ФИРМА "1С"

Средства интеграции и обмена данными в системе "1С:Предприятие 8"

Методические материалы для слушателя сертифицированного курса

Февраль, 2015 г.

Введение

"1С:Предприятие 8" является универсальной системой автоматизации деятельности предприятия. За счет своей универсальности система "1С:Предприятие 8" может быть использована для автоматизации самых различных участков деятельности организаций, предприятий. В том числе с ее помощью можно реализовывать различные обменные механизмы как с другими базами системы "1С:Предприятие 8", так и с совершенно другими программными комплексами.

В рамках данного курса не будут рассматриваться общие положения, связанные с функционированием программного комплекса. Считается, что перед тем как приступать к изучению данного материала, обучаемые уже познакомились с основами работы в системе "1С:Предприятие 8".

В предлагаемом курсе пойдет речь о различных механизмах, их особенностях, с помощью которых программный комплекс "1С:Предприятие 8" может обмениваться данными (загружать, выгружать), взаимодействовать с другими системами (и себе подобными). Т.е. будут рассмотрены те возможности системы, которые непосредственно связаны с термином "обмен".

Задача обмена данными возникла довольно давно. Если у поставщика есть некая система учета своей деятельности, какая-то система может быть у посредника, покупателя. То у посредника или конечного покупателя рано или поздно возникает вполне оправданное желание не заводить документы, получаемые от поставщика "вручную", а "загрузить" их ("упростить себе жизнь"). Здесь следует отметить, что в общем случае задача обмена намного шире.

Очень часто при рассмотрении возможности обмена идет речь именно об обмене данными посредством файлов. В рамках данного курса будет рассматривать механизмы, ориентированные на формирование файлов обмена различных форматов и механизмы "непосредственного" обмена (без использования файлов посредников).

В общем случае почти любую задачу по организации обмена данными можно "разбить" на четыре подзадачи:

- Формат файла обмена (в ряде случаев может отсутствовать)
- Способ "доставки" данных
- Состав выгружаемых данных
- Задача синхронизации данных

Можно сказать, что все задачи решаемые, но самая сложная из них - последняя. В рамках данного курса рассматривать вопросы синхронизации мы не будем. Связано это с тем, что в общем случае данная задача не решается только программными средствами. Обычно это комплекс как организационных, так и программных решений.

Суть задачи синхронизации в том, что в разных базах данных одни и те же номенклатурные позиции (к примеру) могут иметь различные характеристики (наименование, артикул, код и т.д.), либо наоборот разные номенклатурные позиции могут иметь одинаковые характеристики. Требуется, чтобы при обмене не возникало из-за этого проблем.

Способ доставки – одна из самых "простых" задач. Диапазон решений довольно широк: от курьеров с дискетами, до использования Интернета (электронной почты и т.д.). Наиболее популярными способами доставки является использование различных Интернет технологий (ftp, http,e-mail).

Задача формата файла обмена (и соответственно его типа) возникает в том случае, когда для переноса данных используется промежуточный файл (в ряде случаев можно обойтись без него). В данном случае "1С:Предприятие 8" позволяет работать с текстовыми файлами, dbf, html, xml. Можно сказать, что диапазон типов файлов очень широк, т.к. программный комплекс может использовать возможности Automation, COM-соединения (следует отметить, что сами по себе эти механизмы не являются механизмами обмена, скорее их можно отнести в механизмам взаимодействия, но используя данные механизмы можно решать задачи обмена между программными комплексами).

Под форматом понимается структура файла обмена (для файла dbf- это состав и типы полей и т.д.).

В рамках данного курса будет относительно подробно рассматриваться как раз решение задач формирования файлов обмена, и способов их доставки.

Постановка задачи

Необходимо организовать обмен данными в уже существующей конфигурации. Информация должна выгружаться, загружаться в файлы формата txt, dbf, xml, xls. В рамках данной задачи будем организовывать взаимодействие по Automation с другой информационной базой "1С:Предпрятия 8", рассмотрим вариант использования COM-соединения, реализацию механизма обмена с использованием возможностей объекта "ПланыОбмена". Рассмотрим общие принципы функционирования конфигурации "Конвертация данных редакция 2"

1. Общие принципы работы с файлами

Перед тем как приступить к рассмотрению особенностей работы с файлами определенных типов проведем обзор средств программного комплекса "1С:Предприятие 8", которые позволяют работать непосредственно с самими файлами.

