Интеграция системы ELMA с системой «1С: Предприятие»

Краткое руководство

elma

Система управления бизнес-процессами и эффективностью

Оглавление

Введение

Для систем «1С: Предприятие» в составе ELMA имеется специальный отдельно лицензируемый модуль для построения быстрого и эффективного взаимодействия.

Модуль «Интеграция с 1С» позволяет обеспечить интеграцию системы ELMA с «1С: Предприятие» (1С) версий 8.0, 8.1, 8.2 и 8.3, т.е. со всеми активно используемыми в наши дни версиями.

В рамках данного краткого руководства рассмотрим общие принципы взаимодействия системы ELMA и «1С: Предприятие», а также решение задачи по использованию ELMA в качестве средства взаимодействия интернет-магазина и 1С.

В главах данной книги приведена информация по подключению конфигурации, дано общее описание возможностей интеграции через COM-соединение. Также в данной книге подробно описана настройка синхронизации номенклатуры 1С с ELMA, процедура обработки заказов интернет-магазина и особенности работы с Web API ELMA из 1С. Данное руководство предназначено для специалистов, которые планируют профессионально заниматься внедрением системы.

Данная книга предполагает, что пользователь уже знаком с архитектурой системы ELMA и владеет базовыми навыками работы с ней, которые описаны в кратком руководстве по Платформе ELMA BPM. Также предполагается, что пользователь знаком с основами администрирования системы ELMA, которые описаны в кратком руководстве администратора ELMA.

Полное описание функционала системы ELMA приведено в справке по системе. Справка входит в поставку системы, а также всегда доступна в Базе знаний ELMA: http://www.elma-bpm.ru/kb/help/.

Решения многих технических вопросов описаны в Базе знаний ELMA по адресу http://www.elma-bpm.ru/kb/. База знаний постоянно пополняется специалистами компании.

Глава 1. Подключение конфигурации

Для успешной и безотказной интеграции системы ELMA и «1С: Предприятие» версий 8.x необходимо использовать COMConnector. В настройках подключения доступен также режим COMApplication, однако, при работе с ним возможны перебои с инициализацией подключения. Например, перебои, вызванные модальными окнами, возникающими при работе 1С.

Работа возможна как с базой, размещенной на сервере 1С, так и с файловым вариантом размещения. На компьютер, где установлен сервер ELMA необходимо установить платформу 1С (клиентское приложение) той же версии, что и сервер 1С, на котором установлена нужная для интеграции база 1С.

1.1. Настройка компонентов Windows

После установки клиентского приложения 1С необходимо проверить и, при необходимости, настроить службы компонентов в операционной системе на сервере ELMA.

1.1.1. Создание нового приложения

Для создания нового приложения необходимо:

1. Зайти в службы компонентов **Панель управления -> Администрирование -> Службы компонентов**.
2. Выбрать **Службы компонентов -> Компьютеры -> Мой компьютер -> Приложения COM+**.
3. Создать новое приложение, выбрав в контекстном меню пункт **Создать – Приложение** (Рис. 1)

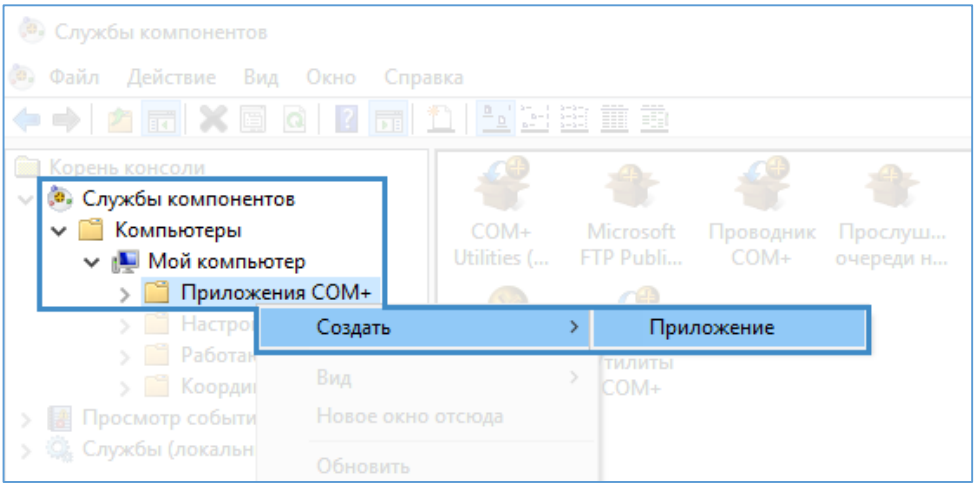


Рис. 1. Настройка компонентов Windows

Откроется Мастер установки приложения COM+. Далее необходимо:

1. В стартовом окне мастера установки (Рис. 2) нажать на кнопку **Далее**.

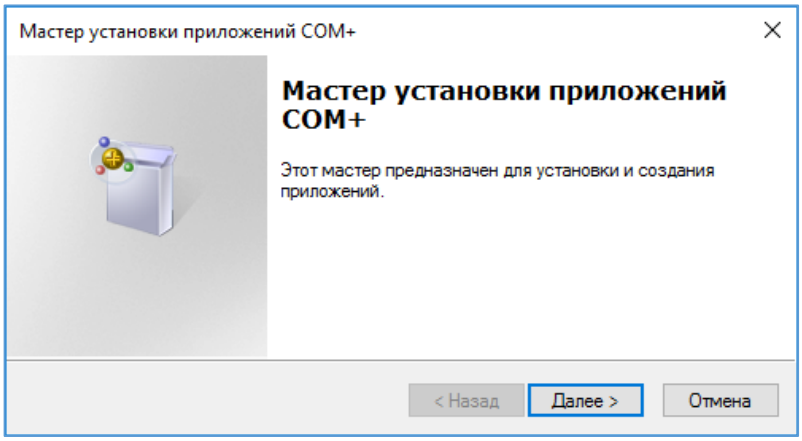


Рис. 2. Мастер установки приложений COM+

2. На следующем шаге (Рис. 3) необходимо выбрать **Создать новое приложение**.

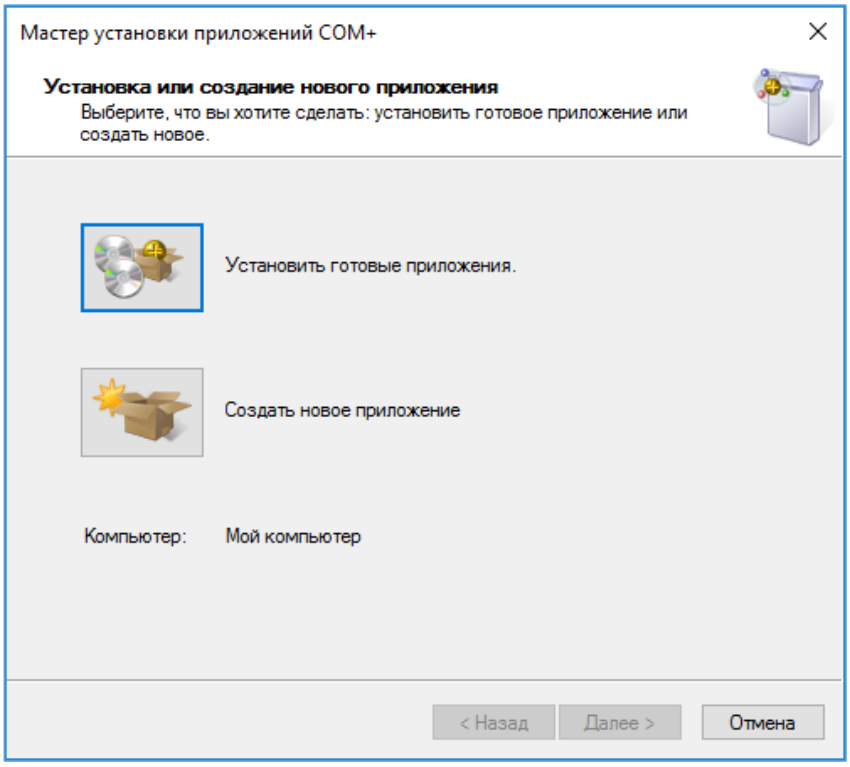


Рис. 3. Мастер установки приложений COM+

3. Далее требуется указать **Имя нового приложения** в виде «V82COMConnector» (не зависит от версии 8.2 или 8.3), способ активации – **Серверное приложение** (Рис. 4) и нажать на кнопку **Далее**.

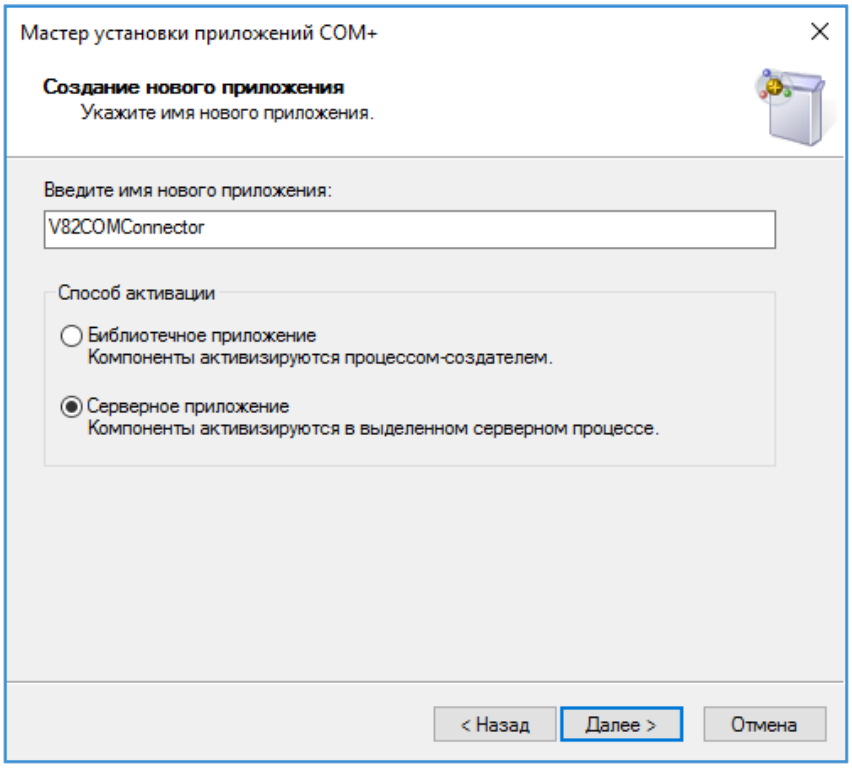


Рис. 4. Мастер установки приложений COM+

4. Далее требуется указать пользователя (Рис. 5), от имени которого будет запускаться приложение и нажать на кнопку **Далее**.

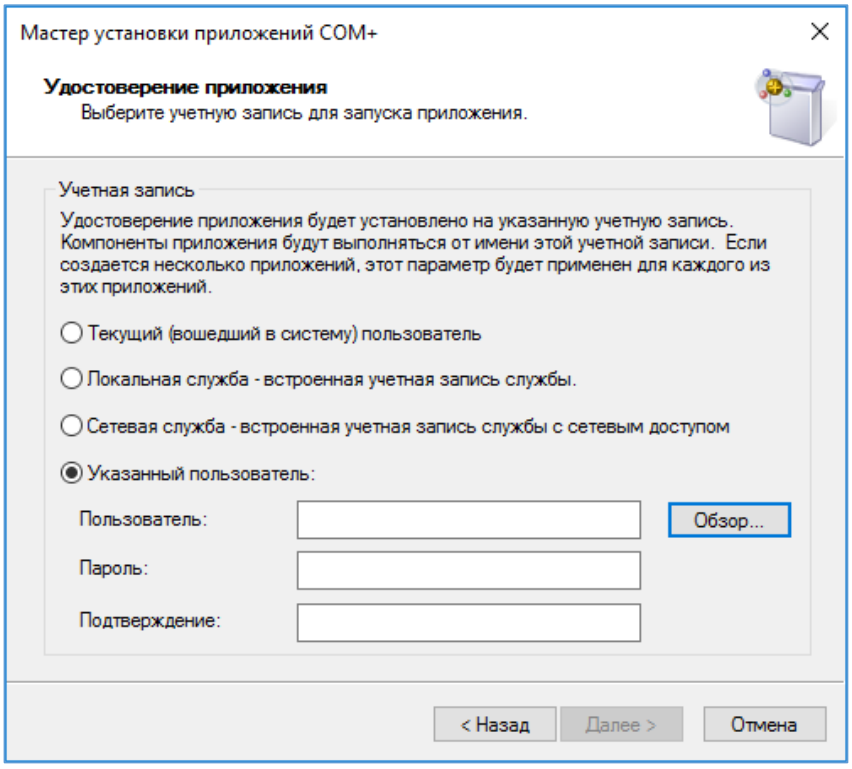


Рис. 5. Мастер установки приложений COM+

5. На остальных шагах необходимо все настройки оставить по умолчанию (пропустить их, нажав на кнопку **Далее**).
6. Для завершения работы мастера необходимо нажать на кнопку **Готово**.

1.1.2. Создание нового компонента

Далее требуется создать новый компонент. Для этого необходимо в **Службах компонентов** перейти в **Компьютеры -> Мой компьютер -> Приложения COM+ -> V82COMConnector -> Компоненты** и в контекстном меню выбрать пункт **Создать – Компонент** (Рис. 6).

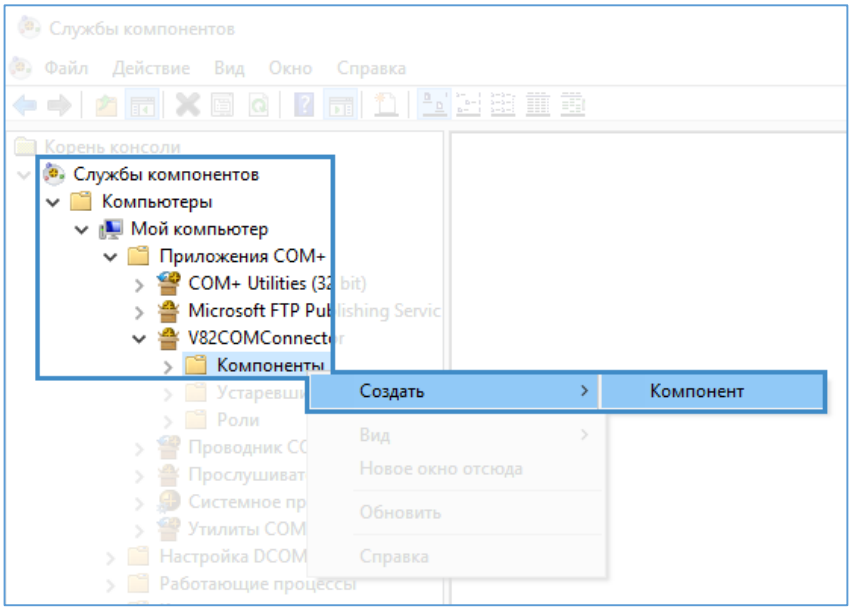


Рис. 6. Настройка компонентов Windows

Откроется Мастер установки компонентов COM+. Далее необходимо:

1. В стартовом окне мастера установки (Рис. 7) нажать на кнопку **Далее**.

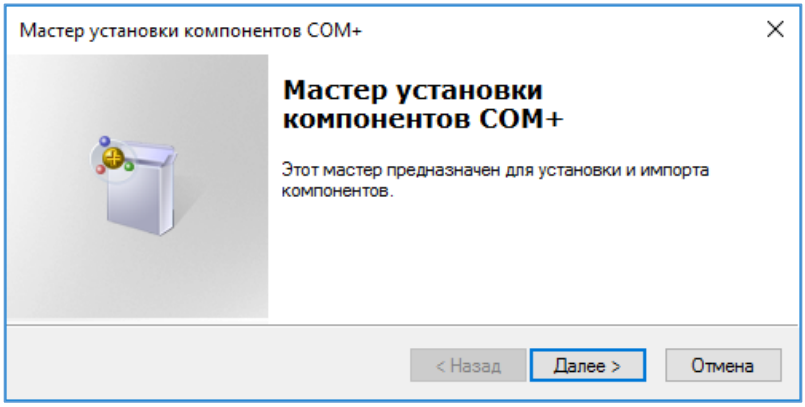


Рис. 7. Мастер установки приложений COM+

2. На следующем шаге (Рис. 8) необходимо выбрать **Установка новых компонентов**.

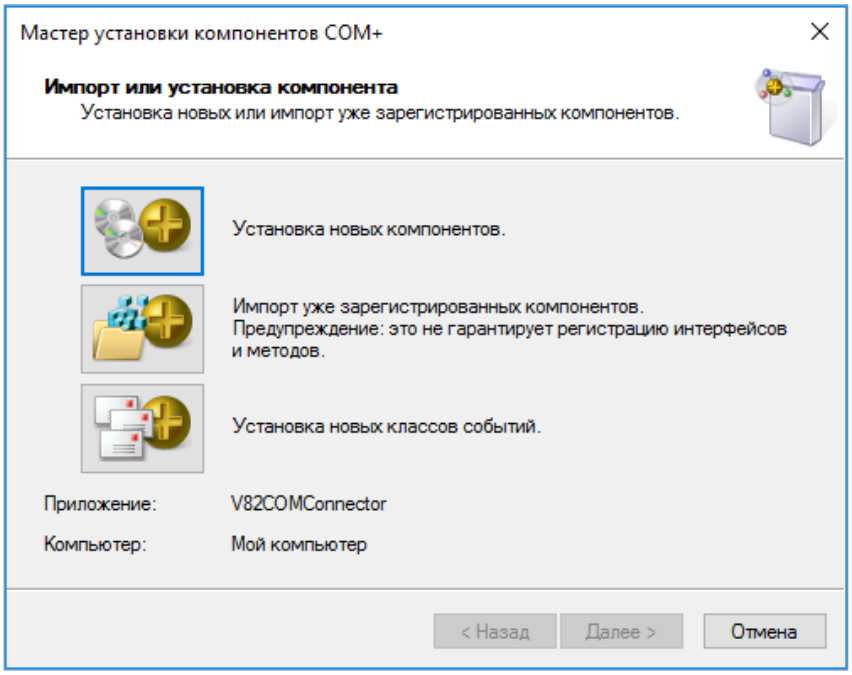


Рис. 8. Мастер установки приложений COM+

При этом необходимо указать путь до файла, расположенного в директории **C:\Program Files(x86)\1cv8\8.3.10.2168\bin\comcntr.dll**. Обратите внимание, пути до файла могут отличаться:

3. Для 32-битных систем или для 64-битной платформы «1С: Предприятие» – **C:\Program Files**.
4. Для версий платформы 8.2 – «1cv82» вместо «1cv8».
5. Вместо «8.3.10.2168» необходимо выбрать папку с нужной версией платформы.

Нажмите на кнопку **Далее**.

6. Для завершения работы мастера необходимо нажать на кнопку **Готово**.

Стоит отметить, что одновременно в системе может быть зарегистрирован только один компонент для одной из версий платформы 1С 8.3 и для 1С 8.2. Поэтому необходимо, чтобы все базы 1С, с которыми осуществляется интеграция, были запущены на одной версии платформы, или же на разных семействах (1С 8.2 и 8.3). Если происходит обновление платформы, компонент необходимо перерегистрировать вручную, предварительно удалив его из дерева приложения **V82COMConnector**.

Также могут возникнуть проблемы, если сервер приложений ELMA установлен на том же сервере, где работает сервер приложений 1С. В этом случае, в настройках приложения **V82COMConnector** (пункт **Свойства** в контекстном меню) на вкладке

Удостоверение необходимо в качестве учетной записи указать **Локальная служба – встроенная учетная запись службы** (Рис. 9).

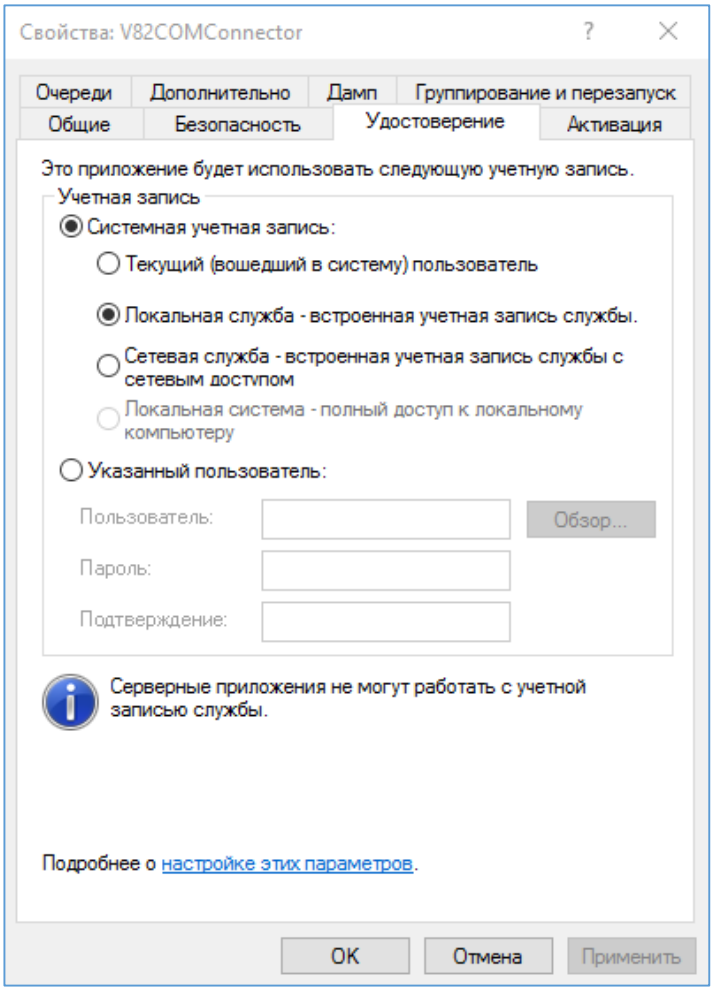


Рис. 9. Настройка учетной записи запуска COM-приложения

1.2. Настройка подключения к базе данных 1С

После настройки компонентов Windows следует настроить подключение к базе данных 1С из ELMA.

Для этого в Дизайнере ELMA переходим в раздел **Меню -> Интеграция с 1С**. В левой части окна нажимаем на кнопку **Добавить**.

В открывшемся диалоговом окне (Рис. 10) указываем необходимые настройки подключения к базе. В зависимости от типа расположения информационной базы потребуется указать путь к папке с информационной базой 1С или настройки кластера и имени базы.

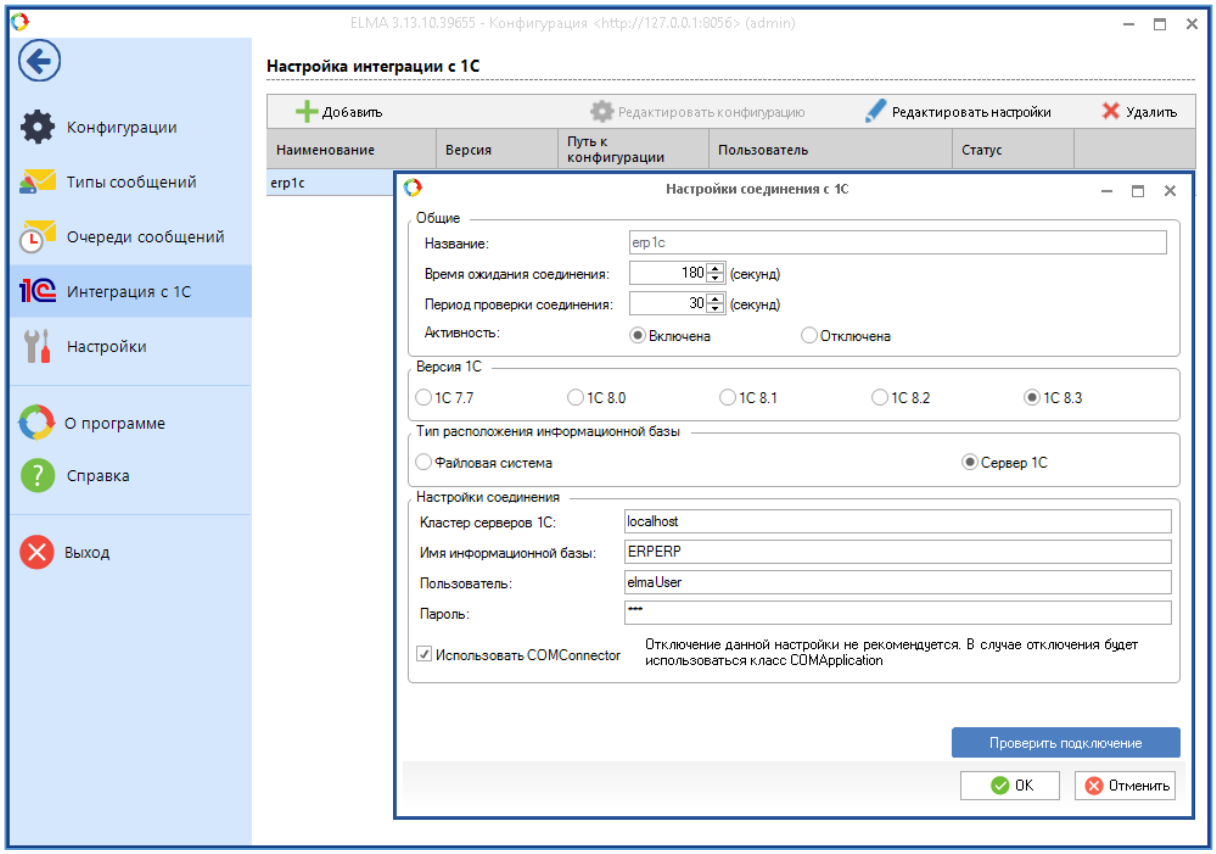


Рис. 10. Настройка подключения ELMA к информационной базе 1С

В качестве названия рекомендуется использовать латинские буквы без пробелов и других специальных знаков (для более наглядного вызова подключения в сценариях ELMA). После ввода настроек при помощи кнопки **Проверить подключение** можно осуществить попытку соединения с 1С. Если проверка прошла успешно, нажмите на кнопку **OK**.

При необходимости можно указать, какие данные из системы «1С: Предприятие» (справочники и документы) требуется использовать в системе ELMA. На этом настройка интеграции систем заканчивается – данные можно использовать.

Для этого в списке подключенных информационных баз 1С устанавливаем курсор на нужную нам базу и нажимаем на кнопку **Редактировать конфигурацию**. В появившемся окне можно выбрать объекты конфигурации 1С, с которыми необходимо осуществлять работу в ELMA (Рис. 11). Настраиваются как справочники, так и документы.

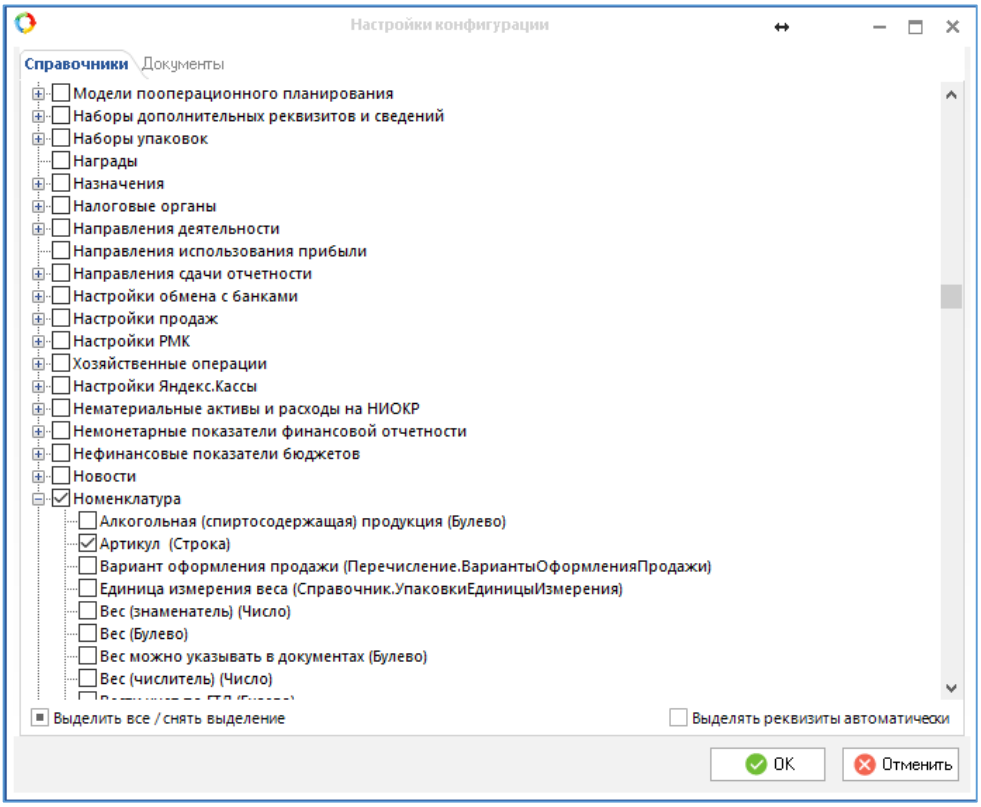


Рис. 11. Редактирование настроек конфигурации 1С

Для каждого объекта необходимо выбрать набор реквизитов, которые будут отображаться в ELMA. Реквизиты «Код» и «Наименование» для справочников, а также «Номер» и «Дата» для документов импортируются автоматически и не выбираются в данном диалоговом окне.

Настройки интеграции с 1С хранятся в директории **<Общая папка с системой ELMA>\UserConfig\Integration1C**.

После выполнения всех требуемых настроек для начала работы с объектами потребуется перезапуск сервера ELMA.

Проверить правильность настроек интеграции можно различными способами.

Так, например, в веб-приложении ELMA можно зайти в раздел **Справочники** и в дереве справочников выбрать пункт **Интеграция с 1С**. В нем будет отображено вложенное дерево с названием подключенной базы (как оно было задано в настройках) и перечнем интегрированных справочников (Рис. 12).

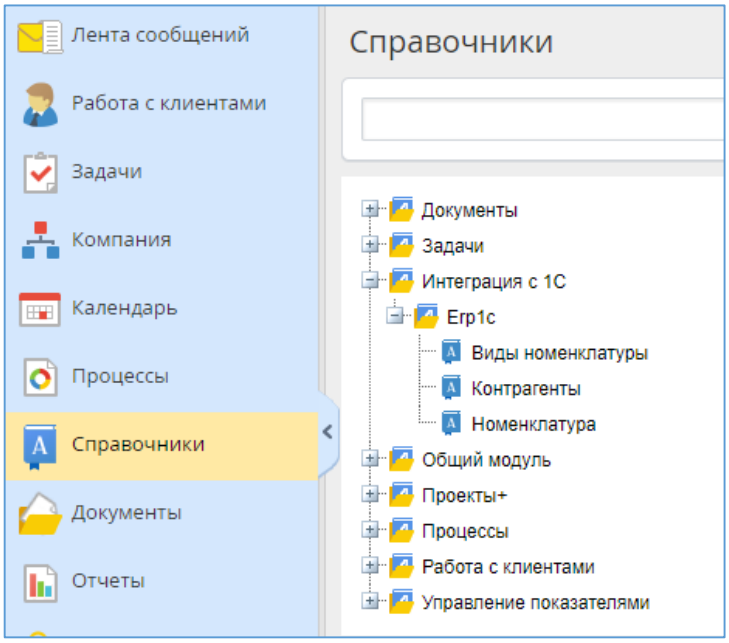


Рис. 12. Представление справочников 1С в веб-приложении ELMA

Зайдя в любой из этих справочников, будет отображен список его элементов с учетом иерархии. В настройках списка можно указать, какие поля (которые были помечены в настройках конфигурации) будут отображены в списке.

Также можно провести эмуляцию работы с 1С в сценарии. Для работы сценария необходимо подключить пространства имен:

```
using EleWise.ELMA.Integration1C;
using EleWise.ELMA.Integration1C.Data;
using EleWise.ELMA.Integration1C.V81;
using EleWise.ELMA.Model.Common;
using EleWise.ELMA.Model.Entities;
using EleWise.ELMA.Model.Managers;
using EleWise.ELMA.Model.Types.Settings;
using EleWise.ELMA.Services;
```

Текст процедуры проверки:

```
public virtual void testConnection1C (Context context)
{
    //инициализируем подключение
    var service = Locator.GetServiceNotNull<Integration1CService>();
```

```
    ComObject connector = service.GetComConnector("erp1c");
    //получаем менеджер объектов 1С
    var Manager1C = (dynamic)connector.Reference;
    //проверяем свойства глобального контекста "ПараметрыСеанса",
которое доступно в большинстве конфигурааций
    Console.WriteLine(Manager1C.ПараметрыСеанса.ТекущийПользователь.Н
аименование);
}
```

Процедуру можно запустить в эмуляции выполнения сценариев, результатом будет выведенное в Консоль сообщений имя пользователя, указанного при настройке интеграции ELMA и 1С.

Глава 2. Общее описание возможностей интеграции через COM-соединение

Интеграция 1С и ELMA при помощи COMConnector имеет свои особенности, которые стоит учитывать при разработке продукта. Для работы всех сценариев (если дополнительно не указано иной информации), представленных в рамках данной главы необходимо подключать пространства имен:

```
using EleWise.ELMA.Services;
using EleWise.ELMA.Integration1C;
using EleWise.ELMA.Integration1C.Data;
using EleWise.ELMA.Integration1C.V82;
```

Также, необходимо подключать сборку:

```
Elewise.ELMA.Integration1C
```

Если у конфигурации 1С, для которой ведется разработка решений по интеграции, используется режим запуска «Управляемое приложение», необходимо при разработке методов, вызываемых в ELMA, указывать перед каждой процедурой и функцией директиву выполнения кода на сервере:

```
&НаСервере
```

2.1. Особенности работы со справочниками и документами 1С в веб-приложении системы ELMA

При выборе объектов конфигурации (справочников и документов) в настройках в ELMA записывается только структура объектов 1С. Если для объектов выбраны реквизиты не примитивных типов (другие справочники и документы, перечисления), то для каждого из них интеграция будет настроена автоматически.

В веб-приложении ELMA возможна работа со справочниками и их импортированными реквизитами с возможностью интерактивного редактирования и добавления элементов справочника. В веб-приложении ELMA справочники 1С отображаются в разделе **Справочники** – **Интеграция с 1С** – **<Папка с именем конфигурации 1С>** (Рис. 13).

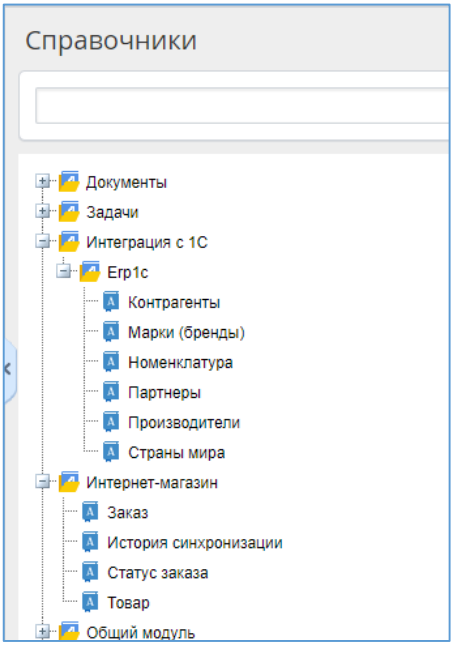


Рис. 13. Справочники 1С, подключенные к ELMA

При открытии карточки элемента справочника будут отображены включенные в настройках интеграции (Рис. 11) реквизиты элемента справочника 1С (Рис. 14).

