа =
Источник.Ссылка.ЗначенияДоступа;
Если Источник.Роли.Количество() <>
СтарыеРоли.Количество()
Или Источник.ВидыДоступа.Количество() <>
СтарыеВидыДоступа.Количество()
Или Источник.ЗначенияДоступа.Количество() <>
СтарыеЗначенияДоступа.Количество() Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
Иначе
// Поэлементное сравнение
ЗначениеИзменено = Ложь;
Для Сч = 0 По СтарыеРоли.Количество() - 1 Цикл
Если СтарыеРоли[Сч].Роль <> Источник.Роли[Сч].Роль
Тогда
ЗначениеИзменено = Истина;
Прервать;
КонецЕсли;
КонецЦикла;
Если ЗначениеИзменено Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
КонецЕсли;
Если Не ЗначениеИзменено Тогда
Для Сч = 0 По СтарыеВидыДоступа.Количество() - 1 Цикл
Если СтарыеВидыДоступа[Сч].ВидДоступа <>
Источник.ВидыДоступа[Сч].ВидДоступа
Или СтарыеВидыДоступа[Сч].ДоступРазрешен <>
Источник.ВидыДоступа[Сч].ДоступРазрешен Тогда
ЗначениеИзменено = Истина;
Прервать;
КонецЕсли;
КонецЦикла;
Если ЗначениеИзменено Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
КонецЕсли;
КонецЕсли;
Если Не ЗначениеИзменено Тогда
Для Сч = 0 По СтарыеЗначенияДоступа.Количество() - 1
Цикл
Если СтарыеЗначенияДоступа[Сч].ВидДоступа <>
Источник.ЗначенияДоступа[Сч].ВидДоступа
Или СтарыеЗначенияДоступа[Сч].ЗначениеДоступа <>
Источник.ЗначенияДоступа[Сч].ЗначениеДоступа Тогда
ЗначениеИзменено = Истина;
Прервать;
КонецЕсли;
КонецЦикла;
Если ЗначениеИзменено Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
КонецЕсли;
КонецЕсли;
КонецЕсли;
КонецЕсли;
КонецПроцедуры
// Обработчик подписки
ДокументооборотПриЗаписиПравообразующихОбъектов
Процедура
ДокументооборотПриЗаписиПравообразующихОбъектов(Источник, Отказ) Экспорт
УстановитьПривилегированныйРежим(Истина);
Если Не
ДокументооборотПраваДоступаПовтИсп.ВключеноИспользованиеПравДоступа() Тогда
Возврат;
КонецЕсли;
// Проверка на то, что правообразующие свойства источника были изменены
Если НЕ
Источник.ДополнительныеСвойства.Свойство("ЗначениеИзменено")
Или НЕ
Источник.ДополнительныеСвойства.ЗначениеИзменено
Тогда
Возврат;
КонецЕсли;
// Обработка источников разных типов
Если
ДокументооборотПраваДоступаПереопределяемый.ДокументооборотПриЗаписиПравообразующихОбъектов(Источник, Отказ) Тогда
Возврат;
КонецЕсли;
ТипИсточника = ТипЗнч(Источник);
Если ТипИсточника =
Тип("СправочникОбъект.ГруппыПользователей") Тогда
// Перезагрузка регл. заданий - для корректной работы функций повторного использования
ОтключитьОбработкуОперативнойОчереди(Истина);
ОтключитьОбработкуДолгойОчереди(Истина);
ТекущийПриоритетОчередиОбновленияПрав =
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав;
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав = 2;
// Долгая очередь
Попытка
Если
Источник.ДополнительныеСвойства.Свойство("СтарыйРодитель") Тогда
ОбновитьПраваОбъектовЗависящихОтГруппыПользователей(Источник.ДополнительныеСвойства.СтарыйРодитель);
КонецЕсли;
ОбновитьПраваОбъектовЗависящихОтГруппыПользователей(Источник.Ссылка);
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
Исключение
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
ВызватьИсключение;
КонецПопытки;
ИначеЕсли ТипИсточника =
Тип("СправочникОбъект.Пользователи") Тогда
Если Источник.ДополнительныеСвойства.ЭтоНовый Тогда
ТекущийПриоритетОчередиОбновленияПрав =
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав;
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав = 2;
// Долгая очередь
Попытка
// При добавлении нового пользователя неявно меняется состав
// предопределенной виртуальной группы "Все пользователи"
ОбновитьПраваОбъектовЗависящихОтГруппыПользователей(
Справочники.ГруппыПользователей.ВсеПользователи);
// Обновление пустых дескрипторов, которые дают разрешения всем пользователям
ОбновитьПраваПустыхДескрипторов();
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
Исключение
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
ВызватьИсключение;
КонецПопытки;
// Изменена пометка на удаление или признак "Недействителен"
ИначеЕсли
Источник.ДополнительныеСвойства.ИзмененаПометкаУдаления
Или
Источник.ДополнительныеСвойства.ИзмененПризнакДействительности Тогда
Немедленно = Не
ДокументооборотПраваДоступаПовтИсп.ОтложенноеОбновлениеПравДоступа();
Если Источник.ПометкаУдаления Или
Источник.Недействителен Тогда
ОбновитьПраваПоПользователю(Источник.Ссылка,
Немедленно);
Иначе
ВосстановитьПраваПользователя(Источник.Ссылка,
Немедленно);
КонецЕсли;
КонецЕсли;
ИначеЕсли ТипИсточника =
Тип("СправочникОбъект.ГруппыДоступа") Тогда
Если Источник.ЭтоГруппа Тогда
Возврат;
КонецЕсли;
// Отключение регл. заданий для корректной работы функций повторного использования
ОтключитьОбработкуОперативнойОчереди(Истина);
ОтключитьОбработкуДолгойОчереди(Истина);
// Обновление сведений о составе групп доступа
РегистрыСведений.ДокументооборотПользователиГруппДоступа.ОбновитьПоГруппеДоступа(Источник.Ссылка);
Если Источник.ЭтоНовый() Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
Иначе
СтарыеЗначенияДоступа = Новый Массив;
Если
Источник.ДополнительныеСвойства.Свойство("СтарыеЗначенияДоступа") Тогда
СтарыеЗначенияДоступа =
Источник.ДополнительныеСвойства.СтарыеЗначенияДоступа;
КонецЕсли;
СтарыйПрофиль = Неопределено;
Если
Источник.ДополнительныеСвойства.Свойство("СтарыйПрофиль") Тогда
СтарыйПрофиль =
Источник.ДополнительныеСвойства.СтарыйПрофиль;
КонецЕсли;
СтарыйСостав = Неопределено;
Если
Источник.ДополнительныеСвойства.Свойство("СтарыйСостав") Тогда
СтарыйСостав =
Источник.ДополнительныеСвойства.СтарыйСостав;
КонецЕсли;
ЗначенияДоступаДляОбработки = Новый Массив;
ПолныйПересчет = Ложь;
// Проверка изменения профиля группы доступа
Если СтарыйПрофиль <> Источник.Профиль Тогда
ПолныйПересчет = Истина;
КонецЕсли;
// Проверка изменения состава участников
Если СтарыйСостав = Неопределено
Или СтарыйСостав.Количество() <>
Источник.Пользователи.Количество() Тогда
ПолныйПересчет = Истина;
Иначе
// Поэлементное сравнение старых и новых пользователей группы
Для Сч = 0 По СтарыйСостав.Количество() - 1 Цикл
Если СтарыйСостав[Сч].Пользователь <>
Источник.Пользователи[Сч].Пользователь Тогда
```

```
ПолныйПересчет = Истина;
Прервать;
КонецЕсли;
КонецЦикла;
КонецЕсли;
НовыеЗначенияДоступа =
ПолучитьЗначенияДоступаГруппыДоступа(Источник.Ссылка);
// Добавление новых значений доступа
Для каждого Эл Из НовыеЗначенияДоступа Цикл
ЗначенияДоступаДляОбработки.Добавить(Эл);
КонецЦикла;
Если ПолныйПересчет Тогда
// Добавление старых значений доступа
Для каждого Эл Из СтарыеЗначенияДоступа Цикл
ЗначенияДоступаДляОбработки.Добавить(Эл);
КонецЦикла;
КонецЕсли;
// Обновление отобранных значений доступа
ТекущийПриоритетОчередиОбновленияПрав =
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав;
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав = 2;
// Долгая очередь
Попытка
ОбновитьПраваПоЗначениямВидовДоступа(ЗначенияДоступаДляОбработки);
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
Исключение
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
ВызватьИсключение;
КонецПопытки;
КонецЕсли;
ИначеЕсли ТипИсточника =
Тип("СправочникОбъект.ПапкиФайлов")
ИЛИ ТипИсточника =
Тип("СправочникОбъект.ПапкиВнутреннихДокументов")
Тогда
ОбновитьПраваДескрипторовЗависящихОтПапки(Источник.Ссылка);
ИначеЕсли ТипИсточника =
Тип("СправочникОбъект.СтруктураАдминистрации") Тогда
ТекущийПриоритетОчередиОбновленияПрав =
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав;
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав = 2;
// Долгая очередь
Попытка
ОбновитьПраваПоПодразделению(Источник.Ссылка);
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
Исключение
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
ВызватьИсключение;
КонецПопытки;
ИначеЕсли ТипИсточника =
Тип("СправочникОбъект.ПрофилиГруппДоступа") Тогда
Если Не Источник.ЭтоНовый() Тогда
ОбновитьПраваДескрипторовЗависящихОтПрофиля(Источник.Ссылка);
КонецЕсли;
КонецЕсли;
КонецПроцедуры
// Обработчик подписки
ДокументооборотПередЗаписьюПравообразующихРегистров
Процедура
ДокументооборотПередЗаписьюПравообразующихРегистров(Источник, Отказ, Замещение) Экспорт
УстановитьПривилегированныйРежим(Истина);
Если Не
ДокументооборотПраваДоступаПовтИсп.ВключеноИспользованиеПравДоступа() Тогда
Возврат;
КонецЕсли;
Если ТипЗнч(Источник) =
Тип("РегистрСведенийНаборЗаписей.СведенияОПользователях") Тогда
// Проверка изменения подразделения пользователя
Пользователь = Источник.Отбор.Пользователь.Значение;
СведенияОПользователе =
РегистрыСведений.СведенияОПользователях.ПолучитьСведенияОПользователе(Пользователь);
Если Источник.Количество() <> 0 Тогда
Для каждого Эл Из Источник Цикл
Если Не
СведенияОПользователе.Свойство("Подразделение") Тогда
Источник.ДополнительныеСвойства.Вставить("СтароеПодразделение", Неопределено);
Иначе
Если Эл.Подразделение <>
СведенияОПользователе.Подразделение Тогда
Источник.ДополнительныеСвойства.Вставить("СтароеПодразделение", СведенияОПользователе.Подразделение);
Прервать;
КонецЕсли;
КонецЕсли;
КонецЦикла;
Иначе
Если Не
СведенияОПользователе.Свойство("Подразделение") Тогда
Источник.ДополнительныеСвойства.Вставить("СтароеПодразделение", Неопределено);
Иначе
Источник.ДополнительныеСвойства.Вставить("СтароеПодразделение", СведенияОПользователе.Подразделение);
КонецЕсли;
КонецЕсли;
КонецЕсли;
КонецПроцедуры
// Обработчик подписки
ДокументооборотПриЗаписиПравообразующихРегистров
Процедура
ДокументооборотПриЗаписиПравообразующихРегистров(Источник, Отказ, Замещение) Экспорт
УстановитьПривилегированныйРежим(Истина);
Если Не
ДокументооборотПраваДоступаПовтИсп.ВключеноИспользованиеПравДоступа() Тогда
Возврат;
КонецЕсли;
Если
ДокументооборотПраваДоступаПереопределяемый.ДокументооборотПриЗаписиПравообразующихРегистров(Источник, Отказ) Тогда
Возврат;
КонецЕсли;
Если ТипЗнч(Источник) =
Тип("РегистрСведенийНаборЗаписей.ДескрипторыДоступаДляОбъектов") Тогда
// Обработка изменения владельца файла
Для каждого Эл Из Источник Цикл
ОбновитьПраваСвязанныхДескрипторовПоОбъекту(Эл.Объект);
КонецЦикла;
ИначеЕсли ТипЗнч(Источник) =
Тип("РегистрСведенийНаборЗаписей.ИсполнителиЗадач")
Тогда
ТекущийПриоритетОчередиОбновленияПрав =
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав;
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав = 2;
// Долгая очередь
Попытка
ПриЗаписиИсполнителейЗадач(Источник);
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
Исключение
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
ВызватьИсключение;
КонецПопытки;
ИначеЕсли ТипЗнч(Источник) =
Тип("РегистрСведенийНаборЗаписей.НастройкиПравОбъектов") Тогда
Папка = Источник.Отбор.Объект.Значение;
ОбновитьПраваПапки(Папка);
ИначеЕсли ТипЗнч(Источник) =
Тип("РегистрСведенийНаборЗаписей.НаследованиеНастроекПравОбъектов") Тогда
Папка = Источник.Отбор.Объект.Значение;
ОбновитьПраваПапки(Папка);
ИначеЕсли ТипЗнч(Источник) =
Тип("РегистрСведенийНаборЗаписей.СведенияОПользователях") Тогда
ТекущийПриоритетОчередиОбновленияПрав =
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав;
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав = 2;
// Долгая очередь
Попытка
ПриЗаписиСведенийОПользователях(Источник);
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
Исключение
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
ВызватьИсключение;
КонецПопытки;
КонецЕсли;
КонецПроцедуры
// Обработчик подписки
ДокументооборотПраваДоступаПередЗаписьюПравообразующихКонстант
Процедура
ДокументооборотПраваДоступаПередЗаписьюПравообразующихКонстант(Источник, Отказ) Экспорт
Если Константы[Источник.Метаданные().Имя].Получить() <> Источник.Значение Тогда
Источник.ДополнительныеСвойства.Вставить("ЗначениеИзменено", Истина);
КонецЕсли;
КонецПроцедуры
// Обработчик подписки
ДокументооборотПраваДоступаПриЗаписиПравообразующихКонстант
Процедура
ДокументооборотПраваДоступаПриЗаписиПравообразующихКонстант(Источник, Отказ) Экспорт
УстановитьПривилегированныйРежим(Истина);
Если НЕ
Источник.ДополнительныеСвойства.Свойство("ЗначениеИзменено")
Или НЕ
Источник.ДополнительныеСвойства.ЗначениеИзменено
Тогда
Возврат;
КонецЕсли;
// Вызов переопределяемого метода
Если
ДокументооборотПраваДоступаПереопределяемый.ДокументооборотПраваДоступаПриЗаписиПравообразующихКонстант(Источник, Отказ) Тогда
Возврат;
КонецЕсли;
ТекущийПриоритетОчередиОбновленияПрав =
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав;
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав = 2;
// Долгая очередь
Попытка
ПриЗаписиКонстанты(Источник);
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
Исключение
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
ТекущийПриоритетОчередиОбновленияПрав;
ВызватьИсключение;
КонецПопытки;
КонецПроцедуры
// Обработчик подписки
ДокументооборотПраваДоступаПередЗаписьюПапок
Процедура
ДокументооборотПраваДоступаПередЗаписьюПапок(Источник, Отказ) Экспорт
УстановитьПривилегированныйРежим(Истина);
РегистрыСведений.НаследованиеНастроекПравОбъектов.Обновить(Источник);
УстановитьПривилегированныйРежим(Ложь);
КонецПроцедуры
// Обработчик регламентного задания
ДокументооборотОбновлениеПравДоступаОперативное
Процедура
ДокументооборотОбновлениеПравДоступаОперативное() Экспорт
ОбновитьПовторноИспользуемыеЗначения();
ОбработатьОчередьОбновленияПравДоступа(1);
КонецПроцедуры
// Обработчик регламентного задания
ДокументооборотОбновлениеПравДоступаДолгое
Процедура
ДокументооборотОбновлениеПравДоступаДолгое() Экспорт
УстановитьПривилегированныйРежим(Истина);
ОбновитьПовторноИспользуемыеЗначения();
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав = 2;
// Проверка на необходимость полного обновления прав доступа
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ ПЕРВЫЕ 1
|	ИСТИНА КАК ЕстьЗаписи
|ИЗ
|	РегистрСведений.ПраваПоДескрипторамДоступа КАК ПраваПоДескрипторамДоступа";
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
// Полное обновление всех прав
ДокументооборотПраваДоступа.УдалитьВсеДанныеОПравахДоступа(Истина);
ДокументооборотПраваДоступа.ОбновитьПраваВсехДанных();
КонецЕсли;
ОбработатьОчередьОбновленияПравДоступа(2);
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав = 1;
КонецПроцедуры
Процедура
ОбработатьОчередьОбновленияПравДоступа(Приоритет)
Экспорт
УстановитьПривилегированныйРежим(Истина);
ПараметрыСеанса.ПриоритетОчередиОбновленияПрав =
Приоритет;
ЗаписьЖурналаРегистрации(
НСтр("ru = 'Права доступа'"),
УровеньЖурналаРегистрации.Информация,,
,
НСтр("ru = 'Начато обновление прав доступа'"));
Количество =
РегистрыСведений.ОчередьОбновленияПравДоступа.ОбработатьПорцию();
Пока Количество <> 0 Цикл
Количество =
РегистрыСведений.ОчередьОбновленияПравДоступа.ОбработатьПорцию();
КонецЦикла;
ЗаписьЖурналаРегистрации(
НСтр("ru = 'Права доступа'"),
УровеньЖурналаРегистрации.Информация,,
,
НСтр("ru = 'Закончено обновление прав доступа'"));
КонецПроцедуры
// Обработчик регламентного задания
ДокументооборотУдалениеУстаревшихПравДоступа
Процедура
ДокументооборотУдалениеУстаревшихПравДоступа()
Экспорт
УстановитьПривилегированныйРежим(Истина);
ЗаписьЖурналаРегистрации(
НСтр("ru = 'Права доступа'"),
УровеньЖурналаРегистрации.Информация,,
,
НСтр("ru = 'Начато удаление устаревших прав доступа'"));
Справочники.ДескрипторыДоступаОбъектов.УдалитьНеиспользуемые();
```