Работа с файлом может начинаться с создания специального объекта "Файл". Следует отметить, что данный объект позволяет работать именно с файлом, и не имеет отношения к его содержимому.

```
МойФайл=Новый Файл(Имя)
```

Имя - полное имя файла или каталога, с которым будет связан конструируемый объект.

Пример использования:

```
Файл = Новый Файл(ПутьДоФайла);

Если Файл.ЭтоФайл() Тогда

     Если Файл.Размер()>300000 Тогда

Сообщить("Не нужно такой файл отправлять по почте!");

     КонецЕсли;

Сообщить("Изменен:
"+Строка(Файл.ПолучитьВремяИзменения()));

Иначе

     Сообщить("Выбран каталог");

КонецЕсли;
```

При работе с файлами очень часто приходится организовывать для пользователя механизм интерактивного выбора нужного ему файла (каталога). Решить данную задачу в системе "1С:Предприятие 8" можно с помощью объекта "ДиалогВыбораФайла"

Как пример использования можно привести следующий фрагмент кода:

```
Режим=РежимДиалогаВыбораФайла.Открытие;

ДиалогОткрытияФайла=Новый ДиалогВыбораФайла(Режим);

ДиалогОткрытияФайла.ПолноеИмяФайла="";

Фильтр="Текст(*,txt)|*.txt";

ДиалогОткрытияФайла.Фильтр=Фильтр;

ДиалогОткрытияФайла.МножественныйВыбор=Ложь;

ДиалогОткрытияФайла.Заголовок="Выберите файл";

Если ДиалогОткрытияФайла.Выбрать() Тогда

        ПутьДоФайла=ДиалогОткрытияФайла.ПолноеИмяФайла;

КонецЕсли
```

Практикум № 1

Для констант "Путь до файлов" и "Путь до приемника" (тип значения "Строка", длина переменная 100 символов) в форме констант определите механизм выбора каталога и записи полного пути до него.

Для полноты картины следует еще рассмотреть процедуры работы с операционной системой:

```
КомандаСистемы(<Строка команды>, <Текущий каталог>);
```

Использование данной процедуры вызывает на исполнение команду операционной системы, как если бы она была введена в командной строке. К таким командам можно отнести: copy, del и т.д. (с указанием в строке команды необходимых параметров).

```
ЗапуститьПриложение(<Строка команды>, <Текущий каталог>,
<Дождаться завершения>);
```

Данная процедура выполняет запуск внешнего приложения, либо открытие файла с использованием ассоциированного с ним приложения.

1.1. Работа с текстовыми файлами

В данном разделе пойдет речь о работе с файлами, имеющими расширение "txt" (Текстовыми файлами). Хочется отметить, что при помощи рассматриваемых механизмов можно работать не только с файлами "*.txt", но и с рядом других типов файлов (например, html, xml и т.д., правда такой способ работы будет далеко не оптимальным).

В системе "1С:Предприятие" работать с такими файлами можно несколькими способами:

- Используя элемент управления "ПолеТекстовогоДокумента"
- Используя объект "Текстовый документ"
- Используя модель последовательного доступа к файлу

Начнем с самого простого

1.1.1. Работа с текстовым документом

Работа с объектом "Текстовый документ" начинается (в общем случае) с его создания при помощи конструктора:

```
ТД=Новый ТекстовыйДокумент;
```

Познакомимся с данным объектом на примере выгрузки справочника "Контрагенты" (обработка "ТекстовыйДокумент").