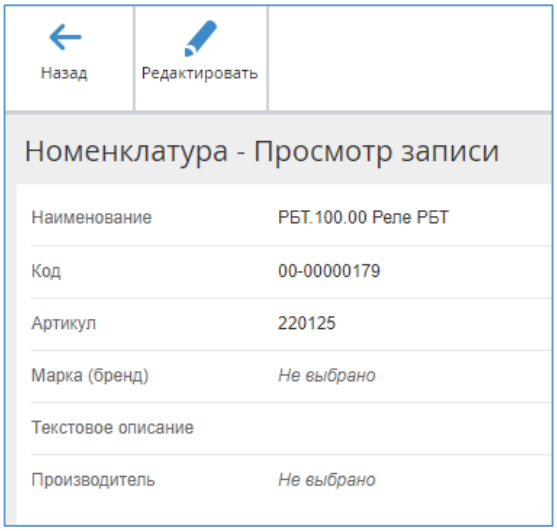


Рис. 14. Форма просмотра элемента справочника 1С в ELMA

По кнопке **Редактировать** откроется окно изменения номенклатуры с возможностью изменения полей, настроенных в интеграции.

Внимание! Крайне не рекомендуется использовать данную возможность.

Дело в том, что при интерактивной работе с формой в клиенте 1С используется множество механизмов заполнения и подстановки значений в недоступные для пользователя реквизиты. Также могут присутствовать обработки при записи объекта. Поэтому любое создание и редактирование элементов справочника 1С рекомендуется производить, только работая в 1С.

2.2. Особенности работы со справочниками и документами 1С при разработке бизнес-процессов

При моделировании процессов подключенные справочники и документы 1С доступны для выбора в качестве типа данных контекстных переменных. Также можно назначать эти типы атрибутам новых и существующих объектов системы. Типы 1С находятся в дереве объектов в разделах **Документ 1С** и **Справочник 1С** (Рис. 15).

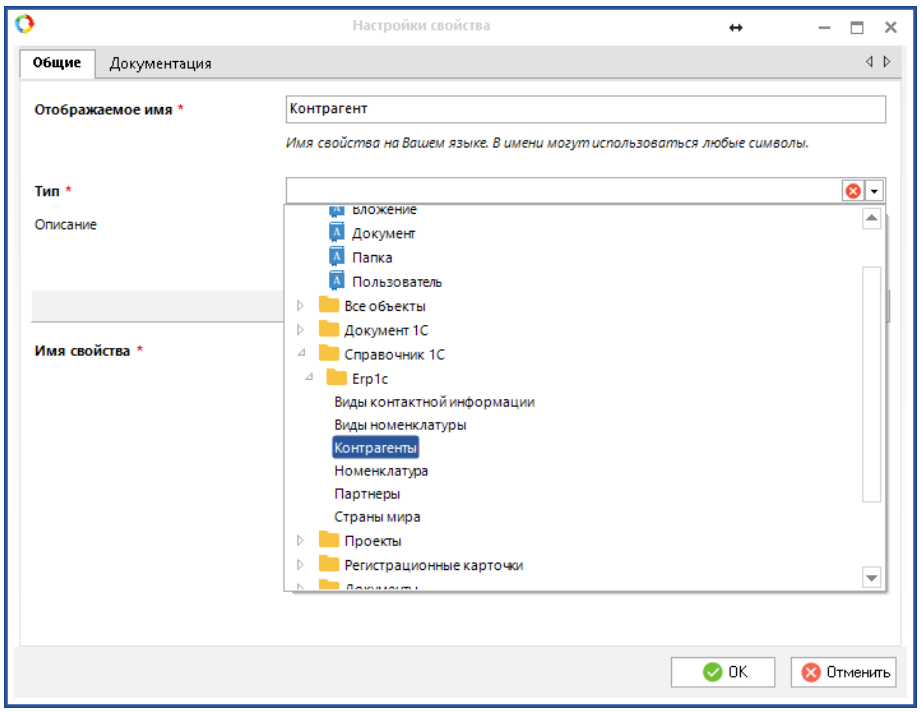


Рис. 15. Выбор типа переменной контекста

В задачах процесса работа со справочниками отличается от работы с документами. На форме доступен только выбор элементов справочника 1С. При этом доступны все способы выбора, которые доступны для стандартных справочников ELMA (Рис. 16): из выпадающего списка (в том числе с использованием быстрого поиска) и из формы списка (при нажатии на кнопку).

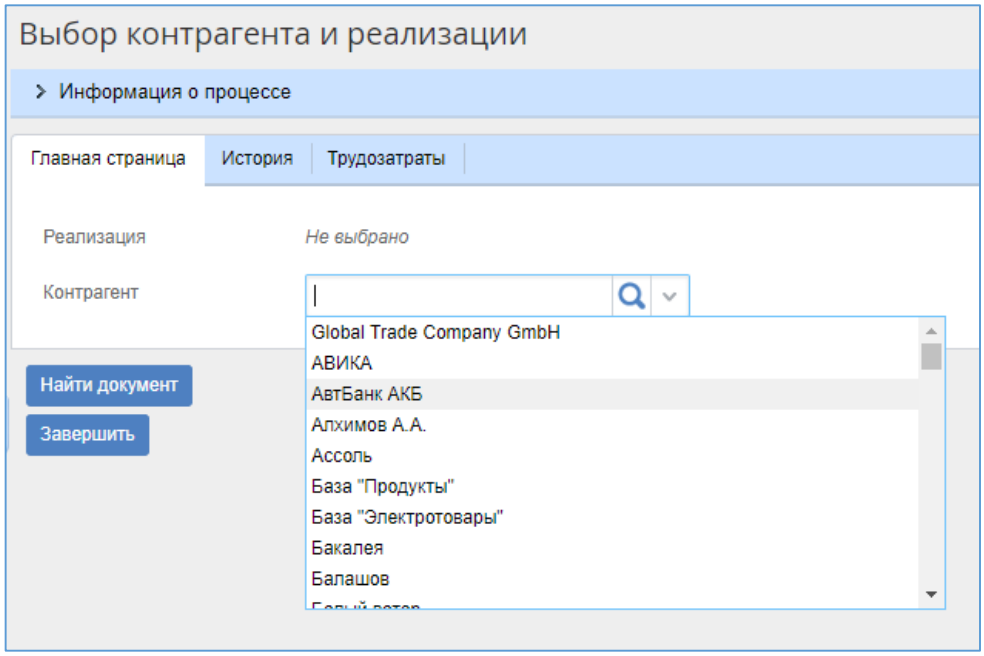


Рис. 16. Выбор элементов справочника 1С в форме задачи процесса

Работа с документами отличается. Для них недоступен непосредственный выбор. Переменную необходимо заполнять при помощи сценария. Заполнить документ можно различными способами:

```
//инициализируем подключение
var service = Locator.GetServiceNotNull<Integration1CService> ();
ComObject connector = service.GetComConnector ("erp1c");
//получаем менеджер объектов 1С
var Manager1C = (dynamic)connector.Reference;
//использование встроеного метода провайдера ELMA
var provider = service.GetProvider("erp1c");// Указываем имя
конфигурации, к которой подключаемся
//загрузка по номеру и дате, аналог метода менеджера документов 1С
"НайтиПоНомеру", только дата передается в виде строки
context.Realisation =
provider.LoadDocumentByNumber<EleWise.ELMA.Integration1C.Configs.Erp1
c.RealizaciyaTovarovUslug>(context.NumberDoc, context.DataDoc.ToStrin
g());
//--
//загрузка через COM-объект
//ищем ссылку на документ через менеджеры 1С любым удобным способом -
через запрос, методами "НайтиПоНомеру" и т.п.
var doc1c =
Manager1C.Документы.РеализацияТоваровУслуг.НайтиПоНомеру(context.Numb
erDoc, context.DataDoc);
//инициализируем COM объект на основе найденной ссылки
var CO = new ComObject(doc1c);
//загружаем элемент в контекстную переменную
```

```
context.Realisation =
provider.LoadDocumentByComObject<EleWise.ELMA.Integration1C.Configs.Erp1c.RealizaciyaTovarovUslug>(CO);
//--
```

context.Realisation – переменная контекста процесса с типом «Реализация товаров и услуг (Документ 1С)».

Помимо этого, у провайдера 1С в ELMA также есть аналогичные методы, позволяющие загрузить справочник по уникальному идентификатору или COM-объекту.

К реквизитам объекта 1С, ссылка на который загружена в контекстную переменную, можно обращаться как к свойствам стандартного объекта ELMA. Доступны будут только те реквизиты, которые были настроены при интеграции. Имена реквизитов представляют из себя транслитерированные имена реквизитов из 1С (Рис. 17).

context.Kontragent – переменная контекста процесса с типом «Контрагенты (Справочник 1С)».

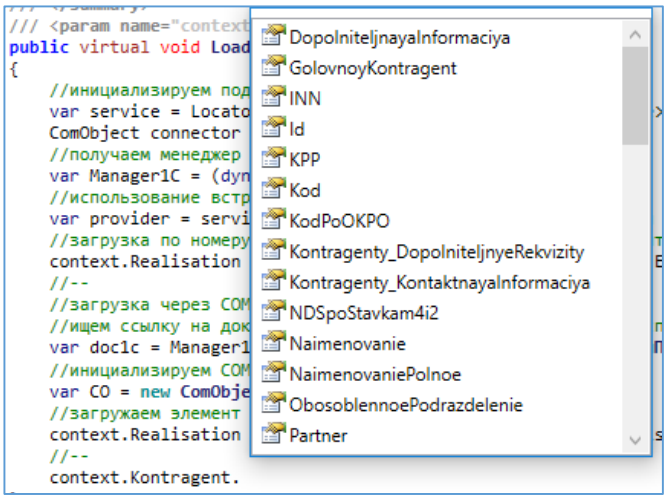


Рис. 17. Работа с реквизитами объекта 1С

Для того, чтобы обратиться к реквизитам и методам самого объекта 1С, используя возможности COM-соединения, необходимо получить на него ссылку. Для этого используется метод **GetComReference()** и его свойство **Ref** для переменной с типом «Документ 1С» или «Справочник 1С». Примеры обращения к реквизиту и к методу ссылки на документ:

```
//получить комментарий
string comment =
context.Realisation.GetComReference().Ref.Комментарий;
//получить уникальный идентификатор документа
```

```
dynamic uidDoc =
context.Realisation.GetComReference().Ref.УникальныйИдентификатор();
```

При программном заполнении объектов 1С из ELMA для присвоения реквизитам ссылочных типов значений контекстных переменных также необходимо использование конструкции **GetComReference().Ref**. Это аналогично использованию реквизита «Ссылка» при программном заполнении объектов при использовании конфигуратора 1С.

При работе с объектами ELMA, в которых есть свойства с типами документа или справочника 1С, присвоение происходит непосредственно. В указанном примере:

- **context.ContractorELMA** – переменная контекста процесса с типом «Контрагент»;
- **context.Kontragent** – переменная контекста с типом «Контрагент (Справочник 1С)».

У объекта типа «Контрагент» (в объектах ELMA) добавлено свойство **Contractor1C** с типом «Контрагент (Справочник 1С)».

```
context.ContractorELMA.Contractor1C = context.Kontragent;
context.ContractorELMA.Save();
```

При разработке также может пригодиться использование автоматической фильтрации справочников 1С на форме задачи. Один из самых распространенных примеров – это выбор договора контрагента. В одном из полей выбирается контрагент, а в другом – список связанных с этим контрагентом договоров.

Добавляем на форму задачи контекстные переменные «Контрагент» и «Договор» с типами «Контрагенты (справочник 1С)» и «Договоры контрагентов (справочник 1С)» соответственно (Рис. 18).

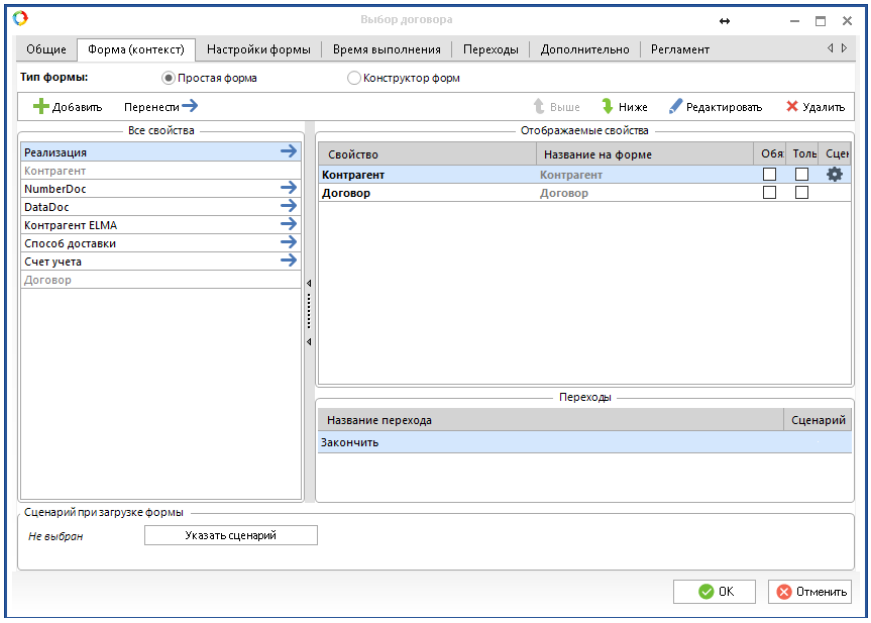


Рис. 18. Настройка формы задачи для использования фильтров

На поле «Контрагент» устанавливаем сценарий (через кнопку **Редактировать** и переходим на вкладку **Дополнительно**). Текст сценария:

```
public virtual void DogovorFilter (Context
context, EleWise.ELMA.Model.Views.FormViewBuilder<Context> form)
{
    //получаем настройки для элемента формы с договором 1С
    var contractSettings = (EntitySettings)context.GetSettingsFor (c
=> c.Dogovor);
    //устанавливаем фильтр для данного элемента с использованием
языка запросов 1С
    contractSettings.FilterQuery = "Ссылка в (ВЫБРАТЬ
ДоговорыКонтрагентов.Ссылка ИЗ Справочник.ДоговорыКонтрагентов КАК
ДоговорыКонтрагентов ГДЕ ДоговорыКонтрагентов.Контрагент.Наименование
= \""+context.Kontragent.Naimenovanie+"\" И
ДоговорыКонтрагентов.Контрагент.ИНН = \""+context.Kontragent.INN+"\"
)";
    contractSettings.Save ();
}
```

У запросов в фильтре есть одна особенность: для них невозможно использовать установку параметров. Поэтому требуется использовать какой-либо реквизит или набор реквизитов объекта, которые могут однозначно определить нужный элемент. В нашем случае – это наименование и ИНН контрагента (реквизит «Код» в конфигурации отсутствует).

Если требуется отфильтровать поле выбора независимо от другого, то подобный код можно выполнить в сценарии до задачи (применить фильтр к переменной контекста процесса).

2.3. Особенности работы с прочими объектами 1С из системы ELMA

Помимо справочников и документов в 1С существуют объекты других типов: перечисления, планы видов характеристик, регистры и т.п.

Если тип реквизита «Перечисление», то, в данном случае все перечисления будут сохранены в объектной модели ELMA. Ознакомиться со списком сохраненных перечислений можно в разделе **Перечисления – Интеграция с 1С** вкладки **Объекты** Дизайнера ELMA (Рис. 19).

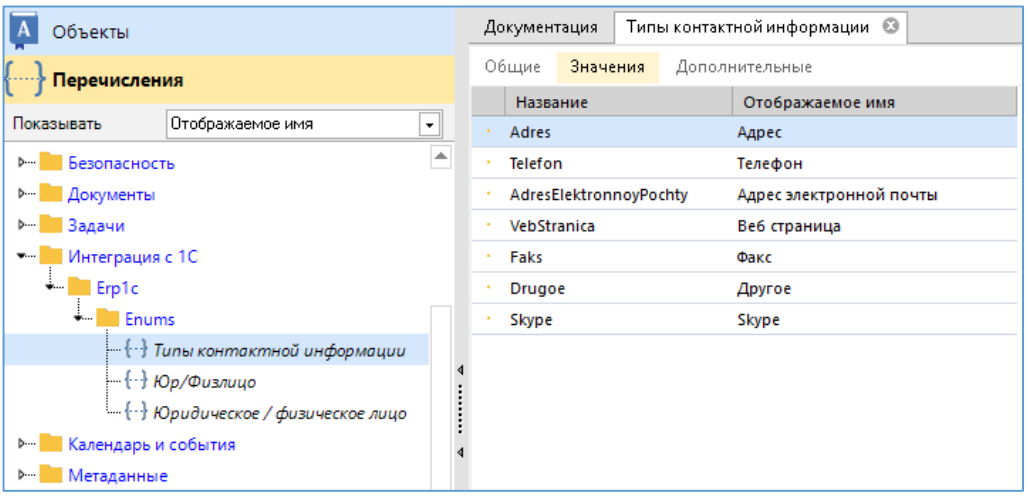


Рис. 19. Дерево перечислений в списке объектов ELMA

При этом, сохраненные в ELMA перечисления никак не связаны с перечислениями 1С, потому что перечисления в 1С являются частью структуры конфигурации и не имеют ссылок.

Распространенными задачами в решениях с интеграцией с 1С могут быть задачи выбора различных объектов 1С при создании или изменении документов, или использовании реквизитов выбранных объектов. Рассмотрим задачу: в задаче процесса в определенном заранее документе «Реализация товаров и услуг» 1С пользователь должен изменить значение реквизита «Способ доставки».

Если разместить на форме задачи процесса элемент с типом «Способ доставки» из дерева объектов **Перечисления – Интеграция с 1С** – во время присвоения его значению реквизита документа 1С и записи документа – получим в 1С пустое значение. Необходимо на форме задачи в ELMA реализовать элемент, в котором однозначно будет определяться перечисление 1С.

Для решения этой задачи воспользуемся элементом типа «Выпадающий список». При добавлении контекстной переменной процесса список значений оставляем пустым (Рис. 20).

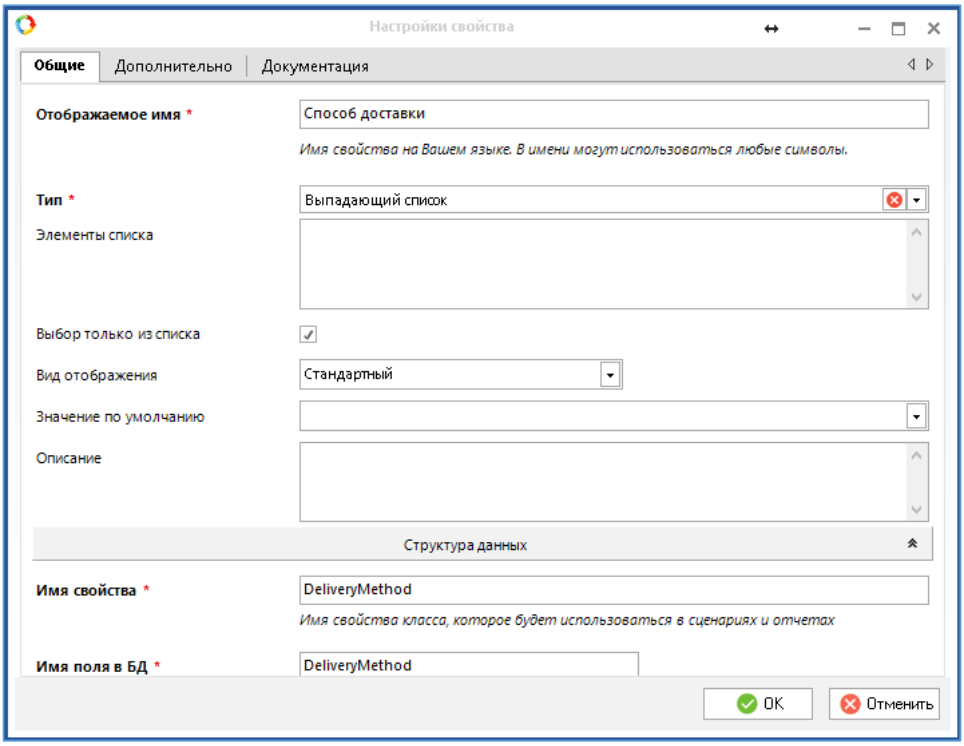


Рис. 20. Создание выпадающего списка для хранения значений перечисления 1С

Решение задачи разобьем на несколько этапов:

1. Определение документа и запись в контекстную переменную.
2. Заполнение выпадающего списка возможными значениями перечислений (для выбранного реквизита объекта).
3. Отображение пользовательской задачи.
4. Запись документа с новым значением перечисления.

Реализуем эти этапы через бизнес-процесс (Рис. 21).

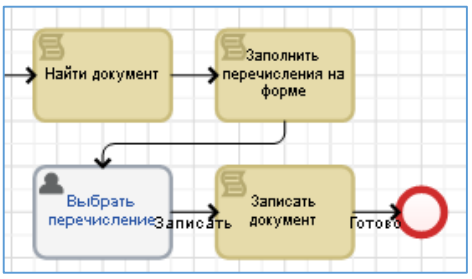


Рис. 21. Решение задачи по изменению пользователем реквизита типа «Перечисление 1С»

Для поиска документа можно использовать любой из описанных выше методов.

```
public virtual void FindRealisation (Context context)
{
    //инициализируем подключение
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
    var provider = service.GetProvider ("erp1c");
    //присваиваем контекстной переменной ссылку на необходимый
документ 1С
    context.Realisation =
provider.LoadDocumentByNumber<EleWise.ELMA.Integration1C.Configs.Erp1
c.RealizaciyaTovarovUslug> ("ПР00-000012", "2017-01-01");
}
```

Для остальных сценариев используется работа с метаданными 1С. Для заполнения контекстной переменной типа «Выпадающий список» используются следующие процедуры:

```
public virtual void FillEnum (Context context)
{
    //инициализируем подключение
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
    //инициализируем доп. переменную – для хранения настроек
выпадающего списка, который будет заполняться
    var dDsettings = (DropDownListSettings)context.GetSettingsFor (c
=> c.DeliveryMethod);
    //в переменную ниже получаем метаданные нужного реквизита
документа (или иного объекта 1С) с типом "перечисление", в нашем
случае это "СпособДоставки"
    var metaData1C =
context.Realisation.GetComReference ().Ref.СпособДоставки.Метаданные
();
    //вызываем процедуру заполнения выпадающего списка
    FillDropOut1SPerechislenie (dDsettings, metaData1C);
    //для удобства пользователя – необходимо на форме задачи процесса
установить текущее значение реквизита документа
    dynamic managerPerechislenie =
connector.GetFunctionValue ("NewObject", "EnumManager." + metaData1C.
Имя);
    //находим значение в коллекции элементов выпадающего списка по
индексу перечисления, сравнивая его с полем key
    context.DeliveryMethod = dDsettings.Items.Find (d => d.Key ==
managerPerechislenie.Индекс (context.Realisation.GetComReference ().R
ef.СпособДоставки).ToString ());
}
//функция заполнения контекстной переменной типа "выпадающий список"
//входные параметры – настройки выпадающего списка и метаданные
реквизита 1С с типом "перечисление"
public void FillDropOut1SPerechislenie (DropDownListSettings
dDsettings, dynamic metaData1C)
{
```

```
//инициализируем подключение
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
    var Manager1C = (dynamic)connector.Reference;
    //получаем менеджер перечислений и обходим его, добавляя новые
элементы в выпадающий список процесса
    dynamic managerPerechislenie
=     connector.GetFunctionValue ("NewObject", "EnumManager." + metaD
ata1C.Имя);
    foreach (var element in managerPerechislenie) {
        //получаем индекс и по нему получаем синоним из коллекции
метаданных
        //в списке хранятся индексы перечислений и их представление
для пользователей
        dDsettings.Items.Add (new DropDownItem (managerPerechislenie.
Индекс (element).ToString ()    , metaData1C.ЗначенияПеречисления.Пол
учить (managerPerechislenie.Индекс (element)).Синоним));
     }
     //сохраняем настройки выпадающего списка
     dDsettings.Save ();
}
```

Пользователю поступит задача с уже определенным документом и предустановленным в элемент выпадающего списка текущим значением реквизита «СпособДоставки» (Рис. 22).

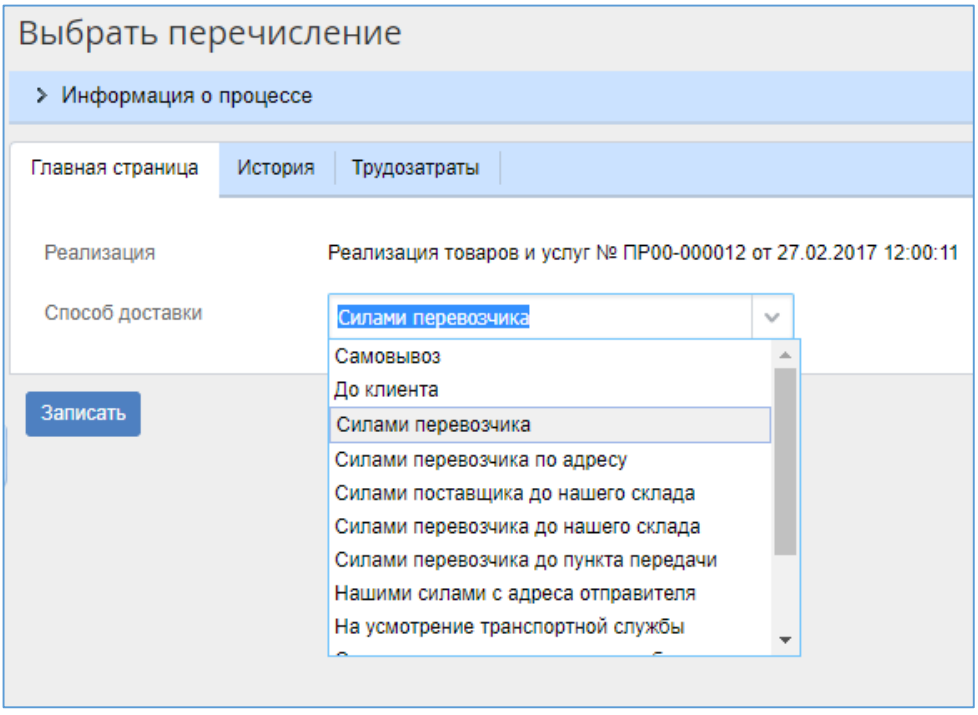


Рис. 22. Форма задачи с выбором из значений перечисления «СпособыДоставки»

Выберем значение «Самовывоз» и нажмем на кнопку **Записать**. Запись документа осуществляется с предварительным определением перечисления из объектов 1С, исходя из выбранного значения выпадающего списка:

```
public virtual void SaveRealisation (Context context)
{
     //для сохранения – получаем объект 1С
     dynamic objDoc =
context.Realisation.GetComReference().Ref.ПолучитьОбъект();
     //вызываем функцию, в которой получаем по индексу выбранного
перечисления ссылку на объект в базе 1С,
     //передаем метаданные нужного реквизита и ключ выбранного в
выпадающем списке элемента
     objDoc.СпособДоставки
= Get1SEnumeration(context.Realisation.GetComReference().Ref.СпособДо
ставки.Метаданные(), context.DeliveryMethod.Key);
     //вызываем метод 1С для записи документа
     objDoc.Записать();
}
//функция получения перечисления 1С по индексу
public dynamic Get1SEnumeration(dynamic Metadata1C, string keyDD)
{
     var service =
Locator.GetServiceNotNull<Integration1CService> ();
     ComObject connector = service.GetComConnector ("erp1c");
     dynamic managerPerechislenie
=    connector.GetFunctionValue("NewObject", "EnumManager."+Metadata1
C.Имя);
     //в поле key элемента выпадающего списка может быть только
строка, поэтому приводим ее в числовой формат
     int ind = -1;
     ind = Convert.ToInt32(keyDD);
     //получаем по индексу (скрытое для пользователя поле "key" в
выпадающем списке) значение перечисления
     dynamic enum1S =    managerPerechislenie.Получить(ind);
     return enum1S;
 }
```

Результат работы можно проверить в 1С, открыв карточку документа и проверив значение реквизита «Способ доставки» (Рис. 23).

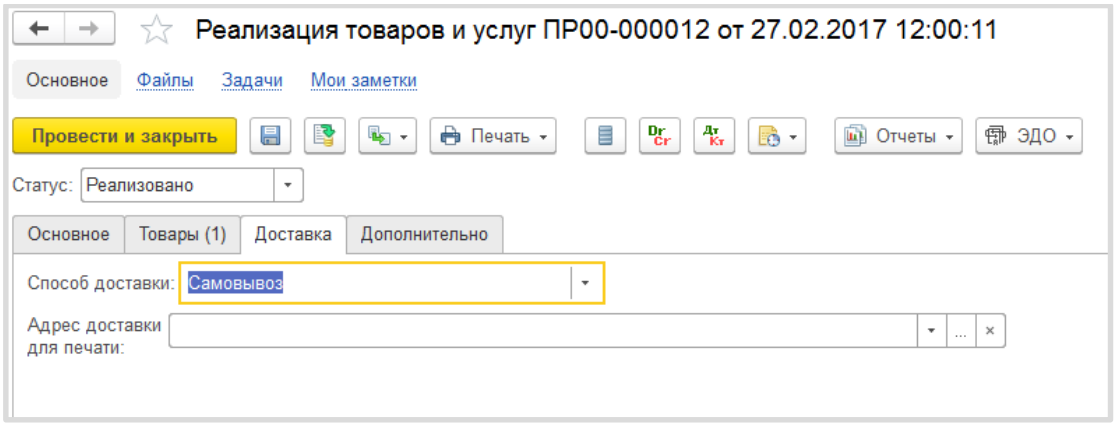


Рис. 23. Проверка результатов работы процесса

В данных сценариях используется работа с метаданными 1С, т.к. из менеджера перечислений, при вызове его из ELMA невозможно получить элементы по строковому представлению (например, Перечисление.СпособыДоставки["Самовывоз"]). Из-за подобных ограничений необходимо использовать более сложные конструкции программного кода. Процедура **FillDropOut1SPerechislenie** является универсальной, в нее можно передать метаданные любого реквизита с типом «Перечисление» из любого объекта 1С.

Для прочих объектов конфигурации 1С и в целом для всех объектов с ссылочными типами также можно использовать универсальное решение по выбору объектов из выпадающего списка. Рассмотрим пример: разместим на форме задачи выпадающий список, в котором будут счета из плана счетов 1С с видом субконто «Банковский счет». По выбранному значению необходимо получить ссылку на сущность в 1С.

Для решения также используется тип переменной контекста «Выпадающий список». В качестве ключа для поля задается строка с уникальным идентификатором объекта в 1С, а в качестве значения – его строковое представление в 1С. Процедуры получения таблицы с данными и заполнения выпадающего списка:

```
public virtual void FillAccount (Context context)
{
    var dDsettings = (DropDownListSettings)context.GetSettingsFor (c
=> c.Account);
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
    var Manager1C = (dynamic)connector.Reference;
    //необходимые, отфильтрованные данные получим через запрос.
Таблицу значений можно сформировать также любым другим способом
    string ТекстЗапроса = "ВЫБРАТЬ РАЗЛИЧНЫЕ
" + "ХозрасчетныйВидыСубконто.Ссылка КАК Ссылка " + "ИЗ
```

```
ПланСчетов.Хозрасчетный.ВидыСубконто КАК ХозрасчетныйВидыСубконто
" + "ГДЕ " + "ХозрасчетныйВидыСубконто.ВидСубконто = &ВидСубконто";
    //Создание нового объекта типа Запрос
    dynamic Запрос =
connector.GetFunctionValue ("NewObject", "Запрос");
    //Заносим текст запроса в соответствующий атрибут Запроса.
    Запрос.Текст = ТекстЗапроса;
    //установим параметр запроса, для фильтрации списка счетов
    Запрос.УстановитьПараметр ("ВидСубконто", Manager1C.ПланыВидовХар
актеристик.ВидыСубконтоХозрасчетные.НайтиПоНаименованию ("Банковские
счета"));
    //выполнение запроса и выгрузка результата в таблицу значений 1С.
    dynamic table1s = Запрос.Выполнить ().Выгрузить ();
    FillDropOut1SLink (dDsettings, table1s);
}

//функция заполнения контекстной переменной типа "выпадающий список"
//входные параметры – настройки выпадающего списка, таблица значений
1С с ссылками,
//поле таблицы с ссылкой обязательно должно называться "Ссылка",
иначе необходимо изменить имя атрибута в коде процедуры
public void FillDropOut1SLink (DropDownListSettings
dDsettings, dynamic table1s)
{
    //инициализируем подключение
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
    var Manager1C = (dynamic)connector.Reference;
    //обходим таблицу с ссылками
    for (var i = 0; i < table1s.Count; i++) {
    var element = table1s.Получить (i);
    //в списке хранятся индексы перечислений и их представление для
пользователей
    //представление выводит метод Manager1C.String, он равнозначен
методу "Строка" глобального контекста 1С
    dDsettings.Items.Add (new DropDownItem (Manager1C.String (element
.Ссылка.УникальныйИдентификатор ()), Manager1C.String (element.Ссылка
)));
    }
   //сохраняем настройки выпадающего списка
    dDsettings.Save ();
}
```

При выборе элемента списка на форме задачи строка с уникальным идентификатором будет в атрибуте **key** контекстной переменной (в описанном примере – **context.Account.Key**). Для того чтобы получить ссылку на сущность 1С, необходимо использовать метод «ПолучитьСсылку» у менеджера нужного объекта:

```
public dynamic Get1SLink (string guid1c)
{
    //инициализируем подключение
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
```

```
    var Manager1C = (dynamic)connector.Reference;
    //получаем объект типа "УникальныйИдентификатор" из переданной в
функцию строки
    dynamic guidObj =
Manager1C.NewObject ("УникальныйИдентификатор", guid1c);
    //получаем менеджер нужных нам объектов
   dynamic managerObj = Manager1C.ПланыСчетов.Хозрасчетный;
    dynamic link1c = managerObj.ПолучитьСсылку (guidObj);
    return link1c;
}
```

Стоит отметить, что описанное выше решение пригодно для всех ссылочных типов 1С: справочники, документы, планы видов характеристик, планы счетов, планы видов расчета, планы обмена, бизнес-процессы, задачи. Из-за ограничений COM-соединения невозможно обращаться к конструкциям 1С через имена объектов в квадратных скобках ([...]), т.е. **Manager1C.ПланыСчетов.Хозрасчетный** нельзя заменить на **Manager1C.ПланыСчетов["Хозрасчетный"]**. Поэтому для каждого типа необходимо написать свою процедуру с получением нужного менеджера объектов.