```
Справочники.ДескрипторыДоступаФайлов.УдалитьНеиспользуемые();
Справочники.ДескрипторыДоступаРегистров.УдалитьНеиспользуемые();
ЗаписьЖурналаРегистрации(
НСтр("ru = 'Права доступа'"),
УровеньЖурналаРегистрации.Информация,,
,
НСтр("ru = 'Закончено удаление устаревших прав доступа'"));
КонецПроцедуры
// Обновляет права всех дескрипторов, которые зависят от указанного дескриптора
Процедура
ОбновитьПраваСвязанныхДескрипторовПоДескриптору(Дескриптор) Экспорт
УстановитьПривилегированныйРежим(Истина);
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаДляОбъектов.Объект
|ИЗ
|	РегистрСведений.ДескрипторыДоступаДляОбъектов КАК ДескрипторыДоступаДляОбъектов
|ГДЕ
|	ДескрипторыДоступаДляОбъектов.Дескриптор = &Дескриптор";
Запрос.УстановитьПараметр("Дескриптор", Дескриптор);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
ОбновитьПраваСвязанныхДескрипторовПоОбъекту(ВыборкаДетальныеЗаписи.Объект);
КонецЦикла;
КонецПроцедуры
// Обновляет права всех дескрипторов, которые зависят от указанных
// групп доступа корреспондентов
Процедура
ОбновитьПраваПоГруппамДоступаКорреспондентов(ГруппыДоступа) Экспорт
Если ГруппыДоступа.Количество() = 0 Тогда
Возврат;
КонецЕсли;
// ДескрипторыДоступаОбъектов
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗЛИЧНЫЕ
|	ДескрипторыДоступаОбъектовКорреспонденты.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов.Корреспонденты КАК ДескрипторыДоступаОбъектовКорреспонденты
|ГДЕ
|	ДескрипторыДоступаОбъектовКорреспонденты.ГруппаДоступа В(&ГруппыДоступа)";
Запрос.УстановитьПараметр("ГруппыДоступа", ГруппыДоступа);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// ДескрипторыДоступаРегистров
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаРегистров.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаРегистров КАК ДескрипторыДоступаРегистров
|ГДЕ
|	ДескрипторыДоступаРегистров.ОбъектДоступа1 В(&ГруппыДоступа)
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	ДескрипторыДоступаРегистров.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаРегистров КАК ДескрипторыДоступаРегистров
|ГДЕ
|	ДескрипторыДоступаРегистров.ОбъектДоступа2 В(&ГруппыДоступа)
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	ДескрипторыДоступаРегистров.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаРегистров КАК ДескрипторыДоступаРегистров
|ГДЕ
|	ДескрипторыДоступаРегистров.ОбъектДоступа3 В(&ГруппыДоступа)";
Запрос.УстановитьПараметр("ГруппыДоступа", ГруппыДоступа);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаРегистров.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
КонецПроцедуры
// Обновляет права всех дескрипторов, которые зависят от указанных
// групп доступа физлиц
Процедура
ОбновитьПраваПоГруппамДоступаФизлиц(ГруппыДоступа) Экспорт
Если ГруппыДоступа.Количество() = 0 Тогда
Возврат;
КонецЕсли;
// ДескрипторыДоступаОбъектов
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗЛИЧНЫЕ
|	ДескрипторыДоступаОбъектовФизическиеЛица.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов.ФизическиеЛица КАК ДескрипторыДоступаОбъектовФизическиеЛица
|ГДЕ
|	ДескрипторыДоступаОбъектовФизическиеЛица.ГруппаДоступа В(&ГруппыДоступа)";
Запрос.УстановитьПараметр("ГруппыДоступа", ГруппыДоступа);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// ДескрипторыДоступаРегистров
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаРегистров.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаРегистров КАК ДескрипторыДоступаРегистров
|ГДЕ
|	ДескрипторыДоступаРегистров.ОбъектДоступа1 В(&ГруппыДоступа)
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	ДескрипторыДоступаРегистров.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаРегистров КАК ДескрипторыДоступаРегистров
|ГДЕ
|	ДескрипторыДоступаРегистров.ОбъектДоступа2 В(&ГруппыДоступа)
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	ДескрипторыДоступаРегистров.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаРегистров КАК ДескрипторыДоступаРегистров
|ГДЕ
|	ДескрипторыДоступаРегистров.ОбъектДоступа3 В(&ГруппыДоступа)";
Запрос.УстановитьПараметр("ГруппыДоступа", ГруппыДоступа);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаРегистров.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
КонецПроцедуры
// Обновляет права файлов, владельцем которых является указанный объект
Процедура ОбновитьПраваФайловОбъекта(Объект) Экспорт
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ ПЕРВЫЕ 1
|	Файлы.Ссылка
|ИЗ
|	Справочник.Файлы КАК Файлы
|ГДЕ
|	Файлы.ВладелецФайла = &ВладелецФайла";
Запрос.УстановитьПараметр("ВладелецФайла", Объект.Ссылка);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Если ВыборкаДетальныеЗаписи.Следующий() Тогда
// Определение дескриптора
Дескриптор = ОпределитьДескрипторДоступаФайлаПоВладельцу(Объект.Ссылка);
// Вычисление права
Если Дескриптор <> Неопределено Тогда
Справочники.ДескрипторыДоступаФайлов.ОбновитьПрава(Дескриптор);
КонецЕсли;
КонецЕсли;
КонецПроцедуры
////////////////////////////////////////////////////////////////////////////////
// Служебные методы
Функция ВсеРуководителиПользователя(Пользователь)
МассивРуководителей = Новый Массив;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	СведенияОПользователях.Подразделение КАК Подразделение
|ИЗ
|	РегистрСведений.СведенияОПользователях КАК СведенияОПользователях
|ГДЕ
|	СведенияОПользователях.Пользователь = &Пользователь";
Запрос.УстановитьПараметр("Пользователь", Пользователь);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат МассивРуководителей;
КонецЕсли;
Выборка = Результат.Выбрать();
Выборка.Следующий();
Подразделение = Выборка.Подразделение;
Пока ЗначениеЗаполнено(Подразделение) Цикл
Руководитель = Подразделение.Руководитель;
Если ЗначениеЗаполнено(Руководитель) И Руководитель <> Пользователь Тогда
МассивРуководителей.Добавить(Руководитель);
КонецЕсли;
Подразделение = Подразделение.Родитель;
КонецЦикла;
Возврат МассивРуководителей;
КонецФункции
Функция ПолучитьДескрипторДоступа(ОбъектДоступа, Немедленно = Ложь)
УстановитьПривилегированныйРежим(Истина);
НовыйДескриптор = Справочники.ДескрипторыДоступаОбъектов.СоздатьНовыйДескриптор(ОбъектДоступа);
ГотовыйДескриптор = Справочники.ДескрипторыДоступаОбъектов.НайтиДескрипторПоОбразцу(НовыйДескриптор);
Если ЗначениеЗаполнено(ГотовыйДескриптор) Тогда
Возврат ГотовыйДескриптор;
КонецЕсли;
// Запись нового дескриптора
Если Немедленно Тогда
НовыйДескриптор.ДополнительныеСвойства.Вставить("НемедленноРассчитатьПрава", Истина);
КонецЕсли;
НовыйДескриптор.Записать();
Возврат НовыйДескриптор.Ссылка;
КонецФункции
Функция ПолучитьДескрипторДоступаФайла(ВладелецФайла, Немедленно = Ложь)
УстановитьПривилегированныйРежим(Истина);
// Поиск существующего дескриптора
ДескрипторВладельца = РегистрыСведений.ДескрипторыДоступаДляОбъектов.НайтиДескрипторДляОбъекта(ВладелецФайла);
// Владелец файла может не участвовать в механизме прав доступа
Если ДескрипторВладельца = Неопределено Тогда
Возврат Неопределено;
КонецЕсли;
ГотовыйДескриптор = НайтиДескрипторФайла(ДескрипторВладельца);
Если ЗначениеЗаполнено(ГотовыйДескриптор) Тогда
// Запись соответствия владельца файла и дескриптора доступа файла
РегистрыСведений.ДескрипторыДоступаДляФайлов.Сохранить(ГотовыйДескриптор.Ссылка, ВладелецФайла.Ссылка);
Возврат ГотовыйДескриптор;
КонецЕсли;
// Формирование нового дескриптора
НовыйДескриптор = Справочники.ДескрипторыДоступаФайлов.СоздатьЭлемент();
НовыйДескриптор.ДескрипторВладельца = ДескрипторВладельца;
// Проверяем, что тип владельца файла указан в допустимах типах
// реквизита ВладелецФайла
Если Не ЗначениеЗаполнено(НовыйДескриптор.ДескрипторВладельца) Тогда
ТекстОшибки = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
```

```
НСтр("ru = 'Не указан тип владельца файла (%1).'"),
ВладелецФайла.Метаданные().ПолноеИмя());
ВызватьИсключение ТекстОшибки;
КонецЕсли;
// Начало транзакции т.к. дескриптор и его привязка к
владельцу файла должны
// записываться строго вместе
ВнешняяТранзакция = ТранзакцияАктивна();
Если Не ВнешняяТранзакция Тогда
НачатьТранзакцию();
КонецЕсли;
// Запись дескриптора
Если Немедленно Тогда
НовыйДескриптор.ДополнительныеСвойства.Вставить("Нем
едленноРассчитатьПрава", Истина);
КонецЕсли;
НовыйДескриптор.Записать();
// Запись соответствия владельца файла и дескриптора
доступа файла
РегистрыСведений.ДескрипторыДоступаДляФайлов.Сохран
ить(НовыйДескриптор.Ссылка, ВладелецФайла.Ссылка);
// Завершение транзакции
Если Не ВнешняяТранзакция Тогда
ЗафиксироватьТранзакцию();
КонецЕсли;
Возврат НовыйДескриптор.Ссылка;
КонецФункции
Функция НайтиДескрипторФайла(ДескрипторВладельца)
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаФайлов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаФайлов КАК
ДескрипторыДоступаФайлов
|ГДЕ
|
	ДескрипторыДоступаФайлов.ДескрипторВлад
ельца = &ДескрипторВладельца";
Запрос.УстановитьПараметр("ДескрипторВладельца",
ДескрипторВладельца);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Если ВыборкаДетальныеЗаписи.Следующий() Тогда
Возврат ВыборкаДетальныеЗаписи.Ссылка;
КонецЕсли;
Возврат Неопределено;
КонецФункции
Функция
ПолучитьДескрипторДоступаЗаписиНабораЗаписей(Иденти
фикаторОбъектаМетаданных, Запись, Немедленно = Ложь)
УстановитьПривилегированныйРежим(Истина);
НовыйДескриптор =
Справочники.ДескрипторыДоступаРегистров.СоздатьНовый
Дескриптор(ИдентификаторОбъектаМетаданных, Запись);
ГотовыйДескриптор =
Справочники.ДескрипторыДоступаРегистров.НайтиДескрип
торПоОбразцу(НовыйДескриптор);
Если ЗначениеЗаполнено(ГотовыйДескриптор) Тогда
Возврат ГотовыйДескриптор;
КонецЕсли;
// Запись нового дескриптора
Если Немедленно Тогда
НовыйДескриптор.ДополнительныеСвойства.Вставить("Нем
едленноРассчитатьПрава", Истина);
КонецЕсли;
НовыйДескриптор.Записать();
Возврат НовыйДескриптор.Ссылка;
КонецФункции
Процедура
ОбновитьПраваСвязанныхДескрипторовПоОбъекту(Объект)
УстановитьПривилегированныйРежим(Истина);
// Вызов переопределяемого метода
ДокументооборотПраваДоступаПереопределяемый.Обновит
ьПраваСвязанныхДескрипторовПоОбъекту(Объект);
// Обновление прав на файлы
ОбновитьПраваФайловОбъекта(Объект);
// Обновление прав на регистры
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаРегистров.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаРегистров
КАК ДескрипторыДоступаРегистров
|ГДЕ
|
	ДескрипторыДоступаРегистров.ОбъектДоступ
а1 = &ОбъектДоступа1
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	ДескрипторыДоступаРегистров.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаРегистров
КАК ДескрипторыДоступаРегистров
|ГДЕ
|
	ДескрипторыДоступаРегистров.ОбъектДоступ
а2 = &ОбъектДоступа2
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	ДескрипторыДоступаРегистров.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаРегистров
КАК ДескрипторыДоступаРегистров
|ГДЕ
|
	ДескрипторыДоступаРегистров.ОбъектДоступ
а3 = &ОбъектДоступа3";
Запрос.УстановитьПараметр("ОбъектДоступа1",
Объект.Ссылка);
Запрос.УстановитьПараметр("ОбъектДоступа2",
Объект.Ссылка);
Запрос.УстановитьПараметр("ОбъектДоступа3",
Объект.Ссылка);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаРегистров.ОбновитьПрав
а(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// Обновление прав на задачи процессов
Если ЭтоБизнесПроцесс(Объект.Ссылка) Тогда
ПраваДоступаНаБизнесПроцессыВызовСервера.ОбновитьП
раваНаЗадачиБизнесПроцесса(Объект.Ссылка);
ПраваДоступаНаБизнесПроцессыВызовСервера.ОбновитьП
раваНаДочерниеБизнесПроцессы(Объект.Ссылка);
КонецЕсли;
КонецПроцедуры
// Обновляет права папки и всех ее подпапок
Процедура ОбновитьПраваПапки(Папка)
Если Не ЗначениеЗаполнено(Папка) Тогда
Возврат;
КонецЕсли;
УстановитьПривилегированныйРежим(Истина);
ИмяТаблицы = Папка.Метаданные().ПолноеИмя();
// Отбор всех папок в иерархии
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ТаблицаПапок.Ссылка
|ИЗ
|	#Таблица КАК ТаблицаПапок
|ГДЕ
|	ТаблицаПапок.Ссылка В ИЕРАРХИИ
(&Папка)";
Запрос.Текст = СтрЗаменить(Запрос.Текст, "#Таблица",
ИмяТаблицы);
Запрос.УстановитьПараметр("Папка", Папка);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
// Обновление прав для каждой папки
ОпределитьДескрипторДоступаОбъекта(ВыборкаДетальные
Записи.Ссылка);
ОбновитьПраваДоступаДляОбъекта(ВыборкаДетальныеЗапи
си.Ссылка);
ОбновитьПраваДескрипторовЗависящихОтПапки(ВыборкаД
етальныеЗаписи.Ссылка);
КонецЦикла;
КонецПроцедуры
Процедура
ОбновитьПраваДескрипторовЗависящихОтПапки(Папка)
УстановитьПривилегированныйРежим(Истина);
// Обновление дескрипторов объектов
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов
КАК ДескрипторыДоступаОбъектов
|ГДЕ
|	ДескрипторыДоступаОбъектов.Папка =
&Папка";
Запрос.УстановитьПараметр("Папка", Папка);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрав
а(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// Обновление дескрипторов файлов
Справочники.ДескрипторыДоступаФайлов.ОбновитьПрава
ПоВладельцуФайла(Папка);
КонецПроцедуры
Процедура
ОбновитьПраваДескрипторовЗависящихОтПрофиля(Профи
ль)
УстановитьПривилегированныйРежим(Истина);
ЗначенияДоступа = Новый Массив;
Для каждого Эл Из Профиль.ВидыДоступа Цикл
// Пропускаем неиспользуемые виды доступа
Если Эл.ВидДоступа =
ПланыВидовХарактеристик.ВидыДоступа.Дополнительные
Сведения
ИЛИ Эл.ВидДоступа =
ПланыВидовХарактеристик.ВидыДоступа.Пользователи
Тогда
Продолжить;
КонецЕсли;
ВсеЗначенияВида =
ПолучитьВсеЗначенияВидаДоступа(Эл.ВидДоступа);
Для каждого ЗначениеВида Из ВсеЗначенияВида Цикл
ЗначенияДоступа.Добавить(ЗначениеВида.Ссылка);
КонецЦикла;
// Добавление пустой ссылки
ЗначенияДоступа.Добавить(ПолучитьПустуюССылкуДляВи
даДоступа(Эл.ВидДоступа));
КонецЦикла;
ОбновитьПраваПоЗначениямВидовДоступа(ЗначенияДосту
па);
КонецПроцедуры
// Возвращает массив значений доступа, права по которым
определяет указанная
// группа доступа
Функция
ПолучитьЗначенияДоступаГруппыДоступа(ГруппаДоступа)
ЗначенияДоступа = Новый Массив;
Для каждого Эл Из ГруппаДоступа.Профиль.ВидыДоступа
Цикл
// Пропускаем неиспользуемые виды доступа
Если Эл.ВидДоступа =
ПланыВидовХарактеристик.ВидыДоступа.Дополнительные
Сведения
ИЛИ Эл.ВидДоступа =
ПланыВидовХарактеристик.ВидыДоступа.Пользователи
Тогда
Продолжить;
КонецЕсли;
ВсеЗначенияВида =
ПолучитьВсеЗначенияВидаДоступа(Эл.ВидДоступа);
Для каждого ЗначениеВида Из ВсеЗначенияВида Цикл
ЗначенияДоступа.Добавить(ЗначениеВида.Ссылка);
КонецЦикла;
// Добавление пустой ссылки
ЗначенияДоступа.Добавить(ПолучитьПустуюССылкуДляВи
даДоступа(Эл.ВидДоступа));
КонецЦикла;
Возврат ЗначенияДоступа;
КонецФункции
Процедура
ЗаполнитьЗначенияГруппыДоступаПоВидуДоступа(Значени
яДоступа, ГруппаДоступа, ВидДоступа)
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ЗначенияГруппДоступа.ЗначениеДоступа КАК
ЗначениеДоступа,
|	ЗначенияГруппДоступа.ТолькоВидДоступа
КАК Разрешено
|ИЗ
|	РегистрСведений.ЗначенияГруппДоступа КАК
ЗначенияГруппДоступа
|ГДЕ
|	ЗначенияГруппДоступа.ГруппаДоступа =
&ГруппаДоступа
|	И ЗначенияГруппДоступа.ВидДоступа =
&ВидДоступа";
Запрос.УстановитьПараметр("ГруппаДоступа",
ГруппаДоступа.Ссылка);
Запрос.УстановитьПараметр("ВидДоступа", ВидДоступа);
ВыборкаДетальныеЗаписи = Запрос.Выполнить().Выбрать();
Если ВыборкаДетальныеЗаписи.Количество() <> 0 Тогда
Если ВыборкаДетальныеЗаписи.Количество() = 1 Тогда
Если ВыборкаДетальныеЗаписи.Следующий() Тогда
Если
ЗначениеЗаполнено(ВыборкаДетальныеЗаписи.ЗначениеДос
тупа) Тогда
// Указано разрешение или запрещение
ЗначениеДоступа = Новый Структура;
ЗначениеДоступа.Вставить("Значение",
ВыборкаДетальныеЗаписи.ЗначениеДоступа);
ЗначениеДоступа.Вставить("Разрешено",
ВыборкаДетальныеЗаписи.Разрешено);
ЗначенияДоступа.Добавить(ЗначениеДоступа);
Иначе
// Все запрещены
ВсеЗначения =
ПолучитьВсеЗначенияВидаДоступа(ВидДоступа);
Для каждого Эл Из ВсеЗначения Цикл
ЗначениеДоступа = Новый Структура;
ЗначениеДоступа.Вставить("Значение", Эл.Ссылка);
ЗначениеДоступа.Вставить("Разрешено", Ложь);
ЗначенияДоступа.Добавить(ЗначениеДоступа);
КонецЦикла;
КонецЕсли;
КонецЕсли;
Иначе
// Указан список разрешенных или запрещенных
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Если ВыборкаДетальныеЗаписи.ЗначениеДоступа <>
Неопределено Тогда
ЗначениеДоступа = Новый Структура;
ЗначениеДоступа.Вставить("Значение",
ВыборкаДетальныеЗаписи.ЗначениеДоступа);
ЗначениеДоступа.Вставить("Разрешено",
ВыборкаДетальныеЗаписи.Разрешено);
ЗначенияДоступа.Добавить(ЗначениеДоступа);
КонецЕсли;
КонецЦикла;
КонецЕсли;
Иначе
// Все разрешены, добавляем все значения, соответствующие
этому виду доступа
ВсеЗначения =
ПолучитьВсеЗначенияВидаДоступа(ВидДоступа);
Для каждого Эл Из ВсеЗначения Цикл
```