"Обычная" выгрузка:

```
Процедура ВыгрузкаОбычная(Кнопка)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    ТД=Новый ТекстовыйДокумент;

Выборка=Справочники.Контрагенты.ВыбратьИерархически();

    Пока Выборка.Следующий() Цикл

        ТД.ДобавитьСтроку(Выборка.Наименование);

    КонецЦикла;

    ТД.Записать(Путь+"kontr.txt");

КонецПроцедуры
```

"Обычная" загрузка:

```
Процедура ЗагрузкаНаименований(Кнопка)
      Путь= Константы.ПутьДоФайлов.Получить()+"\";
      ЭлементыФормы.ПолеТД.Очистить();
      ТД=Новый ТекстовыйДокумент;
      ТД.Прочитать(Путь+"kontr.txt");
      КоличествоСтрокДокумента=ТД.КоличествоСтрок();
      Для счНом=1 По КоличествоСтрокДокумента Цикл
            Стр=ТД.ПолучитьСтроку(счНом);
            Сообщить(Стр);
      КонецЦикла;
КонецПроцедуры
```

Выгрузка с открытием документа

```
Процедура ВыгрузкаСОткрытием(Кнопка)
      Путь= Константы.ПутьДоФайлов.Получить()+"\";
      ТД=Новый ТекстовыйДокумент;
Выборка=Справочники.Контрагенты.ВыбратьИерархически();
      Пока Выборка.Следующий() Цикл
            ТД.ДобавитьСтроку(Выборка.Наименование);
      КонецЦикла;
      ТД.ТолькоПросмотр=Истина;
      ТД.Показать("Для просмотра");
      ТД.Записать(Путь+"kontr.txt");
КонецПроцедуры
```

Формирование документа с запретом на вывод

```
Процедура ВыгрузкаНеДляПечати(Кнопка)
    Путь= Константы.ПутьДоФайлов.Получить()+"\";
    ТД=Новый ТекстовыйДокумент;
Выборка=Справочники.Контрагенты.ВыбратьИерархически();
    Пока Выборка.Следующий() Цикл
        ТД.ДобавитьСтроку(Выборка.Наименование);
    КонецЦикла;
    ТД.Вывод=ИспользованиеВывода.Запретить;
    ТД.Показать("Не для печати");
    Попытка
        ТД.Записать(Путь+"kontr.txt");
    Исключение
        Сообщить("Вывод документа запрещен!");
    КонецПопытки;
КонецПроцедуры
```

Проверьте работоспособность обработки на практике.

1.1.2. Элемент управления "ПолеТекстовогоДокумента"

Не смотря на то, что элемент управления "Поле текстового документа" обладает всеми свойствами и методами объекта "Текстовый документ", он все же заслуживает отдельного рассмотрения.

Рассмотрим обработку "ПолеТекстовогоДокумента".

Заполнение при открытии:

```
Процедура ПриОткрытии()
// обработчик события "При открытии" формы обработки
    ЭлементыФормы.ПолеТекста.УстановитьТекст("Создаем
новый.");
    ЭлементыФормы.ПолеТекста.ДобавитьСтроку("Добавляем
строку");
    ЭлементыФормы.ПолеТекста.ВставитьСтроку(2,"вставляем
строку");
КонецПроцедуры
```

Запись данных в файл

```
Процедура Записать(Кнопка)

// обработчик нажатия на кнопку "Записать"

      Путь=Константы.ПутьДоФайлов.Получить()+"\";

      Стр=ЭлементыФормы.ПолеТекста.ПолучитьВыделенныйТекст()
;

      ЭлементыФормы.ПолеТекста.УстановитьТекст(Стр);

      Предупреждение("Остался только выделенный текст");

      ЭлементыФормы.ПолеТекста.Записать(Путь+"result.txt");

КонецПроцедуры
```

Чтение данных из файла:

```
Процедура КнопкаВыполнитьНажатие(Элемент)

// обработчик нажатия на кнопку "Прочитать"

      Путь=Константы.ПутьДоФайлов.Получить()+"\";

ЭлементыФормы.ПолеТекста.Прочитать(Путь+"result.txt");

      ЭлементыФормы.ПолеТекста.ДобавитьСтроку("------");

      ЭлементыФормы.ПолеТекста.ДобавитьСтроку("Загрузка
завершена!!!");

КонецПроцедуры
```

Очистка элемента управления:

```
Процедура Очистить(Кнопка)

// обработчик нажатия на кнопку "Очистить"

      ЭлементыФормы.ПолеТекста.Очистить();

КонецПроцедуры
```

В качестве комментария хочется добавить, что поле текстового документа предназначено для размещения в форме окна текстового редактора. Объект является элементом управления и помимо всех возможностей текстового документа обладает дополнительными свойствами и методами, связанными с его "видимостью" пользователю (с возможностью интерактивной работы пользователя с этим элементом управления).