При работе с уникальными идентификаторами можно дополнить объекты ELMA соответствием с любыми ссылочными типами 1С, а в качестве связи с использовать атрибут типа «Строка», в котором будет хранится UID элемента 1С. Затем через объявление менеджера нужных объектов в сценариях и метода **ПолучитьСсылку** получаем сопоставленный сущности ELMA объект 1С.

2.4. Особенности сеанса COM-соединения

Настроенное COM-соединение ELMA с базой 1С постоянно активно. Поэтому при работе ELMA нет постоянных переподключений к конфигурации 1С, что существенно ускоряет вызов функций и методов. Однако нужно быть внимательным, если производится динамическое обновление конфигурации. Перезапуск 1С у пользователя в данном случае не обязателен, поэтому ELMA без принудительного переподключения будет работать со старой конфигурацией. Порядок действий для переподключения следующий:

1. В Дизайнере ELMA в разделе **Меню -> Интеграция с 1С** выбираем необходимую конфигурацию 1С и нажимаем на кнопку **Редактировать настройки** (Рис. 10).
2. Устанавливаем в поле **Активность** переключатель в положение **Отключена** и нажимаем на кнопку **OK**.
3. В списке активных пользователей в 1С проверяем, что пользователя, который был указан в настройках подключения ELMA и 1С с типом приложения «COM-соединение», нет в списке.
4. Повторяем шаги 1 и 2, только в поле **Активность** устанавливаем переключатель в положение **Включена**. Если подключение не пройдет с первого раза, снова открываем окно настроек, в поле **Активность** устанавливаем переключатель в положение **Отключена** и нажимаем на кнопку **OK**. После этого снова устанавливаем переключатель в положение **Включена**.

Из описанного выше следует, что для всех вызовов 1С всегда используется только одно подключение. Сценарии процессов с вызовами 1С выполняются в порядке очереди исполнения аналогично обычным сценариям ELMA. Также, если в сценариях используются функции, возвращающие объекты 1С, то в вызывающих процедурах при работе с данными объектами возможно использование всех свойств и методов объекта 1С без каких-либо дополнительных инициализаций COM-соединения.

Еще один важный момент – COM-соединение может работать только с методами платформы 1С с доступностью «Внешнее соединение». Проверить доступность у определенного метода можно во встроенном синтаксис-помощнике конфигуратора 1С (Рис. 24).

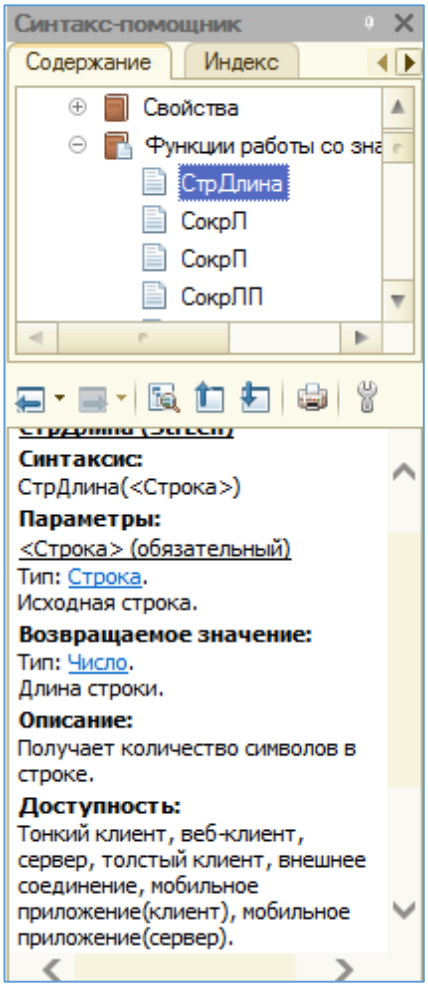


Рис. 24. Описание методов конфигурации в синтаксис-помощнике

Если методы без доступности «Внешнее соединение» вызываются в каких-либо обработчиках событий «ПередЗаписью», «ПриЗаписи» или подписках в 1С, то при инициализации этих событий из ELMA они не выполнятся. Важным ограничением, которое необходимо для безопасности, является невозможность непосредственного вызова методов глобального контекста 1С: «Выполнить» и «Вычислить». В этих функциях возможно выполнение произвольного кода 1С.

При помощи менеджера глобального контекста из ELMA можно вызывать только те методы общих модулей, которые описаны с ключевым словом «Экспорт». Также необходимо наличие определенных флажков, отвечающих за возможности вызова модуля. Важно, чтобы был установлен флажок **Внешнее соединение** или **Вызов сервера** (Рис. 25).

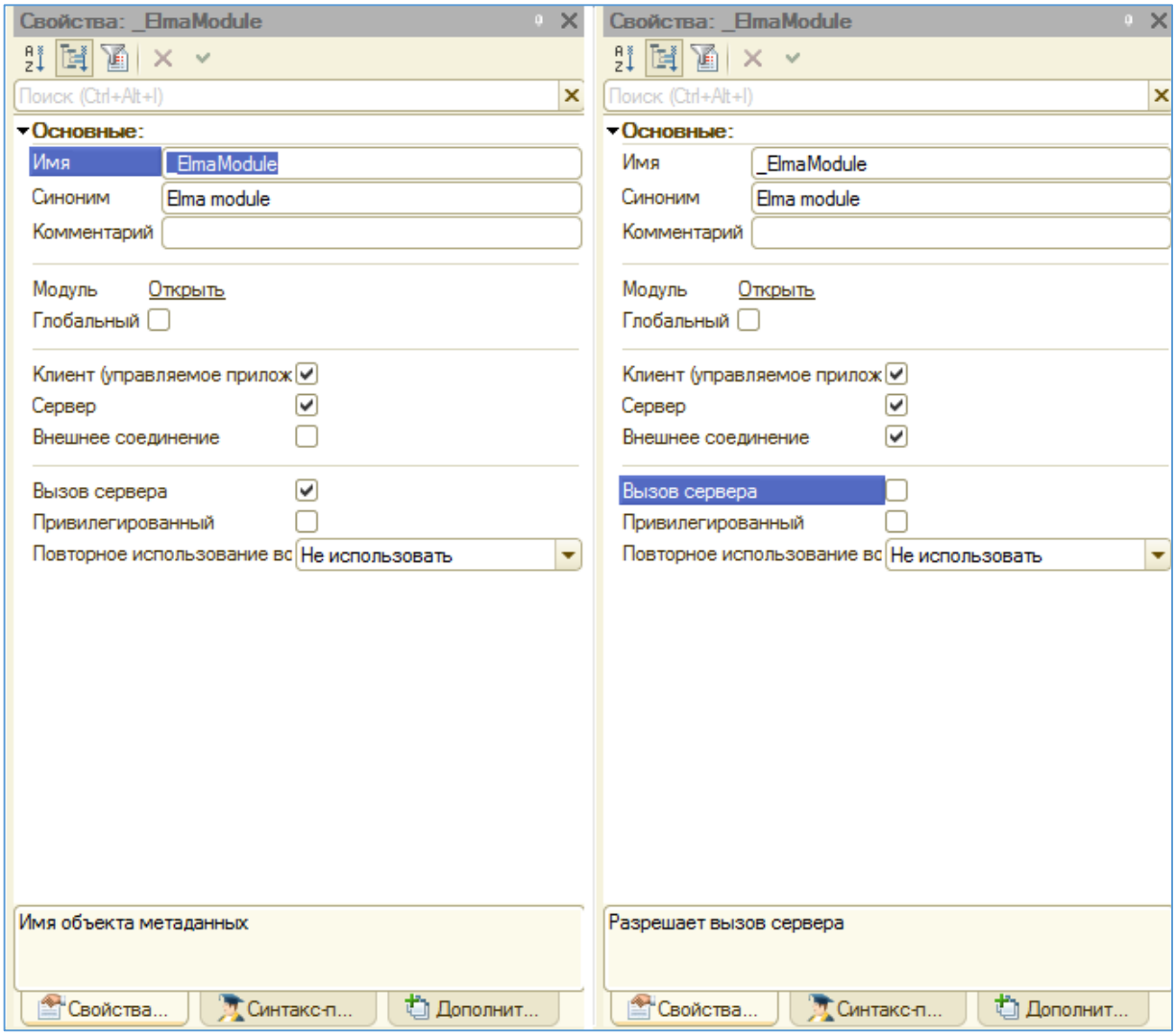


Рис. 25. Возможные настройки общего модуля в 1С для вызова из COM-соединения

2.5. Отладка и контроль ошибок работы с 1С из системы ELMA

Иногда во время разработки сложных решений, связанных с интеграцией, требуется пошагово проверить выполнение вызываемых методов 1С. В платформе 1С существует режим отладки с возможностью пошагового исполнения кода. Для использования отладки для работы с внешним соединением необходимы предварительные настройки.

В папке **..\bin\conf** с установленным клиентом 1С (платформой) необходимо создать файл **comcntrcfg.xml**. Он должен иметь следующее содержание:

```
<config xmlns="http://v8.1c.ru/v8/comcntrcfg">
    <debugconfig debug="true"
   debuggerURL="tcp://localhost:1560" />
</config>
```

Данный файл включает возможность запуска COM-соединения с локальной машины с использованием отладки.

После этого переподключаем ELMA к 1С. Далее в конфигураторе 1С в верхнем меню открываем раздел **Отладка – Подключение...** (Рис. 26).

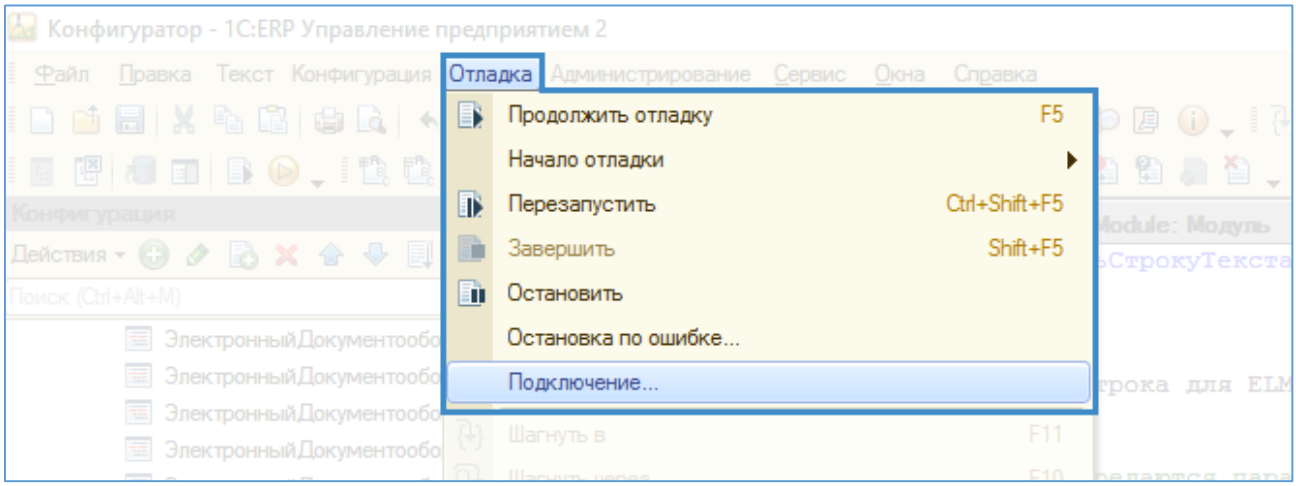


Рис. 26. Подключение отладки в 1С.

В окне доступных предметов отладки должны появиться сеансы с типом **COM-соединение**, их может быть несколько для одного подключения ELMA. Подключаем каждый сеанс (Рис. 27).

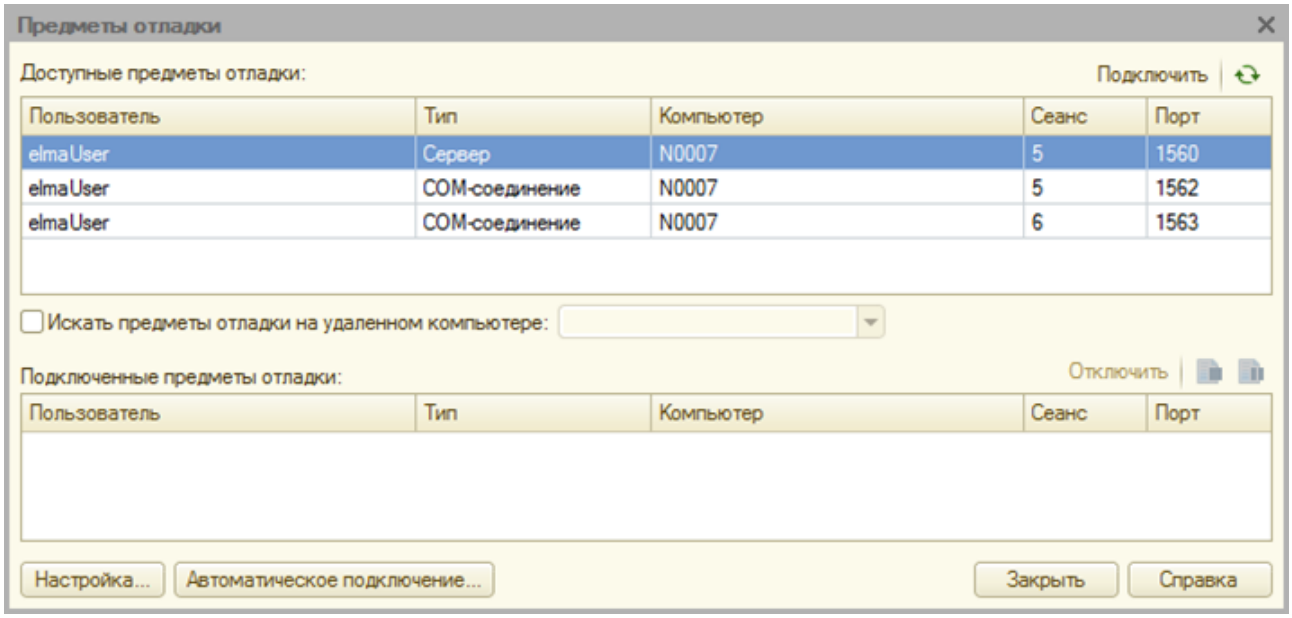


Рис. 27. Сеансы COM-соединений, доступные для отладки

Для проверки отладки устанавливаем точку останова в вызываемом из ELMA методе 1С и выполняем сценарий в ELMA. В веб-приложении ELMA или в эмуляции сценариев для сценария будет отображаться процесс выполнения, а в конфигураторе 1С выполнение будет приостановлено в нужной точке останова (Рис. 28).

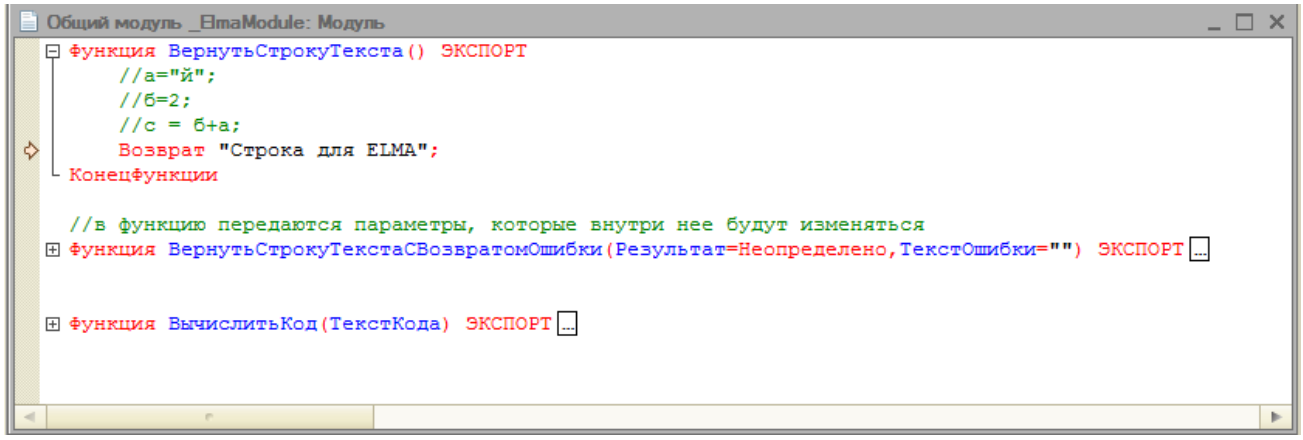


Рис. 28. Отладка вызываемого из ELMA метода в 1С

Если по каким-либо причинам отладка в точке останова не останавливается, выполните следующие действия:

1. Закройте конфигуратор 1С.
2. Переподключите ELMA к 1С.
3. Откройте конфигуратор 1С и заново подключите отладку.

Если по каким-либо причинам вам недоступен конфигуратор 1С для базы данных 1С, на которой ведется разработка, можно повторить в 1С COM-соединение. Это позволит выяснить, нет ли в вызываемом коде методов, использование которых недоступно для внешнего соединения, или исправить неправильный синтаксис вызовов функций в ELMA.

Рассмотрим пример кода, вызывающего из 1С функцию, возвращающую строку. Код ELMA:

```
//инициализируем подключение
var service = Locator.GetServiceNotNull<Integration1CService> ();
ComObject connector = service.GetComConnector ("erp1c");
//получаем менеджер объектов 1С
var Manager1C = (dynamic)connector.Reference;
//вызываем функцию из общего модуля
string testString = Manager1C._ElmaModule.ВернутьСтрокуТекста();
Console.WriteLine (testString);
```

Намеренно включим в процедуру в 1С ошибку. При выполнении кода в ELMA получим ошибку:

[System.NullReferenceException: Ссылка на объект не указывает на экземпляр объекта.]

Как видно, понять причину ошибки невозможно. Для получения развернутого описания ошибки нужно получить ее в 1С. Для этого напишем внешнюю обработку, в которой инициализируем COM-подключение 1С. Для проверки можно создать новую пустую базу 1С, в которой у вас будут полные права, а затем инициализировать подключение к базе данных 1С, на которой ведется разработка (используя данные учетной записи ELMA). В обработке необходимо вызвать нужный метод через менеджер соединения:

```
//создаем объект
COMConnector= Новый COMОбъект("V83.COMConnector");
//задаем параметры подключения к базе
СтрокаПодключения =
"Srvr=""localhost"";Ref=""ERPERP"";Usr=""elmaUser"";Pwd=""123"";";
//выполняем подключение, в данном случае подключение поддерживается
только в рамках локального контекста процедуры
Соединение = COMConnector.Connect(СтрокаПодключения);
//в переменной "Соединиение" хранится менеджер соединения, он
аналогичен переменной "Manager1C" в коде ELMA
ТестСтрока = Соединение._ElmaModule.ВернутьСтрокуТекста();
Сообщить(ТестСтрока);
```

После выполнения данного кода в 1С будет вызвано исключение, в котором будет описана природа ошибки (Рис. 29).

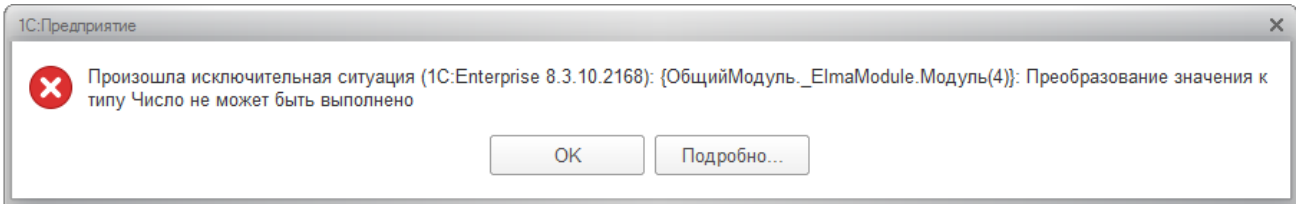


Рис. 29. Ошибка в вызываемом методе общего модуля 1С

В данном случае ошибка не связана с использованием методов, для которых недоступен вызов через внешнее соединение. Однако методика проверки, описанная выше, универсальная и рекомендуется для использования во всех случаях разработки решений на базе ELMA с интеграцией с 1С. Еще один нюанс – при отладке вызова через COM-соединение отладчик не будет переходить в вызываемый из подключаемой конфигурации метод.

Возможны случаи, когда на стороне ELMA требуется отслеживать ошибки 1С. Например, ошибки проведения документов. При этом требуется выполнение различных действий – отправить уведомление, выполнить отличный от обычного переход в процессе и т.п. Для этих случаев целесообразно написать специальную «обертку», в которой мы будем получать описание ошибки из 1С. Код функции в общем модуле 1С:

```
//в функцию передаются параметры, которые внутри нее будут изменяться
Функция
ВернутьСтрокуТекстаСВозвратомОшибки(Результат=Неопределено,ТекстОшибк
и="") ЭКСПОРТ
//требуемый метод конфигурации 1С вызывается через "Попытку"
Попытка
  //если выполнение удачно – в качестве значения функции возвращается
ИСТИНА,
  //параметр "Результат" принимает значение из вызываемой функции
  Результат = ВернутьСтрокуТекста();
  Возврат Истина;
Исключение
  //при ошибке – возвращается ЛОЖЬ, параметр "Результат" не меняется,
  //в качестве параметра "ТекстОшибки" – возвращается описание ошибки
1С
  ТекстОшибки = ОписаниеОшибки();
  Возврат Ложь;
КонецПопытки;
КонецФункции
```

Для обработки в ELMA можно использовать следующую конструкцию:

```
//инициализируем подключение
var service = Locator.GetServiceNotNull<Integration1CService> ();
ComObject connector = service.GetComConnector ("erp1c");
//получаем менеджер объектов 1С
```

```
var Manager1C = (dynamic)connector.Reference;
//инициализируем переменные для передачи в функцию 1С
dynamic result1c = null;
string errInfo = "";
//вызываем функцию из общего модуля
bool status1c =
Manager1C._ElmaModule.ВернутьСтрокуТекстаСВозвратомОшибки(ref result1
c, ref errInfo);
if (status1c)
{
    //ошибки не было
    Console.WriteLine(result1c);
}
Else
{
   //возникла ошибка
    Console.WriteLine("Текст ошибки: "+errInfo);
}
```

Начальное значение переменной **result1c** может быть любого типа, главное объявить ее «dynamic». Преобразование типа пройдет внутри функции 1С и в нее запишется вычисленное в 1С значение. Это могут быть как примитивные типы, так и объекты 1С (документы, таблицы значений и т.п.). При вызове функции обязательно использование ключевого слова **ref** с передаваемыми переменными (для того, чтобы произошло их изменение в функции).

В результате при возникновении ошибки в методе 1С «ВернутьСтрокуТекста()» в ELMA в значении переменной **errInfo** мы получим текст ошибки 1С (Рис. 30).

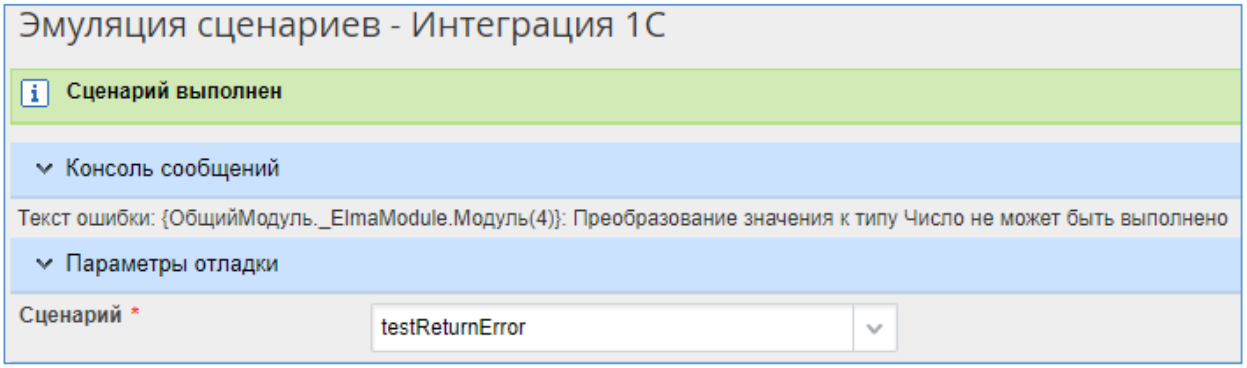


Рис. 30. Эмуляция сценария с обработкой ошибки 1С

Возможны различные варианты подобных «оберток». 1С не контролирует типы переменных и, теоретически, можно сделать функцию, которая в случае ошибки будет возвращать ее текст, а в случае верного выполнения – значение вызываемого метода.

Глава 3. Настройка синхронизации номенклатуры 1С с номенклатурой в системе ELMA

В большинстве торговых организаций товарный учет уже ведется в базе данных 1С. Для хранения используется справочник 1С: «Номенклатура». В справочнике хранится основная информация о товаре. Номенклатура является одним из важнейших разрезов учета компании. Справочник используется для хранения информации о всех товарно-материальных ценностях (ТМЦ) и услугах компании. Для ТМЦ можно выделить основные этапы жизненного цикла в компании:

1. Поступление и оприходование.
2. Хранение.
3. Перемещение (при необходимости).
4. Отгрузка.

Информация о товаре (его визуальные данные и иные характеристики) обычно фиксируются в процессе оприходования товаров сотрудниками склада или иными специалистами по ТМЦ. Данные сохраняются в карточке номенклатуры. Логично, что для осуществления продажи товаров через интернет-магазин эти данные нужно использовать. Также из 1С мы можем получить актуальное количество товара на складе и его цену. Это позволяет избежать двойной работы по внесению информации на сайт и внесению информации в 1С.

Перед началом настройки процесса экспорта данных о товарах на сайт необходимо определиться с характеристиками товаров, которые мы хотим отображать на сайте. Если существующих (внесенных в 1С) данных недостаточно, необходимо добавить и заполнить новые атрибуты в справочник номенклатуры 1С. Это можно сделать как через изменение конфигурации 1С, так и используя дополнительные механизмы: регистры сведений (связать их со справочником «Номенклатура») или оснастку «Дополнительные реквизиты», доступную в новых продуктах 1С (например, «1С: Управление торговлей 11», «1С: ERP Управление предприятием 2»).

Также возможны случаи получения информации о номенклатуре напрямую из прайсов поставщика. В нашем случае за импорт товаров от поставщиков отвечает ELMA. Реализация механизмов импорта описана в **Главе 3.3**.

Таким образом процесс синхронизации товаров между информационными системами разделен на 2 этапа:

1. Записать в 1С информацию о новых товарах, полученных из прайсов поставщиков.
2. Получить информацию о товарах из 1С.

Т.к. новые товары создаются в двух информационных системах, необходимо определить ключевое поле или набор ключевых полей для определения уникального товара. В описанном решении таким полем будет являться атрибут «Артикул». Также это может быть связка атрибутов «Артикул» + «Поставщик», уникальный штрихкод товара и т.п.

3.1. Процесс синхронизации товаров

Синхронизация может осуществляться автоматически и полуавтоматически через периодический запуск процесса синхронизации. В процессе будет осуществляться взаимодействие информационных систем ELMA и 1С.

В рассматриваемом примере в системе ELMA предусмотрено хранение истории синхронизаций. Это необходимо для того, чтобы избежать обработки полного справочника товаров при каждом запуске синхронизации.

Схема процесса представлена на Рис. 31.

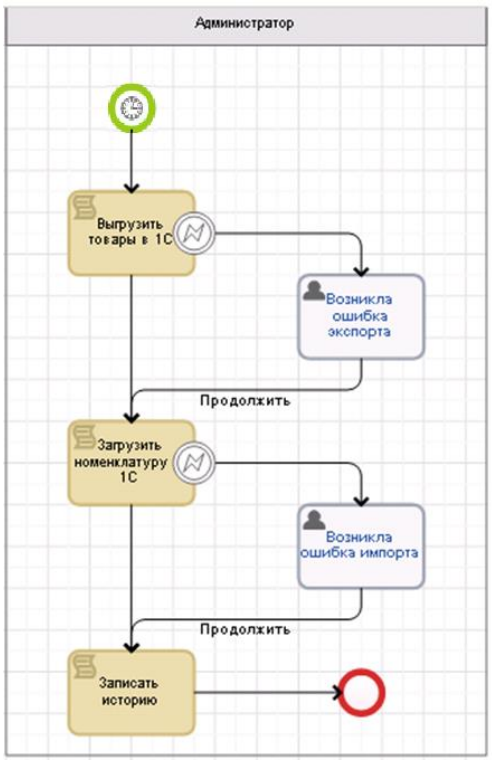


Рис. 31. Карта процесса синхронизации товаров с 1С

В данном процессе предусмотрено полуавтоматическое выполнение. Вручную осуществляется контроль возникших ошибок в сценариях экспорта и импорта. Также при больших объемах данных необходимо разделение операций на итерации. Для полностью автоматического выполнения данного процесса можно предусмотреть журналирование ошибок и исключить операции типа «Пользовательская задача».

Также в рассмотренном примере не происходит периодической синхронизации атрибутов товаров в ELMA и 1С. Характеристики товара могут меняться как в 1С, так и в ELMA. Поэтому необходимо четко определить, какая система будет основной, и

настроить одностороннюю синхронизацию аналогично тому, как присвоение атрибутов сделано в сценарии «Загрузить номенклатуру из 1С».

Для корректной работы сценариев процесса необходимо подключить пространства имен:

```
using EleWise.ELMA.Services;
using EleWise.ELMA.ConfigurationModel;
using EleWise.ELMA.Integration1C;
using EleWise.ELMA.Integration1C.Data;
using EleWise.ELMA.Integration1C.V82;
```

3.2. Выгрузка товаров в 1С

За выгрузку товаров в процессе «Синхронизация номенклатуры» отвечает сценарий «Выгрузить товары в 1С»:

```
public virtual void ExportTo1S (Context context)
{
//1. Получаем товары, измененные после последнего успешного обмена и
не имеющие привязки с номенклатурой 1С
var lastSyncDate = GetLastSyncDate ();
var listGoods = EntityManager<Goods>.Instance.Find (p => p.UpdateDate
>= lastSyncDate && p.Nomenclature1S == null).ToList().GetRange(0,30);
//2. Инициализируем необходимые для работы с 1С переменные
var service = Locator.GetServiceNotNull<Integration1CService> ();
ComObject connector = service.GetComConnector ("erp1c");
var Manager1C = (dynamic)connector.Reference;
var provider = service.GetProvider ("erp1c");
//3. Выполняем операции для каждого из элементов списка товаров
foreach (var element in listGoods) {
  //Ищем по артикулу товара ссылку на элемент справочника
номенклатуры в 1С
  var nomenclature1S = Manager1C._ElmaModule.НайтиПоАртикулу
(element.Code);
  //если 1С может возвращать "null" или "Неопределено", необходимо
проверить тип на System.DBNull
  if (nomenclature1S is System.DBNull) {
        //если не нашли соответствия – создаем новую запись
справочника "Номенклатура" в 1С
        var refNomenclature1S =
Manager1C._ElmaModule.СоздатьНоменклатуруИзELMA (element.Name,
element.Code, element.Unit,
element.Contractor.Contractor1C.GetComReference ().Ref,
element.Dimensions, element.Color, (element.Weight.HasValue) ?
element.Weight.Value:0, element.Description);
        //обработка ошибки – если нам вернулась строка, то добавляем в
список исключений
        if (refNomenclature1S is string)
        {
              context.NotExported.Add(element);
        }
        else
        {
              //присваиваем атрибуту объекта "Товар" ссылку на
номенклатуру
              var com1s = new ComObject (refNomenclature1S);
              element.Nomenclature1S =
provider.LoadBookByComObject<EleWise.ELMA.Integration1C.Configs.Erp1c
.Nomenklatura> (com1s);
              element.UpdateDate = DateTime.Now;
              element.Save ();
              context.Exported.Add(element);
```

```
        }
  }
  else {
        //просто присваиваем атрибуту объекта "Товар" ссылку на
номенклатуру
        var com1s = new ComObject (nomenclature1S);
        element.Nomenclature1S =
provider.LoadBookByComObject<EleWise.ELMA.Integration1C.Configs.Erp1c
.Nomenklatura> (com1s);
        element.UpdateDate = DateTime.Now;
        element.Save ();
        context.Exported.Add(element);
  }
}
if (!context.Exported.IsEmpty)
{
  SetPriceIn1S(context);
}
}
```

Из сценария вызывается процедура получения даты последней успешной синхронизации:

```
public DateTime GetLastSyncDate ()
{
var last = EntityManager<SyncHistory>.Instance.Find (p => p.Status ==
true).OrderByDescending (p => p.SyncDate).FirstOrDefault ();
if (last != null) {
  return last.SyncDate;
}
else {
  return DateTime.MinValue;
}
}
```

После синхронизации вызывается процедура **SetPriceIn1S()**, предназначенная для формирования документа установки цен номенклатуры в 1С по выгруженным товарам:

```
public void SetPriceIn1S(Context context)
{
//из списка товаров формируем таблицу значений и отправляем ее в 1С
//1. Инициализируем необходимые для работы с 1С переменные
var service = Locator.GetServiceNotNull<Integration1CService> ();
ComObject connector = service.GetComConnector ("erp1c");
var Manager1C = (dynamic)connector.Reference;
//2. Создаем таблицу значений
dynamic table1s = connector.GetFunctionValue ("NewObject",
"ТаблицаЗначений");
//типизация колонок при создании не обязательна – типы будут
назначены после добавления первой строки
table1s.Колонки.Добавить("Ссылка");
```

```
table1s.Колонки.Добавить("Цена");
//3. Обходим список товаров и заполняем таблицу значений
foreach(var element in context.Exported)
{
  var nStr = table1s.Добавить();
  nStr.Ссылка = element.Nomenclature1S.GetComReference().Ref;
  nStr.Цена = (element.Price.HasValue)?element.Price.Value:0;
}
//4. Вызываем точку интеграции по созданию документа установки цен
Manager1C._ElmaModule.СоздатьДокументУстановкиЦен(table1s);
}
```

В процедуре экспорта используются вызовы методов, заранее разработанных в конфигурации 1С. Такая методика разработки позволяет сделать код ELMA более читаемым. Кроме этого, реализация методов в конфигурации 1С более удобна (работают механизмы автоподстановки, контроля кода, доступно больше возможностей платформы). Также для такой разработки можно привлечь специалистов-разработчиков 1С, у которых есть опыт работы с конфигурацией, интегрируемой с ELMA.