```
ЗначениеДоступа = Новый Структура;
ЗначениеДоступа.Вставить("Значение", Эл.Ссылка);
ЗначениеДоступа.Вставить("Разрешено", Истина);
ЗначенияДоступа.Добавить(ЗначениеДоступа);
КонецЦикла;
КонецЕсли;
КонецПроцедуры
Функция ПолучитьВсеЗначенияВидаДоступа(ВидДоступа)
Типы = ВидДоступа.ТипЗначения.Типы();
Если Типы.Количество() > 1 Тогда
ВызватьИсключение НСтр("ru = 'Число типов вида доступа больше 1.'");
КонецЕсли;
МетаданныеТипа = Метаданные.НайтиПоТипу(Типы[0]);
ИмяТаблицы = МетаданныеТипа.ПолноеИмя();
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	Таблица.Ссылка
|ИЗ
|	#Таблица КАК Таблица";
Запрос.Текст = СтрЗаменить(Запрос.Текст, "#Таблица", ИмяТаблицы);
Возврат Запрос.Выполнить().Выгрузить();
КонецФункции
Функция ПолучитьПустуюССылкуДляВидаДоступа(ВидДоступа)
ПустаяСсылка = Неопределено;
Типы = ВидДоступа.ТипЗначения.Типы();
Если Типы.Количество() > 1 Тогда
ВызватьИсключение НСтр("ru = 'Число типов вида доступа больше 1.'");
КонецЕсли;
МетаданныеТипа = Метаданные.НайтиПоТипу(Типы[0]);
Менеджер = ОбщегоНазначения.МенеджерОбъектаПоПолномуИмени(МетаданныеТипа.ПолноеИмя());
ПустаяСсылка = Менеджер.ПустаяСсылка();
Возврат ПустаяСсылка;
КонецФункции
Процедура ОбновитьПраваЗависящиеОтГруппыДоступа(ГруппаДоступа)
УстановитьПривилегированныйРежим(Истина);
ЗначенияДоступа = ПолучитьЗначенияДоступаГруппыДоступа(ГруппаДоступа);
ОбновитьПраваПоЗначениямВидовДоступа(ЗначенияДоступа);
КонецПроцедуры
// Обновляет права всех дескрипторов доступа в которых есть ссылки на
// указанные значения доступа
Процедура ОбновитьПраваПоЗначениямВидовДоступа(ЗначенияДоступа)
УстановитьПривилегированныйРежим(Истина);
// Поиск и обновление дескрипторов по реквизитам
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов
|ГДЕ
|	ДескрипторыДоступаОбъектов.ВидОбъекта В(&ЗначенияДоступа)
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов
|ГДЕ
|
|	ДескрипторыДоступаОбъектов.ВопросДеятельности В(&ЗначенияДоступа)
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов
|ГДЕ
|	ДескрипторыДоступаОбъектов.ГрифДоступа В(&ЗначенияДоступа)
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов
|ГДЕ
|	ДескрипторыДоступаОбъектов.Организация В(&ЗначенияДоступа)";
Запрос.УстановитьПараметр("ЗначенияДоступа", ЗначенияДоступа);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// Поиск и обновление дескрипторов по группам доступа корреспондентов
ГруппыДоступаКорреспондентов = Новый Массив();
Для каждого Эл Из ЗначенияДоступа Цикл
Если ТипЗнч(Эл) = Тип("СправочникСсылка.ГруппыДоступаКорреспондентов") Тогда
ГруппыДоступаКорреспондентов.Добавить(Эл);
КонецЕсли;
КонецЦикла;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗЛИЧНЫЕ
|
|	ДескрипторыДоступаОбъектовКорреспонденты.Ссылка
|ИЗ
|
|	Справочник.ДескрипторыДоступаОбъектов.Корреспонденты КАК ДескрипторыДоступаОбъектовКорреспонденты
|ГДЕ
|
|	ДескрипторыДоступаОбъектовКорреспонденты.ГруппаДоступа В(&ГруппыДоступаКорреспондентов)";
Запрос.УстановитьПараметр("ГруппыДоступаКорреспондентов", ГруппыДоступаКорреспондентов);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// Поиск и обновление дескрипторов по группам доступа физических лиц
ГруппыДоступаФизЛиц = Новый Массив();
Для каждого Эл Из ЗначенияДоступа Цикл
Если ТипЗнч(Эл) = Тип("СправочникСсылка.ГруппыДоступаФизическихЛиц") Тогда
ГруппыДоступаФизЛиц.Добавить(Эл);
КонецЕсли;
КонецЦикла;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗЛИЧНЫЕ
|
|	ДескрипторыДоступаОбъектовФизическиеЛица.Ссылка
|ИЗ
|
|	Справочник.ДескрипторыДоступаОбъектов.ФизическиеЛица КАК ДескрипторыДоступаОбъектовФизическиеЛица
|ГДЕ
|
|	ДескрипторыДоступаОбъектовФизическиеЛица.ГруппаДоступа В(&ГруппыДоступаФизЛиц)";
Запрос.УстановитьПараметр("ГруппыДоступаФизЛиц", ГруппыДоступаФизЛиц);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
КонецПроцедуры
Процедура ОбновитьПраваПоПодчиненнымПодразделениям(Подразделение)
УстановитьПривилегированныйРежим(Истина);
// Выбор всех подразделений от указанного (включительно)
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	СтруктураАдминистрации.Ссылка
|ИЗ
|	Справочник.СтруктураАдминистрации КАК СтруктураАдминистрации
|ГДЕ
|	СтруктураАдминистрации.Ссылка В ИЕРАРХИИ(&Родитель)";
Запрос.УстановитьПараметр("Родитель", Подразделение);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
ОбновитьПраваСотрудниковПодразделения(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
КонецПроцедуры
Процедура ОбновитьПраваСотрудниковПодразделения(Подразделение)
УстановитьПривилегированныйРежим(Истина);
// Выбор всех дескрипторов, в правах которых встречаются сотрудники
// указанного подразделения
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ПраваПоДескрипторамДоступа.Дескриптор
|ИЗ
|
|	РегистрСведений.ПраваПоДескрипторамДоступа КАК ПраваПоДескрипторамДоступа
|	ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрСведений.СведенияОПользователях КАК СведенияОПользователях
|	ПО ПраваПоДескрипторамДоступа.Пользователь = СведенияОПользователях.Пользователь
|ГДЕ
|	СведенияОПользователях.Подразделение = &Подразделение";
Запрос.УстановитьПараметр("Подразделение", Подразделение);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
ОбновитьПраваДоступаПоДескриптору(ВыборкаДетальныеЗаписи.Дескриптор);
КонецЦикла;
КонецПроцедуры
// Находит дескриптор указанного объекта доступа и немедленно обновляет права
// найденного дескриптора. Если дескриптор не найден, то создает его.
Процедура ОбновитьПраваДоступаДляОбъекта(ОбъектДоступа)
УстановитьПривилегированныйРежим(Истина);
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаДляОбъектов.Дескриптор
|ИЗ
|
|	РегистрСведений.ДескрипторыДоступаДляОбъектов КАК ДескрипторыДоступаДляОбъектов
|ГДЕ
|	ДескрипторыДоступаДляОбъектов.Объект = &ОбъектДоступа";
Запрос.УстановитьПараметр("ОбъектДоступа", ОбъектДоступа);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Если ВыборкаДетальныеЗаписи.Следующий() Тогда
Немедленно = НеДокументооборотПраваДоступаПовтИсп.ОтложенноеОбновлениеПравДоступа();
ОбновитьПраваДоступаПоДескриптору(ВыборкаДетальныеЗаписи.Дескриптор, Немедленно);
Иначе
ОпределитьДескрипторДоступаОбъекта(ОбъектДоступа);
КонецЕсли;
КонецПроцедуры
Процедура ОбновлениеРолейПередЗаписьюОбъектовПередЗаписью(Источник, Отказ) Экспорт
Если ТипЗнч(Источник) = Тип("СправочникОбъект.Пользователи") Тогда
Источник.ДополнительныеСвойства.Вставить("ЭтоНовый", Источник.ЭтоНовый());
ИначеЕсли ТипЗнч(Источник) = Тип("СправочникОбъект.ГруппыПользователей") Тогда
СоставГруппы = Новый ТаблицаЗначений;
СоставГруппы.Колонки.Добавить("Пользователь");
Если Не Источник.ЭтоНовый() Тогда
СоставГруппы = ДокументооборотПраваДоступаПовтИсп.ПолучитьСоставГруппыПользователей(Источник.Ссылка);
КонецЕсли;
Источник.ДополнительныеСвойства.Вставить("СтарыйСоставГруппы", СоставГруппы);
Если Не Источник.ЭтоНовый() Тогда
СтарыйРодитель = ОбщегоНазначения.ПолучитьЗначениеРеквизита(Источник.Ссылка, "Родитель");
Источник.ДополнительныеСвойства.Вставить("СтарыйРодитель", СтарыйРодитель);
КонецЕсли;
ИначеЕсли ТипЗнч(Источник) = Тип("СправочникОбъект.ГруппыДоступа") Тогда
СоставГруппы = Новый ТаблицаЗначений;
СоставГруппы.Колонки.Добавить("Пользователь");
Если Не Источник.ЭтоНовый() Тогда
СоставГруппы = Источник.Ссылка.Пользователи;
КонецЕсли;
Источник.ДополнительныеСвойства.Вставить("СтарыйСоставГруппы", СоставГруппы);
КонецЕсли;
КонецПроцедуры
Процедура ОбновлениеРолейПриЗаписиОбъектовПриЗаписи(Источник, Отказ) Экспорт
Если ТипЗнч(Источник) = Тип("СправочникОбъект.Пользователи")
И (Источник.ДополнительныеСвойства.ЭтоНовый
Или (Источник.ДополнительныеСвойства.Свойство("ДобавленНовыйПользовательИБ")
```

```
И
Источник.ДополнительныеСвойства.ДобавленНовыйПользо
вательИБ)) Тогда
МассивПользователей = Новый Массив;
МассивПользователей.Добавить(Источник.Ссылка);
УправлениеДоступом.ОбновитьРолиПользователей(Массив
Пользователей);
РегистрыСведений.ДокументооборотПользователиГруппДо
ступа.ОбновитьВсе();
КонецЕсли;
Если ТипЗнч(Источник) =
Тип("СправочникОбъект.ГруппыПользователей") Тогда
МассивИзменений = Новый Массив;
ПолноеОбновление = Ложь;
Если
Источник.ДополнительныеСвойства.Свойство("СтарыйРоди
тель") Тогда
СтарыйРодитель =
Источник.ДополнительныеСвойства.СтарыйРодитель;
Если СтарыйРодитель <> Источник.Родитель Тогда
// Полное обновление
ПолноеОбновление = Истина;
СоставГруппы =
ДокументооборотПраваДоступаПовтИсп.ПолучитьСоставГр
уппыПользователей(Источник.Ссылка);
Для Каждого Строка Из СоставГруппы Цикл
МассивИзменений.Добавить(Строка.Пользователь);
КонецЦикла;
КонецЕсли;
КонецЕсли;
Если Не ПолноеОбновление Тогда
// Поиск и обновление отличий в составе группы
пользователей
НовыйСоставГруппы =
ДокументооборотПраваДоступаПовтИсп.ПолучитьСоставГр
уппыПользователей(Источник.Ссылка);
СтарыйСоставГруппы =
Источник.ДополнительныеСвойства.СтарыйСоставГруппы;
Для Каждого Строка Из НовыйСоставГруппы Цикл
Пользователь = Строка.Пользователь;
Если СтарыйСоставГруппы.Найти(Пользователь,
"Пользователь") = Неопределено Тогда
Если МассивИзменений.Найти(Пользователь) =
Неопределено Тогда
МассивИзменений.Добавить(Пользователь);
КонецЕсли;
КонецЕсли;
КонецЦикла;
Для Каждого Строка Из СтарыйСоставГруппы Цикл
Пользователь = Строка.Пользователь;
Если НовыйСоставГруппы.Найти(Пользователь,
"Пользователь") = Неопределено Тогда
Если МассивИзменений.Найти(Пользователь) =
Неопределено Тогда
МассивИзменений.Добавить(Пользователь);
КонецЕсли;
КонецЕсли;
КонецЦикла;
КонецЕсли;
УправлениеДоступом.ОбновитьРолиПользователей(Массив
Изменений);
РегистрыСведений.ДокументооборотПользователиГруппДо
ступа.ОбновитьВсе();
КонецЕсли;
Если ТипЗнч(Источник) =
Тип("СправочникОбъект.ГруппыДоступа") Тогда
НовыйСоставГруппы = Источник.Пользователи;
СтарыйСоставГруппы =
Источник.ДополнительныеСвойства.СтарыйСоставГруппы;
МассивИзменений = Новый Массив;
Для Каждого Строка Из НовыйСоставГруппы Цикл
Пользователь = Строка.Пользователь;
Если СтарыйСоставГруппы.Найти(Пользователь,
"Пользователь") = Неопределено Тогда
Если МассивИзменений.Найти(Пользователь) =
Неопределено Тогда
МассивИзменений.Добавить(Пользователь);
КонецЕсли;
КонецЕсли;
КонецЦикла;
Для Каждого Строка Из СтарыйСоставГруппы Цикл
Пользователь = Строка.Пользователь;
Если НовыйСоставГруппы.Найти(Пользователь,
"Пользователь") = Неопределено Тогда
Если МассивИзменений.Найти(Пользователь) =
Неопределено Тогда
МассивИзменений.Добавить(Пользователь);
КонецЕсли;
КонецЕсли;
КонецЦикла;
ОбновитьМассивПользователейИГруппПользователей(Масс
ивИзменений);
РегистрыСведений.ДокументооборотПользователиГруппДо
ступа.ОбновитьПоГруппеДоступа(Источник.Ссылка);
КонецЕсли;
Если ТипЗнч(Источник) =
Тип("СправочникОбъект.ПрофилиГруппДоступа") Тогда
// Группы доступа
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ГруппыДоступа.Ссылка КАК Ссылка
|ИЗ
|	Справочник.ГруппыДоступа КАК
ГруппыДоступа
|ГДЕ
|	ГруппыДоступа.Профиль = &Профиль";
Запрос.УстановитьПараметр("Профиль", Источник.Ссылка);
Результат =
Запрос.Выполнить().Выгрузить().ВыгрузитьКолонку("Ссыл
ка");
Для Каждого ГруппаДоступа Из Результат Цикл
МассивИзменений = Новый Массив;
Для Каждого Строка Из ГруппаДоступа.Пользователи Цикл
МассивИзменений.Добавить(Строка.Пользователь);
КонецЦикла;
ОбновитьМассивПользователейИГруппПользователей(Масс
ивИзменений);
РегистрыСведений.ДокументооборотПользователиГруппДо
ступа.ОбновитьПоГруппеДоступа(ГруппаДоступа);
КонецЦикла;
КонецЕсли;
КонецПроцедуры
Процедура
ОбновитьМассивПользователейИГруппПользователей(Масс
ивИзменений)
МассивИзмененийРазвернутый = Новый Массив;
Для Каждого ПользовательИлиГруппа Из МассивИзменений
Цикл
Если ТипЗнч(ПользовательИлиГруппа) =
Тип("СправочникСсылка.Пользователи") Тогда
Если
МассивИзмененийРазвернутый.Найти(ПользовательИлиГру
ппа) = Неопределено Тогда
МассивИзмененийРазвернутый.Добавить(ПользовательИли
Группа);
КонецЕсли;
ИначеЕсли ТипЗнч(ПользовательИлиГруппа) =
Тип("СправочникСсылка.ГруппыПользователей") Тогда
СоставГруппы =
ДокументооборотПраваДоступаПовтИсп.ПолучитьСоставГр
уппыПользователей(ПользовательИлиГруппа);
Для Каждого Строка Из СоставГруппы Цикл
Если
МассивИзмененийРазвернутый.Найти(Строка.Пользователь
) = Неопределено Тогда
МассивИзмененийРазвернутый.Добавить(Строка.Пользоват
ель);
КонецЕсли;
КонецЦикла;
КонецЕсли;
КонецЦикла;
УправлениеДоступом.ОбновитьРолиПользователей(Массив
ИзмененийРазвернутый);
КонецПроцедуры
Процедура
ОбновитьПраваЗависимыхДанных(ОбъектСсылка)
ДокументооборотПраваДоступаПереопределяемый.Обновит
ьПраваЗависимыхДанных(ОбъектСсылка);
КонецПроцедуры
Процедура ОбновитьПраваПустыхДескрипторов()
УстановитьПривилегированныйРежим(Истина);
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗЛИЧНЫЕ
|	ПраваПоДескрипторамДоступа.Дескриптор
|ИЗ
|	РегистрСведений.ПраваПоДескрипторамДосту
па КАК ПраваПоДескрипторамДоступа
|		ВНУТРЕННЕЕ СОЕДИНЕНИЕ
Справочник.ДескрипторыДоступаОбъектов КАК
ДескрипторыДоступаОбъектов
|		ПО
ПраваПоДескрипторамДоступа.Дескриптор =
ДескрипторыДоступаОбъектов.Ссылка
|ГДЕ
|	ДескрипторыДоступаОбъектов.ЭтоПустойДес
криптор";
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Немедленно = Не
ДокументооборотПраваДоступаПовтИсп.ОтложенноеОбнов
лениеПравДоступа();
ОбновитьПраваДоступаПоДескриптору(ВыборкаДетальные
Записи.Дескриптор, Немедленно);
КонецЦикла;
КонецПроцедуры
// Возвращает Истина если переданная ссылка является
ссылкой на любой бизнес-процесс
Функция ЭтоБизнесПроцесс(Ссылка)
МетаданныеОбъекта = Ссылка.Метаданные();
Возврат
Метаданные.БизнесПроцессы.Содержит(МетаданныеОбъект
а);
КонецФункции
// Обрабатывает запись набора записей РС
ИсполнителиЗадач
Процедура ПриЗаписиИсполнителейЗадач(Источник)
Если Источник.Количество() = 0 Тогда
// Обработка удаления набора записей
РольИсполнителя =
Источник.Отбор.РольИсполнителя.Значение;
ОсновнойОбъектАдресации =
Источник.Отбор.ОсновнойОбъектАдресации.Значение;
ДополнительныйОбъектАдресации =
Источник.Отбор.ДополнительныйОбъектАдресации.Значен
ие;
// Поиск дескрипторов по ТЧ Пользователи
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗЛИЧНЫЕ
|	ДескрипторыДоступаОбъектовПользователи.С
сылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов.По
льзователи КАК
ДескрипторыДоступаОбъектовПользователи
|ГДЕ
|	ДескрипторыДоступаОбъектовПользователи.П
ользователь = &Пользователь
|	И
ДескрипторыДоступаОбъектовПользователи.ОсновнойОбъе
ктАдресации = &ОсновнойОбъектАдресации
|	И
ДескрипторыДоступаОбъектовПользователи.Дополнительн
ыйОбъектАдресации =
&ДополнительныйОбъектАдресации";
Запрос.УстановитьПараметр("ДополнительныйОбъектАдрес
ации", ДополнительныйОбъектАдресации);
Запрос.УстановитьПараметр("ОсновнойОбъектАдресации",
ОсновнойОбъектАдресации);
Запрос.УстановитьПараметр("Пользователь",
РольИсполнителя);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрав
а(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// Поиск дескрипторов по ТЧ РабочаяГруппа
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗЛИЧНЫЕ
|	ДескрипторыДоступаОбъектовРабочаяГруппа.
Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов.Ра
бочаяГруппа КАК
ДескрипторыДоступаОбъектовРабочаяГруппа
|ГДЕ
|	ДескрипторыДоступаОбъектовРабочаяГруппа.
Участник = &Участник
|	И
ДескрипторыДоступаОбъектовРабочаяГруппа.ОсновнойОбъ
ектАдресации = &ОсновнойОбъектАдресации
|	И
ДескрипторыДоступаОбъектовРабочаяГруппа.Дополнитель
ныйОбъектАдресации =
&ДополнительныйОбъектАдресации";
Запрос.УстановитьПараметр("ДополнительныйОбъектАдрес
ации", ДополнительныйОбъектАдресации);
Запрос.УстановитьПараметр("ОсновнойОбъектАдресации",
ОсновнойОбъектАдресации);
Запрос.УстановитьПараметр("Участник",
РольИсполнителя);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрав
а(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
Иначе
// Обработка записи нового набора записей
Для каждого Эл Из Источник Цикл
РольИсполнителя = Эл.РольИсполнителя;
ОсновнойОбъектАдресации =
Эл.ОсновнойОбъектАдресации;
ДополнительныйОбъектАдресации =
Эл.ДополнительныйОбъектАдресации;
// Поиск дескрипторов по ТЧ Пользователи
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗЛИЧНЫЕ
|	ДескрипторыДоступаОбъектовПользователи.С
сылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов.По
льзователи КАК
ДескрипторыДоступаОбъектовПользователи
|ГДЕ
|	ДескрипторыДоступаОбъектовПользователи.П
ользователь = &Пользователь
|	И
ДескрипторыДоступаОбъектовПользователи.ОсновнойОбъе
ктАдресации = &ОсновнойОбъектАдресации
|	И
ДескрипторыДоступаОбъектовПользователи.Дополнительн
ыйОбъектАдресации =
&ДополнительныйОбъектАдресации";
```