1.1.3. Организация последовательного доступа к тексту

При работе с объектом "Текстовый документ" (либо на "запись", либо на "чтение") файл загружался полностью. В том случае, когда обрабатываемый файл имеет большой размер, данная особенность объекта "Текстовый документ" может привести к понижению скорости работы всего механизма в целом.

Как выход из подобной ситуации (для организации загрузки данных из файлов большого размера, выгрузки большого объема данных) можно использовать модель последовательного доступа. При использовании данного механизма загружается не весь файл, а только его "часть" и т.д.

Но следует помнить, что при использовании модели последовательно доступа нельзя прочитать нужную строку по номеру, добавить фрагмент текста в произвольное место файла.

При последовательном доступе к файлу используются два объекта:

- "ЧтениеТекста"
- "ЗаписьТекста"

Обратите внимание на тот факт, что у объекта "ЗаписьТекста", что "ЧтениеТекста" определено два конструктора. Отличаются они набором параметров. У конструктора формирующего неинициализированный объект параметров нет. В таком случае для работы с файлом потребуется использовать метод "Открыть" (но при таком способе создания объекта есть возможность работы с большим количеством кодировок).

Для знакомства с данным механизмом рассмотрим обработку "Последовательный текст". В модуле основной формы размещен текст обработчика события нажатия на кнопку "Выгрузка":

```
Процедура КнопкаВыполнитьНажатие(Элемент)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    Текст=Новый
ЗаписьТекста(Путь+"nomen_p.txt",КодировкаТекста.UTF8);

    Выборка=Справочники.Номенклатура.Выбрать();

    Пока Выборка.Следующий() Цикл

Текст.ЗаписатьСтроку(СокрЛП(Выборка.Наименование));

    КонецЦикла;

    Текст.Закрыть();

КонецПроцедуры
```

И обработчик нажатия на кнопку "Загрузка"

```
Процедура Загрузить(Кнопка)

      Путь= Константы.ПутьДоФайлов.Получить()+"\";

      Текст=Новый
ЧтениеТекста(Путь+"nomen_p.txt",КодировкаТекста.UTF8);

      Стр=Текст.ПрочитатьСтроку();

      Пока Стр<>Неопределено Цикл

            ЭлементыФормы.ПолеТД.ДобавитьСтроку(Стр);

            Стр=Текст.ПрочитатьСтроку();

      КонецЦикла;

КонецПроцедуры
```

"ПолеТД" - элемент управление "Поле текстового документа", размещен в диалоге обработки (в него производится чтение данных из открываемого в последовательной модели доступа файла).

Следует обратить внимание на то, что текст можно читать не только по строкам (используя метод "ПрочитатьСтроку()"), но и указывая количество считываемых символов (метод "Прочитать(<Размер>)").

Практикум № 2

Модифицируйте обработку "ЧтениеНоменклатуры". Необходимо добиться того, чтобы чтение из файла производилось по строке длиной 10 символов. Оцените результат.

При работе "ЗаписьТекста" следует обратить внимание на отличие методов "Записать" и "ЗаписатьСтроку".

"Записать" - записывает строку текста в файл. В конце строки разделитель не записывается. При использовании метода "ЗаписатьСтроку" разделитель записывается.

Формирование неинициализированного объекта:

```
Процедура НеинициализированныйОбъект(Кнопка)
    Путь= Константы.ПутьДоФайлов.Получить()+"\";
    Текст=Новый ЗаписьТекста();
    Текст.Открыть(Путь+"nomen_p.txt","ibm-1162");
    Выборка=Справочники.Номенклатура.Выбрать();
    Пока Выборка.Следующий() Цикл
Текст.ЗаписатьСтроку(СокрЛП(Выборка.Наименование));
    КонецЦикла;
    Текст.Закрыть();
КонецПроцедуры
```

1.2. Работа с файлами dbf

Для работы с базами данных формата DBF в системе может использоваться специальный объект "XBase". Механизм работы с базами данных формата DBF предназначен для обеспечения возможности манипулирования ими непосредственно из встроенного языка. Каждый объект XBase может быть связан с одним файлом базы данных и одним индексным файлом.