Метод 1С **СоздатьНоменклатуруИзELMA**:

```
Функция СоздатьНоменклатуруИзELMA(Наименование, Артикул,
ЕдиницаИзмерения, Поставщик, Размер, Цвет, Вес, Описание) ЭКСПОРТ
Попытка
  НоваяНоменклатура = Справочники.Номенклатура.СоздатьЭлемент();
  НоваяНоменклатура.Наименование = Наименование;
  НоваяНоменклатура.Артикул = Артикул;
  НоваяНоменклатура._ElmaРазмер = Размер;
  НоваяНоменклатура._ElmaЦвет = Цвет;
  НоваяНоменклатура._ElmaПоставщик = Поставщик;
  НоваяНоменклатура.ВесЧислитель = Вес;
  НоваяНоменклатура.ВесЗнаменатель = 1;
  НоваяНоменклатура.Описание = Описание;
  //сохраняем номенклатуру в заранее определенную папку -
  НоваяНоменклатура.Родитель =
Справочники.Номенклатура.НайтиПоКоду("000000043");
  НоваяНоменклатура.Записать();
  //создадим единицу измерения
  НоваяНоменклатура.ЕдиницаИзмерения =
СоздатьЕдиницуИзмеренияПоСтроке(ЕдиницаИзмерения,НоваяНоменклатура.Сс
ылка);
  НоваяНоменклатура.ЕдиницаДляОтчетов =
НоваяНоменклатура.ЕдиницаИзмерения;
  НоваяНоменклатура.Записать();
  Возврат НоваяНоменклатура.Ссылка;
Исключение
  Возврат Наименование+", арт. "+Артикул+" - не удалось создать
номенклатуру в 1С: "+ОписаниеОшибки();
КонецПопытки;
```

```
КонецФункции
```

В методе предусмотрена обработка ошибок. В случае успешного выполнения функция возвращает ссылку на созданную номенклатуру, в случае ошибки – строку с описанием ошибки. В данном примере рассмотрено заполнение необходимого минимума атрибутов. В зависимости от версии конфигурации 1С, использовании режима запуска (управляемое или обычное приложение), дополнительных контролей заполнения реквизитов, код может отличаться. В методе вызывается также функция создания единицы измерения. В большинстве конфигураций 1С единицы измерения являются отдельным подчиненным номенклатуре справочником. Однако набор атрибутов этого справочника может быть различным.

```
Функция
СоздатьЕдиницуИзмеренияПоСтроке(ЕдиницаИзмеренияСтрока,Номенклатура)
НоваяЕдиница = Справочники.УпаковкиЕдиницыИзмерения.СоздатьЭлемент();
НоваяЕдиница.Наименование = ЕдиницаИзмеренияСтрока;
НоваяЕдиница.Владелец = Номенклатура;
НоваяЕдиница.Записать();
Возврат НоваяЕдиница.Ссылка
КонецФункции
```

Из процедуры **SetPriceIn1S()** вызывается метод **СоздатьДокументУстановкиЦен**:

```
Процедура СоздатьДокументУстановкиЦен(ТаблицаТоваров) ЭКСПОРТ
//подразумеваем, что в таблице значений номенклатура передается в
колонке "Ссылка", а "Цена" в колонке "Цена"
ВидЦеныРозничная =
Справочники.ВидыЦен.НайтиПоНаименованию("Розничная");
НовыйДокумент = Документы.УстановкаЦенНоменклатуры.СоздатьДокумент();
НовыйДокумент.Дата = ТекущаяДата();
нвид = НовыйДокумент.ВидыЦен.Добавить();
нвид.ВидЦены = ВидЦеныРозничная;
НовыйДокумент.Ответственный = ПараметрыСеанса.ТекущийПользователь;
Для Каждого стр из ТаблицаТоваров Цикл
  нстр = НовыйДокумент.Товары.Добавить();
  нстр.Номенклатура = стр.Ссылка;
  нстр.Упаковка = нстр.Номенклатура.ЕдиницаИзмерения;
  нстр.ВидЦены = ВидЦеныРозничная;
  нстр.Цена = стр.Цена;
КонецЦикла;
НовыйДокумент.Статус =
Перечисления.СтатусыУстановокЦенНоменклатуры.Согласован;
НовыйДокумент.Записать(РежимЗаписиДокумента.Проведение);
КонецПроцедуры
```

В функции заранее предопределены параметры для автоматического заполнения документа: тип цен, текущий ответственный, статусы согласования. Код функции

также будет отличаться в зависимости от механизмов конфигурации, используемых для учета цен номенклатуры.

Процедура **SetPriceIn1S()** может также использоваться как пример универсального взаимодействия ELMA и 1С через таблицу обмена. В рассмотренном примере такой подход более оптимален, т.к. при таком формате обмена создается единый документ установки цен, содержащий все товары, а не один документ на один товар. Это не перегружает информационное наполнение базы 1С. Кроме этого, такой обмен можно использовать, если на стороне 1С необходимо объединить обработку всего массива данных в единую транзакцию.

3.3. Импорт номенклатуры из 1С

В процедуре импорта номенклатуры 1С происходит поиск товаров в существующем справочнике «Товар» в ELMA (последовательно по двум атрибутам). Если соответствия не было найдено, создается новый товар.

```
public virtual void ImportFrom1C(Context context) {
    //1. Инициализируем необходимые для работы с 1С переменные
    var service = Locator.GetServiceNotNull < Integration1CService >
();
    ComObject connector = service.GetComConnector("erp1c");
    var Manager1C = (dynamic) connector.Reference;
    var provider = service.GetProvider("erp1c");
    //2. Получаем из 1С таблицу товаров, тип: "Таблица значений 1С"
    dynamic table1s = Manager1C._ElmaModule.ПолучитьТаблицуТоваров
();
    for (var i = 0; i < table1s.Количество (); i++) {
        var element = table1s.Получить (i);
        //для каждого, полученного из 1С товара проверяем его наличие
в ELMA (поиск по атрибуту "Номенклатура 1С")
        Goods goodsElma = null;
        var com1s = new ComObject(element.Ссылка);
        var elmaNomenclature1S = provider.LoadBookByComObject <
EleWise.ELMA.Integration1C.Configs.Erp1c.Nomenklatura > (com1s);
        goodsElma = EntityManager < Goods > .Instance.Find(p =
>p.Nomenclature1S == elmaNomenclature1S).FirstOrDefault();
        if (goodsElma == null) {
            //если элемент не найден - ищем по артикулу
            string codeArt = element.Артикул.Trim().ToLower();
            goodsElma = EntityManager < Goods > .Instance.Find(p =
>p.Code.Trim().ToLower() == codeArt).FirstOrDefault();
            if (goodsElma == null) {
                //если не найден - создаем новую сущность
                goodsElma = EntityManager < Goods >
.Instance.Create();
                //заполняем атрибуты из данных таблицы
                goodsElma.Code = element.Артикул;
                goodsElma.Unit = element.ЕдиницаИзмерения;
                goodsElma.Name = element.Наименование;
                goodsElma.Price = element.Цена;
                goodsElma.Description = element.Описание;
                goodsElma.Color = element.Цвет;
                goodsElma.Dimensions = element.Размер;
                goodsElma.Weight = element.Вес;
                //поставщика нужно найти по атрибуту "Контрагент 1С"
у объекта "Юридическое лицо"
                var com1sContractor = new
ComObject(element.Поставщик);
                var elmaContractor1S = provider.LoadBookByComObject <
EleWise.ELMA.Integration1C.Configs.Erp1c.Kontragenty >
(com1sContractor);
```

```
                goodsElma.Contractor = EntityManager <
EleWise.ELMA.CRM.Models.ContractorLegal > .Instance.Find(p =
>p.Contractor1C == elmaContractor1S).FirstOrDefault();
                //также заполняем атрибут "Номенклатура 1С"
                com1s = new ComObject(element.Ссылка);
                goodsElma.Nomenclature1S =
provider.LoadBookByComObject <
EleWise.ELMA.Integration1C.Configs.Erp1c.Nomenklatura > (com1s);
                goodsElma.UpdateDate = DateTime.Now;
                goodsElma.Save();
            } else {
                //если найден – заполняем атрибут "Номенклатура 1С"
                com1s = new ComObject(element.Ссылка);
                goodsElma.Nomenclature1S =
provider.LoadBookByComObject <
EleWise.ELMA.Integration1C.Configs.Erp1c.Nomenklatura > (com1s);
                goodsElma.UpdateDate = DateTime.Now;
                goodsElma.Save();
            }
        }
    }
}
```

В примере таблицу товаров получаем, вызывая функцию из 1С. Также запрос в 1С можно сформировать непосредственно из ELMA. Однако, часто формирование массива информации о товарах для интернет-магазина невозможно с использованием запроса. Это связано с тем, что требуется использовать различные механизмы хранения настроек выгрузки на сайт (список полей товаров, дополнительная информация, ключевые поля синхронизации и т.п.). Выборка информации в таком случае происходит через динамически-формируемый запрос или через последовательное наполнение таблицы данных на стороне 1С. Подобные механизмы предусмотрены в модулях выгрузки данных из различных конфигураций 1С в интернет-магазин на CMS «Битрикс».

В нашем случае запрос информации о товарах достаточно простой (выборка реквизитов справочника «Номенклатура» и связь с регистром сведений с ценами номенклатуры).

```
Функция ПолучитьТаблицуТоваров() ЭКСПОРТ
Запрос = Новый Запрос;
Запрос.Текст =  "ВЫБРАТЬ
                | НоменклатураСправочник.Ссылка КАК Ссылка,
                | НоменклатураСправочник.Наименование КАК
Наименование,
                | НоменклатураСправочник.Артикул КАК Артикул,
                | ЕСТЬNULL(ЦеныНоменклатурыСрезПоследних.Цена, 0) КАК
Цена,
```

```
                |
  НоменклатураСправочник.ЕдиницаИзмерения.Наименование КАК
ЕдиницаИзмерения,
                | НоменклатураСправочник._ElmaПоставщик КАК
Поставщик,
                | НоменклатураСправочник._ElmaРазмер КАК Размер,
                | НоменклатураСправочник._ElmaЦвет КАК Цвет,
                | НоменклатураСправочник.ВесЧислитель КАК Вес,
                | НоменклатураСправочник.Описание КАК Описание
                |ИЗ
                | Справочник.Номенклатура КАК НоменклатураСправочник
                |       ЛЕВОЕ СОЕДИНЕНИЕ
РегистрСведений.ЦеныНоменклатуры.СрезПоследних КАК
ЦеныНоменклатурыСрезПоследних
                |       ПО НоменклатураСправочник.Ссылка =
ЦеныНоменклатурыСрезПоследних.Номенклатура
                |            И (ЦеныНоменклатурыСрезПоследних.ВидЦены
= &amp;РозничнаяЦена)
                |ГДЕ
                | НоменклатураСправочник.Ссылка В
ИЕРАРХИИ(&amp;ГруппаНоменклатуры)
                | И НоменклатураСправочник.ЭтоГруппа = ЛОЖЬ";
//в примере – установим настройки типа цен и папок номенклатуры
"жестко"
Запрос.УстановитьПараметр("РозничнаяЦена",Справочники.ВидыЦен.НайтиПо
Наименованию("Розничная"));
СписокПапокНоменклатуры = Новый СписокЗначений;
СписокПапокНоменклатуры.Добавить(Справочники.Номенклатура.НайтиПоКоду
("000000043"));
СписокПапокНоменклатуры.Добавить(Справочники.Номенклатура.НайтиПоКоду
("000000044"));
Запрос.УстановитьПараметр("ГруппаНоменклатуры",СписокПапокНоменклатур
ы);
Выборка = Запрос.Выполнить().Выгрузить();
//возвращаем результат запроса в виде таблицы значений
Возврат Выборка;
КонецФункции
```

В 1С можно также хранить историю последнего экспорта, использовать планы обмена или версионирование для отслеживания измененных позиций справочника номенклатуры. На основании этих данных можно ограничить объем экспортируемых данных и оптимизировать процесс синхронизации.

Еще один сценарий процесса – запись в историю синхронизации. История хранит дату выполнения синхронизации, статус (если синхронизация выполнена успешно – «Истина», иначе – «Ложь»), описание и ссылку на экземпляр процесса. При необходимости можно хранить более развернутую историю, а также отдельно хранить информацию об импорте и экспорте.

```
public virtual void SaveSyncHistory (Context context)
{
    //сохраняем запись в справочнике историй об успешной
синхронизации
    SyncHistory synHis = EntityManager<SyncHistory>.Create();
    synHis.SyncDate = DateTime.Now;
    if (context.ScriptError == null && context.NotExported.IsEmpty)
    {
        //ошибок не было
        synHis.Status = true;
    }
    else
    {
        if (context.ScriptError.ToString().Trim()!="")
        {
            synHis.Status = false;
            synHis.Description = "Произошла ошибка синхронизации";
        }
        else
        {
            synHis.Status = false;
            synHis.Description = "Есть неэкспортированные товары";
        }
    }
    synHis.InstanceWorkflow = context.WorkflowInstance;
    synHis.Save();
}
```

3.4. Результаты работы процесса

Рассмотрим сценарий использования синхронизации.

В систему ELMA добавлен перечень товаров. Это может быть загрузка с прайса поставщика или добавление товаров вручную (Рис. 32).

Наименование	Артикул	Единица измерения	Дата обновления	Поставщик	Номенклатура 1С	
Светильник декоративный СД 1*60 Вт Е27 полукруг белый (в разборе)TDM	SQ0358-0001	шт	08.08.2017 17:16	ТДМ		
Светильник декоративный СД 1*60 Вт Е27 полукруг голубой (в разборе) TDM	SQ0358-0002	шт	08.08.2017 17:16	ТДМ		
Светильник декоративный СД 1*60 Вт Е27 полукруг мрамор TDM	SQ0358-0021	шт	08.08.2017 17:16	ТДМ		
Светильник декоративный СД 1*60 Вт Е27 полукруг янтарный (в разборе) TDM	SQ0358-0003	шт	08.08.2017 17:16	ТДМ		
Светильник декоративный СД 2*60 Вт Е27 круг белый (в разборе) TDM	SQ0358-0004	шт	08.08.2017 17:16	ТДМ		
Светильник декоративный СД 2*60 Вт Е27 круг голубой (в разборе) TDM	SQ0358-0005	шт	08.08.2017 17:16	ТДМ		
Светильник декоративный СД 2*60 Вт Е27 круг мрамор TDM	SQ0358-0022	шт	08.08.2017 17:16	ТДМ		
Светильник декоративный СД 2*60 Вт Е27 круг янтарный (в разборе) TDM	SQ0358-0006	шт	08.08.2017 17:16	ТДМ		

Рис. 32. Новые товары, добавленные в ELMA

После выполнения сценария «Выгрузить товары в 1С» у товаров в ELMA будет заполнен атрибут «Номенклатура 1С». В 1С будет создана номенклатура с заполненной информацией о товаре, взятой из соответствующих атрибутов карточки товара в ELMA. На Рис. 33 представлена карточка товара, экспортированного из ELMA в 1С.

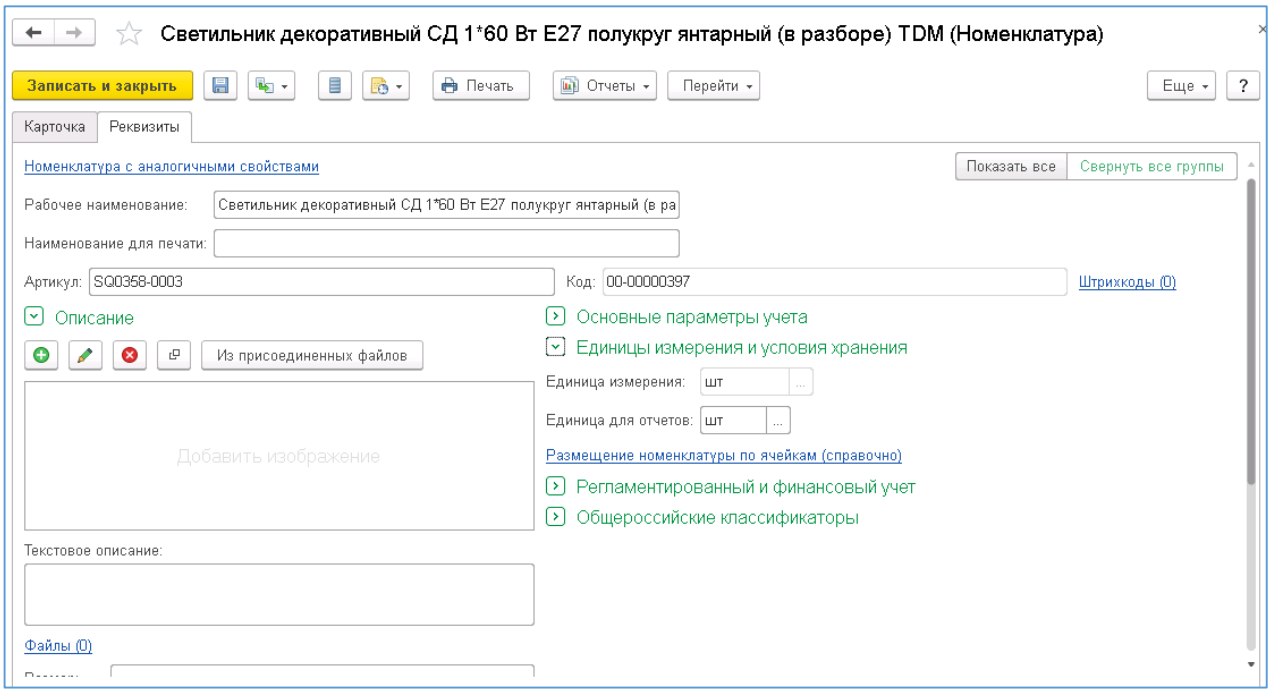


Рис. 33. Экспортированный из ELMA в 1С товар

Также будет создан документ установки цен на новые товары (Рис. 34).

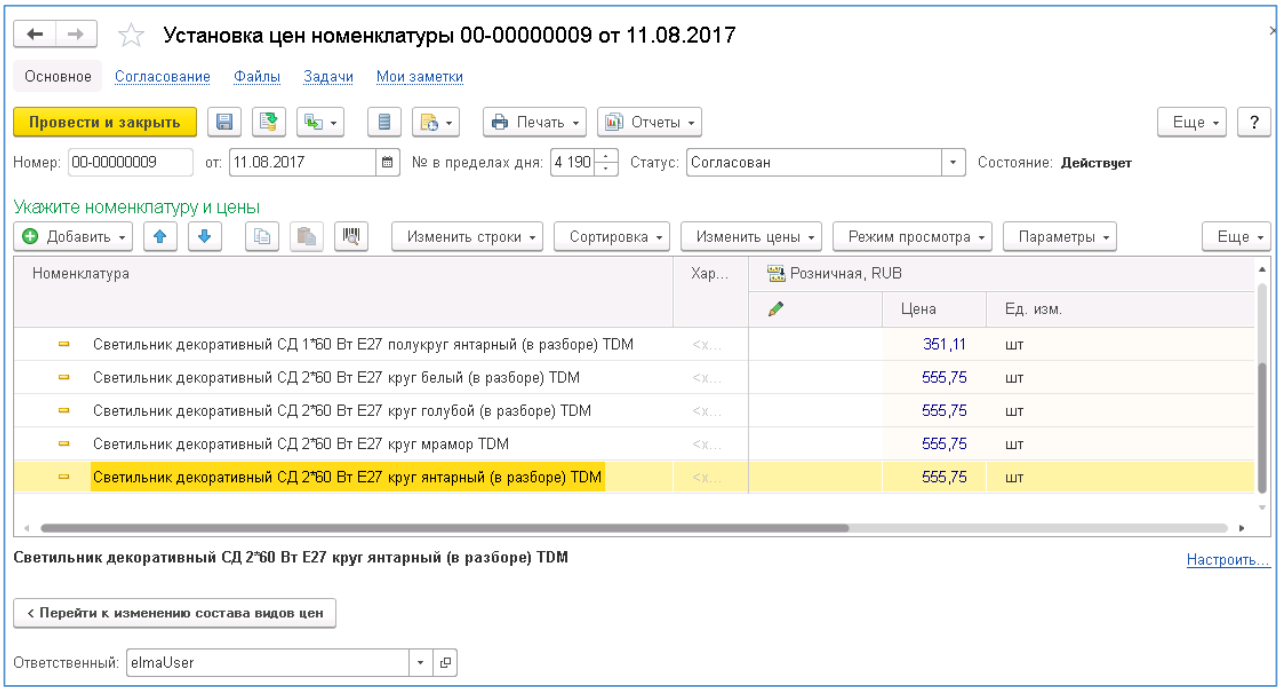


Рис. 34. Документ установки цен номенклатуры на загруженные в 1С товары

Далее выполняется сценарий «Загрузить номенклатуру из 1С». Из 1С выбираются товары из определенных папок справочника «Номенклатура». В рассмотренном примере это папка «Кухонные электроприборы» (Рис. 35).

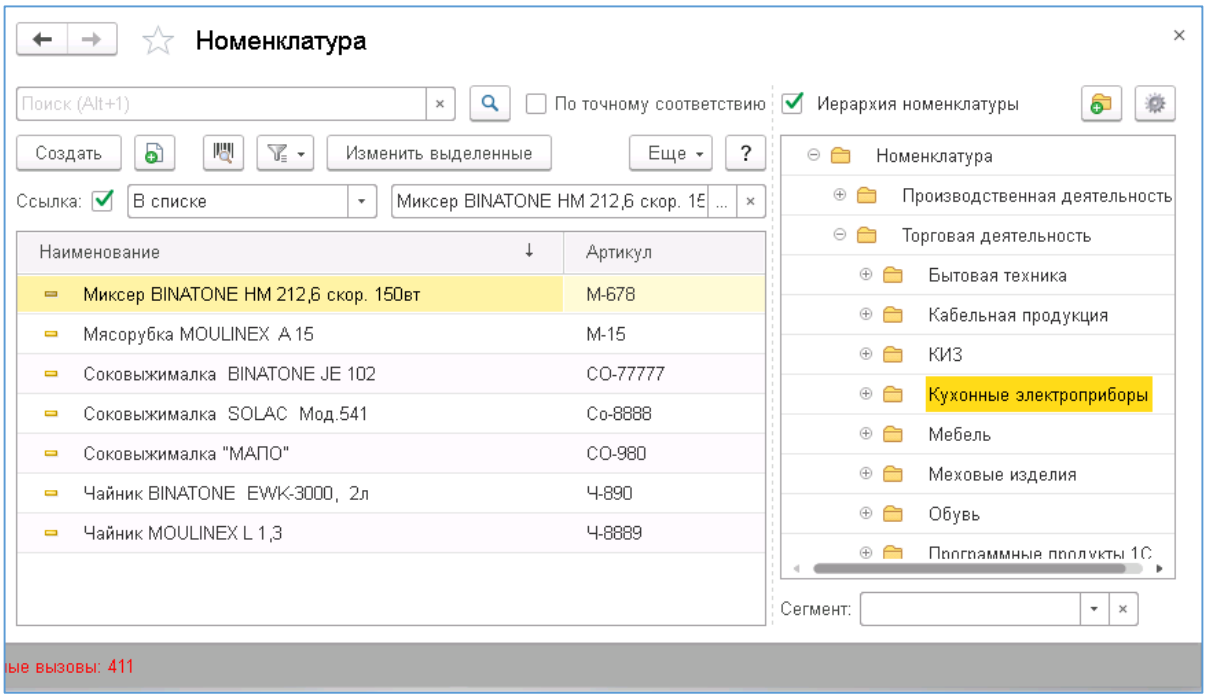


Рис. 35. Справочник «Номенклатура» в 1С

После выполнения сценария «Загрузить номенклатуру из 1С» в справочнике «Товар» в ELMA будут созданы новые записи (Рис. 36).

Наименование	Артикул	Единица измерения	Дата обновления	Поставщик	Номенклатура 1С
Миксер BINATONE HM 212,6 скор. 150вт	M-678	шт	11.08.2017 10:10	Альтаир	Миксер BINATONE HM 212,6 скор. 150вт, Не годен
Соковыжималка SOLAC Мод.541	Co-8888	шт	11.08.2017 10:10	Икар	Соковыжималка SOLAC Мод.541
Соковыжималка "МАПО"	CO-980	шт	11.08.2017 10:10	Все для дома	Соковыжималка "МАПО"
Чайник BINATONE EWK-3000, 2л	Ч-890	шт	11.08.2017 10:10	Альтаир	Чайник BINATONE EWK-3000, 2л
Чайник BINATONE AEJ-1001, 2,2л	Ч-980	шт	11.08.2017 10:10	Все для дома	Чайник BINATONE AEJ-1001, 2,2л
Мясорубка MOULINEX A 15	M-15	шт	11.08.2017 10:10	Ассоль	Мясорубка MOULINEX A 15
Соковыжималка BINATONE JE 102	CO-77777	шт	11.08.2017 10:10	Альтаир	Соковыжималка BINATONE JE 102
Чайник MOULINEX L 1,3	Ч-8889	шт	11.08.2017 10:10	Все для дома	Чайник MOULINEX L 1,3

Рис. 36. Импортированные из 1С товары

Глава 4. Обработка заказов клиентов

После обработки заказа клиента с интернет-магазина в системе ELMA создается запись в справочнике «Заказ». **Заказ** — это объект, объединяющий данные о клиенте, приобретенных им товаров и их стоимости. В 1С для отражения потребности клиента существует документ «Заказ клиента». В различных конфигурациях 1С он называется по-разному: «Заказ покупателя», «Счет на оплату» и т.п.

4.1. Выгрузка заказов покупателей в 1С

Перед выгрузкой информации о заказе клиента в 1С необходимо создать нового клиента-покупателя. Процедура создания клиента аналогична созданию номенклатуры. Однако, в новых конфигурациях 1С («Управление торговлей 11» и «Управление предприятием: ERP 2») вместе с элементом справочника «Контрагенты» должен создаваться элемент справочника «Партнеры» («Партнер» является реквизитом «Контрагент»). Это необходимо из-за особенностей учета данных конфигураций. В других конфигурациях достаточно создать только элемент справочника «Контрагент». После экспорта клиентов в 1С создаются их заказы.

Для хранения результатов выгрузки в атрибутах объекта «Физическое лицо» в ELMA добавлено свойство с типом «Контрагенты (Справочник 1С)». В атрибутах объекта «Заказ» в ELMA добавлено свойство с типом «Заказ клиента (Документ 1С)».

Карта процесса «Экспорт заказов клиентов» представлена на Рис. 37.

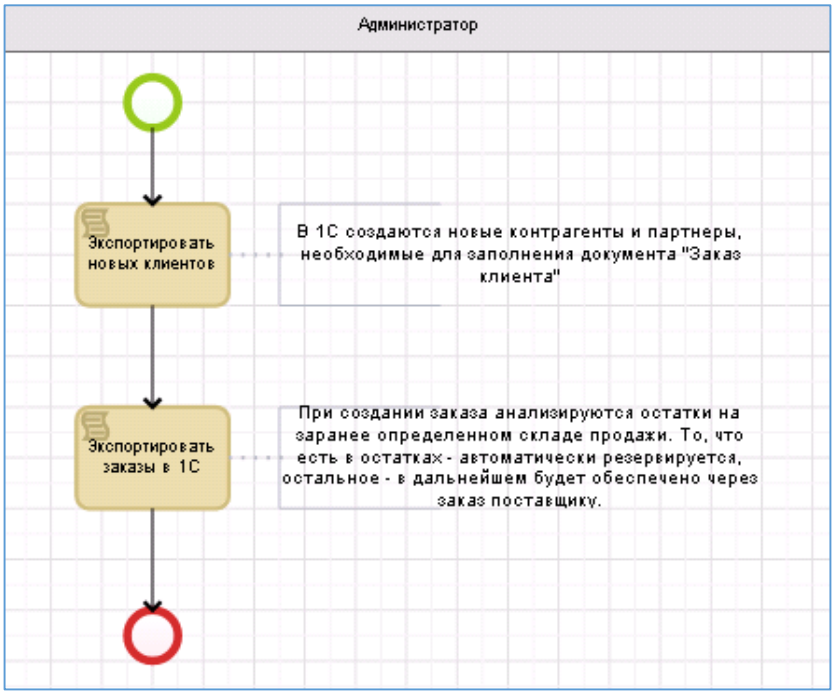


Рис. 37. Карта процесса «Экспорт заказов клиентов»

Для работы сценариев процесса необходимо подключить пространства имен:

```
using EleWise.ELMA.Services;
using EleWise.ELMA.ConfigurationModel;
using EleWise.ELMA.CRM.Models;
using EleWise.ELMA.Integration1C;
using EleWise.ELMA.Integration1C.Data;
using EleWise.ELMA.Integration1C.V82;
```

Сценарий «Экспортировать новых клиентов» выполняет выборку и экспорт клиентов, для которых еще нет установленной связи с клиентом в 1С (пустое поле с типом «Контрагенты (Справочник 1С)».

```
public virtual void ExportCustomers (Context context)
{
    //1. Выбираем клиентов (справочник "Физическое лицо"), у которых
нет связи с контрагентом 1С
    var listIndividual =
EntityManager<ContractorIndividual>.Instance.Find (p =>
p.Contractor1C == null).ToList ();
    //2. Инициализируем необходимые для работы с 1С переменные
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
    var Manager1C = (dynamic)connector.Reference;
    var provider = service.GetProvider ("erp1c");
    //3. Для каждого клиента создаем контрагента и партнера в 1С
    foreach (var element in listIndividual) {
        //создаем при помощи реализованного в конфигурации 1С метода
```

```
        var customer1s = Manager1C._ElmaModule.СоздатьПокупателя
(element.Name);
        //присваиваем атрибуту объекта "Физическое лицо" ссылку на
контрагента
        var com1s = new ComObject (customer1s);
        element.Contractor1C =
provider.LoadBookByComObject<EleWise.ELMA.Integration1C.Configs.Erp1c
.Kontragenty> (com1s);
        element.Save ();
    }
}
```

Сценарий «Экспортировать заказы в 1С» выполняет создание в 1С новых заказов клиентов из данных заказов ELMA, у которых не заполнен атрибут «Заказ 1С».

```
public virtual void ExportOrders (Context context)
{
    //1. Инициализируем необходимые для работы с 1С переменные
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
    var Manager1C = (dynamic)connector.Reference;
    var provider = service.GetProvider ("erp1c");
    //2. Создаем таблицу обмена по товарам заказа
    dynamic table1s = connector.GetFunctionValue ("NewObject",
"ТаблицаЗначений");
    //типизация колонок при создании не обязательна – типы будут
назначены после добавления первой строки
    table1s.Колонки.Добавить("Товар");
    table1s.Колонки.Добавить("Цена");
    table1s.Колонки.Добавить("Количество");
    //3. Выбираем неэкспортированные заказы (пустое поле "Заказ 1С")
и сортируем их по дате заказа
    var listOrder = EntityManager<Order>.Instance.Find (p =>
p.Order1S == null).ToList ().OrderBy(p=>p.OrderDate).ToList();
    //4. По каждому заказу формируем документ "Заказ клиента" в 1С и
проводим его.
    foreach(var element in listOrder)
    {
        //очищаем таблицу
        table1s.Очистить();
        foreach (var goods in element.Product)
        {
            var nStr = table1s.Добавить();
            nStr.Товар =
goods.Product.Nomenclature1S.GetComReference().Ref;
            nStr.Цена = (goods.Price.HasValue)?goods.Price.Value:0;
            nStr.Количество = goods.Quantity;
        }
        var order1s = Manager1C._ElmaModule.СоздатьЗаказПокупателя
(table1s,element.OrderDate,element.Contractor.Contractor1C.GetComRefe
rence().Ref);
        //присваиваем атрибуту объекта "Заказ" ссылку на заказ
клиента в 1с
```

```
        var com1s = new ComObject (order1s);
        element.Order1S =
provider.LoadDocumentByComObject<EleWise.ELMA.Integration1C.Configs.E
rp1c.ZakazKlienta> (com1s);
        element.Save ();
    }
}
```

Оба сценария создают объекты в 1С через реализованные в конфигурации 1С методы: **СоздатьПокупателя** и **СоздатьЗаказПокупателя**.

```
Функция СоздатьПокупателя(ФИО) ЭКСПОРТ
//создаем элемент справочника "Партнеры"
НовыйПартнер = Справочники.Партнеры.СоздатьЭлемент();
НовыйПартнер.Клиент = Истина;
НовыйПартнер.Комментарий = "Клиент интернет-магазина";
НовыйПартнер.Наименование = ФИО;
НовыйПартнер.НаименованиеПолное = ФИО;
НовыйПартнер.Записать();
//создаем элемент справочника "Контрагенты"
НовыйКонтрагент = Справочники.Контрагенты.СоздатьЭлемент();
НовыйКонтрагент.Наименование = ФИО;
НовыйКонтрагент.НаименованиеПолное = ФИО;
НовыйКонтрагент.ЮрФизЛицо = Перечисления.ЮрФизЛицо.ФизЛицо;
НовыйКонтрагент.ЮридическоеФизическоеЛицо =
Перечисления.ЮридическоеФизическоеЛицо.ФизическоеЛицо;
НовыйКонтрагент.Партнер = НовыйПартнер.Ссылка;
НовыйКонтрагент.Записать();
Возврат НовыйКонтрагент.Ссылка;
КонецФункции
```

Вместе с элементом справочника «Контрагенты» в 1С также создается элемент справочника «Партнеры» с установленным флажком «Клиент».