```
Запрос.УстановитьПараметр("ДополнительныйОбъектАдресации", ДополнительныйОбъектАдресации);
Запрос.УстановитьПараметр("ОсновнойОбъектАдресации", ОсновнойОбъектАдресации);
Запрос.УстановитьПараметр("Пользователь", РольИсполнителя);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// Поиск дескрипторов по ТЧ РабочаяГруппа
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗЛИЧНЫЕ
|
	ДескрипторыДоступаОбъектовРабочаяГруппа.Ссылка
|ИЗ
|
	Справочник.ДескрипторыДоступаОбъектов.РабочаяГруппа КАК ДескрипторыДоступаОбъектовРабочаяГруппа
|ГДЕ
|
	ДескрипторыДоступаОбъектовРабочаяГруппа.Участник = &Участник
|	И ДескрипторыДоступаОбъектовРабочаяГруппа.ОсновнойОбъектАдресации = &ОсновнойОбъектАдресации
|	И ДескрипторыДоступаОбъектовРабочаяГруппа.ДополнительныйОбъектАдресации = &ДополнительныйОбъектАдресации";
Запрос.УстановитьПараметр("ДополнительныйОбъектАдресации", ДополнительныйОбъектАдресации);
Запрос.УстановитьПараметр("ОсновнойОбъектАдресации", ОсновнойОбъектАдресации);
Запрос.УстановитьПараметр("Участник", РольИсполнителя);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
КонецЦикла;
КонецЕсли;
КонецПроцедуры
// Обрабатывает запись набора записей РС СведенияОПользователях
Процедура ПриЗаписиСведенийОПользователях(Источник)
Если Константы.ДобавлятьРуководителямДоступПодчиненных.Получить() Тогда
Если Источник.ДополнительныеСвойства.Свойство("СтароеПодразделение") Тогда
// Если подразделение пользователя было изменено, то
// выполняется обновление прав, зависящих от пользователя
Немедленно = Не ДокументооборотПраваДоступаПовтИсп.ОтложенноеОбновлениеПравДоступа();
ЗначениеОтбора = Источник.Отбор.Пользователь.Значение;
Если ЗначениеЗаполнено(ЗначениеОтбора) Тогда
ОбновитьПраваПоПользователю(ЗначениеОтбора, Немедленно);
Иначе
Для каждого Эл Из Источник Цикл
ОбновитьПраваПоПользователю(Эл.Пользователь, Немедленно);
КонецЦикла;
КонецЕсли;
КонецЕсли;
КонецЕсли;
КонецПроцедуры
// Обрабатывает запись правообразующих констант
Процедура ПриЗаписиКонстанты(Источник)
Если ТипЗнч(Источник) = Тип("КонстантаМенеджерЗначения.ИспользоватьУчетПоОрганизациям") Тогда
// Обработка всех дескрипторов, у которых установлен реквизит Организация
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов
|ГДЕ
|	ДескрипторыДоступаОбъектов.Организация <>&ПустаяОрганизация";
Запрос.УстановитьПараметр("ПустаяОрганизация", Справочники.Организации.ПустаяСсылка());
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Дескриптор = ВыборкаДетальныеЗаписи.Ссылка;
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(Дескриптор);
ОбновитьПраваСвязанныхДескрипторовПоДескриптору(Дескриптор);
КонецЦикла;
ИначеЕсли ТипЗнч(Источник) = Тип("КонстантаМенеджерЗначения.ИспользоватьГрифыДоступа") Тогда
// Обработка всех дескрипторов, у которых установлен реквизит ГрифДоступа
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов
|ГДЕ
|	ДескрипторыДоступаОбъектов.ГрифДоступа <>&ГрифДоступа";
Запрос.УстановитьПараметр("ГрифДоступа", Справочники.ГрифыДоступа.ПустаяСсылка());
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Дескриптор = ВыборкаДетальныеЗаписи.Ссылка;
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(Дескриптор);
ОбновитьПраваСвязанныхДескрипторовПоДескриптору(Дескриптор);
КонецЦикла;
ИначеЕсли ТипЗнч(Источник) = Тип("КонстантаМенеджерЗначения.ИспользоватьВопросыДеятельности") Тогда
// Обработка всех дескрипторов, у которых установлен реквизит ВопросДеятельности
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов
|ГДЕ
|
	ДескрипторыДоступаОбъектов.ВопросДеятельности <>&ВопросДеятельности";
Запрос.УстановитьПараметр("ВопросДеятельности", Справочники.ВопросыДеятельности.ПустаяСсылка());
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Дескриптор = ВыборкаДетальныеЗаписи.Ссылка;
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(Дескриптор);
ОбновитьПраваСвязанныхДескрипторовПоДескриптору(Дескриптор);
КонецЦикла;
ИначеЕсли ТипЗнч(Источник) = Тип("КонстантаМенеджерЗначения.ИспользоватьВидыИсходящихДокументов") Тогда
// Обработка всех дескрипторов, у которых установлен реквизит ВидОбъекта
// типа Справочник.ВидыИсходящихДокументов
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов
|ГДЕ
|	ДескрипторыДоступаОбъектов.ВидОбъекта <>&ВидОбъекта
|	И ТИПЗНАЧЕНИЯ(ДескрипторыДоступаОбъектов.ВидОбъекта) = ТИП(Справочник.ВидыИсходящихДокументов)";
Запрос.УстановитьПараметр("ВидОбъекта", Справочники.ВидыИсходящихДокументов.ПустаяСсылка());
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Дескриптор = ВыборкаДетальныеЗаписи.Ссылка;
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(Дескриптор);
ОбновитьПраваСвязанныхДескрипторовПоДескриптору(Дескриптор);
КонецЦикла;
ИначеЕсли ТипЗнч(Источник) = Тип("КонстантаМенеджерЗначения.ИспользоватьВидыВходящихДокументов") Тогда
// Обработка всех дескрипторов, у которых установлен реквизит ВидОбъекта
// типа Справочник.ВидыВходящихДокументов
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов
|ГДЕ
|	ДескрипторыДоступаОбъектов.ВидОбъекта <>&ВидОбъекта
|	И ТИПЗНАЧЕНИЯ(ДескрипторыДоступаОбъектов.ВидОбъекта) = ТИП(Справочник.ВидыВходящихДокументов)";
Запрос.УстановитьПараметр("ВидОбъекта", Справочники.ВидыВходящихДокументов.ПустаяСсылка());
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Дескриптор = ВыборкаДетальныеЗаписи.Ссылка;
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(Дескриптор);
ОбновитьПраваСвязанныхДескрипторовПоДескриптору(Дескриптор);
КонецЦикла;
ИначеЕсли ТипЗнч(Источник) = Тип("КонстантаМенеджерЗначения.ИспользоватьВидыВнутреннихДокументов") Тогда
// Обработка всех дескрипторов, у которых установлен реквизит ВидОбъекта
// типа Справочник.ВидыВнутреннихДокументов
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов
|ГДЕ
|	ДескрипторыДоступаОбъектов.ВидОбъекта <>&ВидОбъекта
|	И ТИПЗНАЧЕНИЯ(ДескрипторыДоступаОбъектов.ВидОбъекта) = ТИП(Справочник.ВидыВнутреннихДокументов)";
Запрос.УстановитьПараметр("ВидОбъекта", Справочники.ВидыВнутреннихДокументов.ПустаяСсылка());
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Дескриптор = ВыборкаДетальныеЗаписи.Ссылка;
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(Дескриптор);
ОбновитьПраваСвязанныхДескрипторовПоДескриптору(Дескриптор);
КонецЦикла;
ИначеЕсли ТипЗнч(Источник) = Тип("КонстантаМенеджерЗначения.ДобавлятьРуководителямДоступПодчиненных") Тогда
// Обновление всех дескрипторов
// ДескрипторыДоступаОбъектов
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаОбъектов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаОбъектов КАК ДескрипторыДоступаОбъектов";
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаОбъектов.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// ДескрипторыДоступаРегистров
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаРегистров.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаРегистров КАК ДескрипторыДоступаРегистров";
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаРегистров.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
// ДескрипторыДоступаФайлов
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ДескрипторыДоступаФайлов.Ссылка
|ИЗ
|	Справочник.ДескрипторыДоступаФайлов КАК ДескрипторыДоступаФайлов";
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Справочники.ДескрипторыДоступаФайлов.ОбновитьПрава(ВыборкаДетальныеЗаписи.Ссылка);
КонецЦикла;
КонецЕсли;
КонецПроцедуры
// Извлечь текст из файла и поместить в ХранилищеЗначения
Функция ИзвлечьТекстВХранилищеЗначения(ПолноеИмяФайла) Экспорт
Текст = "";
ИзвлекатьТекстыФайловНаСервере = Константы.ИзвлекатьТекстыФайловНаСервере.Получить();
Если Не ИзвлекатьТекстыФайловНаСервере Тогда
ТипПлатформыСервера = ОбщегоНазначенияПовтИсп.ТипПлатформыСервера();
Если ТипПлатформыСервера = ТипПлатформы.Windows_x86 ИЛИ ТипПлатформыСервера = ТипПлатформы.Windows_x86_64 Тогда
```

```
Текст =
ФайловыеФункцииКлиентСервер.ИзвлечьТекст(ПолноеИмя
Файла);
КонецЕсли;
КонецЕсли;
Если ПустаяСтрока(Текст) Тогда
Возврат "";
КонецЕсли;
ХранилищеЗначения = Новый ХранилищеЗначения(Текст);
Возврат ХранилищеЗначения;
КонецФункции
// Разыменовать lnk файл
Функция РазыменоватьLnkФайл(ВыбранныйФайл) Экспорт
ТипПлатформыСервера =
ОбщегоНазначенияПовтИсп.ТипПлатформыСервера();
Если ТипПлатформыСервера =
ТипПлатформы.Windows_x86 ИЛИ ТипПлатформыСервера
= ТипПлатформы.Windows_x86_64 Тогда
ShellApp = Новый COMОбъект("shell.application");
FolderObj = ShellApp.NameSpace(ВыбранныйФайл.Путь);
                                        // полный (только) путьна lnk-
файл
FolderObjItem =
FolderObj.items().item(ВыбранныйФайл.Имя);              //
толькоимя lnk-файла
Link = FolderObjItem.GetLink();
ВозвратНовыйФайл(Link.path);
КонецЕсли;
Возврат ВыбранныйФайл;
КонецФункции
// Рекурсивная функция импорта файлов с диска - принимает
массив файлов (или каталогов)
// - если файл, просто добавляет его,   если каталог - создает
группу и рекурсивно вызывает саму себя
Процедура ИмпортФайлов(
Владелец,
Путь,
ФайлыАргумент,
Индикатор,
МассивИменФайловСОшибками,
МассивСтруктурВсехФайлов,
Комментарий,
ХранитьВерсии,
Рекурсивно,
КоличествоСуммарное,
Счетчик,
ИдентификаторФормы,
Знач ПсевдоФайловаяСистема,
ДобавленныеФайлы,
МассивВсехПапок,
РежимЗагрузки,
Пользователь,
ПараметрыРаспознавания) Экспорт
Перем ПервыйПапкаСТакимЖеИменем;
Перем ДокГруппаСсылка;
МаксРазмерФайла =
ФайловыеФункции.ПолучитьМаксимальныйРазмерФайла();
ЗапретЗагрузкиФайловПоРасширению =
ФайловыеФункции.ПолучитьЗапретЗагрузкиФайловПоРасш
ирению();
СписокЗапрещенныхРасширений =
ФайловыеФункции.ПолучитьСписокЗапрещенныхРасширен
ий();
Для Каждого ВыбранныйФайл Из ФайлыАргумент Цикл
Попытка
Если ВыбранныйФайл.Существует() Тогда
Если ВыбранныйФайл.Расширение = ".lnk" Тогда
ВыбранныйФайл =
РазыменоватьLnkФайл(ВыбранныйФайл);
КонецЕсли;
Если ВыбранныйФайл.ЭтоКаталог() Тогда
Если Рекурсивно = Истина Тогда
НовыйПуть = Строка(ВыбранныйФайл.Путь);
ОбщегоНазначенияКлиентСервер.ДобавитьСлешЕслиНужн
о(НовыйПуть,
ОбщегоНазначенияПовтИсп.ТипПлатформыСервера());
НовыйПуть = НовыйПуть + Строка(ВыбранныйФайл.Имя);
МассивФайлов =
ФайловыеФункцииКлиентСервер.НайтиФайлыПсевдо(Псев
доФайловаяСистема, НовыйПуть);
// Создаем группу в справочнике - эквивалент папки на
диске
Если МассивФайлов.Количество() <> 0 Тогда
ИмяФайла = ВыбранныйФайл.Имя;
Если
РаботаСФайламиВызовСервера.ЕстьПапкаСТакимИменем(
ИмяФайла, Владелец, ПервыйПапкаСТакимЖеИменем)
Тогда
ДокГруппаСсылка = ПервыйПапкаСТакимЖеИменем;
Иначе
ДокГруппаСсылка =
РаботаСФайламиВызовСервера.СправочникиПапкиСоздать
Элемент(ИмяФайла, Владелец, Пользователь);
КонецЕсли;
ИмпортФайлов(
ДокГруппаСсылка,
НовыйПуть,
МассивФайлов,
Индикатор,
МассивИменФайловСОшибками,
МассивСтруктурВсехФайлов,
Комментарий,
ХранитьВерсии,
Рекурсивно,
КоличествоСуммарное,
Счетчик,
ИдентификаторФормы,
ПсевдоФайловаяСистема,
ДобавленныеФайлы,
МассивВсехПапок,
РежимЗагрузки,
Пользователь,
ПараметрыРаспознавания);
МассивВсехПапок.Добавить(НовыйПуть);
КонецЕсли;
КонецЕсли;
Продолжить;
КонецЕсли;
Если Не
ФайловыеФункцииКлиентСервер.ФайлМожноЗагружать(Вы
бранныйФайл, МаксРазмерФайла,
ЗапретЗагрузкиФайловПоРасширению,
СписокЗапрещенныхРасширений,
МассивИменФайловСОшибками) Тогда
Продолжить;
КонецЕсли;
// Создаем Элемент справочника Файлы
ИмяБезРасширения = ВыбранныйФайл.ИмяБезРасширения;
Расширение = ВыбранныйФайл.Расширение;
Если РежимЗагрузки Тогда
Если
РаботаСФайламиВызовСервера.ЕстьФайлСТакимИменем(И
мяБезРасширения, Владелец) Тогда
Запись = Новый Структура;
Запись.Вставить("ИмяФайла",
ВыбранныйФайл.ПолноеИмя);
ТекстОшибки =
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = 'Файл с таким именем уже есть в
информационной базе в папке ""%1""'"),
Строка(Владелец));
Запись.Вставить("Ошибка", ТекстОшибки);
МассивИменФайловСОшибками.Добавить(Запись);
Продолжить;
КонецЕсли;
КонецЕсли;
АдресВременногоХранилищаФайла =
ВыбранныйФайл.ПолноеИмя;
АдресВременногоХранилищаТекста =
ИзвлечьТекстВХранилищеЗначения(
ВыбранныйФайл.ПолноеИмя);
// Создаем элемент справочника Файлы
РаботаСФайламиКлиентСервер.СоздатьЭлементСправочник
аФайлы(ВыбранныйФайл, МассивСтруктурВсехФайлов,
Владелец, ИдентификаторФормы, Комментарий,
ХранитьВерсии, ДобавленныеФайлы,
АдресВременногоХранилищаФайла,
АдресВременногоХранилищаТекста, Пользователь,
ПараметрыРаспознавания);
Иначе
Запись = Новый Структура;
Запись.Вставить("ИмяФайла",
ВыбранныйФайл.ПолноеИмя);
Запись.Вставить("Ошибка", НСтр("ru = 'Файл отсутствует на
диске'"));
МассивИменФайловСОшибками.Добавить(Запись);
КонецЕсли;
Исключение
ОшибкаИнфо = ИнформацияОбОшибке();
СообщениеОбОшибке = "";
Если ОшибкаИнфо.Причина = Неопределено Тогда
СообщениеОбОшибке =ОшибкаИнфо.Описание;
Иначе
СообщениеОбОшибке = ОшибкаИнфо.Причина.Описание;
КонецЕсли;
Запись = Новый Структура;
Запись.Вставить("ИмяФайла",
ВыбранныйФайл.ПолноеИмя);
Запись.Вставить("Ошибка", СообщениеОбОшибке);
МассивИменФайловСОшибками.Добавить(Запись);
КонецПопытки;
КонецЦикла;
КонецПроцедуры
// Импорт - с вспомогательными операциями типа проверки
предельного размера и впоследствии удаления файлов и
показа ошибок
//   при импорте только одной папки - вернет на нее ссылку
Функция ИмпортФайловВыполнить(
ПапкаДляДобавления,
ВыбранныеФайлы,
Комментарий,
ХранитьВерсии,
УдалятьФайлыПослеДобавления,
Рекурсивно,
ИдентификаторФормы,
Знач ПсевдоФайловаяСистема,
ДобавленныеФайлы,
РежимЗагрузки,
Пользователь,
МассивИменФайловСОшибками,
СтратегияРаспознавания,
ЯзыкРаспознавания) Экспорт
Перем ПервыйПапкаСТакимЖеИменем;
Перем ПапкаДляДобавленияТекущая;
ВыбранКаталог = Ложь;
Путь = "";
КоличествоСуммарное = 0;
МассивФайлов = Новый Массив;
Счетчик = 0;
Индикатор = 0;
МассивСтруктурВсехФайлов = Новый Массив;
МассивВсехПапок = Новый Массив;
ПапкаДляДобавленияТекущая = Неопределено;
// параметры распознавания по умолчанию
РаспознатьПослеДобавления = Истина;
Если СтратегияРаспознавания =
Перечисления.СтратегииРаспознаванияТекста.НеРаспознава
ть Тогда
РаспознатьПослеДобавления = Ложь;
КонецЕсли;
ИспользоватьРаспознавание =
РаботаСФайламиВызовСервера.ПолучитьИспользоватьРасп
ознавание();
Если ИспользоватьРаспознавание = Ложь Тогда
РаспознатьПослеДобавления = Ложь;
КонецЕсли;
ПараметрыРаспознавания = Новый
Структура("СтратегияРаспознавания, ЯзыкРаспознавания,
РаспознатьПослеДобавления",
СтратегияРаспознавания, ЯзыкРаспознавания,
РаспознатьПослеДобавления);
Для Каждого ИмяФайла Из ВыбранныеФайлы Цикл
Попытка
ВыбранныйФайл = Новый Файл(ИмяФайла.Значение);
ВыбранКаталог = Ложь;
Если ВыбранныйФайл.Существует() Тогда
ВыбранКаталог = ВыбранныйФайл.ЭтоКаталог();
КонецЕсли;
Если ВыбранКаталог Тогда
Путь = ИмяФайла.Значение;
МассивФайловЭтогоКаталога =
ФайловыеФункцииКлиентСервер.НайтиФайлыПсевдо(Псев
доФайловаяСистема, Путь);
ИмяПапки = ВыбранныйФайл.Имя;
Если
РаботаСФайламиВызовСервера.ЕстьПапкаСТакимИменем(
ИмяПапки, ПапкаДляДобавления,
ПервыйПапкаСТакимЖеИменем) Тогда
ПапкаДляДобавленияТекущая =
ПервыйПапкаСТакимЖеИменем;
Иначе
ПапкаДляДобавленияТекущая =
РаботаСФайламиВызовСервера.СправочникиПапкиСоздать
Элемент(ИмяПапки, ПапкаДляДобавления, Пользователь);
КонецЕсли;
// Собственно импорт
ИмпортФайлов(
ПапкаДляДобавленияТекущая,
Путь,
МассивФайловЭтогоКаталога,
Индикатор,
МассивИменФайловСОшибками,
МассивСтруктурВсехФайлов,
Комментарий,
ХранитьВерсии,
Рекурсивно,
КоличествоСуммарное,
Счетчик,
ИдентификаторФормы,
ПсевдоФайловаяСистема,
ДобавленныеФайлы,
МассивВсехПапок,
РежимЗагрузки,
Пользователь,
ПараметрыРаспознавания);
МассивВсехПапок.Добавить(Путь);
Иначе
МассивФайлов.Добавить(ВыбранныйФайл);
КонецЕсли;
Исключение
// запись в журнал регистрации
ОписаниеОшибки =
КраткоеПредставлениеОшибки(ИнформацияОбОшибке());
ТекстСообщения =
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = 'Ошибка автоматической загрузки файлов:
""%1""'"),
ОписаниеОшибки);
ЗаписьЖурналаРегистрации("Автоматическая загрузка
файлов",
УровеньЖурналаРегистрации.Ошибка, , ,
ТекстСообщения);
КонецПопытки;
КонецЦикла;
Если МассивФайлов.Количество() <> 0 Тогда
// Собственно импорт
ИмпортФайлов(
ПапкаДляДобавления,
Путь,
МассивФайлов,
Индикатор,
МассивИменФайловСОшибками,
МассивСтруктурВсехФайлов,
Комментарий,
ХранитьВерсии,
Рекурсивно,
КоличествоСуммарное,
Счетчик,
ИдентификаторФормы,
ПсевдоФайловаяСистема,
```