Разберем общие принципы работы обработки "КонтрагентыDBF".

Текст процедуры, производящей выгрузку справочника "Контрагенты", следующий:

```
Процедура КнопкаВыполнитьНажатие(Элемент)
    Путь= Константы.ПутьДоФайлов.Получить()+"\";
    БД=Новый XBase;
    БД.Поля.Добавить("CODE","S",5);
    БД.Поля.Добавить("NAME","S",40);
    БД.СоздатьФайл(Путь+"start.dbf",Путь+"index.cdx");
    БД.Индексы.Добавить("IDXCODE","CODE");
    ФлИБД=БД.СоздатьИндексныйФайл(Путь+"index.cdx");
    БД.АвтоСохранение=Истина;
Выборка=Справочники.Контрагенты.ВыбратьИерархически();
    Пока Выборка.Следующий() Цикл
```

```
            БД.Добавить();

            БД.CODE=Выборка.Код;

            БД.NAME=Выборка.Наименование;

      КонецЦикла;

      БД.ЗакрытьФайл();

КонецПроцедуры
```

В тексте данного модуля дополнительное внимание хотелось бы обратить на метод добавляющий индексы:

```
Добавить(<Имя>, <Выражение>, <Уникальность>, <Убывание>,
<Фильтр>);
```

Параметр <Убывание> (необязательный) определяет направление формирования индекса (Истина - убывает, Ложь - возрастает. Значение по умолчанию: Ложь.

Текст процедуры загрузки данных

```
Процедура Загрузка(Кнопка)

      Путь= Константы.ПутьДоФайлов.Получить()+"\";

      БД=Новый XBase;

      БД.ОткрытьФайл (Путь+"start.dbf",Путь+"index.cdx");

      Сообщить(БД.Name);

      Пока БД.Следующая() Цикл

            Сообщить(БД.Name);

      КонецЦикла;

      БД.ЗакрытьФайл();

КонецПроцедурыИспользование
```

Обратите внимание на тот факт, что после открытия файла объект УЖЕ СПОЗИЦИОНИРОВАН на первой записи файла. Т.е. если после открытия обходить данные в цикле, в условии которого вызывается метод "Следующая()", то можно пропустить данные первой записи.

Текст поиска значений удовлетворяющих определенному условию

```
Процедура Поиск(Кнопка)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    БД=Новый XBase;

    БД.ОткрытьФайл (Путь+"start.dbf",Путь+"index.cdx");

    БД.ТекущийИндекс=БД.Индексы.IDXCODE;

    Если БД.Найти("2",">=") Тогда

        Сообщить(БД.Name);

        Пока БД.Следующая() Цикл

            Сообщить(БД.Name);

        КонецЦикла;

    КонецЕсли;

    БД.ЗакрытьФайл();

КонецПроцедуры
```

1.3. Документы html

Можно сказать, что html документ представляет собой текстовый файл, имеющий определенную структуру и которому принудительно дано расширение htm или html. На самом деле более правильно говорить о языке html (язык разметки гипертекста). С этой точки зрения html документ, это документ с текстовым содержимым, написанный на языке html.

"Структура" html документа задается с помощью так называемых "тегов" (выражений заключенных в угловые скобки). Набор тегов и их свойств (значений свойств) зафиксирован. Изучение языка html сводится к изучению конечного множества тегов, свойств тегов и их значений.

Если говорить о "задаче, решаемой языком html", то это форматирование отображаемых данных самого документа. В этом смысле теги можно разбить на "теги структуры документа", "теги форматирования" и "другие".