При создании заказа покупателя также осуществляется проверка свободных остатков с целью правильно расставить способы обеспечения («Резерв» или «Обособленное обеспечение»).

```
Функция СоздатьЗаказПокупателя(ТаблицаТовары,Дата,Контрагент) ЭКСПОРТ
//заказ создается по таблице содержащей колонки "Товар", "Цена",
"Количество"
//склад определяется заранее
СкладЗаказов = Справочники.Склады.НайтиПоНаименованию("Магазин
"+Символ(34)+"Бытовая техника"+Символ(34)+" ");
НовыйЗаказ = Документы.ЗаказКлиента.СоздатьДокумент();
НовыйЗаказ.Дата = Дата;
НовыйЗаказ.Автор = ПараметрыСеанса.ТекущийПользователь;
НовыйЗаказ.Партнер = Контрагент.Партнер;
НовыйЗаказ.Контрагент = Контрагент;
//заполняем предопределенные, необходимые для проведения документа
данные
```

```
НовыйЗаказ.Соглашение =
Справочники.СоглашенияСКлиентами.НайтиПоНаименованию("Розничные
продажи");
НовыйЗаказ.Организация =
Справочники.Организации.НайтиПоНаименованию("Металл-Сервис");
НовыйЗаказ.Комментарий = "Импортирован из ELMA";
НовыйЗаказ.Статус = Перечисления.СтатусыЗаказовКлиентов.КОбеспечению;
НовыйЗаказ.Склад = СкладЗаказов;
НовыйЗаказ.Приоритет =
Справочники.Приоритеты.НайтиПоНаименованию("Средний");
НовыйЗаказ.Валюта = Справочники.Валюты.НайтиПоКоду("643");
НовыйЗаказ.НеОтгружатьЧастями = ИСТИНА;
НовыйЗаказ.СпособДоставки = Перечисления.СпособыДоставки.Самовывоз;
НовыйЗаказ.НалогообложениеНДС =
Перечисления.ТипыНалогообложенияНДС.ПродажаОблагаетсяНДС;
НовыйЗаказ.ФормаОплаты = Перечисления.ФормыОплаты.Наличная;
НовыйЗаказ.ПорядокОплаты =
Перечисления.ПорядокОплатыПоСоглашениям.РасчетыВРубляхОплатаВРублях;
НовыйЗаказ.ХозяйственнаяОперация =
Перечисления.ХозяйственныеОперации.РеализацияКлиенту;
//перед заполнением товаров необходимо оценить остатки на складах
МассивТоваров = ТаблицаТовары.ВыгрузитьКолонку("Товар");
ТаблицаОстатков =
ПолучитьТаблицуОстатков(МассивТоваров,СкладЗаказов);
Для Каждого строка из ТаблицаТовары Цикл
  нстр = НовыйЗаказ.Товары.Добавить();
  нстр.Номенклатура = строка.Товар;
  нстр.Цена = строка.Цена;
  нстр.Количество = строка.Количество;
  нстр.Склад = СкладЗаказов;
  нстр.СтавкаНДС = Перечисления.СтавкиНДС.НДС18;
  нстр.Упаковка = нстр.Номенклатура.ЕдиницаИзмерения;
  //анализируем остатки товаров и выбираем вариант обеспечения
  СтрокаСОстатками =
ТаблицаОстатков.Найти(строка.Товар,"Номенклатура");
  дстр = неопределено;
  Если СтрокаСОстатками<>Неопределено Тогда
        Если нстр.Количество<=СтрокаСОстатками.СвободныйОстаток Тогда
              //свободного остатка достаточно, ставим в резерв
              нстр.ВариантОбеспечения =
Перечисления.ВариантыОбеспечения.СоСклада;
        Иначе
              //не все количество доступно - разбиваем строку
              ОстатокКоличества = нстр.Количество-
СтрокаСОстатками.СвободныйОстаток;
              //в резерве оставляем то, чт.е. в остатке
              нстр.Количество = СтрокаСОстатками.СвободныйОстаток;
              нстр.ВариантОбеспечения =
Перечисления.ВариантыОбеспечения.СоСклада;
              //исходное количество - остаток устанавливаем в новую
строку к обеспечению
              дстр = НовыйЗаказ.Товары.Добавить();
              ЗаполнитьЗначенияСвойств(дстр,нстр);
```

```
            дстр.Количество = ОстатокКоличества;
            дстр.КоличествоУпаковок = дстр.Количество;
            дстр.ВариантОбеспечения =
Перечисления.ВариантыОбеспечения.Обособленно;
            дстр.Сумма = нстр.Цена*нстр.Количество;
            дстр.СуммаСНДС = дстр.Сумма;
            дстр.СуммаНДС =дстр.Цена*дстр.Количество*0.18;
        КонецЕсли;
  Иначе
        //свободного остатка по товарам нет
        нстр.ВариантОбеспечения =
Перечисления.ВариантыОбеспечения.Обособленно;
  КонецЕсли;
  нстр.КоличествоУпаковок = нстр.Количество;
  нстр.Сумма = нстр.Цена*нстр.Количество;
  нстр.СуммаСНДС = нстр.Сумма;
  нстр.СуммаНДС =нстр.Цена*нстр.Количество*0.18;
КонецЦикла;
//анализируем состав заказа
//если все товары есть в наличии – отгрузка через 3 дня после заказа
//если есть товары к обеспечению – отгрузка через 15 дней после
заказа
//также от этого зависит статус заказа
ПараметрыОтбора = Новый Структура();
ПараметрыОтбора.Вставить("ВариантОбеспечения",Перечисления.ВариантыОб
еспечения.Обособленно);
МассивСтрокОбеспечение =
НовыйЗаказ.Товары.НайтиСтроки(ПараметрыОтбора);
Если МассивСтрокОбеспечение.Количество()>0 Тогда
  //есть строки к обеспечению
  НовыйЗаказ.ДатаОтгрузки = НовыйЗаказ.Дата+86400*15;
  НовыйЗаказ.Статус =
Перечисления.СтатусыЗаказовКлиентов.КОбеспечению;
Иначе
  НовыйЗаказ.ДатаОтгрузки = НовыйЗаказ.Дата+86400*3;
  НовыйЗаказ.Статус = Перечисления.СтатусыЗаказовКлиентов.КОтгрузке;
КонецЕсли;
НовыйЗаказ.Записать(РежимЗаписиДокумента.Проведение);
Возврат НовыйЗаказ.Ссылка;
КонецФункции

Функция ПолучитьТаблицуОстатков(МассивТоваров,Склад) ЭКСПОРТ
//в ERP 2.2 для оперативного учета свободных остатков используется
отдельный регистр, выборку осуществляем из него
Запрос = Новый Запрос;
Запрос.Текст =  "ВЫБРАТЬ
                | СвободныеОстаткиОстатки.Номенклатура КАК
Номенклатура,
                | СвободныеОстаткиОстатки.ВНаличииОстаток –
СвободныеОстаткиОстатки.ВРезервеСоСкладаОстаток –
СвободныеОстаткиОстатки.ВРезервеПодЗаказОстаток КАК СвободныйОстаток
                |ИЗ
```

```
                    | РегистрНакопления.СвободныеОстатки.Остатки КАК
СвободныеОстаткиОстатки
                    |ГДЕ
                    | СвободныеОстаткиОстатки.Номенклатура
В(&amp;СписокНоменклатуры)
                    | И СвободныеОстаткиОстатки.Склад = &amp;Склад" ;
Запрос.УстановитьПараметр("СписокНоменклатуры",МассивТоваров);
Запрос.УстановитьПараметр("Склад",Склад);
Выборка = Запрос.Выполнить().Выгрузить();
Возврат Выборка;
КонецФункции
```

Процедуры создания покупателя, заказа клиента и контроля свободных остатков приведены для типовой конфигурации «1С: ERP Управление предприятием 2». Также подобный набор объектов присутствует в «1С: Управление торговлей 11». В распространенных конфигурациях «1С: Управление торговлей 10.3», «1С: Управление предприятием 1.3» используется иная, более простая объектная модель. Также для контроля свободных остатков там необходимо проверять остатки сразу по нескольким регистрам.

На Рис. 38 представлена страница заказа в веб-приложении ELMA, на Рис. 39 – карточка документа «Заказ клиента», экспортированного из ELMA в 1С.

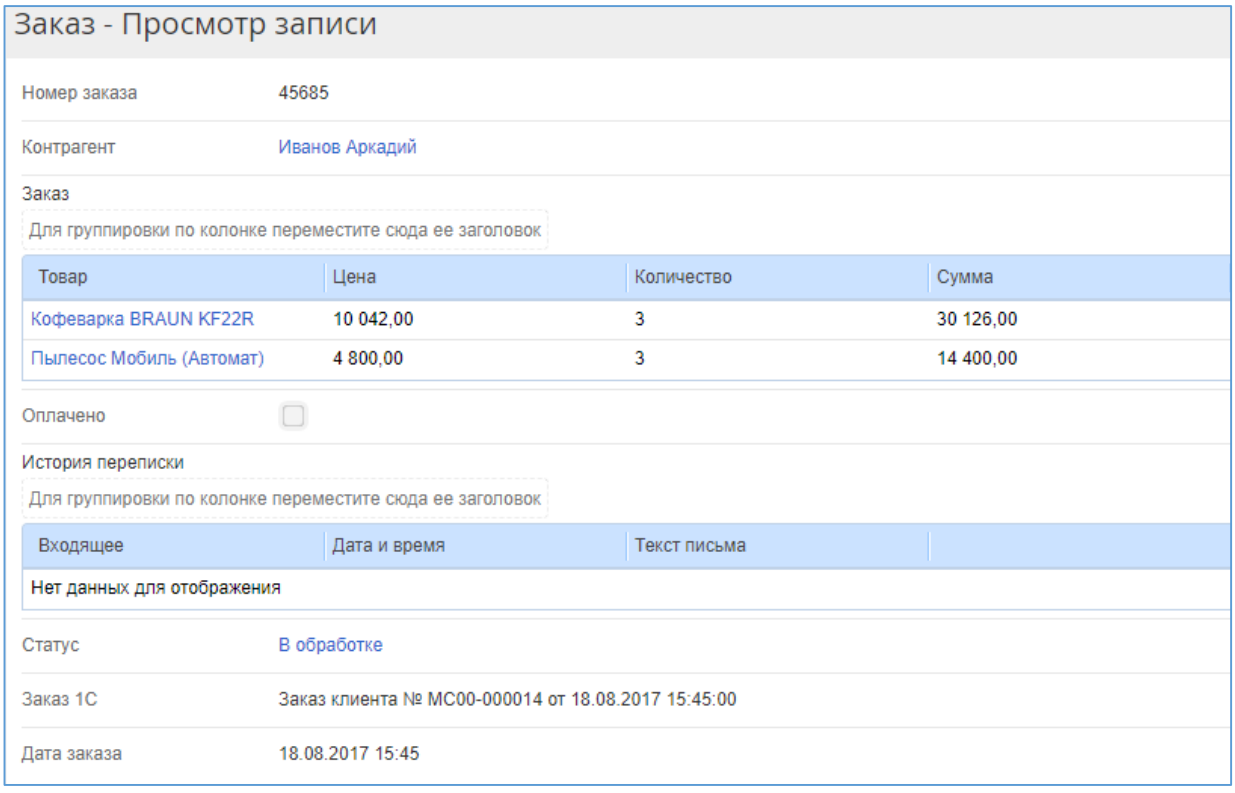


Рис. 38. Заказ в ELMA

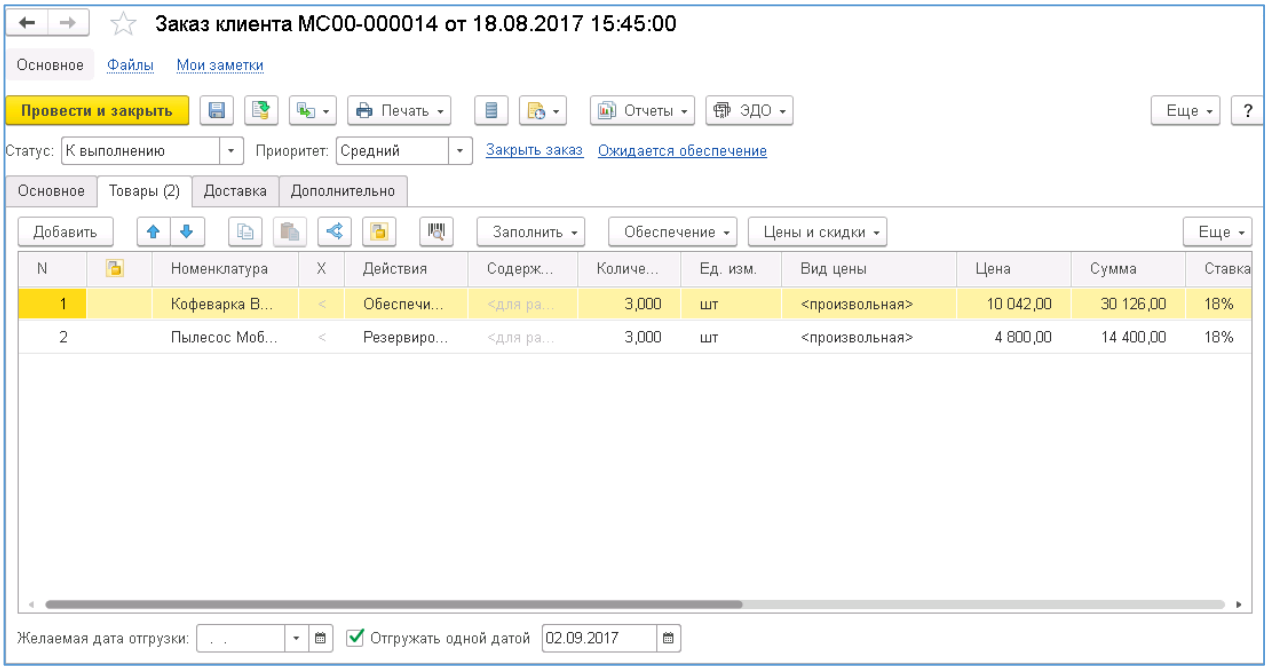


Рис. 39. Заказ клиента в 1С

Как видно, по одному из товаров на складе не было остатка, поэтому для него выбрано «Обособленное обеспечение». Другой имеющийся на складе товар был зарезервирован.

4.2. Обеспечение потребностей по заказам покупателей

В приведенном примере в заказе покупателя для товаров, которых нет в наличии, выбирается способ обеспечения потребности «Обособленное обеспечение». При таком способе используется дополнительный учет обеспечения потребностей в разрезе «Назначение» (от этапа формирования потребностей до поступления и отгрузки), в котором указывается созданный заказ клиента. Таким образом мы можем однозначно определить, под какой заказ клиента поступил товар, и какой мы уже отгрузили.

В большинстве торгово-производственных компаний все операции по обеспечению потребностей осуществляются в учетной системе. Логично обеспечивать потребности интернет-магазина вместе с остальным пулом потребностей компании (заказы на производство, потребности розничной сети, потребности оптовых продаж и т.п.).

Существуют различные способы обеспечения потребностей. Они зависят от особенностей деятельности компании. В каких-то случаях используется обычная схема резервирования – когда между источниками обеспечения и исходными потребностями есть явные связи. Более сложная система обеспечения – это резервирование в графике, т.е. когда источники обеспечения и потребности анализируются в целом для каждого момента времени.

Для определения факта обеспечения потребностей интернет-магазина будем использовать специальный сервисный процесс. Каждый раз при запуске процесса для заказов с определенными статусами будет проверяться поступление товаров по потребностям заказа клиента в 1С.

Результатом работы процесса будет являться актуализация статусов заказов. Новый статус позволит определить состояние обеспеченности заказа.

Процесс актуализации статуса, исходя из обеспеченности заказа, можно свести к автоматическому запуску единственного сценария (Рис. 40).

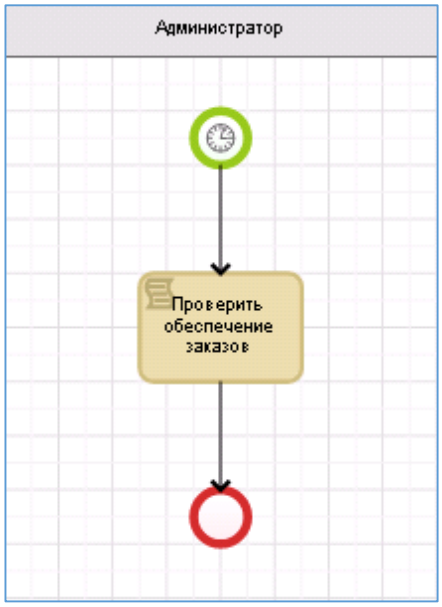


Рис. 40. Процесс проверки обеспечения заказов

Содержание сценария «Проверить обеспеченность заказов»:

```
public virtual void CheckOrderProcuring (Context context)
{
    //1. Получаем список заказов в статусах "Ожидание" "В обработке"
"Ожидает поступления на склад", экспортированные в 1с
    var listOrder = EntityManager<Order>.Instance.Find (p =>
p.Order1S != null && (p.Status.Name == "Ожидание" || p.Status.Name ==
"В обработке" || p.Status.Name == "Ожидает поступления на
склад")).ToList ().OrderBy(p=>p.OrderDate).ToList();
    //2. Инициализируем необходимые для работы с 1С переменные
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
    var Manager1C = (dynamic)connector.Reference;
    var provider = service.GetProvider ("erp1c");
    //3. По каждому заказу вызываем функцию из 1С, в которой
проверяем состояние его обеспеченности
    foreach(var element in listOrder)
    {
        Console.WriteLine(element.Number);
        string status1s =
Manager1C._ElmaModule.ПроверитьОбеспечение(element.Order1S.GetComRefe
rence().Ref);
        Console.WriteLine(status1s);
        //если вернулся пустой статус - не меняем статус заказа
        if (status1s!="")
        {
            var statusOrder =
EntityManager<OrderStatus>.Instance.Find(p => p.Name ==
status1s).FirstOrDefault();
            element.Status = statusOrder;
            element.Save();
```

```
        }
    }
}
```

В сценарии осуществляется вызов реализованного в конфигурации 1С метода **ПроверитьОбеспечение**, параметром которого является ссылка на заказ клиента 1С.

```
Функция ПроверитьОбеспечение(ЗаказСсылка) ЭКСПОРТ
//СписокЗаказов = СтруктураПередачи.СписокЗаказов;
//в запросе анализируются товары по заказам и состояние их
обеспечения
Запрос = Новый Запрос;
Запрос.Текст = "ВЫБРАТЬ
               |  ОбеспечениеЗаказовОбороты.Назначение.Заказ КАК
НазначениеЗаказ,
               |  ОбеспечениеЗаказовОбороты.Номенклатура КАК
Номенклатура,
               |  ОбеспечениеЗаказовОбороты.ПотребностьПриход КАК
ЗаказаноКлиентом,
               |
  ЕСТЬNULL(ТоварыКПоступлениюОстаткиИОбороты.КПоступлениюПриход, 0)
КАК ЗаказаноУПоставщика,
               |
  ЕСТЬNULL(ТоварыКПоступлениюОстаткиИОбороты.КПоступлениюРасход, 0)
КАК ПоступлениеПоЗаказу
               |ИЗ
               |  РегистрНакопления.ОбеспечениеЗаказов.Обороты(, , ,
) КАК ОбеспечениеЗаказовОбороты
               |       ЛЕВОЕ СОЕДИНЕНИЕ
РегистрНакопления.ТоварыКПоступлению.ОстаткиИОбороты КАК
ТоварыКПоступлениюОстаткиИОбороты
               |       ПО ОбеспечениеЗаказовОбороты.Номенклатура =
ТоварыКПоступлениюОстаткиИОбороты.Номенклатура
               |             И ОбеспечениеЗаказовОбороты.Назначение =
ТоварыКПоступлениюОстаткиИОбороты.Назначение
               |ГДЕ
               |  ОбеспечениеЗаказовОбороты.Назначение.Заказ =
&amp;Заказ
               |ИТОГИ
               |  СУММА(ЗаказаноКлиентом),
               |  СУММА(ЗаказаноУПоставщика),
               |  СУММА(ПоступлениеПоЗаказу)
               |ПО
               |  НазначениеЗаказ";
Запрос.УстановитьПараметр("Заказ",ЗаказСсылка);
//результаты группируются по заказам
Выборка =
Запрос.Выполнить().Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);
Если Выборка.Следующий() Тогда
  //для примера анализируем количество в целом по заказу
  Если Выборка.ПоступлениеПоЗаказу >= Выборка.ЗаказаноКлиентом Тогда
       //весь товар по заказу поступил - статус  "На складе"
       Возврат "На складе";
```

```
  ИначеЕсли Выборка.ПоступлениеПоЗаказу < Выборка.ЗаказаноКлиентом
        И  Выборка.ЗаказаноУПоставщика >= Выборка.ЗаказаноКлиентом
Тогда
        //товар не поступил в полном объеме – но заказан у поставщика
– статус "Ожидает поступления на склад"
        Возврат "Ожидает поступления на склад";
  Иначе
        //товар не заказан у поставщика – будет возвращена пустая
строка
        Возврат "";
  КонецЕсли;
Иначе
  Возврат "";
КонецЕсли;
КонецФункции
```

В функции осуществляется позаказный анализ через запрос. Возможно анализировать список заказов, однако, при работе конфигурации 1С в управляемом режиме есть сложности с передачей списка заказов с клиента на сервер. Поэтому в коде для того, чтобы избежать лишнего использования дополнительных структур данных, используется выполнение запроса для одного заказа. Также в запросе предусмотрена выборка обеспеченности по каждому из товаров, а не только в целом по заказу. Это может быть использовано при расширенной аналитике обеспеченности заказа со стороны ELMA. Статус можно детализировать до уровня товаров в заказе.

Рассмотрим пример: до выполнения сценария в ELMA сформированы 2 заказа и экспортированы в 1С (Рис. 41).

Номер заказа	Контрагент	Статус	Заказ 1С
45685	Иванов Аркадий	В обработке	Заказ клиента № МС00-000014 от 18.08.2017 15:45:00
77789	Иванов Аркадий	В обработке	Заказ клиента № МС00-000015 от 21.08.2017 20:00:00

Рис. 41. Исходные заказы в ELMA

После выполнения процесса проверки обеспечения заказов статусы будут актуализированы, исходя из данных о движении товаров в системе 1С (Рис. 42).

Номер заказа	Контрагент	Статус	Заказ 1С
45685	Иванов Аркадий	На складе	Заказ клиента № МС00-000014 от 18.08.2017 15:45:00
77789	Иванов Аркадий	Ожидает поступления на склад	Заказ клиента № МС00-000015 от 21.08.2017 20:00:00

Рис. 42. Результаты работы процесса проверки обеспечения статусов заказов

Для первого заказа из списка в 1С создана соответствующая цепочка обеспечения потребностей заказа: Заказ клиента – Заказ поставщику – Поступление товаров – Приходный ордер на товары (Рис. 43).

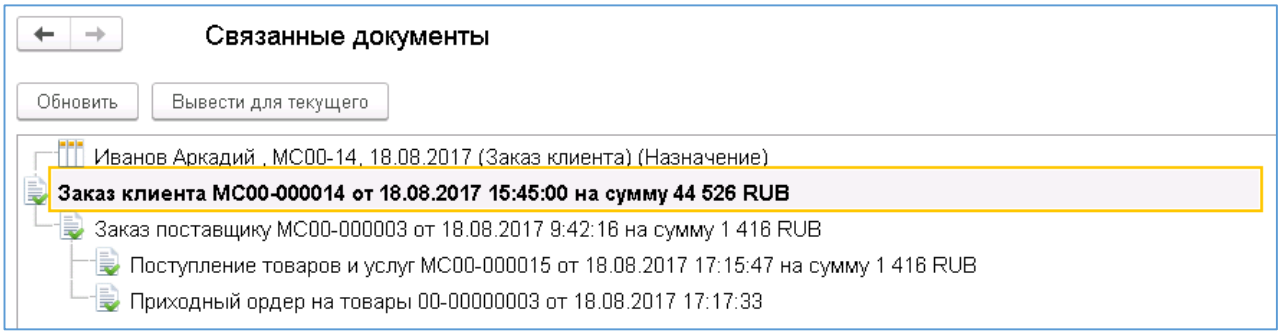


Рис. 43. Цепочка документов обеспечения потребностей

Запрос анализа обеспечения потребностей может очень сильно отличаться в зависимости от используемого в организации метода обеспечения. Не всегда получается учитывать сквозную аналитику от исходной потребности до поступления товаров. Также в разных конфигурациях 1С для хранения данных о движении товаров используются различные регистры и аналитики. При реализации механизма проверки обеспечения необходимо определить четкие условия. Хранение неактуальной информации по состоянию заказов может привести к серьезному сбою в работе интернет-магазина.

Также подобным образом из 1С можно получать более расширенный набор статусов, например, отслеживать отправку заказа в транспортную компанию (после отгрузки). При расширении объектной модели и доработки функции 1С можно также отслеживать дату ожидаемого поступления товаров на склад. Также, как было описано выше, можно предоставлять клиенту информацию о том, какие товары уже на складе, обеспечив возможность частичного получения заказа.

4.3. Отображение в системе ELMA остатков на складах

В некоторых случаях возможна обработка заказов из интернет-магазина в ручном режиме. Оператор интернет-магазина может принимать решение, например, о частичной поставке товаров из заказа или предложить клиенту какой-либо аналогичный товар вместо отсутствующего. Для решения подобных задач необходимо отображать информацию о текущих остатках товаров на складе в интерфейсе работы оператора интернет-магазина в ELMA. Это существенно сократит время работы сотрудника, т.к. ему не потребуется запускать дополнительную информационную систему 1С и вводить в ней данные для отбора нужной информации.

Задачу решим с учетом следующего сценария использования:

1. Оператор в процессе выбирает необходимый заказ.
2. На форме задачи отображаются данные по товарам в заказе и их остаток на текущую дату.

Как правило, два этих этапа могут быть частью более сложного процесса ручной обработки заказа клиента. Карта с этапами процесса представлена на Рис. 44.

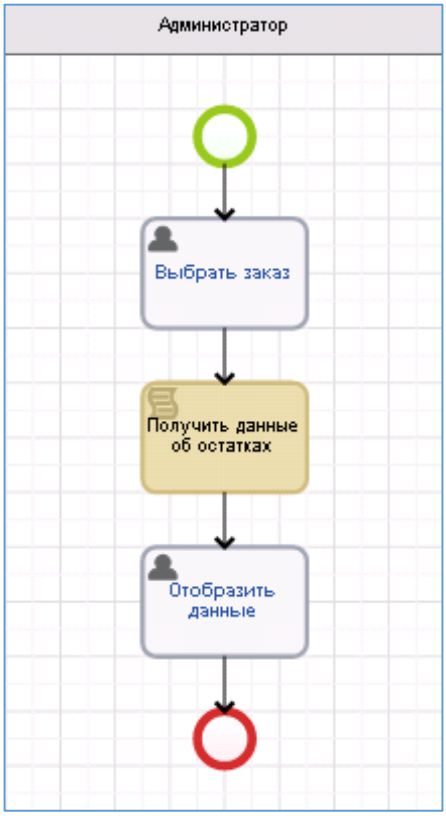


Рис. 44. Карта процесса с получением информации по остаткам

В контексте будем использовать переменную с выбранным заказом клиента и блок с товарами, их количеством в заказе и остатком на складах (Рис. 45).

Отображаемое имя	Имя свойства	Тип
Базовые свойства		
Заказ	Order	Заказ (Объект)
Список товаров	GoodsList	Блок
Товар	Good	Товар (Объект)
Заказано	Ordered	Дробное число
Остаток на складе	Balance	Дробное число

Рис. 45. Необходимые переменные для работы процесса

Для демонстрации примера исполнения запроса 1С из ELMA реализуем всю обработку получения данных на стороне ELMA. Для отбора данных из регистра остатков выполняем запрос по всем товарам из выбранного в ELMA заказа.

Необходимо дополнительно подключить пространства имен:

```
using EleWise.ELMA.Model.Services;
using EleWise.ELMA.Services;
using EleWise.ELMA.ConfigurationModel;
using EleWise.ELMA.CRM.Models;
using EleWise.ELMA.Integration1C;
using EleWise.ELMA.Integration1C.Data;
using EleWise.ELMA.Integration1C.V82;
```

Текст процедуры:

```
public virtual void GetGoodsBalance (Context context)
{
    //1. Инициализируем необходимые для работы с 1С переменные
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
    var Manager1C = (dynamic)connector.Reference;
    var provider = service.GetProvider ("erp1c");
    //2. Инициализируем запрос
    dynamic query1c = connector.GetFunctionValue ("NewObject",
"Запрос");
    //3. Установим текст запроса
    string textQuery = "ВЫБРАТЬ "+
    "СвободныеОстаткиОстатки.Номенклатура КАК Номенклатура, "+
    "СвободныеОстаткиОстатки.ВНаличииОстаток –
СвободныеОстаткиОстатки.ВРезервеСоСкладаОстаток –
СвободныеОстаткиОстатки.ВРезервеПодЗаказОстаток КАК СвободныйОстаток
"+
    "ИЗ "+
    "РегистрНакопления.СвободныеОстатки.Остатки КАК
СвободныеОстаткиОстатки "+
    "ГДЕ "+
```

```
    "СвободныеОстаткиОстатки.Номенклатура В(&СписокНоменклатуры) ";
    query1c.Текст = textQuery;
    //4. В качестве параметра запроса установим список значений 1С,
содержащий список ссылок на номенклатуру из заказа
    dynamic listOfGoods1s = connector.GetFunctionValue ("NewObject",
"СписокЗначений");
    foreach(var element in context.Order.Product)
    {

listOfGoods1s.Добавить(element.Product.Nomenclature1S.GetComReference
().Ref());
    }
    //5. Устанавливаем полученный список в качестве параметра запроса
    query1c.УстановитьПараметр("СписокНоменклатуры", listOfGoods1s);
    //6. Выполняем запрос и выгружаем результат в таблицу значений
    dynamic table1s = query1c.Выполнить().Выгрузить();
    //7. Очищаем и заполняем блок в контексте процесса
    context.GoodsList.Clear();
    foreach(var element in context.Order.Product)
    {
        var newRow =
InterfaceActivator.Create<P_OtobrazhenieOstatkov_GoodsList>();
        newRow.Good = element.Product;
        newRow.Ordered = element.Quantity;
        //для того, чтобы найти остаток по товару из таблицы-
результата запроса – воспользуемся методом таблицы значений 1С
"НайтиСтроки"
        //инициализируем структуру отбора
        dynamic filterStruct = connector.GetFunctionValue
("NewObject", "Структура");
        //вставляем свойство – его имя должно совпадать с именем
колонки с полем поиска в таблице-результате запроса
filterStruct.Вставить("Номенклатура",element.Product.Nomenclature1S.G
etComReference().Ref);
        //выполняем метод, результат – коллекция строк таблицы
значений
        dynamic arrOfRows = table1s.НайтиСтроки(filterStruct);
        //если есть строки в результате – то это одна строка по
одному товару, берем из нее данные
        if (arrOfRows.Количество()>0)
        {
            newRow.Balance = arrOfRows.Получить(0).СвободныйОстаток;
        }
        //добавляем строку в блок
        context.GoodsList.Add(newRow);
    }
}
```

Результаты работы процесса (задача «Отобразить данные») представлены на Рис. 46.

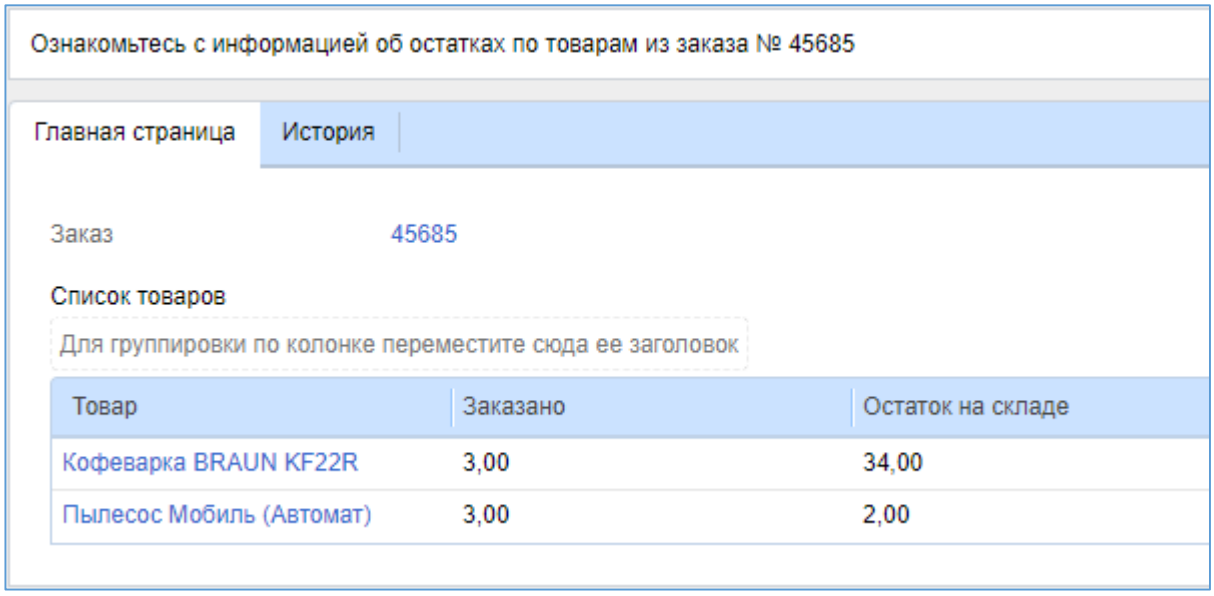


Рис. 46. Отображение остатков товаров на форме задачи процесса

В приведенном выше сценарии используется выполнение запроса 1С на стороне ELMA. Если разработка запроса происходит в 1С при помощи консоли запросов, необходимо дополнительное форматирование текста. Требуется расставить переносы строк или дополнительные пробелы (аналогично запросам в синтаксисе SQL). Также, если используется пакет запросов, между разными виртуальными таблицами конструктор вставляет строку вида «//////////». И, если не расставить переносы строк, эта строка будет воспринята 1С как конструкция комментария и текст после этой конструкции не будет принят к выполнению.

Для соотнесения исходных и полученных в результате запроса данных используется метод **НайтиСтроки** таблицы значений 1С. Он позволяет однозначно отобрать необходимую информацию из результата запроса, в том числе и по разному набору ключевых полей. Например, если используется аналитика остатков по складам, в полях запроса и структуры отбора необходимо добавить поле «Склад».

Также для коллекций 1С в обработке результата недоступны методы обращения напрямую по индексу (через оператор квадратных скобок «[...]»). Практически у всех коллекций в 1С для этого есть метод **Получить()**, где в скобках задается числовой индекс элемента в коллекции.

Глава 5. Работа с Web API ELMA из 1С

В системе существует набор корневых сервисов, которые служат для взаимодействия с сервером на уровне SDK. Основные возможности, которые предоставляют сервисы: авторизация, получение метаданных системы, чтение данных сущностей, отслеживание изменений сущностей на сервере, запуск процессов.

Ознакомиться со справкой и другой информацией по Web API ELMA можно на странице **http://<адрес сервера ELMA>:<порт>/API/Help**.

5.1. Запуск процесса из 1С в системе ELMA при оплате заказа

При работе с заказами клиентов помимо статуса товарного обеспечения заказов необходимо отслеживать статус оплаты заказа.

В рамках системы ELMA за статус оплаты отвечает флажок-атрибут **Оплачено** справочника «Заказ» (Рис. 47).

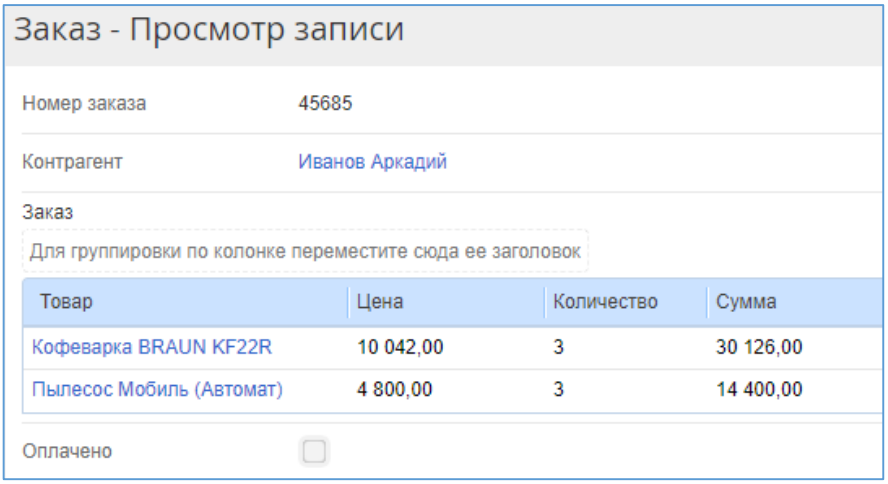


Рис. 47. Признак оплаты у заказа

Один из вариантов отслеживания статуса оплаты – контроль регистров движения денежных средств аналогично контролю регистра товарных движений.

Однако возможен более наглядный и оперативный способ – автоматический запуск процесса в ELMA из 1С. В рамках процесса возможна как автоматическая установка флажка **Оплачено** нужному заказу, так и дополнительный «сигнал» в виде задачи оператору интернет-магазина.

1С позволяет работать со сторонними веб-сервисами при помощи встроенного объекта конфигурации «WS-ссылка». **WS-ссылка 1С** – это описание стороннего веб-сервиса по его WSDL-описанию (его реквизиты и типы данных для дальнейшей интеграции с ним).