```
ДобавленныеФайлы,
МассивВсехПапок,
РежимЗагрузки,
Пользователь,
ПараметрыРаспознавания);
КонецЕсли;
Если УдалятьФайлыПослеДобавления = Истина Тогда
ФайловыеФункцииКлиентСервер.УдалитьФайлыПослеДоба
вления(МассивСтруктурВсехФайлов, МассивВсехПапок,
РежимЗагрузки);
КонецЕсли;
// Вывод сообщений об ошибках
Для Каждого Выборка Из МассивИменФайловСОшибками
Цикл
ТекстСообщения =
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = 'Ошибка автоматической загрузки файла ""%1"":
""%2""'"),
Выборка.ИмяФайла, Выборка.Ошибка);
ЗаписьЖурналаРегистрации("Автоматическая загрузка
файлов",
УровеньЖурналаРегистрации.Ошибка, , ,
ТекстСообщения);
КонецЦикла;
Если ВыбранныеФайлы.Количество() <> 1 Тогда
ПапкаДляДобавленияТекущая = Неопределено;
КонецЕсли;
Возврат ПапкаДляДобавленияТекущая;
КонецФункции
Функция ЕстьПравоДоступа(Пользователь, Папка)
// если не включена функциональная опция "использовать
права доступа" - то не делаем проверки.
Если Не
УправлениеДоступом.ОграничиватьДоступНаУровнеЗаписе
й() Тогда
Возврат Истина;
КонецЕсли;
Возврат
ДокументооборотПраваДоступа.ПолучитьПраваПоОбъекту(
Папка, Пользователь).Добавление;
КонецФункции
Процедура ЗагрузкаФайлов(МассивНастроек) Экспорт
Если ТипЗнч(МассивНастроек) <> Тип("Массив") Тогда
Возврат;
КонецЕсли;
Для Каждого Настройка Из МассивНастроек Цикл
КаталогНаДиске = "";
ТипПлатформыСервера =
ОбщегоНазначенияПовтИсп.ТипПлатформыСервера();
Если ТипПлатформыСервера =
ТипПлатформы.Windows_x86 ИЛИ ТипПлатформыСервера
= ТипПлатформы.Windows_x86_64 Тогда
КаталогНаДиске = Настройка.КаталогWindows;
Иначе
КаталогНаДиске = Настройка.КаталогLinux;
КонецЕсли;
Папка = Настройка.Папка;
Пользователь = Настройка.Пользователь;
СтратегияРаспознавания = Неопределено;
ЯзыкРаспознавания = Неопределено;
Если Настройка.Свойство("СтратегияРаспознавания") Тогда
СтратегияРаспознавания =
Настройка.СтратегияРаспознавания;
Иначе
СтратегияРаспознавания =
Перечисления.СтратегииРаспознаванияТекста.ПоместитьТо
лькоВТекстовыйОбраз;
КонецЕсли;
Если Настройка.Свойство("ЯзыкРаспознавания") Тогда
ЯзыкРаспознавания = Настройка.ЯзыкРаспознавания;
Иначе
ЯзыкРаспознавания =
РаботаСФайламиВызовСервера.ПолучитьЯзыкРаспознавани
я();
КонецЕсли;
Если ПустаяСтрока(КаталогНаДиске) Тогда
ЗаписьЖурналаРегистрации("Автоматическая загрузка
файлов",
УровеньЖурналаРегистрации.Ошибка, , ,
НСтр("ru = 'Не указан каталог на диске'"));
Продолжить;
КонецЕсли;
ФайлКаталога = Новый Файл(КаталогНаДиске);
Если Не ФайлКаталога.Существует() Тогда
ТекстОшибки =
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = 'Неверный путь к каталогу на диске: ""%1""'"),
КаталогНаДиске);
ЗаписьЖурналаРегистрации("Автоматическая загрузка
файлов",
УровеньЖурналаРегистрации.Ошибка, , ,
ТекстОшибки);
Продолжить;
КонецЕсли;
Если Папка.Пустая() Тогда
ЗаписьЖурналаРегистрации("Автоматическая загрузка
файлов",
УровеньЖурналаРегистрации.Ошибка, , ,
НСтр("ru = 'Не указана папка'"));
Продолжить;
КонецЕсли;
Если Пользователь.Пустая() Тогда
ЗаписьЖурналаРегистрации("Автоматическая загрузка
файлов",
УровеньЖурналаРегистрации.Ошибка, , ,
НСтр("ru = 'Не указан пользователь'"));
Продолжить;
КонецЕсли;
Если Не ЕстьПравоДоступа(Пользователь, Папка) Тогда
ТекстОшибки =
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = 'У пользователя ""%1"" нет прав на добавление
файлов в папку ""%2""'"),
Пользователь, Папка);
ЗаписьЖурналаРегистрации("Автоматическая загрузка
файлов",
УровеньЖурналаРегистрации.Ошибка, , ,
ТекстОшибки);
Продолжить;
КонецЕсли;
Попытка
ВыбранныеФайлы = Новый СписокЗначений;
НайденныеФайлы = НайтиФайлы(КаталогНаДиске, "*.*");
Для Каждого ФайлВложенный Из НайденныеФайлы Цикл
ВыбранныеФайлы.Добавить(ФайлВложенный.ПолноеИмя);
КонецЦикла;
ПсевдоФайловаяСистема = Новый Соответствие; //
соответствие путь к директории - файлы и папки в ней
ДобавленныеФайлы = Новый Массив;
МассивИменФайловСОшибками = Новый Массив;
Описание = "";
ХранитьВерсии = Истина;
УдалятьФайлыПослеДобавления = Истина;
ПапкаДляДобавленияТекущая = ИмпортФайловВыполнить(
Папка,
ВыбранныеФайлы,
Описание,
ХранитьВерсии,
УдалятьФайлыПослеДобавления,
Истина,
Неопределено, //УникальныйИдентификатор,
ПсевдоФайловаяСистема,
ДобавленныеФайлы,
Истина,  // режим загрузки
Пользователь,
МассивИменФайловСОшибками,
СтратегияРаспознавания,
ЯзыкРаспознавания);
ТекстСообщения =
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = 'Закончена автоматическая загрузка файлов из
каталога ""%1"" в папку ""%2"". Загружено файлов: %3. Не
удалось загрузить файлов: %4.'"),
КаталогНаДиске, Папка, ДобавленныеФайлы.Количество(),
МассивИменФайловСОшибками.Количество());
ЗаписьЖурналаРегистрации("Автоматическая загрузка
файлов",
УровеньЖурналаРегистрации.Информация, , ,
ТекстСообщения);
Исключение
// запись в журнал регистрации
ОписаниеОшибки =
КраткоеПредставлениеОшибки(ИнформацияОбОшибке());
ТекстСообщения =
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = 'Ошибка автоматической загрузки файлов:
""%1""  из каталога ""%2"" в папку ""%3""'"),
ОписаниеОшибки, КаталогНаДиске, Папка);
ЗаписьЖурналаРегистрации("Автоматическая загрузка
файлов",
УровеньЖурналаРегистрации.Ошибка, , ,
ТекстСообщения);
КонецПопытки;
КонецЦикла;
КонецПроцедуры
Процедура ЗаписатьСобытие(Задача, Событие,
Комментарий = "") Экспорт
УстановитьПривилегированныйРежим(Истина);
МенеджерЗаписи =
РегистрыСведений.ИсторияСобытийЗадач.СоздатьМенедже
рЗаписи();
МенеджерЗаписи.Задача = Задача;
МенеджерЗаписи.ДатаСобытия = ТекущаяДата();
МенеджерЗаписи.Пользователь =
ПараметрыСеанса.ТекущийПользователь;
МенеджерЗаписи.Событие = Событие;
МенеджерЗаписи.Комментарий = Комментарий;
МенеджерЗаписи.Записать();
КонецПроцедуры
Процедура ЗаписатьСобытиеОткрытаКарточка(Задача)
Экспорт
ЗаписатьСобытие(Задача,
Перечисления.ВидыСобытийЗадач.ОткрытаКарточка);
КонецПроцедуры
// Выводит состояние документа (проведен, не проведен,
записан)
Процедура ОбновитьСостояниеДокумента(Ссылка,
СостояниеДокумента, КартинкаСостоянияДокумента)
Экспорт
Проведен = Ссылка.Проведен;
ПометкаУдаления = Ссылка.ПометкаУдаления;
РазрешеноПроведение = (Ссылка.Метаданные().Проведение
= Метаданные.СвойстваОбъектов.Проведение.Разрешить);
Если ПометкаУдаления Тогда
СостояниеДокумента = "Помечен на удаление";
КартинкаСостоянияДокумента = 2;
ИначеЕсли Проведен Тогда
СостояниеДокумента = "Проведен";
КартинкаСостоянияДокумента = 1;
ИначеЕсли РазрешеноПроведение Тогда
СостояниеДокумента = "Не проведен";
КартинкаСостоянияДокумента = 0;
Иначе
СостояниеДокумента = "Записан";
КартинкаСостоянияДокумента = 3;
КонецЕсли;
КонецПроцедуры
// Продляет срок действия документов из регламентного
задания
Процедура АвтоматическоеПродлениеДоговоров() Экспорт
УстановитьПривилегированныйРежим(Истина);
ДокументыОтветственных = Новый ТаблицаЗначений;
ДокументыОтветственных.Колонки.Добавить("Документ");
ДокументыОтветственных.Колонки.Добавить("Ответственн
ый");
ДокументыОтветственных.Колонки.Добавить("СтараяДатаО
кончания");
ДокументыОтветственных.Колонки.Добавить("НоваяДатаО
кончания");
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ВнутренниеДокументы.Ссылка,
|	ВнутренниеДокументы.ПорядокПродления,
|	ВнутренниеДокументы.Ответственный
|ИЗ
|	Справочник.ВнутренниеДокументы КАК
ВнутренниеДокументы
|ГДЕ
|
	ВнутренниеДокументы.ВидДокумента.Учитыв
атьСрокДействия
|	И (НЕ ВнутренниеДокументы.Бессрочный)
|	И ВнутренниеДокументы.ПорядокПродления
В(&ПорядокПродления)
|	И
КОНЕЦПЕРИОДА(ВнутренниеДокументы.ДатаОкончания
Действия, ДЕНЬ) <&ТекущаяДата";
ПорядокПродления = Новый Массив;
ПорядокПродления.Добавить(Перечисления.ПорядокПродле
ния.АвтоматическиНаМесяц);
ПорядокПродления.Добавить(Перечисления.ПорядокПродле
ния.АвтоматическиНаКвартал);
ПорядокПродления.Добавить(Перечисления.ПорядокПродле
ния.АвтоматическиНаПолугодие);
ПорядокПродления.Добавить(Перечисления.ПорядокПродле
ния.АвтоматическиНаГод);
ПорядокПродления.Добавить(Перечисления.ПорядокПродле
ния.АвтоматическиНаНеопределенныйСрок);
Запрос.УстановитьПараметр("ПорядокПродления",
ПорядокПродления);
Запрос.УстановитьПараметр("ТекущаяДата",
ТекущаяДата());
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
Если Выборка.ПорядокПродления =
Перечисления.ПорядокПродления.АвтоматическиНаМесяц
Тогда
НаСрок = 1;
ИначеЕсли Выборка.ПорядокПродления =
Перечисления.ПорядокПродления.АвтоматическиНаКвартал
Тогда
НаСрок = 3;
ИначеЕсли Выборка.ПорядокПродления =
Перечисления.ПорядокПродления.АвтоматическиНаПолуго
дие Тогда
НаСрок = 6;
ИначеЕсли Выборка.ПорядокПродления =
Перечисления.ПорядокПродления.АвтоматическиНаГод
Тогда
НаСрок = 12;
КонецЕсли;
Объект = Выборка.Ссылка.ПолучитьОбъект();
СтараяДатаОкончания = Объект.ДатаОкончанияДействия;
Если Выборка.ПорядокПродления =
Перечисления.ПорядокПродления.АвтоматическиНаНеопре
деленныйСрок Тогда
Объект.ДатаОкончанияДействия = '00010101';
Объект.Бессрочный = Истина;
НоваяДатаОкончания = "Бессрочный";
Иначе
Если КонецДня(Объект.ДатаОкончанияДействия) =
КонецМесяца(Объект.ДатаОкончанияДействия) Тогда
Объект.ДатаОкончанияДействия =
КонецМесяца(ДобавитьМесяц(Объект.ДатаОкончанияДейст
вия, НаСрок));
Иначе
Объект.ДатаОкончанияДействия =
ДобавитьМесяц(Объект.ДатаОкончанияДействия, НаСрок);
КонецЕсли;
НоваяДатаОкончания = Объект.ДатаОкончанияДействия;
КонецЕсли;

ТекстСообщения = "";
Попытка
ЗаблокироватьДанныеДляРедактирования(Объект.Ссылка);
Объект.Записать();
```

```
// Записываем информацию о продлении в историю срока действия документа.
ПараметрыЗаписи = Новый Структура;
ПараметрыЗаписи.Вставить("Документ", Объект.Ссылка);
ПараметрыЗаписи.Вставить("ДатаНачалаДействия", Объект.ДатаНачалаДействия);
ПараметрыЗаписи.Вставить("ДатаОкончанияДействия", Объект.ДатаОкончанияДействия);
ПараметрыЗаписи.Вставить("Бессрочный", Объект.Бессрочный);
ПараметрыЗаписи.Вставить("ПорядокПродления", Объект.ПорядокПродления);
ПараметрыЗаписи.Вставить("ДокументИсточникИзменения", Неопределено);
ПараметрыЗаписи.Вставить("Комментарий", НСтр("ru = 'Срок действия документа автоматически продлен.'"));
РегистрыСведений.ИсторияСроковДействияДокументов.ДобавитьЗапись(ПараметрыЗаписи);
Исключение
ТекстСообщения = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Ошибка при автоматическом продлении срока действия документа: %1.'"),
ПодробноеПредставлениеОшибки(ИнформацияОбОшибке()));
УровеньВажностиСобытия = УровеньЖурналаРегистрации.Ошибка;
КонецПопытки;
Если ПустаяСтрока(ТекстСообщения) Тогда
ТекстСообщения = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Автоматически продлен срок действия документа до %1.'"),
Формат(НоваяДатаОкончания, "ДФ=dd.ММ.уууу"));
УровеньВажностиСобытия = УровеньЖурналаРегистрации.Информация;
Если ЗначениеЗаполнено(Выборка.Ответственный) Тогда
НоваяСтрока = ДокументыОтветственных.Добавить();
НоваяСтрока.Документ = Выборка.Ссылка;
НоваяСтрока.Ответственный = Выборка.Ответственный;
НоваяСтрока.СтараяДатаОкончания = СтараяДатаОкончания;
НоваяСтрока.НоваяДатаОкончания = НоваяДатаОкончания;
КонецЕсли;
КонецЕсли;
ЗаписьЖурналаРегистрации(НСтр("ru = 'Автоматическое продление договоров.'"), УровеньВажностиСобытия,
Выборка.Ссылка.Метаданные(), Выборка.Ссылка,
ТекстСообщения);
КонецЦикла;