Основные теги, задающие структуру html документа следующие:

```
<html>
<head>
<meta content="text/html; charset=windows-1251">
<title>Заголовок страницы</title>
</head>
<body bgcolor=lightyellow text=darkblue>
     "Тело" документа
</body>
</html>
```

В теле документа могут располагаться различные "форматирующие теги". Это непосредственно теги форматирования текста

```
<h1>Заголовок 1-го уровня (можно до 6-го)</h1>
<b>Полужирный<i> полужирный и наклонный </b> наклонный<i>
обычный текст <p> новый абзац.
```

Данные в html документе можно организовывать в таблицы:

```
<table>
<tr><td>первое</td><td>второе</td></tr>
<tr><td>третье</td><td>четвертое</td></tr>
</table>
```

Для "навигации" (переходу к другим ресурсам) могут использоваться гиперссылки

```
<a href="ресурс">текст гиперссылки</a>
```

Все это вместе при просмотре "специальными" программами выглядит следующим образом:

Заголовок 1-го уровня (можно до 6-го)

Полужирный ***полужирный и наклонный*** *наклонный обычный текст*

новый абзац.

первое второе

третье четвертое

текст гиперссылки

"1С:Предприятие 8" имеет средства просмотра html-документов. Познакомимся с такой возможностью на примере обработки "HTML".

У данной обработки определен реквизит URL (тип "строка", 50). При создании основной формы, помимо элемента управления, связанного с реквизитом URL, в диалоге был размещен элемент управления "ПолеHTMLДокумента" (имя "ПолеHTML"), определена командная панель (в качестве источника определен "ПолеHTML").

В качестве обработчика события "Изменение" для элемента управления "URL" используется процедура:

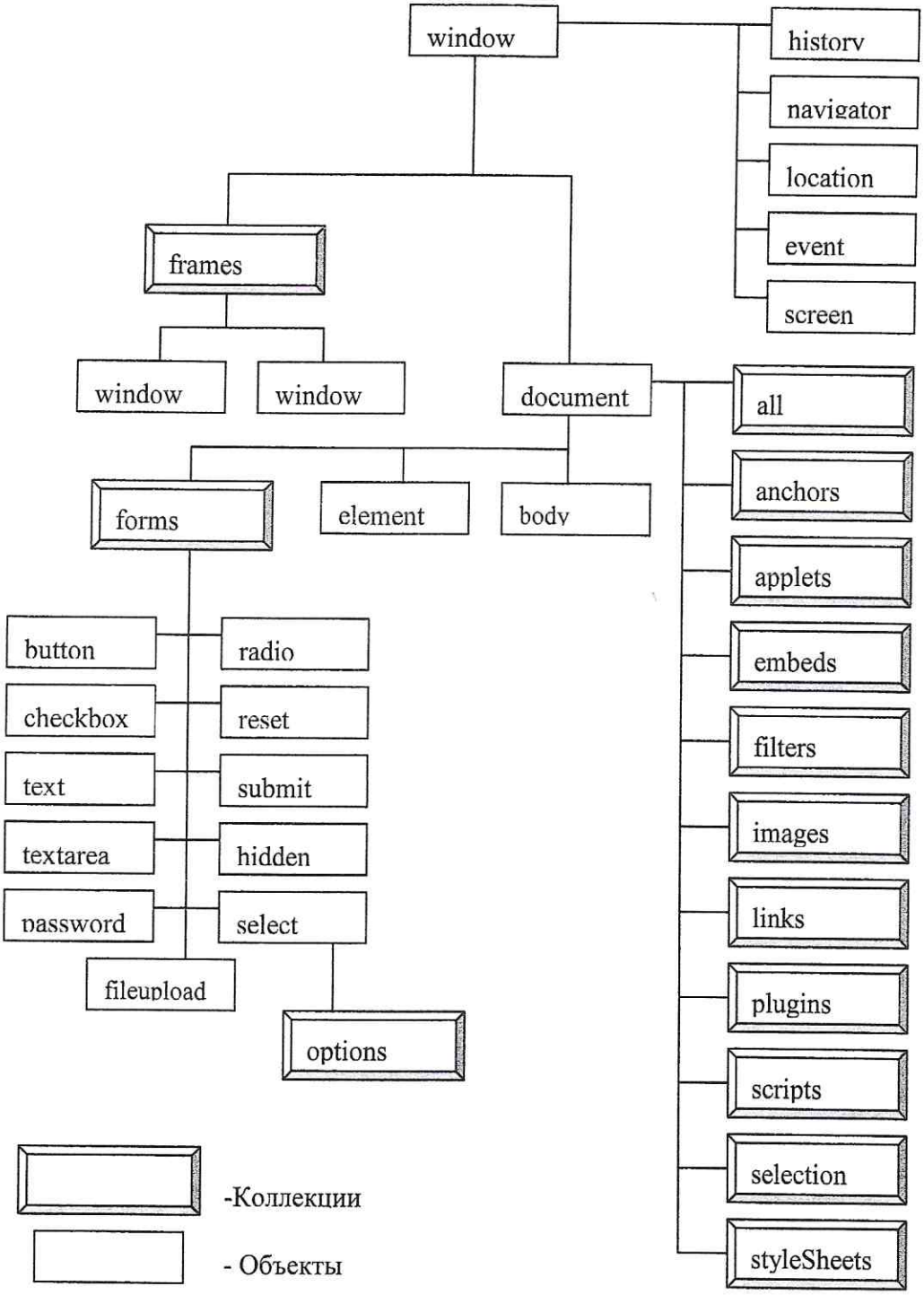
```
Процедура URLПриИзменении(Элемент)

    ЭлементыФормы.ПолеHTML.Перейти(URL);

КонецПроцедуры
```