В системе ELMA помимо сервисов, описанных в справке по Web API (**http://<адрес сервера ELMA>:<порт>/API/Help**), существует специальный сервис для работы с процессами – **WFPWebService**. Его адрес: **http://<адрес сервера ELMA>:<порт>/API/Help/Modules/EleWise.ELMA.Workflow.Processes.Web/WFPWebService.asmx?WSDL**. Например, в нашем примере сервер ELMA находится по адресу http://localhost. При этом ссылка на описание WSDL для сервиса для работы с процессами имеет следующий вид: **http://localhost/Modules/EleWise.ELMA.Workflow.Processes.Web/WFPWebService.asmx?WSDL**.

Для добавления ее в конфигурацию 1С нажмите правой кнопкой мыши в дереве конфигурации 1С по разделу **WS-ссылки** и из контекстного меню выберите пункт **Добавить**. В открывшемся диалоговом окне (Рис. 48) укажите скопированный выше адрес WSDL.

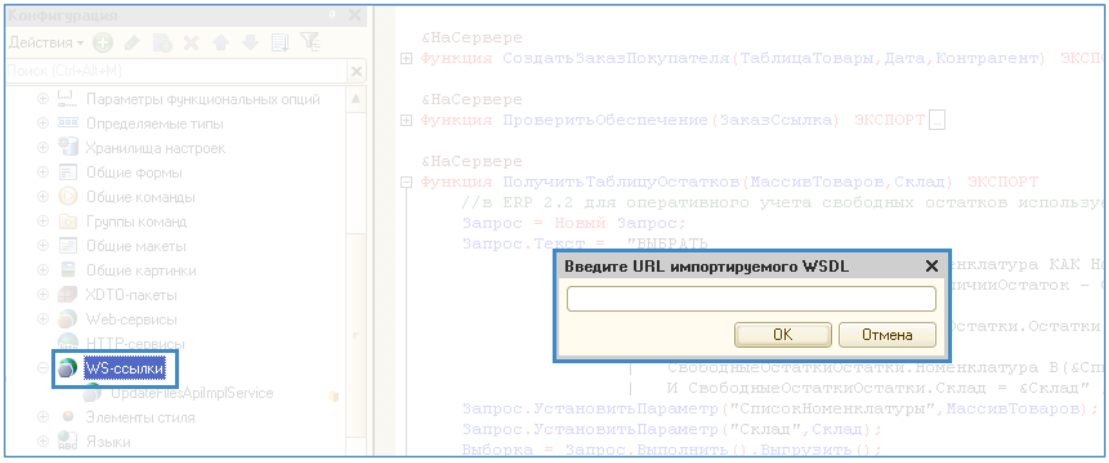


Рис. 48. Добавление WS-ссылки на веб-сервис ELMA в 1С

После добавления дайте ссылке более понятное название, например, **ELMAWorkflow**.

Преимущества данного веб-сервиса в том, что он не требует подстановки специальных заголовков (headers) в тело запроса. При помощи средства работы с WS-ссылками в 1С это сделать невозможно.

В качестве примера реализуем следующий механизм проверки оплаты с заранее определенными условиями и допущениями:

1. В 1С создается платежный документ на основании заказа клиента. Сумма документа всегда равна сумме заказа (механизмы частичной оплаты не используются), также один платеж осуществляется только по одному заказу.
2. При проведении документа в ELMA запускается процесс, на вход которого передаются строки с уникальным идентификатором документа заказа-клиента и документа оплаты в 1С.
3. В сценариях процесса по уникальному идентификатору документа находится сам документ 1С и сопоставленный ему элемент справочника «Заказ» в ELMA.
4. Если в ELMA не существует такого заказа, то процесс завершается, иначе – оператору интернет-магазина назначается задача-уведомление о том, что по заказу получена оплата. Также в задаче выводится информация из документа оплаты 1С.

Для того чтобы отследить событие проведения документа в 1С и при этом не изменять типовые процедуры проведения документа, воспользуемся средством платформы 1С «Подписка на событие» (аналог Listener в ELMA). Для этого:

1. В дереве конфигурации переходим в раздел **Общие – Подписки на события**. Добавляем новую подписку (пункт **Добавить** в контекстном меню раздела).
2. В свойствах подписки указываем необходимые данные (Рис. 49):
    - **Имя** – любое;
    - **Источник** – ДокументОбъект.ПоступлениеБезналичныхДенежныхСредств.
    - **Событие** – «ОбработкаПроведения».
    - **Обработчик** – нужный модуль, где будет размещена процедура-обработчик.

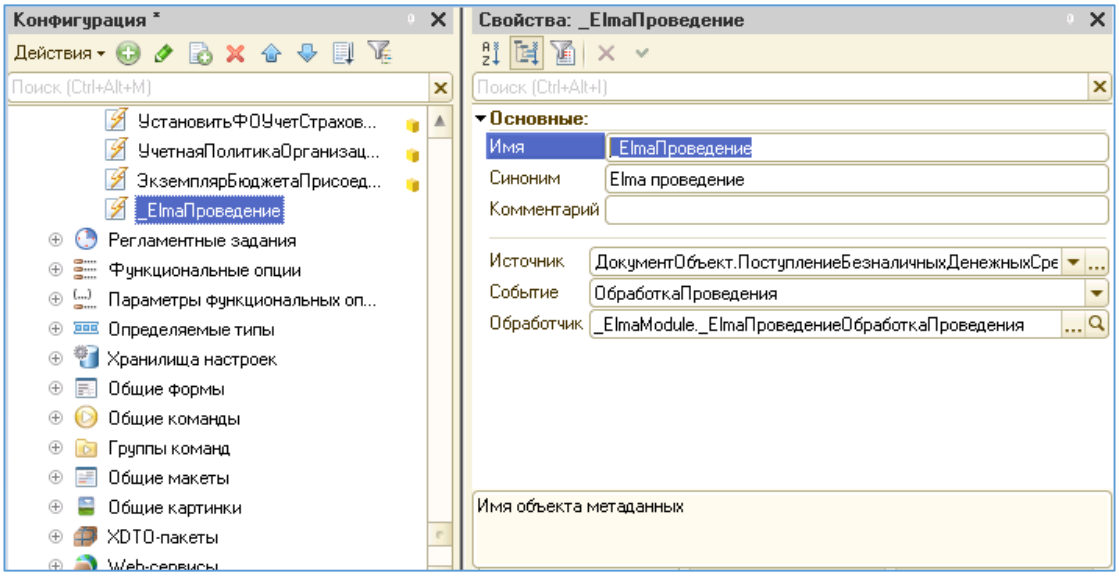


Рис. 49. Настройка подписки на событие в 1С

В пункте 2 подписка может быть создана для различных типов документов, также в разных конфигурациях 1С за платежи отвечают различные типы документов. Данная настройка зависит от того, какими документами осуществляется подтверждение оплаты заказов в учете конкретной организации.

В ELMA необходимо создать процесс и настроить в нем параметры запуска из внешних систем. Для этого в окне редактирования процесса в Дизайнере ELMA перейдите на вкладку **Настройки**, на панели **Варианты запуска процесса** установите флажок **Запуск из внешних систем** и сгенерируйте или введите свой токен процесса (Рис. 50).

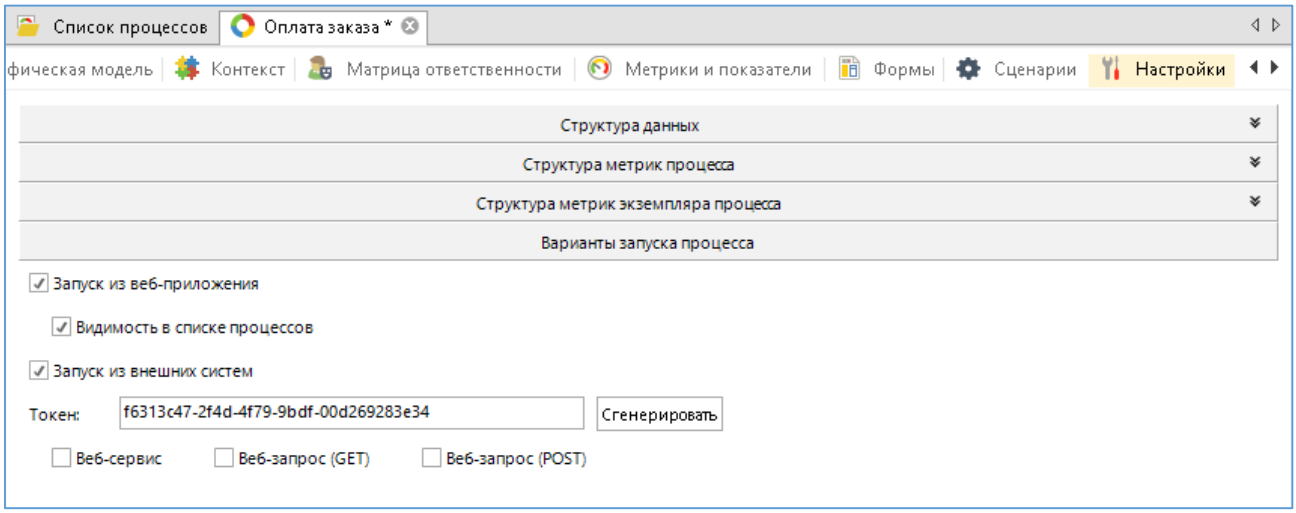


Рис. 50. Настройки запуска процесса из внешних систем

Также в процессе необходимо создать 2 контекстные переменные с типом «Строка». В них будут передаваться строки с уникальными идентификаторами документов в 1С: документа заказа-клиента и документа оплаты. В нашем случае имя свойства данной контекстной переменной «UIDOfDoc1S» (для заказа) и «UIDOfDocPay1S» (для платежного документа). В дальнейшем обе переменные будут использованы в коде вызова веб-сервиса в 1С. Флажок **Входное** устанавливать не обязательно.

При публикации процесса также необходимо установить флажок **Веб-сервис** в настройках (Рис. 51).

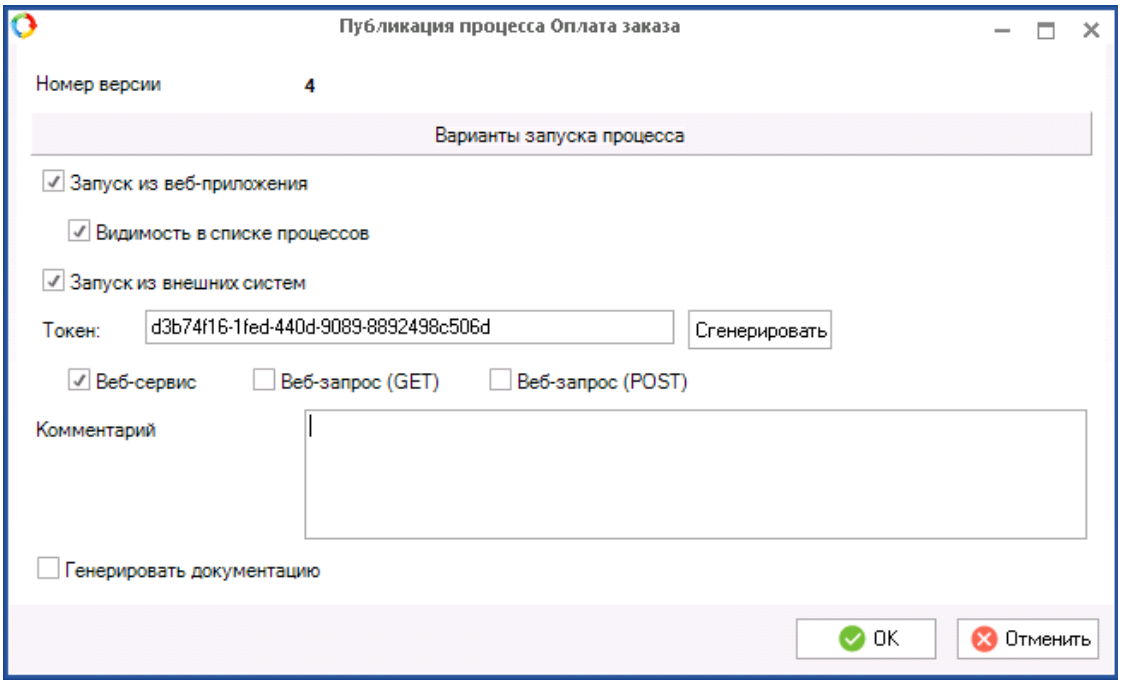


Рис. 51. Дополнительные настройки при публикации процесса

В обработчике подписки проверяются все необходимые условия на соответствие документа нужному типу и запускается процесс в ELMA с передачей всех необходимых параметров.

```
Процедура _ElmaПроведениеОбработкаПроведения(Источник, Отказ,
РежимПроведения) Экспорт
//проверяем, что это "Платежка" нужного нам типа
Если Источник.ТипПлатежногоДокумента =
Перечисления.ТипыПлатежныхДокументов.ПлатежноеПоручение
  И Источник.ХозяйственнаяОперация =
Перечисления.ХозяйственныеОперации.ПоступлениеОплатыОтКлиента Тогда
  //анализируем аналитику платежного поручения по заказам,
  //рассмотрим простой пример - одна платежка сделана по одному
заказу клиента
  Если Источник.РасшифровкаПлатежа.Количество()>0 Тогда
```

```
        Если ЗначениеЗаполнено(Источник.РасшифровкаПлатежа[0].Заказ)
Тогда
            //проверяем также тип заказа, это должен быть
"ЗаказКлиента"
            Если ТипЗнч(Источник.РасшифровкаПлатежа[0].Заказ) =
Тип("ДокументСсылка.ЗаказКлиента") Тогда
                Попытка
                    ИдЭкземпляра =
ЗапуститьПроцессBELMA(Строка(Источник.РасшифровкаПлатежа[0].Заказ.Уни
кальныйИдентификатор()),Источник.РасшифровкаПлатежа[0].Заказ.Номер,Ис
точник.РасшифровкаПлатежа[0].Заказ.Дата,Строка(Источник.Ссылка.Уникал
ьныйИдентификатор()));
                    Сообщить("В ELMA запущен процесс с
Id="+Строка(ИдЭкземпляра));
                Исключение
                    Сообщить(ОписаниеОшибки());
                КонецПопытки;
            КонецЕсли;
        КонецЕсли;
  КонецЕсли;
КонецЕсли;
КонецПроцедуры
```

Из данной процедуры вызывается функция **ЗапуститьПроцессBELMA**.

```
Функция
ЗапуститьПроцессBELMA(УИДДокумента,НомерДокумента,ДатаДокумента,УИДПл
атежногоДокумента)
//инициализация необходимых для подключения переменных
//1. Пространство имен
URI = "http://www.elma-bpm.ru/WFPWebService/";
//2. Другие необходимые параметры
НаименованиеВебСервиса = "WFPWebService";
НаименованиеПорта = "WFPWebServiceSoap";
//используем метод WS-ссылки "СоздатьWSПрокси"
Сервис = WSСсылки.ELMAWorkflow.СоздатьWSПрокси(URI,
НаименованиеВебСервиса, НаименованиеПорта);
//ФабрикаXDTO осуществляет запись данных в пакеты для отправки Веб-
сервису
Фабрика = Сервис.ФабрикаXDTO;
Пакет = Фабрика.Пакеты.Получить(URI);
//получение типов параметров
ТипWebData = Пакет.Получить("WebData");
    ТипWebDataItem = Пакет.Получить("WebDataItem");
  ТипArrayWebDataItem = Пакет.Получить("ArrayOfWebDataItem");
  WebDataItem = Фабрика.Создать(ТипWebDataItem, );
//передаем необходимые параметры на вход процесса
//1. "Name" - это имя свойства контекстной переменной процесса
  WebDataItem.Name = "UIDOfDoc1S";
//2. "Value" - ее значение, возможна передача только примитивных
типов
  WebDataItem.Value = УИДДокумента;
//аналогично создаем еще одну структуру, для передачи УИДа платежного
документа
```

```
WebDataItemPay = Фабрика.Создать(ТипWebDataItem, );
WebDataItemPay.Name = "UIDOfDocPay1S";
WebDataItemPay.Value = УИДПлатежногоДокумента;
//полученные структуры WebDataItem "упаковываем" в тип WebData,
который требуется на входе вызываемого метода Веб-сервиса
  ItemsArr = Фабрика.Создать(ТипArrayWebDataItem, );
  ItemsArr.WebDataItem.Добавить(WebDataItem);
ItemsArr.WebDataItem.Добавить(WebDataItemPay);
  WebData = Фабрика.Создать(ТипWebData, );
  WebData.ItemsArr = ItemsArr;
//другие параметры вызываемого метода – имеют примитивные типы
//учетная запись для запуска процесса, у данной учетной записи должны
быть права на запуск требуемого процесса
UserName = "admin";
  Password = "123";
//Токен процесса – задается на вкладке "Настройки" – "Варианты
запуска процесса" в ELMA
  Token = "d3b74f16-1fed-440d-9089-8892498c506d";
//Задаем имя экземпляра процесса
  ProcessName = "Проведение оплаты заказа в 1С "+НомерДокумента+" от
"+Формат(ДатаДокумента,"ДФ=dd.ММ.уууу");
  //Вызываем метод запуска процесса, он возвращает Id экземпляра
процесса
IdInstance = Сервис.Run(UserName, Password, Token, ProcessName,
WebData); //Метод возвращает Id запущенного процесса
Возврат IdInstance;
КонецФункции
```

В описанной функции запуска бизнес-процесса в ELMA мы заранее определили все необходимые параметры. Однако, ее можно сделать более универсальной (для вызова в других обработчиках в 1С). Например, можно передавать на вход функции токен нужного процесса и структуру параметров для передачи в контекстные переменные процесса. Также можно использовать регистр сведений для хранения паролей с установленным флажком **Режим пароля** для поля, содержащего значение пароля. Это позволит избежать упоминания пароля в открытом виде в коде.

Значения атрибутов пространства имен (URI), наименования веб-сервиса и порта также берутся из данных WS-ссылки. Для этого в дереве конфигурации 1С нажмите правой кнопкой мыши на добавленную WS-ссылку (**ELMAWorkflow**) и выберите пункт **Просмотр**. Будет открыто окно с деревом свойств и методов веб-сервиса (Рис. 52).

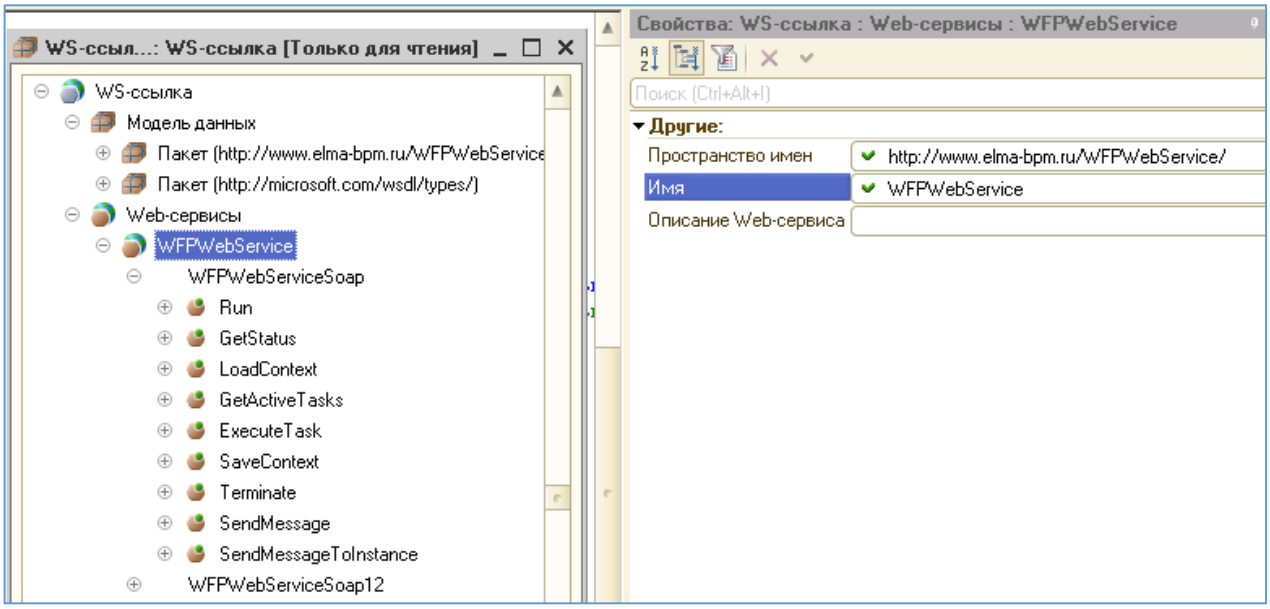


Рис. 52. Просмотр сведений о WS-ссылке в 1С

В поддереве **Модель данных** можно ознакомиться с типами данных для пространства имен. В поддереве **Веб-сервисы** будут представлены имена доступных веб-сервисов и порты каждого из них. В свойствах веб-сервиса есть поле **Пространство имен**. Полученные параметры необходимо использовать при создании нового WS-прокси в 1С. Указанный способ пригоден для работы со всеми WS-ссылками в 1С.

Кроме того, можно динамически обратиться к внешнему веб-сервису, т.е. без добавления объекта WS-ссылка в конфигурацию 1С.

Для этого в описанной выше функции необходимо строку:

```
Сервис = WSСсылки.ELMAWorkflow.СоздатьWSПрокси(URI,
НаименованиеВебСервиса, НаименованиеПорта);
```

Заменить двумя строками:

```
Определения = Новый
WSОпределения("http://localhost/Modules/EleWise.ELMA.Workflow.Process
es.Web/WFPWebService.asmx?WSDL ");
Сервис =  Новый WSПрокси(Определения, URI, НаименованиеВебСервиса,
НаименованиеПорта);
```

Сначала мы создаем так называемые определения для веб-сервиса из его WSDL. Затем также создаем прокси для обращения к нему.

После реализации на стороне 1С готовим необходимый функционал в ELMA. В бизнес-процессе на стороне ELMA требуется реализовать получение необходимых

данных документа и их отображение на форме задачи оператора интернет-магазина. Карта процесса, реализующего необходимую логику, представлена на Рис. 53.

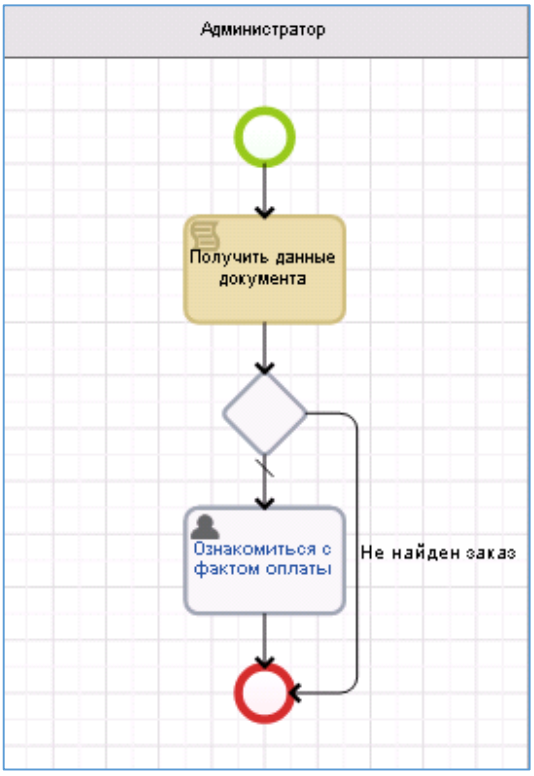


Рис. 53. Процесс обработки оплаты в ELMA

Сценарий «Получить данные документа» осуществляет поиск заказа в ELMA, исходя из уникального идентификатора связанного с ним документа «ЗаказКлиента» в 1С. Также заполняются необходимые переменные контекста с информацией из документа оплаты.

```
public virtual void GetDocInfo (Context context)
{
    //инициализируем подключение
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
    var Manager1C = (dynamic)connector.Reference;
    var provider = service.GetProvider("erp1c");
    //получаем объект типа "УникальныйИдентификатор" из полученной из
1С строки с уникальным идентификатором
    dynamic guidObj = Manager1C.NewObject ("УникальныйИдентификатор",
context.UIDOfDoc1S);
    //получаем ссылку на нужный нам документ ЗаказКлиента
    dynamic link1c = Manager1C.Документы.ЗаказКлиента.ПолучитьСсылку
(guidObj);
    //получаем переменную с типом "Документ 1С", чтобы по ней
выполнить поиск среди заказов в ELMA
    var com1s = new ComObject(link1c);
```

```
    var elmaOrder1S = provider.LoadDocumentByComObject <
EleWise.ELMA.Integration1C.Configs.Erp1c.ZakazKlienta > (com1s);
    context.Order = EntityManager<Order>.Instance.Find(p=>p.Order1S
== elmaOrder1S).FirstOrDefault();
    //если нашли заказ в ELMA – заполняем информацию из документа
оплаты, для вывода на форму и устанавливаем признак оплаты
    if (context.Order!=null)
    {
        context.HaveOrder = true;
        //меняем признак оплаты в документе
        context.Order.Paid = true;
        context.Order.Save();
        //получаем ссылку 1С на документ оплаты
        dynamic guidObjPay = Manager1C.NewObject
("УникальныйИдентификатор", context.UIDOfDocPay1S);
           //получаем ссылку на нужный нам документ оплаты
        dynamic link1cPay =
Manager1C.Документы.ПоступлениеБезналичныхДенежныхСредств.ПолучитьСсы
лку (guidObjPay);
        //заполняем поле с номером и датой документа
        context.DocNumber1S = link1cPay.Номер + " от " +
link1cPay.Дата.ToShortDateString();
        //номер и дата банковской выписки
        context.BankNumber = link1cPay.НомерВходящегоДокумента + " от
" + link1cPay.ДатаВходящегоДокумента.ToShortDateString();
        //комментарий документа
        context.Comment = link1cPay.Комментарий;
    }
    else
    {
        context.HaveOrder = false;
    }
}
```

При проведении документа оплаты в 1С по заказу, который был выгружен в 1С из ELMA, запускается бизнес-процесс. Об успешном запуске будет свидетельствовать сообщение в 1С «Запущен процесс с Id = ». Если по каким-либо причинам (неверные параметры учетной записи или неверный токен процесса) процесс не будет запущен, будет выведен текст ошибки. Форма проведенного документа в 1С представлена на Рис. 54.

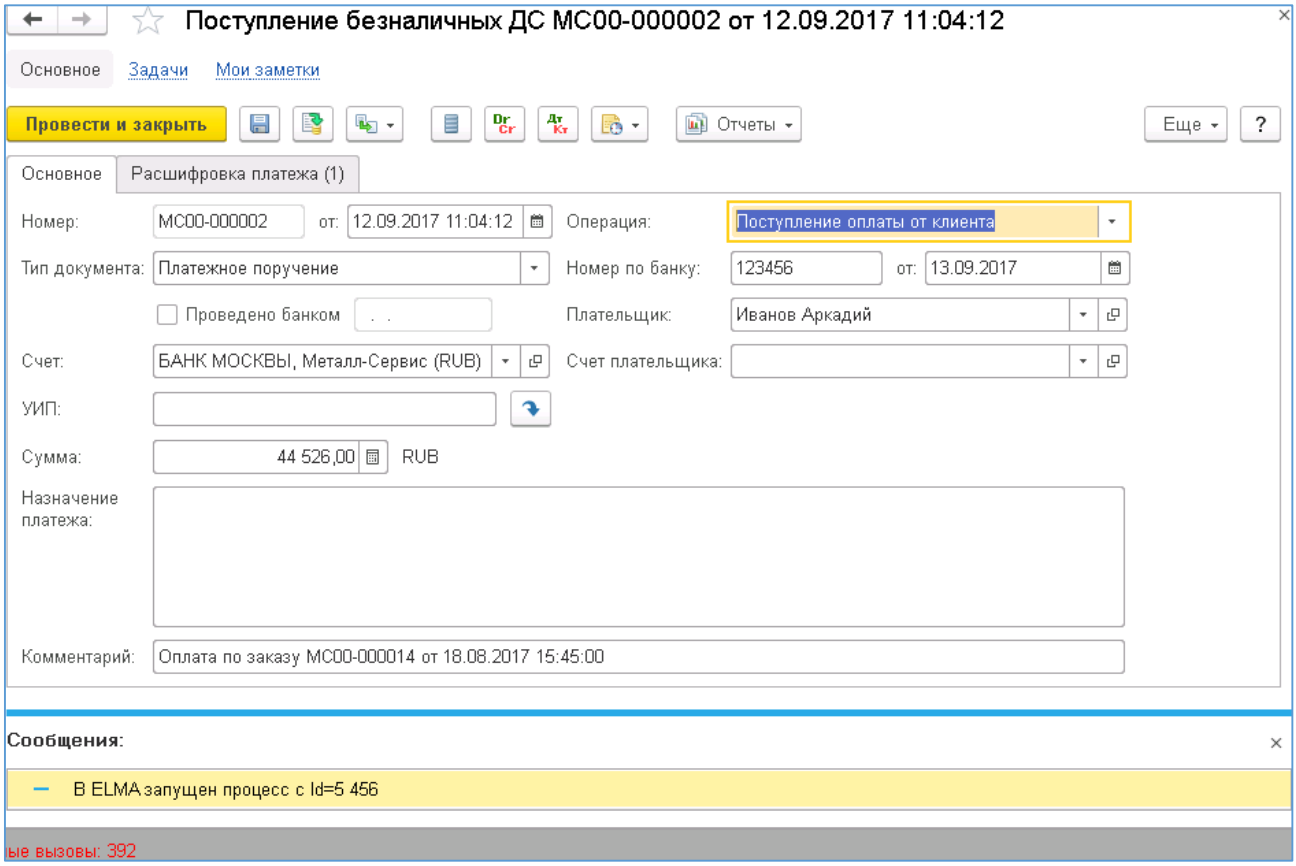


Рис. 54. Проведенный документ оплаты по заказу в 1С

В системе ELMA для оператора интернет-магазина в рамках процесса будет создана задача. В задаче будет указана информация об уникальном идентификаторе заказа-клиента из 1С, а также данные из реквизитов документа оплаты 1С: номер и дата документа, номер и дата банковского документа, комментарий документа (Рис. 55).

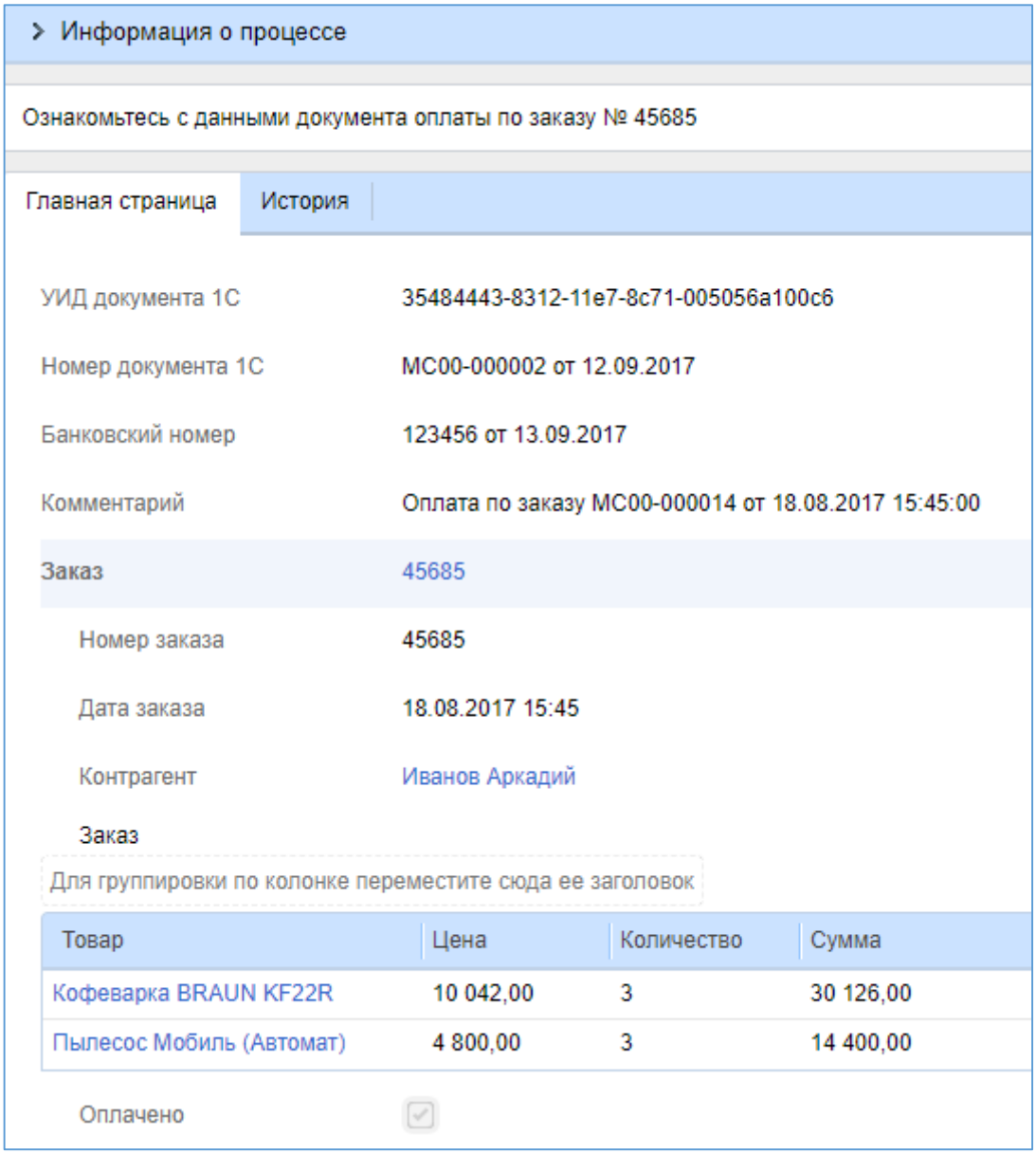


Рис. 55. Задача с информацией об оплате в ELMA

Следует отметить, что данное решение реализовано с некоторыми ограничениями. Однако, описанные механизмы взаимодействия 1С и ELMA, при помощи веб-сервиса работы с процессами позволяют реализовать более масштабируемое решение. Некоторые из возможностей расширения:

1. Контроль изменения документов. При отмене проведения платежного документа возможен запуск иного процесса или этого же процесса с иным набором параметров, влияющих на исполнение процесса.

2. Возможность частичной оплаты заказа. Сумма оплаты является реквизитом документа оплаты (как «Номер» или «Комментарий»), исходя из значения этого реквизита можно указывать, какая часть заказа оплачена.
3. Также необходимо реализовать «защиту» от повторных проведений документа. Это может быть регистр сведений в 1С, где будут храниться, например, соответствия Id запущенных экземпляров процесса в ELMA и уже проведенных документов оплаты в 1С.
4. Возможность проведения одного платежного поручения по многим заказам. На вход процесса можно подавать список уникальных идентификаторов документа, разделенных, например, точкой с запятой. Для каждого из таких UID можно выполнить отдельное получение информации.
5. Также можно реализовать аналогичное решение для оперативного учета обеспечения потребностей по заказам, привязав запуск определенного процесса к проведению документа поступления товаров.