// отправка уведомлений по почте
Ответственные = ДокументыОтветственных.Скопировать();
Ответственные.Свернуть("Ответственный");
Для Каждого Строка Из Ответственные Цикл
Ответственный = Строка.Ответственный;
ПочтовыйАдрес = УправлениеКонтактнойИнформацией.ПолучитьКонтактнуюИнформациюОбъекта(Ответственный,
Справочники.ВидыКонтактнойИнформации.EmailПользователя);
Если ПустаяСтрока(ПочтовыйАдрес) Тогда
ТекстСообщения = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Уведомление не было отправлено, так как у пользователя %1 не задан адрес электронной почты.'"),
Строка(Ответственный));
ЗаписьЖурналаРегистрации(НСтр("ru = 'Уведомление об автоматическом продлении договоров'"),
УровеньЖурналаРегистрации.Информация,,,
ТекстСообщения);
Продолжить;
КонецЕсли;
ПараметрыПисьма = Новый Структура;
ПараметрыПисьма.Вставить("Кому", ПочтовыйАдрес);
ТелоПисьма =
НСтр("ru = 'У следующих документов был автоматически продлен срок действия:
|
|'");
НайденныеСтроки = ДокументыОтветственных.НайтиСтроки(Новый Структура("Ответственный", Ответственный));
Для Каждого НайденнаяСтрока Из НайденныеСтроки Цикл
ТелоПисьма = ТелоПисьма
+ НайденнаяСтрока.Документ.Заголовок
+
?(ЗначениеЗаполнено(НайденнаяСтрока.Документ.РегистрационныйНомер), " № " +
НайденнаяСтрока.Документ.РегистрационныйНомер, "")
+
?(ЗначениеЗаполнено(НайденнаяСтрока.Документ.ДатаРегистрации), " от " +
Формат(НайденнаяСтрока.Документ.ДатаРегистрации,"ДФ=dd.ММ.уууу"), "")
+
?(ЗначениеЗаполнено(НайденнаяСтрока.Документ.Корреспондент), ", корреспондент " +
Строка(НайденнаяСтрока.Документ.Корреспондент), "")
+ НСтр("ru = ' - срок действия продлен до '") +
Формат(НайденнаяСтрока.НоваяДатаОкончания, "ДФ=dd.ММ.уууу") + Символы.ПС;
КонецЦикла;
ПараметрыПисьма.Вставить("Текст", ТелоПисьма);
ТемаПисьма = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Автоматически продлены сроки действия документов (%1)'"),
НайденныеСтроки.Количество());
ПараметрыПисьма.Вставить("Тема", ТемаПисьма);

ТекстСообщения = "";
Попытка
ЛегкаяПочтаСервер.ОтправитьИнтернетПочта(ПараметрыПисьма);
Исключение
ТекстСообщения = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Ошибка при отправке уведомления об автоматическом продлении договоров: %1.'"),
ПодробноеПредставлениеОшибки(ИнформацияОбОшибке()));
УровеньВажностиСобытия = УровеньЖурналаРегистрации.Ошибка;
КонецПопытки;
Если ПустаяСтрока(ТекстСообщения) Тогда
ТекстСообщения = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
НСтр("ru = 'Уведомление об автоматическом продлении договоров успешно отправлено на адрес %1.'"),
ПочтовыйАдрес);
УровеньВажностиСобытия = УровеньЖурналаРегистрации.Информация;
КонецЕсли;
ЗаписьЖурналаРегистрации(НСтр("ru = 'Автоматическое продление договоров'"), УровеньВажностиСобытия,,,
ТекстСообщения);
КонецЦикла;

КонецПроцедуры
////////////////////////////////////////////////////////////////////////////////
// РАБОТА С ИМЕНЕМ ФАЙЛА
// Возвращает структуру
// Результат (Структура)
// - Имя (Строка)
// - Расширение (Строка)
//
// Параметры:
// - ИмяФайла (Строка)
//
Функция РазложитьИмяФайла(ИмяФайла) Экспорт
Расширение = ИмяФайла;
Поз = Найти(Расширение, ".");
Если Поз = 0 Тогда
Возврат Новый Структура("Имя, Расширение", СокрЛП(ИмяФайла), "");
КонецЕсли;
Пока Поз > 0 Цикл
Расширение = Сред(Расширение, Поз + 1);
Поз = Найти(Расширение, ".");
КонецЦикла;
Имя = Лев(ИмяФайла, СтрДлина(ИмяФайла) - СтрДлина(Расширение) - 1);
Возврат Новый Структура("Имя, Расширение", СокрЛП(Имя), СокрЛП(Расширение));
КонецФункции
// Возвращает структуру
// Результат (Структура)
// - Каталог (Строка) - без последнего слеша
// - ИмяФайла (Строка) - имя файла с расширением
// - Имя (Строка)
// - Расширение (Строка) - без точки
//
// Параметры:
// - ПолноеИмяФайла (Строка)
//
Функция РазложитьПолноеИмяФайла(ПолноеИмяФайла) Экспорт
ИмяФайла = ПолноеИмяФайла;
Каталог = "";
Пока Истина Цикл
Поз = Макс(Найти(ИмяФайла, "\"), Найти(ИмяФайла, "/"));
Если Поз = 0 Тогда
Прервать;
КонецЕсли;
Каталог = Каталог + Лев(ИмяФайла, Поз);
ИмяФайла = Сред(ИмяФайла, Поз+1);
КонецЦикла;
ПоследнийСимволКаталога = Прав(Каталог, 1);
Если (ПоследнийСимволКаталога = "\") Или (ПоследнийСимволКаталога = "/") Тогда
Каталог = Лев(Каталог, СтрДлина(Каталог) - 1);
КонецЕсли;
ИияФайлаИнфо = РазложитьИмяФайла(ИмяФайла);
ИияФайлаИнфо.Вставить("ИмяФайла", ИмяФайла);
ИияФайлаИнфо.Вставить("Каталог", Каталог);
Возврат ИияФайлаИнфо;
КонецФункции

////////////////////////////////////////////////////////////////////////////////
// РАБОТА С АДРЕСАМИ ЭЛЕКТРОННОЙ ПОЧТЫ
// Принимает строку почтовых адресов в виде
// "name1 <addr1@dom1>, name2 <addr2@dom2>, ..., nameN <addrN@domN>"
// или
// "name1 <addr1@dom1>; name2 <addr2@dom2>; ...; nameN <addrN@domN>"
// Возвращает:
// Результат (Массив)
// - Элемент (Структура)
//   - Адрес (Строка)
//   - Представление (Строка)
//
Функция РазложитьСтрокуПочтовыхАдресов(Знач ПочтовыеАдресаСтр) Экспорт
Результат = Новый Массив;
ПочтовыеАдресаСтр= СтрЗаменить(ПочтовыеАдресаСтр, ";", Символы.ПС);
ПочтовыеАдресаСтр = СтрЗаменить(ПочтовыеАдресаСтр, ",", Символы.ПС);
Для Счетчик = 1 По СтрЧислоСтрок(ПочтовыеАдресаСтр) Цикл
АдресСтр = СокрЛП(СтрПолучитьСтроку(ПочтовыеАдресаСтр, Счетчик));
Если ПустаяСтрока(АдресСтр) Тогда
Продолжить;
КонецЕсли;
АдресЭлектроннойПочтыИнфо = РазложитьПредставлениеАдресаЭлектроннойПочты(АдресСтр);
Результат.Добавить(АдресЭлектроннойПочтыИнфо);
КонецЦикла;
Возврат Результат;
КонецФункции
// Принимает строку почтового адреса в виде
// "name <addr@dom>"
// Возвращает:
// Результат (Структура)
// - Адрес (Строка) - addr@dom
// - ОтображаемоеИмя (Строка) - name
// - Пользователь (Строка)-  addr
// - Домен (Строка) - dom
//
Функция РазложитьПредставлениеАдресаЭлектроннойПочты(Знач АдресЭлектроннойПочтыСтр) Экспорт
Результат = Новый Структура("Адрес, ОтображаемоеИмя, Пользователь, Домен", "", "", "", "");
АдресЭлектроннойПочтыСтр = СокрЛП(АдресЭлектроннойПочтыСтр);
Поз = Найти(АдресЭлектроннойПочтыСтр, "<");
Результат.ОтображаемоеИмя = "";
Результат.Адрес = АдресЭлектроннойПочтыСтр;
Если Поз > 0 Тогда
Результат.ОтображаемоеИмя = СокрЛП(Лев(АдресЭлектроннойПочтыСтр, Поз - 1));
АдресЭлектроннойПочтыСтр = Сред(АдресЭлектроннойПочтыСтр, Поз + 1);
Поз = Найти(АдресЭлектроннойПочтыСтр, ">");
Если Поз > 0 Тогда
Результат.Адрес = СокрЛП(Лев(АдресЭлектроннойПочтыСтр, Поз - 1));
Иначе
Результат.Адрес = "";
КонецЕсли;
КонецЕсли;
Если Не ЭтоАдресЭлектроннойПочты(Результат.Адрес) Тогда
Результат.Адрес = "";
Иначе
Поз = Найти(Результат.Адрес, "@");
Результат.Пользователь = Лев(Результат.Адрес, Поз - 1);
Результат.Домен = Сред(Результат.Адрес, Поз + 1);
КонецЕсли;
Возврат Результат;
КонецФункции
// Проверяет строку на формат адреса электронной почты
// проверка не точная но основные элементы на месте
//
Функция ЭтоАдресЭлектроннойПочты(АдресЭлектроннойПочты) Экспорт
Поз = Найти(АдресЭлектроннойПочты, "@");
Если Поз = 0 Тогда
Возврат Ложь;
КонецЕсли;
Если СтрЧислоВхождений(АдресЭлектроннойПочты, "@") <> 1 Тогда
Возврат Ложь;
КонецЕсли;
Если Прав(АдресЭлектроннойПочты, 1) = "."
Или Лев(АдресЭлектроннойПочты, 1) = "." Тогда
Возврат Ложь;
КонецЕсли;
Если Найти(АдресЭлектроннойПочты, "..") > 0 Тогда
Возврат Ложь;
КонецЕсли;
ДопустимыеСимволы = "-.0123456789@ABCDEFGHIJKLMNOPQRSTUVWXYZ_abcdefghijklmnopqrstuvwxyz";
Для НомерСимвола = 1 По СтрДлина(АдресЭлектроннойПочты) Цикл
```

```
Если Найти(ДопустимыеСимволы,
Сред(АдресЭлектроннойПочты, НомерСимвола, 1)) = 0
Тогда
Возврат Ложь;
КонецЕсли;
КонецЦикла;
Пользователь = Лев(АдресЭлектроннойПочты, Поз - 1);
Если СтрДлина(Пользователь) = 0 Тогда
Возврат Ложь;
КонецЕсли;
Сервер = Сред(АдресЭлектроннойПочты, Поз + 1);
Если СтрДлина(Сервер) = 0 Тогда
Возврат Ложь;
КонецЕсли;
Поз = Найти(Сервер, ".");
Если Поз = 0 Тогда
Возврат Ложь;
КонецЕсли;
Возврат Истина;
КонецФункции
//Формирует представление адресата электронной почты
//
//Параметры
//  Имя     - Строка - имя адресата
//  Адрес   - Строка - адрес электронной почты адресата
//  Контакт - СправочникСсылка - контакт, которому
принадлежит имя и адрес почты.
//
// Результат (Строка) - "Имя <Адрес>" или "Адрес"
//
Функция
ПолучитьПредставлениеАдресаЭлектроннойПочты(Имя,
Адрес) Экспорт
Если ПустаяСтрока(Имя) Или Имя = Адрес Тогда
Результат = Адрес;
ИначеЕсли ПустаяСтрока(Адрес) Тогда
Результат = Имя;
Иначе
Результат = Имя + " <" + Адрес + ">";
КонецЕсли;
Возврат Результат;
КонецФункции

////////////////////////////////////////////////////////////////////////////////
// РАБОТА С ТЕКСТОМ
// Удаляет недопустимые символы в XML-строке
// Параметры:
// - Текст (Строка)
// - СимволЗамены (Строка)
//
// Результат (Строка)
//
Функция УдалитьНедопустимыеСимволыXML(Знач Текст)
Экспорт
#Если Не ВебКлиент Тогда
ПозицияНачала = 1;
Пока Истина Цикл
Если ПозицияНачала > СтрДлина(Текст) Тогда
Прервать;
КонецЕсли;
Позиция = НайтиНедопустимыеСимволыXML(Текст,
ПозицияНачала);
Если Позиция = 0 Тогда
Прервать;
КонецЕсли;
// Если возвращается позиция, больше чем должна быть, то
корректируем ее.
Если Позиция > 1 Тогда
НедопустимыйСимвол = Сред(Текст, Позиция - 1, 1);
Если
НайтиНедопустимыеСимволыXML(НедопустимыйСимвол)
> 0 Тогда
Текст = СтрЗаменить(Текст, НедопустимыйСимвол, "");
КонецЕсли;
КонецЕсли;
НедопустимыйСимвол = Сред(Текст, Позиция, 1);
Если
НайтиНедопустимыеСимволыXML(НедопустимыйСимвол)
> 0 Тогда
Текст = СтрЗаменить(Текст, НедопустимыйСимвол, "");
КонецЕсли;
ПозицияНачала = Макс(1, Позиция - 1);
КонецЦикла;
#КонецЕсли
Возврат Текст;
КонецФункции
// Удаляет пустые строки в тексте.
//
Процедура УдалитьПустыеСтроки(Текст) Экспорт
Результат = "";
Для Индекс = 1 По СтрЧислоСтрок(Текст) Цикл
Строка = СтрПолучитьСтроку(Текст, Индекс);
Если ЗначениеЗаполнено(Строка) Тогда
ДобавитьЗначениеКСтрокеЧерезРазделитель(
Результат,
Символы.ПС,
Строка);
КонецЕсли;
КонецЦикла;
Текст = Результат;
КонецПроцедуры
// Добавляет к каждой строке спереди символ квотирования.
//
Процедура ДобавитьКвотирование(Текст, Знач
СимволКвотирования) Экспорт
Результат = "";
Для Индекс = 1 По СтрЧислоСтрок(Текст) Цикл
Строка = СтрПолучитьСтроку(Текст, Индекс);
ДобавитьЗначениеКСтрокеЧерезРазделитель(
Результат,
Символы.ПС,
СимволКвотирования + Строка);
КонецЦикла;
Текст = Результат;
КонецПроцедуры
// Заменяет в переданном тексте ошибочные
последовательности переноса строк.
// Все последовательности [ВК]+ПС заменяются на ВК+ПС.
// Все одиночные символы ВК заменяются на ВК+ПС.
//
Процедура ИсправитьПереносыСтрок(Текст) Экспорт
Пока Найти(Текст, Символы.ВК + Символы.ПС) > 0 Цикл
Текст = СтрЗаменить(Текст, Символы.ВК + Символы.ПС,
Символы.ПС);
КонецЦикла;
Если Найти(Текст, Символы.ВК) > 0 Тогда
Текст = СтрЗаменить(Текст, Символы.ВК, Символы.ПС);
КонецЕсли;
КонецПроцедуры

////////////////////////////////////////////////////////////////////////////////
// РАБОТА С ПРЕДСТАВЛЕНИЕМ РАЗМЕРОВ
// Принимает размер в байтах.
// Возвращает строку, например: 7.2 Кбайт, 35 Кбайт, 5.5
Мбайт, 12 Мбайт
Функция ПолучитьРазмерСтрокой(Размер) Экспорт
Если Размер = 0 Тогда
Возврат "-";
ИначеЕсли Размер < 1024 * 10 Тогда // < 10 Кб
Возврат
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = '%1 Кб'"),
Формат(Макс(1, Окр(Размер / 1024, 1, 1)), "ЧГ=0"));
ИначеЕсли Размер < 1024 * 1024 Тогда // < 1 Мб
Возврат
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = '%1 Кб'"),
Формат(Цел(Размер / 1024), "ЧГ=0"));
ИначеЕсли Размер < 1024 * 1024 * 10 Тогда // < 10 Мб
Возврат
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = '%1 Мб'"),
Формат(Окр(Размер / 1024 / 1024, 1, 1), "ЧГ=0"));
Иначе // >= 10 Мб
Возврат
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = '%1 Мб'"),
Формат(Цел(Размер / 1024 / 1024), "ЧГ=0"));
КонецЕсли;
КонецФункции

////////////////////////////////////////////////////////////////////////////////
// РАБОТА С КОДИРОВКАМИ
// Функция возвращает таблицу имен кодировок
// Возвращаемое значение:
// Результат (СписокЗначений)
// - Значение (Строка) - например "ibm852"
// - Представление (Строка) - например "ibm852
(Центральноевропейская DOS)"
//
Функция ПолучитьСписокКодировок() Экспорт
СписокКодировок = Новый СписокЗначений;
СписокКодировок.Добавить("ibm852",
                "IBM852 (Центральноевропейская DOS)");
СписокКодировок.Добавить("ibm866",
                "IBM866 (Кириллица DOS)");
СписокКодировок.Добавить("iso-8859-1",
                "ISO-8859-1 (Западноевропейская ISO)");
СписокКодировок.Добавить("iso-8859-2",
                "ISO-8859-2 (Центральноевропейская ISO)");
СписокКодировок.Добавить("iso-8859-3",
                "ISO-8859-3 (Латиница 3 ISO)");
СписокКодировок.Добавить("iso-8859-4",
                "ISO-8859-4 (Балтийская ISO)");
СписокКодировок.Добавить("iso-8859-5",
                "ISO-8859-5 (Кириллица ISO)");
СписокКодировок.Добавить("iso-8859-7",
                "ISO-8859-7 (Греческая ISO)");
СписокКодировок.Добавить("iso-8859-9",
                "ISO-8859-9 (Турецкая ISO)");
СписокКодировок.Добавить("iso-8859-15",
                "ISO-8859-15 (Латиница 9 ISO)");
СписокКодировок.Добавить("koi8-r",
                "KOI8-R (Кириллица KOI8-R)");
СписокКодировок.Добавить("koi8-u",
                "KOI8-U (Кириллица KOI8-U)");
СписокКодировок.Добавить("us-ascii",                "US-
ASCII США");
СписокКодировок.Добавить("utf-8",
                "UTF-8 (Юникод UTF-8)");
СписокКодировок.Добавить("windows-1250",
                "Windows-1250 (Центральноевропейская
Windows)");
СписокКодировок.Добавить("windows-1251",
                "windows-1251 (Кириллица Windows)");
СписокКодировок.Добавить("windows-1252",
                "Windows-1252 (Западноевропейская
Windows)");
СписокКодировок.Добавить("windows-1253",
                "Windows-1253 (Греческая Windows)");
СписокКодировок.Добавить("windows-1254",
                "Windows-1254 (Турецкая Windows)");
СписокКодировок.Добавить("windows-1257",
                "Windows-1257 (Балтийская Windows)");
Возврат СписокКодировок;
КонецФункции

////////////////////////////////////////////////////////////////////////////////
// ПРОЧИЕ ФУНКЦИИ
// Выделяет продстроку в скобках.
// Напрмер: ВыделитьПодстрокуВСкобках("Name
<name@company.ru>", "<", ">") = "name@company.ru"
//
Функция ВыделитьПодстрокуВСкобках(Знач Строка, Знач
ЛеваяСкобка, Знач ПраваяСкобка) Экспорт
Поз = Найти(Строка, ЛеваяСкобка);
Если Поз <> 0 Тогда
Строка = Сред(Строка, Поз+1);
КонецЕсли;
Поз = Найти(Строка, ПраваяСкобка);
Если Поз <> 0 Тогда
Строка = Лев(Строка, Поз-1);
КонецЕсли;
Возврат Строка;
КонецФункции
// Ищет подстроку в строке, после указанной позиции
//
Функция НайтиПосле(Строка, Подстрока,
НачальнаяПозиция = 0) Экспорт
Позиция = Найти(Сред(Строка, НачальнаяПозиция + 1),
Подстрока);
Если Позиция = 0 Тогда
Возврат 0;
КонецЕсли;
Возврат НачальнаяПозиция + Позиция;
КонецФункции
Функция ПолучитьСвязанныйДокумент(Документ, Знач
ТипСвязи) Экспорт
Если ТипЗнч(ТипСвязи) = Тип("Строка") Тогда
ТипСвязи = Справочники.ТипыСвязей[ТипСвязи];
КонецЕсли;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ ПЕРВЫЕ 1
|               СвязиДокументов.СвязанныйДокумент,
|               СвязиДокументов.Комментарий,
|               СвязиДокументов.Установил,
|               СвязиДокументов.ДатаУстановки
|ИЗ
|               РегистрСведений.СвязиДокументов КАК
СвязиДокументов
|ГДЕ
|               СвязиДокументов.Документ = &Документ
|               И СвязиДокументов.ТипСвязи = &ТипСвязи";
Запрос.УстановитьПараметр("Документ", Документ);
Запрос.УстановитьПараметр("ТипСвязи", ТипСвязи);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
Возврат Неопределено;
КонецЕсли;
Выборка = Результат.Выбрать();
Выборка.Следующий();
Возврат Выборка.СвязанныйДокумент;
КонецФункции
Функция ПолучитьСвязанныеДокументы(Документ, Знач
ТипСвязи) Экспорт
Если ТипЗнч(ТипСвязи) = Тип("Строка") Тогда
ТипСвязи = Справочники.ТипыСвязей[ТипСвязи];
КонецЕсли;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
|               СвязиДокументов.СвязанныйДокумент,
|               СвязиДокументов.Комментарий,
|               СвязиДокументов.Установил,
|               СвязиДокументов.ДатаУстановки
|ИЗ
|               РегистрСведений.СвязиДокументов КАК
СвязиДокументов
|ГДЕ
|               СвязиДокументов.Документ = &Документ
|               И СвязиДокументов.ТипСвязи = &ТипСвязи";
Запрос.УстановитьПараметр("Документ", Документ);
Запрос.УстановитьПараметр("ТипСвязи", ТипСвязи);
Возврат
Запрос.Выполнить().Выгрузить().ВыгрузитьКолонку("Связа
нныйДокумент");
КонецФункции
Процедура СоздатьНастройкуСвязи(
ТипСвязи,
ССылкаИз,
ССылкаНа,
ХарактерСвязи,
ТипОбратнойСвязи = Неопределено,
ХарактерОбратнойСвязи = Неопределено,
Предопределенная = Ложь,
Комментарий = "") Экспорт
```

```
МенеджерЗаписи =
РегистрыСведений.НастройкаСвязей.СоздатьМенеджерЗапи
си();
МенеджерЗаписи.ТипСвязи = ТипСвязи;
МенеджерЗаписи.ССылкаИз = ССылкаИз;
МенеджерЗаписи.ССылкаНа = ССылкаНа;
МенеджерЗаписи.ХарактерСвязи = ХарактерСвязи;
МенеджерЗаписи.ТипОбратнойСвязи = ТипОбратнойСвязи;
МенеджерЗаписи.ХарактерОбратнойСвязи =
ХарактерОбратнойСвязи;
МенеджерЗаписи.Предопределенная = Предопределенная;
МенеджерЗаписи.Комментарий = Комментарий;
МенеджерЗаписи.Записать();
Если ЗначениеЗаполнено(ТипОбратнойСвязи) Тогда
МенеджерЗаписи =
РегистрыСведений.НастройкаСвязей.СоздатьМенеджерЗапи
си();
МенеджерЗаписи.ТипСвязи = ТипОбратнойСвязи ;
МенеджерЗаписи.ССылкаИз = ССылкаНа;
МенеджерЗаписи.ССылкаНа = ССылкаИз;
МенеджерЗаписи.ХарактерСвязи =
ХарактерОбратнойСвязи;
МенеджерЗаписи.ТипОбратнойСвязи = ТипСвязи;
МенеджерЗаписи.ХарактерОбратнойСвязи =
ХарактерСвязи;
МенеджерЗаписи.Предопределенная = Предопределенная;
МенеджерЗаписи.Комментарий = Комментарий;
МенеджерЗаписи.Записать();
КонецЕсли;
КонецПроцедуры
Процедура СоздатьСвязь(
Документ,
СвязанныйДокумент,
ТипСвязи,
Установил = Неопределено,
ДатаУстановки = Неопределено,
Комментарий = "",
Порядок = 0) Экспорт
НачатьТранзакцию();
Попытка
МенеджерЗаписи =
РегистрыСведений.СвязиДокументов.СоздатьМенеджерЗап
иси();
МенеджерЗаписи.Документ = Документ;
МенеджерЗаписи.СвязанныйДокумент =
СвязанныйДокумент;
МенеджерЗаписи.ТипСвязи = ТипСвязи;
МенеджерЗаписи.Комментарий = Комментарий;
МенеджерЗаписи.Порядок = Порядок;
Если Установил = Неопределено Тогда
МенеджерЗаписи.Установил =
ОбщегоНазначения.ТекущийПользователь();
Иначе
МенеджерЗаписи.Установил = Установил;
КонецЕсли;
Если ДатаУстановки = Неопределено Тогда
МенеджерЗаписи.ДатаУстановки = ТекущаяДата();
Иначе
МенеджерЗаписи.ДатаУстановки = ДатаУстановки;
КонецЕсли;
МенеджерЗаписи.Записать();
НастройкаСвязи = ПолучитьНастройкуСвязи(Документ,
СвязанныйДокумент, ТипСвязи);
Если НастройкаСвязи = Неопределено Тогда
ТекстСообщения =
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = 'Тип связи %1 не настроен для %2'"),
Строка(ТипСвязи), Строка(СвязанныйДокумент));
ВызватьИсключение ТекстСообщения;
КонецЕсли;
Если
ЗначениеЗаполнено(НастройкаСвязи.ТипОбратнойСвязи)
Тогда
МенеджерОбратнойЗаписи =
РегистрыСведений.СвязиДокументов.СоздатьМенеджерЗап
иси();
МенеджерОбратнойЗаписи.Документ =
МенеджерЗаписи.СвязанныйДокумент;
МенеджерОбратнойЗаписи.СвязанныйДокумент =
МенеджерЗаписи.Документ;
МенеджерОбратнойЗаписи.ТипСвязи =
НастройкаСвязи.ТипОбратнойСвязи;
МенеджерОбратнойЗаписи.Установил =
МенеджерЗаписи.Установил;
МенеджерОбратнойЗаписи.ДатаУстановки =
МенеджерЗаписи.ДатаУстановки;
МенеджерОбратнойЗаписи.Комментарий =
МенеджерЗаписи.Комментарий;
МенеджерОбратнойЗаписи.Порядок =
МенеджерЗаписи.Порядок;
МенеджерОбратнойЗаписи.Записать();
КонецЕсли;
ЗафиксироватьТранзакцию();
Исключение
ОтменитьТранзакцию();
ВызватьИсключение;
КонецПопытки;
КонецПроцедуры
Процедура УдалитьСвязь(Документ, СвязанныйДокумент,
ТипСвязи) Экспорт
МенеджерЗаписи =
РегистрыСведений.СвязиДокументов.СоздатьМенеджерЗап
иси();
МенеджерЗаписи.Документ = Документ;
МенеджерЗаписи.СвязанныйДокумент =
СвязанныйДокумент;
МенеджерЗаписи.ТипСвязи = ТипСвязи;
МенеджерЗаписи.Удалить();
НастройкаСвязи = ПолучитьНастройкуСвязи(Документ,
СвязанныйДокумент, ТипСвязи);
Если НастройкаСвязи = Неопределено Тогда
ТекстСообщения =
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВС
троку(
НСтр("ru = 'Тип связи %1 не настроен для %2'"),
Строка(ТипСвязи), Строка(СвязанныйДокумент));
ВызватьИсключение ТекстСообщения;
КонецЕсли;
Если
ЗначениеЗаполнено(НастройкаСвязи.ТипОбратнойСвязи)
Тогда
МенеджерЗаписи =
РегистрыСведений.СвязиДокументов.СоздатьМенеджерЗап
иси();
МенеджерЗаписи.Документ = СвязанныйДокумент;
МенеджерЗаписи.СвязанныйДокумент = Документ;
МенеджерЗаписи.ТипСвязи =
НастройкаСвязи.ТипОбратнойСвязи;
МенеджерЗаписи.Удалить();
КонецЕсли;
КонецПроцедуры
Процедура УстановитьКомментарийТипаСвязи(ТипСвязи,
Комментарий) Экспорт
ТипСвязиОбъект = ТипСвязи.ПолучитьОбъект();
ТипСвязиОбъект.Комментарий = Комментарий;
ТипСвязиОбъект.Записать();
КонецПроцедуры
Функция ПолучитьРодителей(Элемент) Экспорт
Родители = Новый Массив;
ТекущийРодитель = Элемент.Родитель;
Пока Не ТекущийРодитель.Пустая() Цикл
Родители.Добавить(ТекущийРодитель);
ТекущийРодитель = ТекущийРодитель.Родитель;
КонецЦикла;
Возврат Родители;
КонецФункции
Функция ПолучитьНастройкуСвязи(Документ,
СвязанныйДокумент, ТипСвязи) Экспорт
СтруктураВозврата = Новый Структура("ТипСвязи,
ССылкаИз, ССылкаНа, ТипОбратнойСвязи, ХарактерСвязи,
ХарактерОбратнойСвязи");
НастройкиСвязи = Новый ТаблицаЗначений;
НастройкиСвязи.Колонки.Добавить("ТипСвязи");
НастройкиСвязи.Колонки.Добавить("ССылкаИз");
НастройкиСвязи.Колонки.Добавить("УровеньССылкаИз");
НастройкиСвязи.Колонки.Добавить("ССылкаНа");
НастройкиСвязи.Колонки.Добавить("УровеньССылкаНа");
НастройкиСвязи.Колонки.Добавить("Уровень");
НастройкиСвязи.Колонки.Добавить("ХарактерСвязи");
НастройкиСвязи.Колонки.Добавить("ТипОбратнойСвязи");
НастройкиСвязи.Колонки.Добавить("ХарактерОбратнойСвя
зи");
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	НастройкаСвязей.ТипСвязи,
|	НастройкаСвязей.ССылкаИз,
|	НастройкаСвязей.ССылкаНа,
|	НастройкаСвязей.ХарактерСвязи,
|	НастройкаСвязей.ТипОбратнойСвязи,
|	НастройкаСвязей.ХарактерОбратнойСвязи,
|	НастройкаСвязей.Комментарий
|ИЗ
|	РегистрСведений.НастройкаСвязей КАК
НастройкаСвязей
|ГДЕ
|	НастройкаСвязей.ТипСвязи = &ТипСвязи";
Запрос.УстановитьПараметр("ТипСвязи", ТипСвязи);
Если ТипЗнч(Документ) =
Тип("СправочникСсылка.Файлы") Тогда
Запрос.Текст = Запрос.Текст +
" И (ТИПЗНАЧЕНИЯ(НастройкаСвязей.ССылкаИз) =
ТИПЗНАЧЕНИЯ(&Документ)) ";
Запрос.УстановитьПараметр("Документ", Документ);
Иначе
Запрос.Текст = Запрос.Текст +
" И (НастройкаСвязей.ССылкаИз = &ВидДокумента
| ИЛИ НастройкаСвязей.ССылкаИз В (&Родители)
| ИЛИ НастройкаСвязей.ССылкаИз = &ПустаяССылка) ";
ВидДокументаССылкаИз = Документ.ВидДокумента;
ПустаяССылкаИз =
Справочники[ВидДокументаССылкаИз.Метаданные().Имя].
ПустаяССылка();
РодителиССылкаИз =
ПолучитьРодителей(ВидДокументаССылкаИз);
Запрос.УстановитьПараметр("ВидДокумента",
ВидДокументаССылкаИз);
Запрос.УстановитьПараметр("Родители",
РодителиССылкаИз);
Запрос.УстановитьПараметр("ПустаяССылка",
ПустаяССылкаИз);
КонецЕсли;
Если ТипЗнч(СвязанныйДокумент) = Тип("Строка")
Или ТипЗнч(СвязанныйДокумент) =
Тип("СправочникСсылка.Файлы") Тогда
Запрос.Текст = Запрос.Текст +
" И (ТИПЗНАЧЕНИЯ(НастройкаСвязей.ССылкаНа) =
ТИПЗНАЧЕНИЯ(&СвязанныйДокумент)) ";
Запрос.УстановитьПараметр("СвязанныйДокумент",
СвязанныйДокумент);
Иначе
Запрос.Текст = Запрос.Текст +
" И (НастройкаСвязей.ССылкаНа =
&ВидДокументаССылкаНа
|	ИЛИ НастройкаСвязей.ССылкаНа В
(&РодителиССылкаНа)
|	ИЛИ НастройкаСвязей.ССылкаНа =
&ПустаяССылкаНа) ";
ВидДокументаССылкаНа =
СвязанныйДокумент.ВидДокумента;
РодителиССылкаНа =
ПолучитьРодителей(ВидДокументаССылкаНа);
ПустаяССылкаНа =
Справочники[ВидДокументаССылкаНа.Метаданные().Имя].
ПустаяССылка();
Запрос.УстановитьПараметр("ВидДокументаССылкаНа",
ВидДокументаССылкаНа);
Запрос.УстановитьПараметр("РодителиССылкаНа",
РодителиССылкаНа);
Запрос.УстановитьПараметр("ПустаяССылкаНа",
ПустаяССылкаНа);
КонецЕсли;
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
НоваяСтрока = НастройкиСвязи.Добавить();
НоваяСтрока.ТипСвязи = Выборка.ТипСвязи;
НоваяСтрока.ССылкаИз = Выборка.ССылкаИз;
НоваяСтрока.ССылкаНа = Выборка.ССылкаНа;
Если ТипЗнч(Выборка.ССылкаИз) =
Тип("СправочникСсылка.Файлы") Тогда
НоваяСтрока.УровеньССылкаИз = 0;
Иначе
Если Выборка.ССылкаИз = ВидДокументаССылкаИз Тогда
НоваяСтрока.УровеньССылкаИз = 0;
ИначеЕсли Выборка.ССылкаИз = ПустаяССылкаИз Тогда
НоваяСтрока.УровеньССылкаИз = 1000;
Иначе
НоваяСтрока.УровеньССылкаИз =
РодителиССылкаИз.Найти(Выборка.ССылкаИз) + 1;
КонецЕсли;
КонецЕсли;
Если ТипЗнч(Выборка.ССылкаНа) = Тип("Строка")
Или ТипЗнч(Выборка.ССылкаНа) =
Тип("СправочникСсылка.Файлы") Тогда
НоваяСтрока.УровеньССылкаНа = 0;
Иначе
Если Выборка.ССылкаНа = ВидДокументаССылкаНа Тогда
НоваяСтрока.УровеньССылкаНа = 0;
ИначеЕсли Выборка.ССылкаНа = ПустаяССылкаНа Тогда
НоваяСтрока.УровеньССылкаНа = 1000;
Иначе
НоваяСтрока.УровеньССылкаНа =
РодителиССылкаНа.Найти(Выборка.ССылкаНа) + 1;
КонецЕсли;
КонецЕсли;
НоваяСтрока.Уровень = НоваяСтрока.УровеньССылкаИз +
НоваяСтрока.УровеньССылкаНа;
НоваяСтрока.ХарактерСвязи = Выборка.ХарактерСвязи;
НоваяСтрока.ТипОбратнойСвязи =
Выборка.ТипОбратнойСвязи;
НоваяСтрока.ХарактерОбратнойСвязи =
Выборка.ХарактерОбратнойСвязи;
КонецЦикла;
НастройкиСвязи.Сортировать("Уровень Возр");
Если НастройкиСвязи.Количество() > 0 Тогда
ЗаполнитьЗначенияСвойств(СтруктураВозврата,
НастройкиСвязи[0]);
Возврат СтруктураВозврата;
КонецЕсли;
Возврат Неопределено;
КонецФункции
Функция ПолучитьНастройкиСвязи(Документ,
СвязанныйДокумент = Неопределено) Экспорт
НастройкиСвязи = Новый ТаблицаЗначений;
НастройкиСвязи.Колонки.Добавить("ТипСвязи");
НастройкиСвязи.Колонки.Добавить("ССылкаИз");
НастройкиСвязи.Колонки.Добавить("ТипССылкаИз");
НастройкиСвязи.Колонки.Добавить("ТипССылкаИзПредстав
ление");
НастройкиСвязи.Колонки.Добавить("ССылкаНа");
НастройкиСвязи.Колонки.Добавить("ТипССылкаНа");
НастройкиСвязи.Колонки.Добавить("ТипССылкаНаПредстав
ление");
НастройкиСвязи.Колонки.Добавить("ХарактерСвязи");
НастройкиСвязи.Колонки.Добавить("ТипОбратнойСвязи");
НастройкиСвязи.Колонки.Добавить("ХарактерОбратнойСвя
зи");
Если Не ЗначениеЗаполнено(Документ) Тогда
Возврат НастройкиСвязи;
КонецЕсли;
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	НастройкаСвязей.ТипСвязи,
|	НастройкаСвязей.ССылкаИз,
|	НастройкаСвязей.ССылкаНа,
|	НастройкаСвязей.ХарактерСвязи,
|	НастройкаСвязей.ТипОбратнойСвязи,
|	НастройкаСвязей.ХарактерОбратнойСвязи,
|	НастройкаСвязей.Комментарий
```