5.2. Запись контрагента из 1С в систему ELMA при помощи веб-сервиса

При использовании в организации двух различных информационных систем часто встает вопрос об обеспечении идентичности данных в обоих системах.

Один из распространенных примеров – синхронизация клиентов между учетной системой и какой-либо специализированной CRM.

В системе ELMA существует дополнительное приложение **ELMA CRM+**, предназначенное для максимальной автоматизации процесса продаж в компании.

Рассмотрим пример, когда данное приложение используется для работы с уже существующими клиентами компании. Клиенты – юридические лица. Первичной системой, в которую заносят контрагентов, является 1С. В системе ELMA осуществляются операции взаимодействия с клиентом (встречи, звонки, учет статусов сделок). Для того чтобы обеспечить полное использование возможностей CRM, необходимо в ELMA создавать контрагента с атрибутами, аналогичными атрибутам контрагента в 1С.

Для этого можно использовать сервисный процесс в ELMA, создающий новый объект «Юридическое лицо» и получающий данные из реквизитов объекта «Контрагент» в 1С с использованием COMConnector. Однако, мы рассмотрим пример использования Web API ELMA.

В этом случае не требуется никакой дополнительной разработки в процессах и модулях ELMA. Также этот способ пригоден для передачи большого объема данных.

Перед разработкой на стороне 1С необходимо провести подготовительные мероприятия: ознакомиться с особенностями работы и передачи данных с использованием веб-сервисов.

Для работы с сущностями используется специальный веб-сервис **IEntityService**. Информацию об этом и других сервисах можно получить на странице **http://<адрес сервера ELMA>:<порт>/API/Help/Services** (Рис. 57). У данного сервиса имеется WSDL-описание, однако работать в 1С с WS-ссылкой на основании этого описания не получится. Это связано с тем, что при работе с объектом «WSПрокси» в 1С нет возможности изменять структуру заголовков (headers) у отправляемого запроса. Поэтому работа будет осуществляться с использованием веб-запросов по правилам REST-систем.

Получить описание доступных методов с их адресами вызовов можно по ссылке **Перейти к описанию REST запросов к методам** в списке доступных веб-сервисов (Рис. 57).

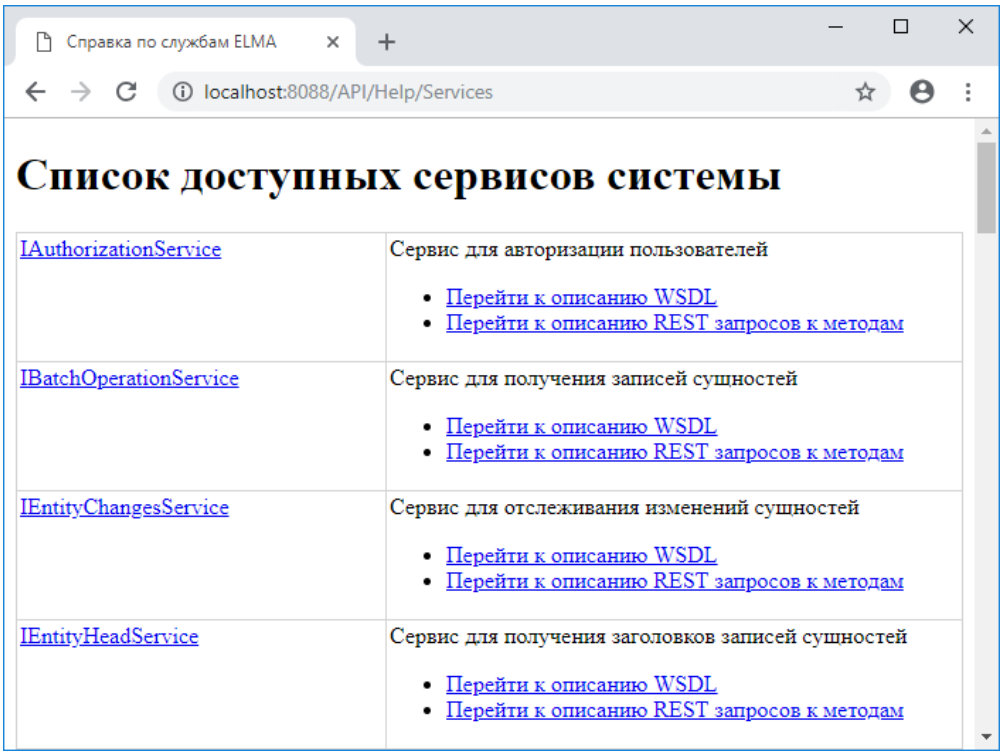


Рис. 56. Список доступных веб-сервисов системы ELMA

При этом откроется список доступных Uri методов сервиса с описанием (Рис. 57).

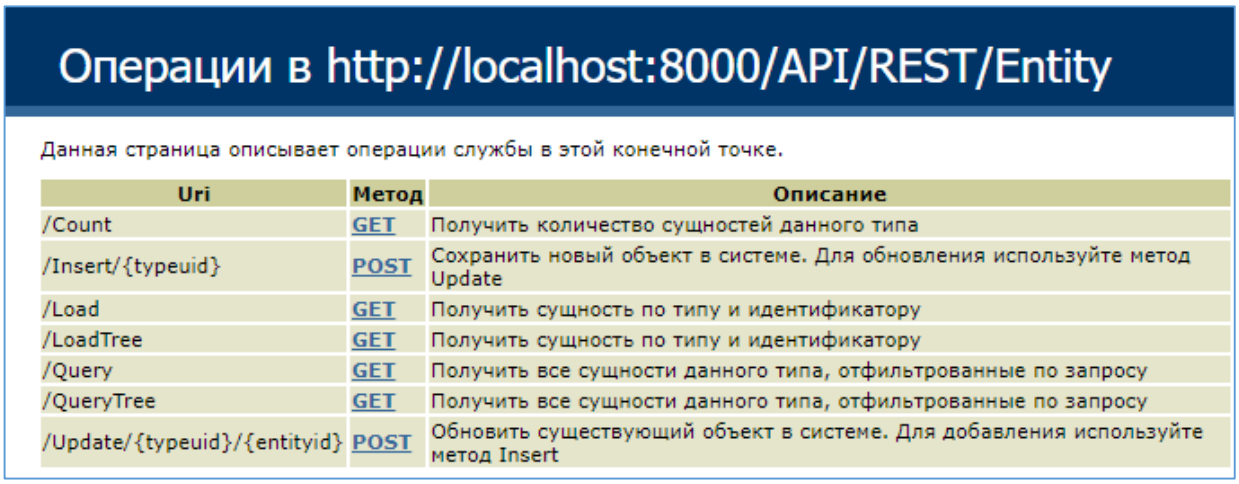
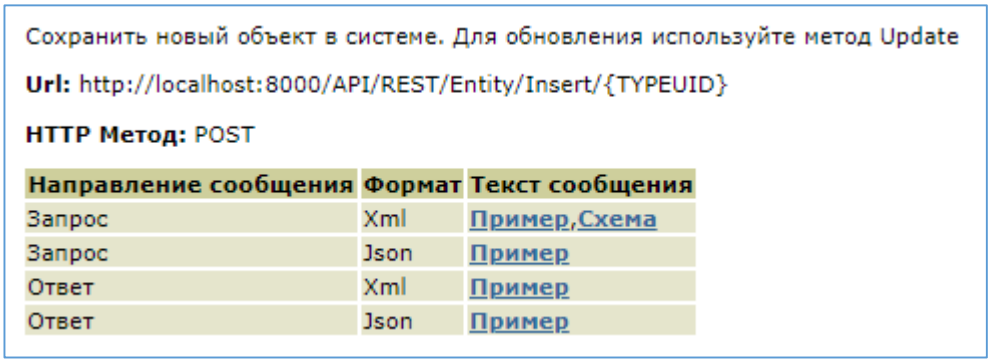


Рис. 57. Информация о доступных методах выбранного веб-сервиса

Если открыть страницу по ссылке в колонке **Метод** (напротив нужного нам метода), будет представлена подробная информация о том, по какому адресу

вызывается метод, какой структуры он должен содержать тело запроса и какая будет структура ответа (Рис. 58).


Сохранить новый объект в системе. Для обновления используйте метод Update

Url: http://localhost:8000/API/REST/Entity/Insert/{TYPEUID}

HTTP Метод: POST

Направление сообщения	Формат	Текст сообщения
Запрос	Xml	Пример,Схема
Запрос	Json	Пример
Ответ	Xml	Пример
Ответ	Json	Пример

Рис. 58. Подробная информация о методе веб-запроса

Более подробную общую информацию о Web API ELMA можно найти в соответствующей статье Базы знаний ELMA.

В первую очередь, для того чтобы 1С выступала в качестве доверенного приложения для ELMA, необходимо сгенерировать для нее токен. Для этого нужно в веб-приложении ELMA перейти в раздел **Администрирование – Система – Внешние приложения** и в верхнем меню страницы нажать на кнопку **Добавить**. На открывшейся странице (Рис. 59) необходимо заполнить требуемые данные и нажать на кнопку **Сохранить**. Они никак не влияют на приложение и носят только информационный характер.

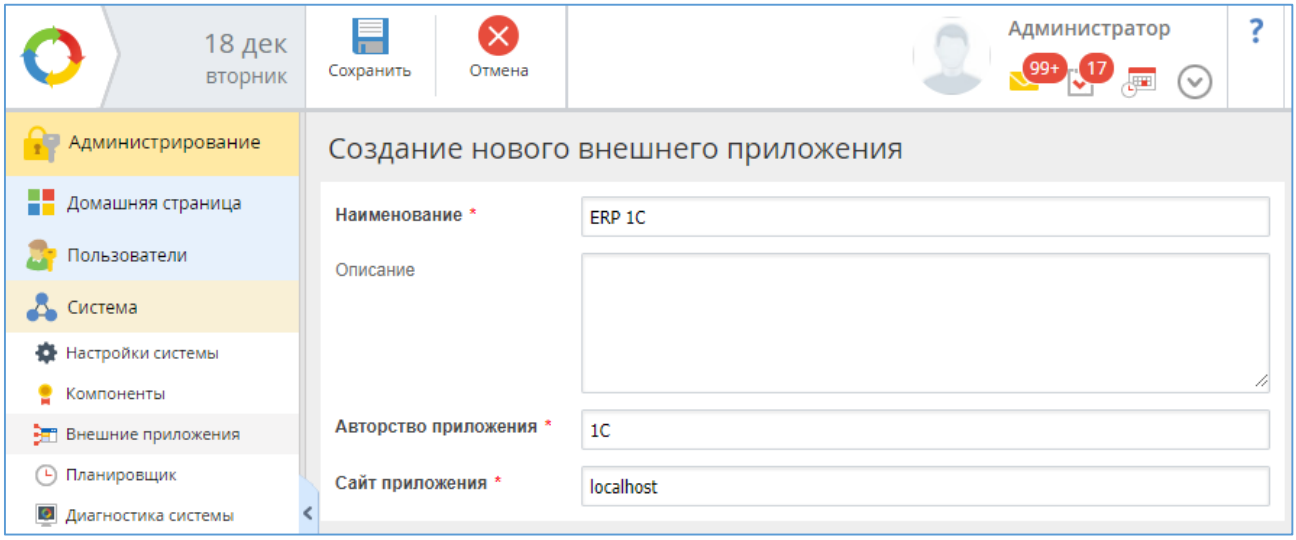


Рис. 59. Добавление нового внешнего приложения для генерации токена

После сохранения информации о новом внешнем приложении необходимо перейти к его карточке (нажатием на его название в списке приложений). В карточке приложения перейдите на вкладку **Токены приложения** (Рис. 60).

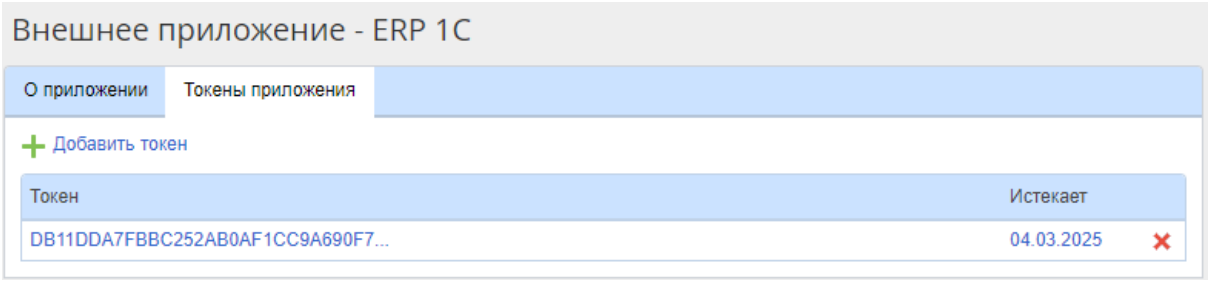


Рис. 60. Информация о токенах внешнего приложения

В столбце **Токен** при нажатии на строку с токеном он будет представлен полностью. В дальнейшем его необходимо скопировать в заголовки формируемого веб-запроса. При нажатии на строку в столбце **Истекает** можно задать новый срок действия токена.

Далее приступаем к разработке на стороне 1С. Для наглядности разработаем внешнюю обработку 1С, в которой необходимо выбрать контрагента. По нажатию кнопки он будет выгружен в ELMA. Также предусмотрим, что, если контрагент в ELMA уже существует, будет происходить обновление информации в существующем контрагенте, а не создание нового. Ключевыми полями для поиска определим поля «Наименование» и «ИНН» у контрагента.

В целом для реализации наших задач необходимо использовать 2 различных веб-сервиса, которые предоставляет ELMA:

1. Сервис для авторизации пользователей **IAuthorizationService** – используется для получения токена авторизации, который используется в заголовках отправляемых веб-запросов.
2. Сервис для получения записей сущностей **IEntityService** – используется для поиска сущностей по фильтру и сохранения/обновления сущности в ELMA.

Все описания методов и функций можно получить на странице справки по веб-сервисам ELMA **http://<адрес сервера ELMA>:<порт>/API/Help** (Рис. 56 – Рис. 58). Кроме того, для идентификации объектов и типов используются такие атрибуты, как «Идентификатор типа» (свойство TypeUid) и «Идентификатор элемента» (свойство Id). С информацией об атрибуте **TypeUid** можно ознакомиться на странице ELMA **http://<адрес сервера ELMA>:<порт>/API/Help/Types** (Рис. 61).

Идентификатор типа объекта: 3325eab1-fe46-4900-a617-c6fb54ac24c0

Юридическое лицо

Id	Идентификатор объекта (Int64)
TypeUid	Идентификатор типа объекта (Guid)
MarketingGroup	Группа мероприятий. (тип: Группа маркетинговых мероприятий (Объект))
	Id — Значение идентификатора сущности
	Name — Строковое представление сущности
	Uid — Уникальный идентификатор сущности
MarketingActivity	Маркетинговое мероприятие. (тип: Маркетинговое мероприятие (Объект))
	Id — Значение идентификатора сущности
	Name — Строковое представление сущности
	Uid — Уникальный идентификатор сущности
Uid	Уникальный идентификатор. (тип: Уникальный идентификатор (GUID))
Name	Наименование. (тип: Строка)
Type	Тип. (тип: Тип клиента (Объект))
	Id — Значение идентификатора сущности
	Name — Строковое представление сущности
	Uid — Уникальный идентификатор сущности

Рис. 61. Информация о TypeUid нужного типа данных

Информацию об Id элемента обычно можно взять из адресной строки браузера, открыв карточку нужного элемента в ELMA (Рис. 62).

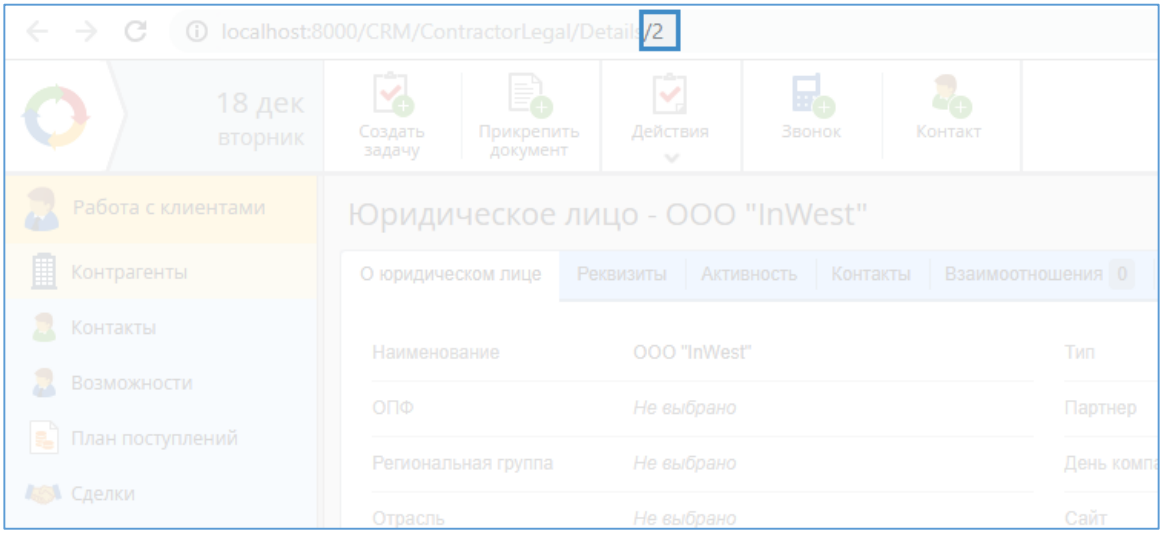


Рис. 62. Просмотр Id элемента в строке браузера

Веб-сервисы ELMA могут отправлять/принимать данные в формате JSON или XML. Применение того или иного формата указывается в заголовках передаваемых веб-запросов. В целом, в платформе 1С существуют средства для работы как с JSON, так и с XML. Поэтому выбор формата для использования в каждом конкретном случае

остается на усмотрение разработчика решения по обмену информацией. В примере мы рассмотрим обмен с использованием JSON.

Набор данных содержится в структуре «тела» веб-запроса или ответа. Для каждого из методов необходимую структуру данных можно посмотреть на странице подробного описания метода веб-сервиса (ранее по тексту – Рис. 58). Например, для метода записи нового объекта (Insert) будет указан такой текст:

```
{
"Items":[{
  "Data":{
        "Items":[{
              "Data":{
                    "Items":null,
                    "Value":"Строковое содержимое"
              },
              "DataArray":[{
                    "Items":null,
                    "Value":"Строковое содержимое"
              }],
              "Name":"Строковое содержимое",
              "Value":"Строковое содержимое"
        }],
        "Value":"Строковое содержимое"
  },
  "DataArray":[{
        "Items":[{
              "Data":{
                    "Items":null,
                    "Value":"Строковое содержимое"
              },
              "DataArray":[{
                    "Items":null,
                    "Value":"Строковое содержимое"
              }],
              "Name":"Строковое содержимое",
              "Value":"Строковое содержимое"
        }],
        "Value":"Строковое содержимое"
  }],
  "Name":"Строковое содержимое",
  "Value":"Строковое содержимое"
}],
"Value":"Строковое содержимое"
}
```

Подобная структура в текстовом виде очень сложна для восприятия, поэтому разумно преобразовать ее в более понятное и читаемое дерево. Сделать это можно при помощи разнообразных сервисов конвертации JSON, например, http://jsoneditoronline.org/ (Рис. 63).

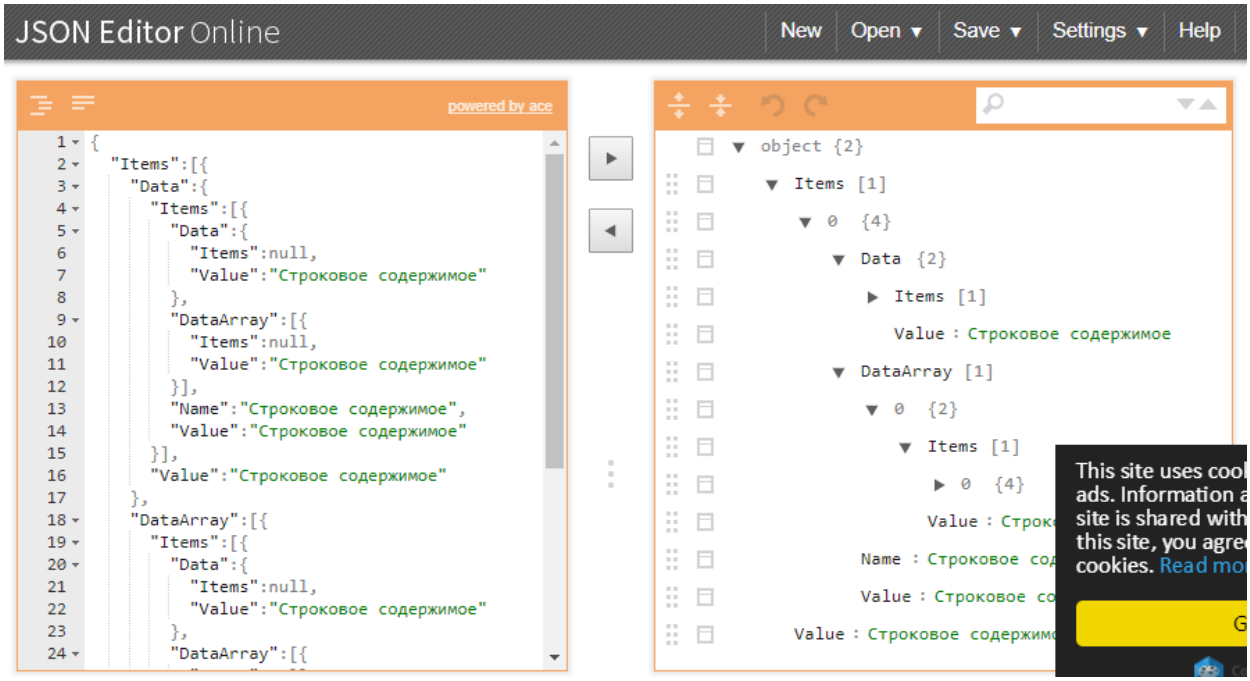


Рис. 63. Сервис конвертации JSON

В левой части окна – исходный текст в формате JSON, в правой – структура (дерево), которую формирует данный текст.

При разработке возможны две ситуации: когда необходимо принять ответ веб-сервиса в виде JSON и когда необходимо отправить текст JSON в теле веб-запроса. В платформе 1С существует метод автоматической десериализации JSON – представления данных в виде структуры (дерева), аналогично той, которую можно получить при помощи сервиса конвертации JSON. С этой структурой можно работать аналогично работе с другими объектами типа «Структура 1С». Запись в формат JSON из 1С также возможна при помощи автоматического сериализатора, однако формат этих данных не подходит для передачи в ELMA, там нужна особая иерархия. Поэтому, для записи мы воспользуемся методом потоковой (последовательной) записи. В этом случае мы формируем структуру передачи последовательно, начиная с самого верхнего уровня.

Для удобства чтения кода все повторяющиеся операции максимально разнесены по различным процедурам и функциям. Более подробные описания каждой операции представлены в комментариях.

```
&НаКлиенте
Процедура ВызовELMA(Команда)
//обработка нажатия кнопки на форме
ВызовELMAНаСервере();
КонецПроцедуры
```

```
&НаСервере
Процедура ВызовELMAНаСервере()
//указываем адрес сервера (строка)
Сервер = "127.0.0.1";
//и порт (число)
Порт = 8088;
СоединениеELMA = Новый HTTPСоединение(Сервер,Число(Порт));
//указываем все предопределенные данные
//1. Идентификатор типа объекта "Юридическое лицо"
УИДТипаЮрЛицо = "3325eab1-fe46-4900-a617-c6fb54ac24c0";
//2. Токен приложения, сгенерированный в ELMA
    ТокенПриложения =
"DB11DDA7FBBC252AB0AF1CC9A690F72C371350A6B2DBD4D8701DC00730AB967D7A17
E97278EB18ADC9B1FE071D0DD5299E4108B556C420EC52176FEDC1915651";
//3. Переменные для обработки ошибок
ВозниклаОшибка = ЛОЖЬ;
ТекстОшибки = "";
//4. Данные для авторизации
Логин = "admin";
Пароль = "123";
//5. Для заголовков запроса – получаем токен авторизации из логина и
пароля
    Токен =
ПолучитьТокенАвторизации(Логин,Пароль,СоединениеELMA,ТокенПриложения,
ВозниклаОшибка,ТекстОшибки);
//готовим структуру с примитивными типами для преобразования ее в
JSON
СтруктураПередачи = ПодготовитьСтруктуруДанных(Контрагент);
//в атрибутах создаваемого объекта есть также сложные типы – ссылки
на другие объекты
//для них необходимо подготовить отдельную таблицу с полями
ТаблицаССылок = ПодготовитьТаблицуСложныхПараметров(Контрагент);
Если НЕ(ВозниклаОшибка) Тогда
  ОтветСервера = ЗаписатьОбновитьСущность(СтруктураПередачи,
ТаблицаССылок, УИДТипаЮрЛицо, СоединениеELMA, ТокенПриложения, Токен,
ВозниклаОшибка,ТекстОшибки);
Иначе
  ОтветСервера = ТекстОшибки;
КонецЕсли;
КонецПроцедуры

&НаСервере
Функция
ПолучитьТокенАвторизации(Логин,Пароль,СоединениеELMA,ApplicationToken
,ВозниклаОшибка = ЛОЖЬ,ТекстОшибки="")
//получаем через веб-сервис токен авторизации пользователя для
использования в других веб-сервисах
АдресВебАвторизации =
"/API/REST/Authorization/LoginWith?username="+Строка(Логин);
Заголовки = Новый Соответствие;
Заголовки.Вставить("ApplicationToken",ApplicationToken);
//указываем, что данные передаем/получаем в формате JSON
```

```
Заголовки.Вставить("Content-type","application/json");
ХттпЗапрос = Новый HTTPЗапрос(АдресВебАвторизации,Заголовки);
//в теле запроса должна быть строка с паролем
ХттпЗапрос.УстановитьТелоИзСтроки(Строка(Пароль),"UTF-8");
Попытка
  Результат = СоединениеELMA.ОтправитьДляОбработки(ХттпЗапрос);
  ТелоЗапроса = Результат.ПолучитьТелоКакСтроку("UTF-8");
    ЧтениеJSON = Новый ЧтениеJSON;
    ЧтениеJSON.УстановитьСтроку(ТелоЗапроса);
  //осуществляем сразу чтение всего массива в структуру
  СтруктураОтвета = ПрочитатьJSON(ЧтениеJSON);
  Если СтруктураОтвета.Свойство("AuthToken") Тогда
        Возврат СтруктураОтвета.AuthToken;
  Иначе
        ВозниклаОшибка = ИСТИНА;
        ТекстОшибки = "Не удалось обнаружить свойство AuthToken в
ответе";
  КОнецЕсли
Исключение
  ВозниклаОшибка = ИСТИНА;
  ТекстОшибки = ОписаниеОшибки();
КонецПопытки
КонецФункции

&НаСервере
Функция ЗаписатьОбновитьСущность(СтруктураПередачи,
ТаблицаСложныхПараметров, УИДТипа, СоединениеELMA, ApplicationToken,
Authtoken, ВозниклаОшибка = ЛОЖЬ,ТекстОшибки="")
//переменные для определения поведения при отправке данных
ТолькоОбновить = Ложь;
ИДНайденного = "";
//сначала необходимо определить – существует ли уже данный контрагент
в ELMA
//для этого разумно использовать какое-либо дополнительное поле в 1С,
которое будет содержать ИД элемента в ELMA
//или же можно использовать связку каких-либо "уникальных" полей, по
которым можно однозначно идентифицировать Юридическое лицо
//в нашем примере будем использовать ИНН+Наименование

//инициалицируем необходимые для веб-запроса заголовки
Заголовки = Новый Соответствие;
Заголовки.Вставить("ApplicationToken",ApplicationToken);
//указываем, что данные передаем/получаем в формате JSON
Заголовки.Вставить("Content-type","application/json");
Заголовки.Вставить("AuthToken",Authtoken);
//выполним веб-запрос метода
/API/REST/Entity/Query?type={TYPEUID}&q={EQLQUERY}
//сформируем строку EQL-запроса
СтрокаEQL = "Name = '"+СтруктураПередачи.Name+"' AND INN =
'"+СтруктураПередачи.INN+"'";
АдресВебВыполнитьПоиск =
"/API/REST/Entity/Query?type="+УИДТипа+"&amp;q="+СтрокаEQL;
//создаем веб-запрос
```

```
ХттпЗапросПоиск = Новый HTTPЗапрос(АдресВебВыполнитьПоиск,Заголовки);
//Выполняем GET-запрос (Получить)
Попытка
  РезультатПоиск = СоединениеELMA.Получить(ХттпЗапросПоиск);
Исключение
  ВозниклаОшибка = ИСТИНА;
  ТекстОшибки = "Не удалось выполнить запрос поиска
"+ОписаниеОшибки();
  Возврат Неопределено;
КонецПопытки;
//разбираем полученный результат поиска
ТелоЗапросаПоиска = РезультатПоиск.ПолучитьТелоКакСтроку("UTF-8");
ЧтениеJSON = Новый ЧтениеJSON;
ЧтениеJSON.УстановитьСтроку(ТелоЗапросаПоиска);
//осуществляем сразу чтение всего массива в структуру
//структура представляет собой преобразованное "дерево" JSON в массив
других структур и элементов
//в ней будут данные по всем атрибутам всех найденных объектов
СтруктураОтвета = ПрочитатьJSON(ЧтениеJSON);
//т.к. мы знаем состав структуры (заранее смотрим его через отладку)
- мы можем сразу "извлечь" нужное значение
//если в структуре ответа нет данных - значит элементы по указанному
фильтру не найдены
Если СтруктураОтвета.Количество() = 0 Тогда
  ТолькоОбновить = Ложь;
Иначе
    ТолькоОбновить = Истина;
  //СтруктураОтвета[0] - в примере берем первый найденный элемент
  //Items[0] - в первом его свойстве ВСЕГДА будет Id
  //Value - в этом поле содержится его значение
  ИДНайденного = СтруктураОтвета[0].Items[0].Value;
КонецЕсли;
//необходима потоковая запись данных в JSON посылку по формату
/API/REST/Entity/help/operations/Insert#request-json
ТекстТела = ГенерироватьJSON(СтруктураПередачи,
ТаблицаСложныхПараметров);
//адреса сервисов по записи и обновлению сущности немного отличаются
Если ТолькоОбновить Тогда
  АдресВебЗаписатьОбновитьСущность =
"/API/REST/Entity/Update/"+СокрЛП(УИДТипа)+"/"+СокрЛП(ИДНайденного);
Иначе
  АдресВебЗаписатьОбновитьСущность =
"/API/REST/Entity/Insert/"+СокрЛП(УИДТипа);
КонецЕсли;
ХттпЗапросСоздатьОбновить = Новый
HTTPЗапрос(АдресВебЗаписатьОбновитьСущность,Заголовки);
ХттпЗапросСоздатьОбновить.УстановитьТелоИзСтроки(ТекстТела,"UTF-8");
//выполнение веб-запроса POST это метод "ОтправитьДляОбработки"
РезультатСоздатьОбновить =
СоединениеELMA.ОтправитьДляОбработки(ХттпЗапросСоздатьОбновить);
Возврат ?(ТолькоОбновить,"Обновлен ","Записан ")+" контрагент-
юридическое лицо с
Id="+РезультатСоздатьОбновить.ПолучитьТелоКакСТроку();
```

```
КонецФункции

&НаСервере
Функция ГенерироватьJSON(СтруктураПередачи, ТаблицаСложныхПараметров)
//функция подготовки текста-JSON из исходных данных о конрагенте
Запись = Новый ЗаписьJSON;
Запись.УстановитьСтроку();
//"Items":[{ - корневой объект
Запись.ЗаписатьНачалоОбъекта();
Запись.ЗаписатьИмяСвойства("Items");
//Items - массив реквизитов записываемого объекта
Запись.ЗаписатьНачалоМассива();
    //в каждом объекте - отдельный реквизит
//все реквизиты простых типов заранее сохранены в структуру,
//ключ - название поля (имя свойства у объекта ELMA), а значение -
его значение
Для Каждого КлючИЗначение из СтруктураПередачи Цикл

  ЗаписатьПростоеСвойство(Запись,КлючИЗначение.Ключ,КлючИЗначение.Зна
чение);
КонецЦикла;
//после простых типов обрабатываем таблицу со сложными данными
Для Каждого СтрТаблицы из ТаблицаСложныхПараметров Цикл
  ЗаписатьСложноеСвойство(Запись,СтрТаблицы);
КонецЦикла;
//}], завершаем запись массива атрибутов
Запись.ЗаписатьКонецМассива();
//это просто необходимое свойство в корневом объекте
Запись.ЗаписатьИмяСвойства("Value");
//"Неопределено" дает в итоговом JSON - Null
Запись.ЗаписатьЗначение(Неопределено);
Запись.ЗаписатьКонецОбъекта();
Результат = Запись.Закрыть();
//результат - текст с JSON
Возврат Результат;
КонецФункции

//реализуем запись данных в JSON в виде отдельных процедур
//запись простых типов и запись ссылочных типов отличается, поэтому
воспользуемся различными процедурами
&НаСервере
Процедура
ЗаписатьПростоеСвойство(ЗаписываемыйJSON,ИмяСвойства,ЗначениеСвойства
)
//Если ТипЗнч(ЗначениеСвойства) <> Тип("Дата") Тогда
  //для простых свойств в каждом из объектов массива реквизитов -
достаточно заполнять только поля Name и Value
  ЗаписываемыйJSON.ЗаписатьНачалоОбъекта();
  //объект должен содержать 4 свойства
  ЗаписываемыйJSON.ЗаписатьИмяСвойства("Data");
  ЗаписываемыйJSON.ЗаписатьЗначение(Неопределено);

  ЗаписываемыйJSON.ЗаписатьИмяСвойства("DataArray");
```

```
  ЗаписываемыйJSON.ЗаписатьНачалоМассива();
  ЗаписываемыйJSON.ЗаписатьКонецМассива();

  ЗаписываемыйJSON.ЗаписатьИмяСвойства("Name");
  ЗаписываемыйJSON.ЗаписатьЗначение(ИмяСвойства);

  ЗаписываемыйJSON.ЗаписатьИмяСвойства("Value");
  //тип "Дата" необходимо записывать в формате MM/dd/yyyy HH:mm:ss
  Если ТипЗнч(ЗначениеСвойства) = Тип("Дата") Тогда

  ЗаписываемыйJSON.ЗаписатьЗначение(Формат(ЗначениеСвойства,"ДФ='MM/d
d/yyyy HH:mm:ss'"));
  Иначе
        ЗаписываемыйJSON.ЗаписатьЗначение(Строка(ЗначениеСвойства));
  КонецЕсли;

  ЗаписываемыйJSON.ЗаписатьКонецОбъекта();
//КОнецЕсли;
КонецПроцедуры

&НаСервере
Процедура ЗаписатьСложноеСвойство(ЗаписываемыйJSON,СтрокаТаблицы)
ЗаписываемыйJSON.ЗаписатьНачалоОбъекта();
  //в атрибутах с ссылочными типами необходимо заполнять свойство
"Data"
    ЗаписываемыйJSON.ЗаписатьИмяСвойства("Data");
  ЗаписываемыйJSON.ЗаписатьНачалоОбъекта();
  //в объекте Data - два свойства, как в корневом -
  //массив с данными объекта - значения атрибута
  //в массиве - один объект это один атрибут
    ЗаписываемыйJSON.ЗаписатьИмяСвойства("Items");
        ЗаписываемыйJSON.ЗаписатьНачалоМассива();
        //Имена свойств у нас заранее определены, а значения
содержатся в параметре "СтрокаТаблицы"
        //при этом объект также должен содержать 4 свойства
аналогичные простому типу

  ЗаписатьПростоеСвойство(ЗаписываемыйJSON,"Id",СтрокаТаблицы.Id);

  ЗаписатьПростоеСвойство(ЗаписываемыйJSON,"TypeUid",СтрокаТаблицы.Ty
peUid);
        ЗаписываемыйJSON.ЗаписатьКонецМассива();
  //ЗаписываемыйJSON.ЗаписатьКонецОбъекта();
  //и свойство Value=Null
  ЗаписываемыйJSON.ЗаписатьИмяСвойства("Value");
    ЗаписываемыйJSON.ЗаписатьЗначение(Неопределено);
    ЗаписываемыйJSON.ЗаписатьКонецОбъекта();

  ЗаписываемыйJSON.ЗаписатьИмяСвойства("DataArray");
  ЗаписываемыйJSON.ЗаписатьНачалоМассива();
  ЗаписываемыйJSON.ЗаписатьКонецМассива();

    ЗаписываемыйJSON.ЗаписатьИмяСвойства("Name");
```

```
  ЗаписываемыйJSON.ЗаписатьЗначение(СтрокаТаблицы.ИмяСвойства);
  //свойство "Value" не заполняется
  ЗаписываемыйJSON.ЗаписатьИмяСвойства("Value");
    ЗаписываемыйJSON.ЗаписатьЗначение(Неопределено);
    ЗаписываемыйJSON.ЗаписатьКонецОбъекта();
КонецПроцедуры

&НаСервере
Функция ПодготовитьСтруктуруДанных(Контрагент)
//выполняем запрос данных по контрагенту
Запрос = Новый Запрос;
Запрос.Текст = "ВЫБРАТЬ
               |  Контрагенты.Наименование КАК Name,
               |  Контрагенты.ИНН КАК INN,
               |  Контрагенты.КПП КАК KPP,
               |  БанковскиеСчетаКонтрагентов.Банк.Наименование КАК
BANK,
               |  БанковскиеСчетаКонтрагентов.Банк.Код КАК BIK,
               |  БанковскиеСчетаКонтрагентов.Банк.КоррСчет КАК KS,
               |  БанковскиеСчетаКонтрагентов.НомерСчета КАК RS,
               |  ""Экспортирован из 1С"" КАК Description,
               |  &amp;ТекущаяДата КАК CreationDate,
               |  &amp;ТекущаяДата КАК ChangeDate
               |ИЗ
               |  Справочник.Контрагенты КАК Контрагенты
               |       ЛЕВОЕ СОЕДИНЕНИЕ
Справочник.БанковскиеСчетаКонтрагентов КАК
БанковскиеСчетаКонтрагентов
               |       ПО Контрагенты.Ссылка =
БанковскиеСчетаКонтрагентов.Владелец
               |ГДЕ
               |  Контрагенты.Ссылка = &amp;Контрагент";
Запрос.УстановитьПараметр("Контрагент",Контрагент);
Запрос.УстановитьПараметр("ТекущаяДата",ТекущаяДата());
Результат = Запрос.Выполнить().Выгрузить();
Если Результат.Количество() > 0 Тогда
  //в результате всегда будет одна строка, преобразуем ее в
структуру,
  //где ключами будут имена колонок таблицы-результата
  СтруктураВозврата = Новый Структура;
  Для Каждого Колонка из Результат.Колонки Цикл

  СтруктураВозврата.Вставить(Колонка.Имя,Результат[0][Колонка.Имя]);
  КонецЦикла;
  Возврат СтруктураВозврата;
Иначе
  Возврат Неопределено;
КонецЕсли;
КонецФункции

&НаСервере
Функция ПодготовитьТаблицуСложныхПараметров(Контрагент)
//параметр контрагента не используется в примере
```

```
//добавлен для сохранения "однообразия" кода
Таблица = Новый ТаблицаЗначений;
//имя свойства атрибута создаваемого объекта
Таблица.Колонки.Добавить("ИмяСвойства");
//Id элемента нужного типа записываемого атрибута
Таблица.Колонки.Добавить("Id");
//Uid типа элемента записываемого атрибута
Таблица.Колонки.Добавить("TypeUid");
//Uid элемента нужного типа записываемого атрибута
Таблица.Колонки.Добавить("Uid");
//Имя – представление элемента
Таблица.Колонки.Добавить("Name");
//ограничимся добавлением только 2 параметров: тип клиента,
//заранее узнаем все параметры этих элементов в ELMA
//устанавливаем Тип клиента – "Партнер"
Нстр = Таблица.Добавить();
Нстр.ИмяСвойства = "Type";
Нстр.Id = "1";
Нстр.TypeUid = "4dd4556a-cec2-4ec6-b951-40e8aae52fe3";
//устанвливаем ответственного пользователя "Администратор"
Нстр = Таблица.Добавить();
Нстр.ИмяСвойства = "Responsible";
Нстр.Id = "1";
Нстр.TypeUid = "18faf3ae-03c9-4e64-b02a-95dd63e54c4d";

Возврат Таблица;
КонецФункции
```

В данном примере не рассмотрена запись таких атрибутов юридического лица, как, например, «Адрес» и «Email». В рамках объектной модели ELMA – это отдельные объекты. Поэтому их необходимо предварительно создать, а затем записать их как атрибуты объекта «Юридическое лицо» аналогично, например, атрибуту «Тип клиента».

Рассмотрим работу данной обработки. При открытии обработки в 1С необходимо выбрать контрагента, которого нужно экспортировать в 1С. После нажатия на кнопку **Записать/обновить контрагента в ELMA** будет запущена обработка. Метод веб-сервиса ELMA «Insert» и метод «Update» возвращают строку с Id элемента (который был создан или который был перезаписан), поэтому его можно использовать для представления «читаемого» для пользователя ответа веб-сервиса. Ответ выведен на форму обработки (Рис. 64).

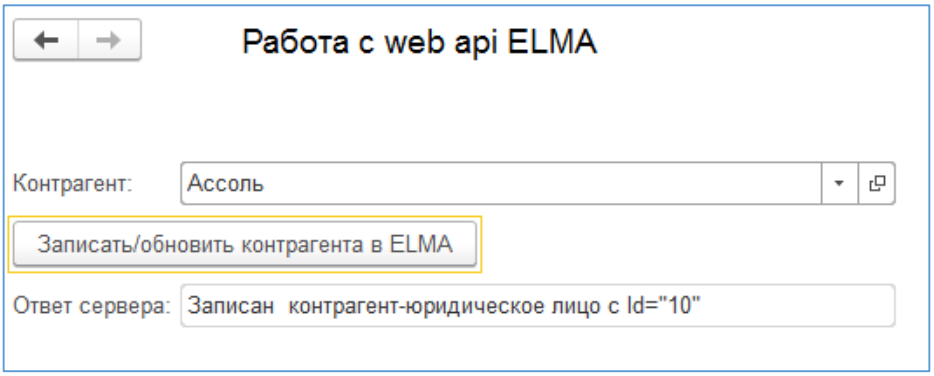


Рис. 64. Окно обработки по выгрузке контрагентов из 1С в ELMA

На Рис. 65 и Рис. 66 представлены карточки с реквизитами созданного в ELMA юридического лица и исходного контрагента в 1С.

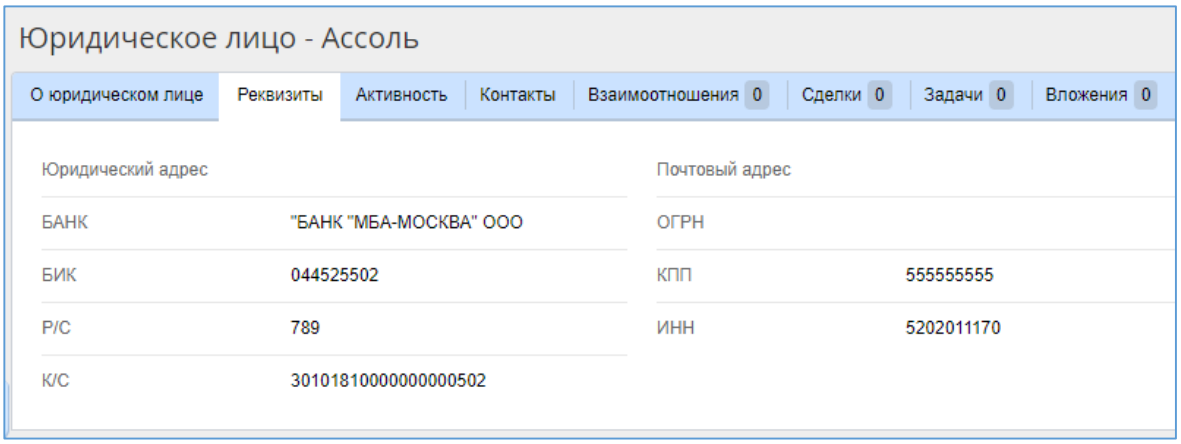


Рис. 65. Реквизиты созданного в ELMA юридического лица

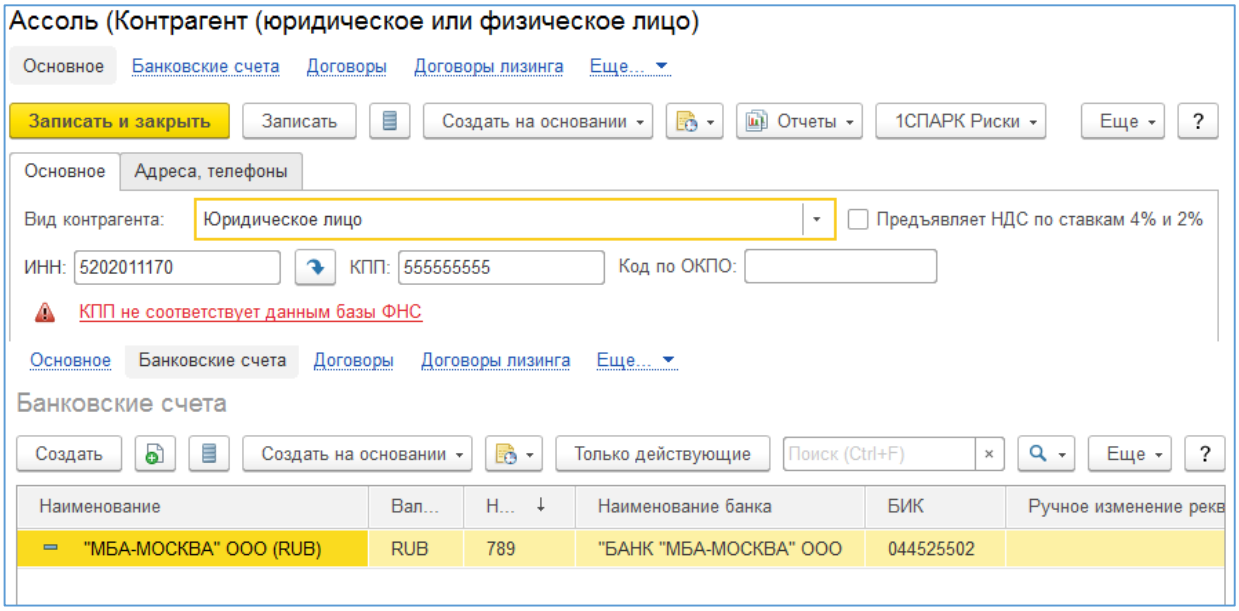


Рис. 66. Реквизиты контрагента в 1С

Например, если изменить у контрагента в 1С реквизит «КПП» (в примере изменим на «123456789») и повторно выгрузить контрагента, то текущее юридическое лицо будет обновлено (Рис. 67).

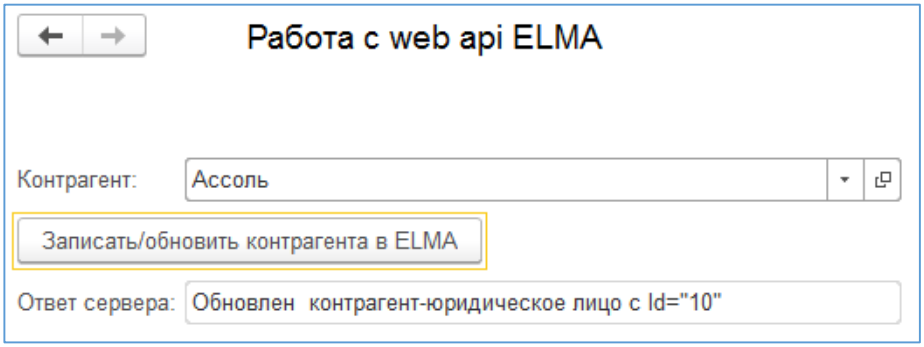


Рис. 67. Обновление контрагента при помощи обработки

В реквизитах карточки юридического лица также будет изменен требуемый атрибут «КПП» (Рис. 68).

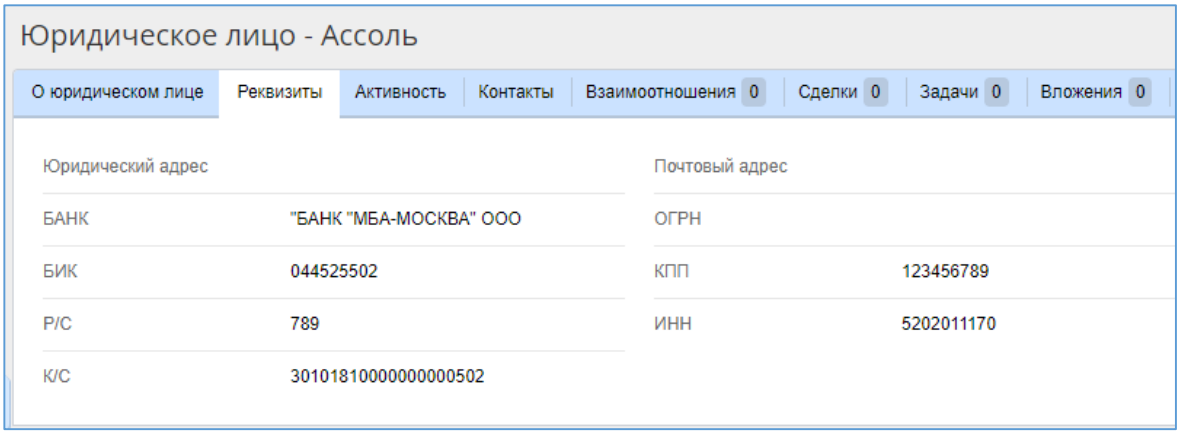


Рис. 68. Обновленный реквизит КПП в карточке юридического лица

Приемы, использованные при разработке данного решения, можно использовать для взаимодействия с любым другим веб-сервисом ELMA. Также в реальных задачах синхронизации данных рекомендуется расширить объект «Контрагент» или «Юридическое лицо» в ELMA, добавив в него атрибут, в котором будет сохраняться уникальный идентификатор нужного элемента справочника 1С.

В целом, у подобного способа обмена информацией есть как преимущества, так и недостатки.

К преимуществам можно отнести отсутствие необходимости в сервисном процессе в ELMA, который осуществляет синхронизацию данных в определенный промежуток времени. Причем, для оперативности выгрузку контрагента из 1С можно привязать к подписке на событие записи контрагента. Тогда актуальные данные будут сразу экспортированы в ELMA.

К недостаткам в первую очередь нужно отнести достаточно сложную разработку решений, связанных с использованием веб-сервисов. При формировании тела запроса при помощи программного кода нужно очень внимательно следить за записью структуры данных, начала и окончания каждого узла. Для проверки каждого шага приходится использовать отладку, т.к. для большинства разработчиков будет сложно сразу (абстрактно) разработать весь необходимый код. Также выполнение обращения к веб-сервису происходит на стороне 1С. Поэтому, если выгрузка привязана к событию записи контрагента, время записи каждого элемента увеличится. При этом, если возникли какие-либо ошибки (соединение с ELMA и т.п.), их также необходимо обработать на стороне 1С и уведомить об этом пользователя.

5.3. Реализация механизмов документооборота ELMA для 1С

Иногда по ходу текущей деятельности в компании возникают ситуации, когда требуется осуществлять согласование определенных типов документов или ознакомление с документами. Во многих распространенных типовых конфигурациях 1С механизмы согласования документов и механизмы постановки задач пользователям реализованы очень поверхностно и требуют серьезной доработки конфигурации.

В рамках деятельности интернет-магазина оператору магазина иногда бывает необходимо распечатать товарный чек или иную форму подтверждения заказа покупателя и вручить ее клиенту или отправить по почте. Печатная форма должна содержать список товаров заказа и другие необходимые атрибуты. Т.к. заказ клиента экспортируется в систему 1С, то разумно использовать стандартную печатную форму, которая используется для документа «Заказ клиента» в 1С.

Используя ELMA и механизмы интеграции 1С, решить эту задачу можно, не привлекая сотрудника (оператора интернет-магазина) к работе в двух информационных системах (ELMA – для обработки заказов, 1С – для их печати). В ELMA можно создать документ с прикрепленным файлом-версией, содержащим печатную форму документа, выполненную по стандартному макету (шаблону) документа в 1С. При этом никакого отдельного шаблона в ELMA разрабатывать не потребуется.

Для этого реализуем небольшой бизнес-процесс в ELMA, содержащий операции документооборота (создание документа и ознакомление с документом). Карта процесса представлена на Рис. 69.

Рис. 69. Процесс создания документа по заказу клиента в ELMA

В процессе для автоматического создания документа используется операция документооборота «Создание документа» (Рис. 70).

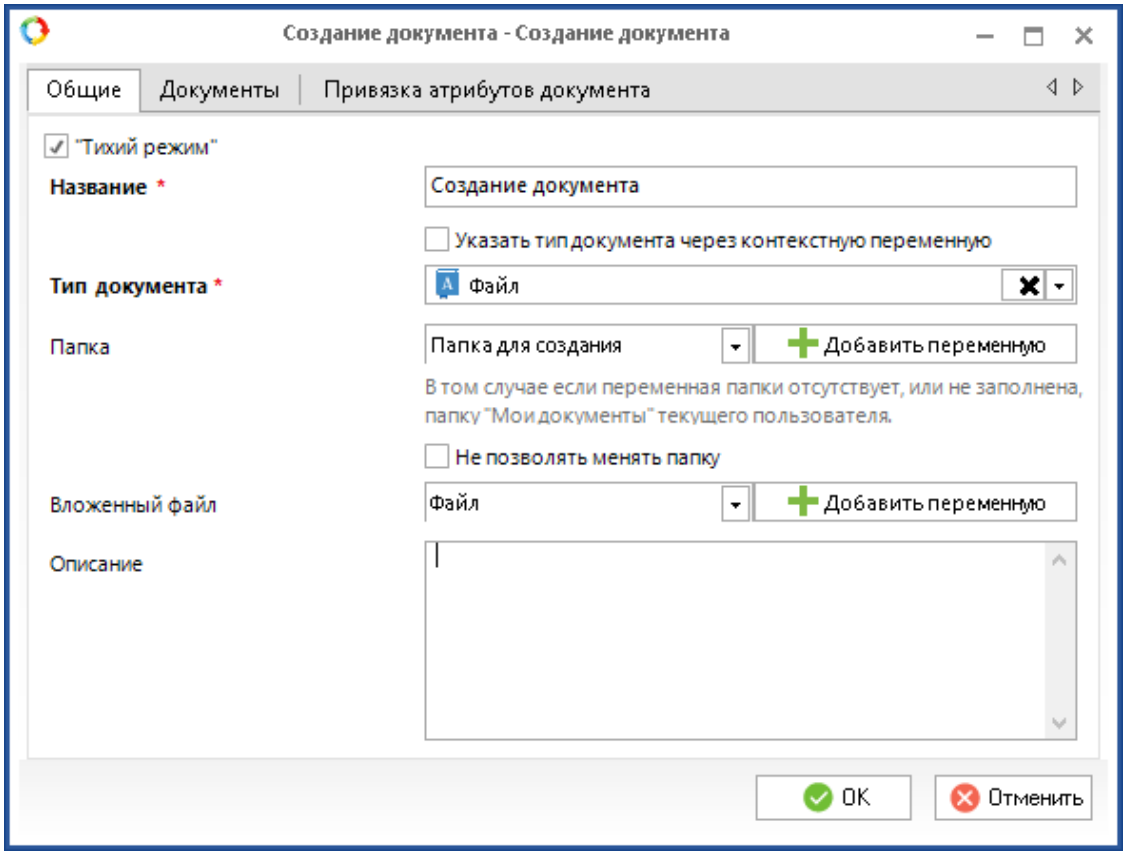


Рис. 70. Настройки операции автоматического создания документа

Для того чтобы документ создавался без участия пользователя, в настройках операции необходимо установить флажок **Тихий режим** и сопоставить необходимые атрибуты контекста процесса с атрибутами создаваемого документа. В поле **Вложенный файл** мы указываем переменную контекста с типом «Файл», куда будет записана сгенерированная печатная форма.

В задаче «Выбрать заказ» от пользователя требуется выбрать нужный элемент справочника «Заказ» в ELMA, печатная форма будет сгенерирована по привязанному к нему заказу клиента в 1С.

В сценарии «Подготовить файл в ELMA» осуществляется вызов необходимых методов из 1С, генерирующих файл печатной формы согласно макету 1С, привязанному к данному типу документа. Полученный файл привязывается к контексту процесса, а также в сценарии заполняются необходимые для автоматического создания документа переменные.

Для работы с файлами в сценариях потребуется подключить дополнительные пространства имен:

```
using EleWise.ELMA.Services;
```

```
using EleWise.ELMA.Integration1C;
using EleWise.ELMA.Integration1C.Data;
using EleWise.ELMA.Integration1C.V82;
using EleWise.ELMA.Files;
using EleWise.ELMA.Runtime.Managers;
using EleWise.ELMA.Common.Managers;
using EleWise.ELMA.Model.Services;
```

Текст сценария «Подготовить файл в ELMA»:

```
public virtual void PrepareDoc (Context context)
{
    //1. Инициализируем необходимые для работы с 1С переменные
    var service = Locator.GetServiceNotNull<Integration1CService> ();
    ComObject connector = service.GetComConnector ("erp1c");
    var Manager1C = (dynamic)connector.Reference;
    //2. Работу с файлами осуществляем методами платформы 1С, все
файлы и пути привязаны к клиенту 1С – т.е. к серверу ELMA
    //для хранения печатной формы генерируем временный файл, который
затем будет автоматически очищен средствами ОС
    string tmpFilePath = Manager1C.ПолучитьИмяВременногоФайла("pdf");
    //3. Заранее разработанная функция в общем модуле конфигурации
1С, возвращаются по ссылке на документ – объект типа "Табличный
документ"
    //ссылку на "Заказ клиента" 1С берем из атрибута "Заказ" ELMA
    dynamic tableDoc =
Manager1C._ElmaModule.ПолучитьТабличныйДокументЗаказаКлиента(context.
Order.Order1S.GetComReference().Ref);
    //4. Метод табличного документа "Записать" позволяет сохранить
его в наиболее популярные форматы
    //для примера используем pdf, доступны также doc, xls
    tableDoc.Записать(tmpFilePath, "pdf");
    //5. Записываем файл в контекст процесса из файла на диске,
созданным при помощи 1С
    context.FileOrder = InterfaceActivator.Create<BinaryFile>();
    context.FileOrder.Name = "Order "+context.Order.Number+".pdf";
    context.FileOrder.CreateDate = DateTime.Now;
    context.FileOrder.InitializeContentFilePath();
    System.IO.File.Copy(tmpFilePath,
context.FileOrder.ContentFilePath);

Locator.GetServiceNotNull<IFileManager>().SaveFile(context.FileOrder)
;
    //6. Инициализируем переменные контекста, необходимые для
создания документа без участия пользователя
    context.DocFolder =
EntityManager<EleWise.ELMA.Documents.Models.Folder>.Instance.LoadOrNu
ll(1L);
    context.DocName = "Печатная форма по заказу №" +
context.Order.Number;
}
```

В процедуре вызывается метод 1С: **ПолучитьТабличныйДокументЗаказаКлиента**, который заранее реализован в конфигурации 1С.

```
Функция ПолучитьТабличныйДокументЗаказаКлиента(ДокументЗаказ) ЭКСПОРТ
Попытка
  МассивДок = Новый Массив;
  МассивДок.Добавить(ДокументЗаказ);
  ПараметрыПечати = Новый Структура;

  ПараметрыПечати.Вставить("ДополнитьКомплектВнешнимиПечатнымиФормами
",ЛОЖЬ);
  ПараметрыПечати.Вставить("Тип","ЗаказКлиента");
  КоллекцияПечатных =
УправлениеПечатью.ПодготовитьКоллекциюПечатныхФорм("ЗаказКлиента");

  ПараметрыВывода =
УправлениеПечатью.ПодготовитьСтруктуруПараметровВывода();

  Обработки.ПечатьЗаказовНаТоварыУслуги.Печать(МассивДок,ПараметрыПеч
ати,КоллекцияПечатных,Новый СписокЗначений,ПараметрыВывода);
  //возвращаем табличный документ
  Возврат КоллекцияПечатных[0].ТабличныйДокумент;
Исключение
  ВОзврат ОписаниеОшибки();
КонецПопытки;
КонецФункции
```

В данной функции «сымитирована» подстановка необходимых параметров и вызов типовой функции печати документа, возвращающей объект типа «Табличный документ». В конфигурации «Управление предприятием: ERP 2», в которой реализована данная функция, предусмотрены универсальные механизмы массовой печати документов, поэтому необходимы подобные доработки с целью более удобного вызова функции из ELMA. В более ранних версиях конфигураций 1С механизмы печати реализованы по-другому (где-то может потребоваться более значительная доработка).

Рассмотрим пример. В конфигурации «Управление торговлей 10.3» функция получения табличного документа вынесена в модуль объекта, который требуется напечатать. Однако, у нее не указано служебное слово «Экспорт», позволяющее вызывать ее из внешней среды. Поэтому для того чтобы иметь возможность вызвать данную функцию из внешнего приложения (в нашем случае – ELMA), можно добавить свою функцию, выполняющую этот же код, или указать слово «Экспорт» при определении функции.

```
// Функция формирует табличный документ с печатной формой заказа или
счета,
```

```
// разработанного методистами
// Тип – строка с типом печатной формы «Счет» или «Заказ»
// Возвращаемое значение:
//  Табличный документ – сформированная печатная форма
//
Функция ПечатьСчетаЗаказа(Тип) ЭКСПОРТ

Возврат СоздатьТабличныйДокументПечатиСчетаЗаказа(Тип,
ПолучитьПараметрыПечатиСчетаЗаказа(Тип));

КонецФункции
```

Вызов функции из ELMA также будет немного отличаться. Т.к. функция привязана к объекту 1С, необходимо инициализировать объект в ELMA, а затем вызвать его метод.

```
dynamic tableDoc =
context.Order.Order1S.GetComReference().Ref.ПолучитьОбъект().ПечатьСч
етаЗаказа("Заказ");
```

В целом подобные доработки с разной степенью сложности можно реализовать в отдельных общих модулях конфигурации 1С. Это может потребоваться, например, чтобы не потерять возможность автоматического обновления типовых конфигураций.

После выполнения сценария и операции создания документа в ELMA мы получим на ознакомление документ типа «Файл» с прикрепленным файлом-версией сформированной печатной формы (в формате .pdf). Вкладка **Предпросмотр** карточки документа представлена на Рис. 71.

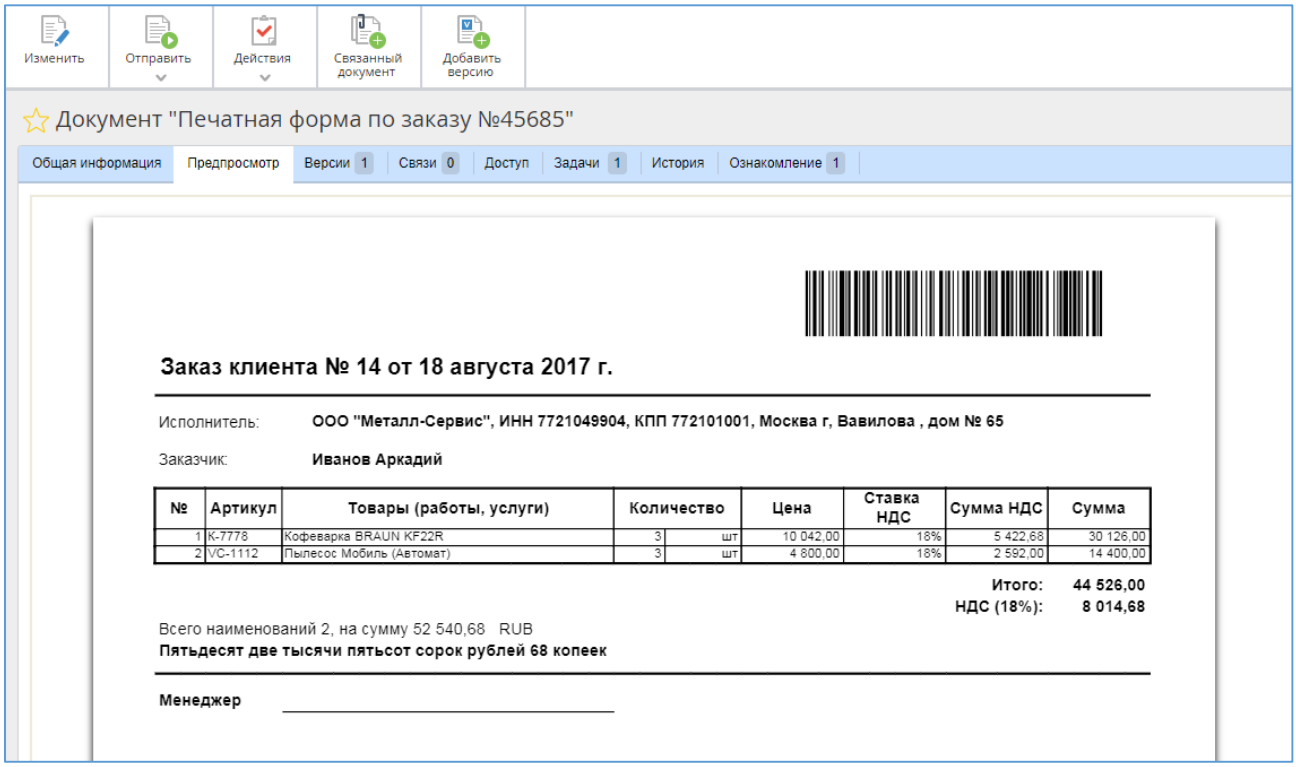


Рис. 71. Документ ELMA с печатной формой заказа клиента из 1С

Используя данный механизм, можно реализовать надстройку, использующую возможности документооборота ELMA с формированием версий документов на базе макетов 1С. Это может применяться, когда необходимо предоставлять документы на ознакомление или согласование сотрудникам, которые не ведут иной деятельности в учетной системе предприятия. Гораздо проще организовать их работу через более легкий интерфейс веб-браузера, а не устанавливать им отдельное приложение-клиент для 1С.

Для согласования потребуется небольшая доработка конфигурации 1С, например, осуществлять проведение документа только после завершения процесса его согласования. Однако эта доработка гораздо менее трудозатратна, чем реализация механизма согласования на стороне 1С.

Подводя итоги, отметим, что интеграция системы ELMA с системой «1С: Предприятие» позволяет:

1. Работать со справочниками и документами системы «1С: Предприятие».
2. Запускать функции системы «1С: Предприятие».
3. Обращаться к объектной модели системы «1С: Предприятие».

4. Осуществлять запуск бизнес-процессов и отсылку сообщений из системы «1С: Предприятие» в систему ELMA.

Обеспечение успешной и безотказной интеграции системы ELMA и «1С: Предприятие» достигается за счет использования COMConnector. Он обеспечивает идентичность данных при использовании в организации нескольких различных информационных систем. В нашем случае – позволяет системе ELMA обращаться к свойствам и методам объектов «1С: Предприятие».

Глава 6.Полезные ресурсы

Помимо текущего руководства, посвященного общим принципам взаимодействия системы ELMA и «1С: Предприятие» и решению задач по использованию ELMA в качестве средства взаимодействия интернет-магазина и 1С, существуют аналогичные издания, в которых описываются основные возможности приложений системы ELMA:

- Краткое руководство по Платформе ELMA BPM
- Краткое руководство по внутреннему порталу ELMA
- Краткое руководство по приложению ELMA ECM+
- Краткое руководство по приложению ELMA CRM+
- Краткое руководство по приложению ELMA Проекты+
- Краткое руководство по приложению ELMA KPI
- Краткое руководство администратора ELMA
- Краткое руководство по настройке работы системы ELMA на высоких нагрузках

Данные руководства знакомят читателя с ключевыми особенностями системы, подробное и исчерпывающее описание функционала системы ELMA содержится в справке, которая входит в поставку системы, а также всегда доступна в сети Интернет: http://www.elma-bpm.ru/kb/help.

Справочные материалы по каждому приложению разбиты на три категории: для пользователя, для внедрения и для администратора, что позволяет быстро найти нужную информацию.

Общее описание приложений и условия их приобретения доступны на **сайте ELMA**: http://www.elma-bpm.ru. Также на данном сайте всегда можно обратиться в компанию ELMA с помощью кнопки **Задать вопрос**, расположенной в верхнем правом углу.

Ключевые возможности приложений и основные способы их использования продемонстрированы в **on-line демоверсии** http://www.elma-bpm.ru/download. Если же вы хотите подробнее изучить какое-либо из приложений, по этой же ссылке доступно скачивание демоверсии с такими же настройками, как и в on-line версии.

Система ELMA постоянно развивается, и на базе Платформы и приложений разрабатываются компоненты, предназначенные для решения различных более узких и конкретных задач. Со списком и условиями приобретения таких готовых решений Вы можете ознакомиться в **ELMA Store**: https://store.elma-bpm.ru.

При разработке собственных решений полезными окажутся материалы **Базы знаний ELMA**: http://www.elma-bpm.ru/kb.

Если же при работе в системе возникли вопросы технического характера, можно обратиться на сайт технической поддержки ELMA: http://support.elma-bpm.ru.

Для получений консультаций по системе ELMA или по сотрудничеству с компанией ELMA позвоните нам:

- Ижевск: +7 (3412) 93-66-93
- Москва: +7 (499) 921-02-87
- Казань: +7 (843) 567-17-69
- Киев: +38 (067) 788-47-12
- Алматы: +7 (727) 313-15-04