```
|ИЗ
|	РегистрСведений.НастройкаСвязей КАК НастройкаСвязей";
Если ТипЗнч(Документ) = Тип("СправочникСсылка.Файлы") Тогда
Запрос.Текст = Запрос.Текст +
" ГДЕ (ТИПЗНАЧЕНИЯ(НастройкаСвязей.ССылкаИз) = ТИПЗНАЧЕНИЯ(&Документ)) ";
Запрос.УстановитьПараметр("Документ", Документ);
Иначе
Запрос.Текст = Запрос.Текст +
" ГДЕ (НастройкаСвязей.ССылкаИз = &ВидДокумента
| ИЛИ НастройкаСвязей.ССылкаИз В (&Родители)
| ИЛИ НастройкаСвязей.ССылкаИз = &ПустаяССылка) ";
ВидДокументаССылкаИз = Документ.ВидДокумента;
ПустаяССылкаИз = Справочники[ВидДокументаССылкаИз.Метаданные().Имя].ПустаяССылка();
РодителиИз = ПолучитьРодителей(ВидДокументаССылкаИз);
Запрос.УстановитьПараметр("ВидДокумента", ВидДокументаССылкаИз);
Запрос.УстановитьПараметр("Родители", РодителиИз);
Запрос.УстановитьПараметр("ПустаяССылка", ПустаяССылкаИз);
КонецЕсли;
Если СвязанныйДокумент <> Неопределено Тогда
Если ТипЗнч(СвязанныйДокумент) = Тип("Строка")
Или ТипЗнч(СвязанныйДокумент) = Тип("СправочникСсылка.Файлы") Тогда
Запрос.Текст = Запрос.Текст +
" И (ТИПЗНАЧЕНИЯ(НастройкаСвязей.ССылкаНа) = ТИПЗНАЧЕНИЯ(&СвязанныйДокумент)) ";
Запрос.УстановитьПараметр("СвязанныйДокумент", СвязанныйДокумент);
Иначе
Запрос.Текст = Запрос.Текст +
" И (НастройкаСвязей.ССылкаНа = &ВидДокументаССылкаНа
|	ИЛИ НастройкаСвязей.ССылкаНа В (&РодителиССылкаНа)
|	ИЛИ НастройкаСвязей.ССылкаНа = &ПустаяССылкаНа) ";
ВидДокументаССылкаНа = СвязанныйДокумент.ВидДокумента;
РодителиССылкаНа = ПолучитьРодителей(ВидДокументаССылкаНа);
ПустаяССылкаНа = Справочники[ВидДокументаССылкаНа.Метаданные().Имя].ПустаяССылка();
Запрос.УстановитьПараметр("ВидДокументаССылкаНа", ВидДокументаССылкаНа);
Запрос.УстановитьПараметр("РодителиССылкаНа", РодителиССылкаНа);
Запрос.УстановитьПараметр("ПустаяССылкаНа", ПустаяССылкаНа);
КонецЕсли;
КонецЕсли;
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
НоваяСтрока = НастройкиСвязи.Добавить();
НоваяСтрока.ТипСвязи = Выборка.ТипСвязи;
НоваяСтрока.ССылкаИз = Выборка.ССылкаИз;
НоваяСтрока.ССылкаНа = Выборка.ССылкаНа;
Если ТипЗнч(Выборка.ССылкаИз) = Тип("СправочникСсылка.ВидыВходящихДокументов") Тогда
НоваяСтрока.ТипССылкаИз = "СправочникСсылка.ВходящиеДокументы";
НоваяСтрока.ТипССылкаИзПредставление = "Входящий документ";
ИначеЕсли ТипЗнч(Выборка.ССылкаИз) = Тип("СправочникСсылка.ВидыИсходящихДокументов") Тогда
НоваяСтрока.ТипССылкаИз = "СправочникСсылка.ИсходящиеДокументы";
НоваяСтрока.ТипССылкаИзПредставление = "Исходящий документ";
ИначеЕсли ТипЗнч(Выборка.ССылкаИз) = Тип("СправочникСсылка.ВидыВнутренниеДокументов") Тогда
НоваяСтрока.ТипССылкаИз = "СправочникСсылка.ВнутренниеДокументы";
НоваяСтрока.ТипССылкаИзПредставление = "Внутренний документ";
ИначеЕсли ТипЗнч(Выборка.ССылкаИз) = Тип("СправочникСсылка.Файлы") Тогда
НоваяСтрока.ТипССылкаИз = "СправочникСсылка.Файлы";
НоваяСтрока.ТипССылкаИзПредставление = "Файл";
КонецЕсли;
Если ТипЗнч(Выборка.ССылкаНа) = Тип("СправочникСсылка.ВидыВходящихДокументов") Тогда
НоваяСтрока.ТипССылкаНа = "СправочникСсылка.ВходящиеДокументы";
НоваяСтрока.ТипССылкаНаПредставление = "Входящий документ";
ИначеЕсли ТипЗнч(Выборка.ССылкаНа) = Тип("СправочникСсылка.ВидыИсходящихДокументов") Тогда
НоваяСтрока.ТипССылкаНа = "СправочникСсылка.ИсходящиеДокументы";
НоваяСтрока.ТипССылкаНаПредставление = "Исходящий документ";
ИначеЕсли ТипЗнч(Выборка.ССылкаНа) = Тип("СправочникСсылка.ВидыВнутренниеДокументов") Тогда
НоваяСтрока.ТипССылкаНа = "СправочникСсылка.ВнутренниеДокументы";
НоваяСтрока.ТипССылкаНаПредставление = "Внутренний документ";
ИначеЕсли ТипЗнч(Выборка.ССылкаНа) = Тип("Строка") Тогда
НоваяСтрока.ТипССылкаНа = "Строка";
НоваяСтрока.ТипССылкаНаПредставление = "Внешний ресурс";
ИначеЕсли ТипЗнч(Выборка.ССылкаНа) = Тип("СправочникСсылка.Файлы") Тогда
НоваяСтрока.ТипССылкаНа = "СправочникСсылка.Файлы";
НоваяСтрока.ТипССылкаНаПредставление = "Файл";
КонецЕсли;
НоваяСтрока.ХарактерСвязи = Выборка.ХарактерСвязи;
НоваяСтрока.ТипОбратнойСвязи = Выборка.ТипОбратнойСвязи;
НоваяСтрока.ХарактерОбратнойСвязи = Выборка.ХарактерОбратнойСвязи;
КонецЦикла;
Возврат НастройкиСвязи;
КонецФункции
Процедура УстановитьСвязь(
Документ,
НачальныйСвязанныйДокумент,
СвязанныйДокумент,
ТипСвязи,
Установил = Неопределено,
ДатаУстановки = Неопределено,
Комментарий = "") Экспорт
Если НачальныйСвязанныйДокумент = СвязанныйДокумент Тогда
Возврат;
КонецЕсли;
Если ЗначениеЗаполнено(НачальныйСвязанныйДокумент) Тогда
Если ЗначениеЗаполнено(СвязанныйДокумент) Тогда
СвязиДокументов.УдалитьСвязь(Документ, НачальныйСвязанныйДокумент, ТипСвязи);
СвязиДокументов.СоздатьСвязь(Документ, СвязанныйДокумент, ТипСвязи, Установил, ДатаУстановки, Комментарий);
Иначе
СвязиДокументов.УдалитьСвязь(Документ, НачальныйСвязанныйДокумент, ТипСвязи);
КонецЕсли;
Иначе
Если ЗначениеЗаполнено(СвязанныйДокумент) Тогда
СвязиДокументов.СоздатьСвязь(Документ, СвязанныйДокумент, ТипСвязи, Установил, ДатаУстановки, Комментарий);
КонецЕсли;
КонецЕсли;
НачальныйСвязанныйДокумент = СвязанныйДокумент;
КонецПроцедуры
Процедура ОбновитьСвязиДокумента(СвязанныйДокумент) Экспорт
УстановитьПривилегированныйРежим(Истина);
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	СвязиДокументов.Документ,
|	СвязиДокументов.ТипСвязи,
|	СвязиДокументов.СвязанныйДокумент
|ИЗ
|	РегистрСведений.СвязиДокументов КАК СвязиДокументов
|ГДЕ
|	СвязиДокументов.СвязанныйДокумент = &СвязанныйДокумент";
Запрос.УстановитьПараметр("СвязанныйДокумент", СвязанныйДокумент);
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
МенеджерЗаписи = РегистрыСведений.СвязиДокументов.СоздатьМенеджерЗаписи();
МенеджерЗаписи.Документ = Выборка.Документ;
МенеджерЗаписи.ТипСвязи = Выборка.ТипСвязи;
МенеджерЗаписи.СвязанныйДокумент = Выборка.СвязанныйДокумент;
МенеджерЗаписи.УдалитьСвязанныйДокумент = Неопределено;
МенеджерЗаписи.СвязаннаяСтрока = "";
МенеджерЗаписи.Прочитать();
Если ТипЗнч(СвязанныйДокумент) = Тип("СправочникСсылка.ИсходящиеДокументы") Тогда
Получатели = СвязанныйДокумент.Получатели;
Отправлен = Ложь;
Если Получатели.Количество() = 1 Тогда
Отправлен = Получатели[0].Отправлен;
ИначеЕсли Получатели.Найти(Ложь, "Отправлен") = Неопределено Тогда
Отправлен = Истина;
ИначеЕсли ТипЗнч(МенеджерЗаписи.Документ) = Тип("СправочникСсылка.ВходящиеДокументы") Тогда
ВходящийДокумент = МенеджерЗаписи.Документ;
ПараметрыОтбора = Новый Структура("Получатель", ВходящийДокумент.Отправитель);
НайденныеСтроки = Получатели.НайтиСтроки(ПараметрыОтбора);
Если НайденныеСтроки.Количество() = 1 Тогда
Отправлен = НайденныеСтроки[0].Отправлен;
Иначе
ПараметрыОтбора = Новый Структура("Получатель, Адресат", ВходящийДокумент.Отправитель, ВходящийДокумент.Подписал);
НайденныеСтроки = Получатели.НайтиСтроки(ПараметрыОтбора);
Если НайденныеСтроки.Количество() = 1 Тогда
Отправлен = НайденныеСтроки[0].Отправлен;
КонецЕсли;
КонецЕсли;
КонецЕсли;
Если Отправлен <> МенеджерЗаписи.СвязанныйДокументОтправлен Тогда
МенеджерЗаписи.Записать();
КонецЕсли;
КонецЕсли;
КонецЦикла;
КонецПроцедуры
Функция УстановитьРеквизитыПриДобавленииСвязи(ДокументССылка, УникальныйИдентификаторДокумента, ТипСвязи) Экспорт
РеквизитыИзменены = Ложь;
Если ТипЗнч(ДокументССылка) = Тип("СправочникСсылка.ВходящиеДокументы") Тогда
Если (ТипСвязи = Справочники.ТипыСвязей.ПервичноеОбращение) И (Не ДокументССылка.Повторное) Тогда
ДокументОбъект = ДокументССылка.ПолучитьОбъект();
ЗаблокироватьДанныеДляРедактирования(ДокументОбъект.ССылка, , УникальныйИдентификаторДокумента);
ДокументОбъект.Повторное = Истина;
ДокументОбъект.Записать();
РазблокироватьДанныеДляРедактирования(ДокументОбъект.ССылка, УникальныйИдентификаторДокумента);
РеквизитыИзменены = Истина;
КонецЕсли;
Если (ТипСвязи = Справочники.ТипыСвязей.ОсновноеОбращение) И (Не ДокументССылка.Дубликат) Тогда
ДокументОбъект = ДокументССылка.ПолучитьОбъект();
ЗаблокироватьДанныеДляРедактирования(ДокументОбъект.ССылка,, УникальныйИдентификаторДокумента);
ДокументОбъект.Дубликат = Истина;
ДокументОбъект.Записать();
РазблокироватьДанныеДляРедактирования(ДокументОбъект.ССылка, УникальныйИдентификаторДокумента);
РеквизитыИзменены = Истина;
КонецЕсли;
ИначеЕсли ТипЗнч(ДокументССылка) = Тип("СправочникСсылка.ВнутренниеДокументы") Тогда
Если (ТипСвязи = Справочники.ТипыСвязей.НеДействуетВСоответствии) И (Не ДокументССылка.НеДействует) Тогда
ДокументОбъект = ДокументССылка.ПолучитьОбъект();
ЗаблокироватьДанныеДляРедактирования(ДокументОбъект.ССылка,, УникальныйИдентификаторДокумента);
ДокументОбъект.НеДействует = Истина;
ДокументОбъект.Записать();
РазблокироватьДанныеДляРедактирования(ДокументОбъект.ССылка, УникальныйИдентификаторДокумента);
РеквизитыИзменены = Истина;
КонецЕсли;
КонецЕсли;
Возврат РеквизитыИзменены;
КонецФункции
Функция УстановитьРеквизитыПриУдаленииСвязи(ДокументССылка, УникальныйИдентификаторДокумента, ТипСвязи) Экспорт
РеквизитыИзменены = Ложь;
Если ТипЗнч(ДокументССылка) = Тип("СправочникСсылка.ВходящиеДокументы") Тогда
Если (ТипСвязи = Справочники.ТипыСвязей.ПервичноеОбращение) И ДокументССылка.Повторное Тогда
ДокументОбъект = ДокументССылка.ПолучитьОбъект();
ЗаблокироватьДанныеДляРедактирования(ДокументОбъект.ССылка,, УникальныйИдентификаторДокумента);
ДокументОбъект.Повторное = Ложь;
ДокументОбъект.Записать();
РазблокироватьДанныеДляРедактирования(ДокументОбъект.ССылка, УникальныйИдентификаторДокумента);
РеквизитыИзменены = Истина;
КонецЕсли;
Если (ТипСвязи = Справочники.ТипыСвязей.ОсновноеОбращение) И ДокументССылка.Дубликат Тогда
ДокументОбъект = ДокументССылка.ПолучитьОбъект();
ЗаблокироватьДанныеДляРедактирования(ДокументОбъект.ССылка,, УникальныйИдентификаторДокумента);
ДокументОбъект.Дубликат = Ложь;
ДокументОбъект.Записать();
РазблокироватьДанныеДляРедактирования(ДокументОбъект.ССылка, УникальныйИдентификаторДокумента);
РеквизитыИзменены = Истина;
КонецЕсли;
ИначеЕсли ТипЗнч(ДокументССылка) = Тип("СправочникСсылка.ВнутренниеДокументы") Тогда
Если (ТипСвязи = Справочники.ТипыСвязей.НеДействуетВСоответствии) И ДокументССылка.НеДействует Тогда
ДокументОбъект = ДокументССылка.ПолучитьОбъект();
```

```
ЗаблокироватьДанныеДляРедактирования(ДокументОбъект.Ссылка,, УникальныйИдентификаторДокумента);
ДокументОбъект.НеДействует = Ложь;
ДокументОбъект.Записать();
РазблокироватьДанныеДляРедактирования(ДокументОбъект.Ссылка, УникальныйИдентификаторДокумента);
РеквизитыИзменены = Истина;
КонецЕсли;
КонецЕсли;
Возврат РеквизитыИзменены;
КонецФункции
// Обработчик обновления информационной базы
//
Процедура ПерейтиНаВерсию_1_2_1_3() Экспорт
Набор = РегистрыСведений.СвязиДокументов.СоздатьНаборЗаписей();
Набор.Отбор.ТипСвязи.Установить(Справочники.ТипыСвязей.Содержит);
Набор.Прочитать();
Набор.Очистить();
Набор.Записать(Истина);
Набор = РегистрыСведений.СвязиДокументов.СоздатьНаборЗаписей();
Набор.Отбор.ТипСвязи.Установить(Справочники.ТипыСвязей.ВходитВКомплект);
Набор.Прочитать();
Набор.Очистить();
Набор.Записать(Истина);
УстановитьКомментарийТипаСвязи(Справочники.ТипыСвязей.ВходитВКомплект, НСтр("ru = 'Ссылка на комплект документов'"));
УстановитьКомментарийТипаСвязи(Справочники.ТипыСвязей.Содержит, НСтр("ru = 'Ссылка на элемент комплекта'"));
СоздатьНастройкуСвязи(Справочники.ТипыСвязей.Содержит,
Справочники.ВидыВнутреннихДокументов.ПустаяСсылка(),
Справочники.ВидыВнутреннихДокументов.ПустаяСсылка(),
Перечисления.ХарактерСвязей.Множественная,
Справочники.ТипыСвязей.ВходитВКомплект,
Перечисления.ХарактерСвязей.Множественная,
Истина);
СоздатьНастройкуСвязи(Справочники.ТипыСвязей.Содержит,
Справочники.ВидыВнутреннихДокументов.ПустаяСсылка(),
Справочники.ВидыВходящихДокументов.ПустаяСсылка(),
Перечисления.ХарактерСвязей.Множественная,
Справочники.ТипыСвязей.ВходитВКомплект,
Перечисления.ХарактерСвязей.Множественная,
Истина);
СоздатьНастройкуСвязи(Справочники.ТипыСвязей.Содержит,
Справочники.ВидыВнутреннихДокументов.ПустаяСсылка(),
Справочники.ВидыИсходящихДокументов.ПустаяСсылка(),
Перечисления.ХарактерСвязей.Множественная,
Справочники.ТипыСвязей.ВходитВКомплект,
Перечисления.ХарактерСвязей.Множественная,
Истина);
СоздатьНастройкуСвязи(Справочники.ТипыСвязей.Содержит,
Справочники.ВидыВнутреннихДокументов.ПустаяСсылка(),
Справочники.Файлы.ПустаяСсылка(),
Перечисления.ХарактерСвязей.Множественная,,,
Истина);
КонецПроцедуры
Процедура ОбновитьСвязиПриРазделенииИзмерения() Экспорт
// Разделение данных в регистре сведений СвязиДокументов на два измерения
Если Метаданные.РегистрыСведений.СвязиДокументов.Измерения.Найти("УдалитьСвязанныйДокумент") <> Неопределено Тогда
ВыборкаТипыСвязей = Справочники.ТипыСвязей.Выбрать();
Пока ВыборкаТипыСвязей.Следующий() Цикл
ТипСвязи = ВыборкаТипыСвязей.Ссылка;
НаборСвязиДокументов = РегистрыСведений.СвязиДокументов.СоздатьНаборЗаписей();
НаборСвязиДокументов.Отбор.ТипСвязи.Установить(ТипСвязи);
НаборСвязиДокументов.Прочитать();
ЗаписатьНабор = Ложь;
Для каждого Запись Из НаборСвязиДокументов Цикл
Если ЗначениеЗаполнено(Запись.СвязанныйДокумент)
Или ЗначениеЗаполнено(Запись.СвязаннаяСтрока)
Или Не ЗначениеЗаполнено(Запись.УдалитьСвязанныйДокумент)
Тогда
Продолжить;
КонецЕсли;
Если ТипЗнч(Запись.УдалитьСвязанныйДокумент) = Тип("Строка") Тогда
Запись.СвязаннаяСтрока = Запись.УдалитьСвязанныйДокумент;
Иначе
Запись.СвязанныйДокумент = Запись.УдалитьСвязанныйДокумент;
КонецЕсли;
Запись.УдалитьСвязанныйДокумент = Неопределено;
ЗаписатьНабор = Истина;
КонецЦикла;
Если ЗаписатьНабор Тогда
НаборСвязиДокументов.ОбменДанными.Загрузка = Истина;
НаборСвязиДокументов.Записать();
КонецЕсли;
КонецЦикла;
КонецЕсли;
КонецПроцедуры
```