Данные несложные действия позволяют создать "собственный внутренний браузер".

Следует отметить событие "ДокументСформирован" и свойство "Документ". Свойство "Документ" содержит СОМОбъект и предоставляет доступ к html документу (его объектной модели).

Штатными средствами просмотра html документов являются специализированные программы – браузеры. Наиболее популярный из них (мнение автора может не совпадать с вашим) является Microsoft IE. Браузер при работе с html документом реализует так называемую "объектную модель". Такую же объектную модель "поддерживает" элемент управления "ПолеHTMLДокумента".

Объектная модель, поддерживаемая браузерами, представлена на схеме приведенной ниже:



Свойство "Документ" позволяет работать с содержимым структурной единицы "document" (т.е. с коллекцией форм, и т.д.)

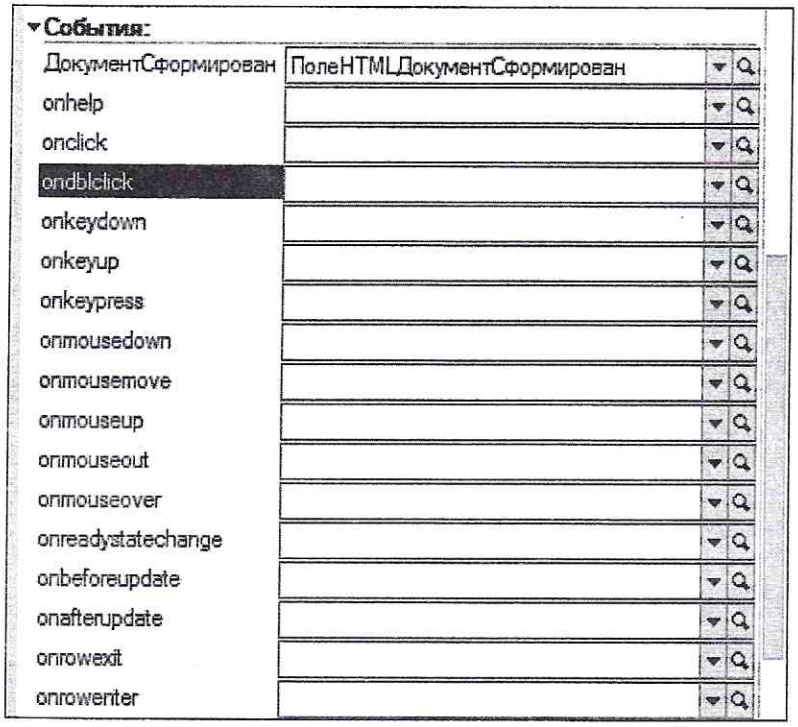
Событие "ДокументЗагружен" позволяет отследить момент полной загрузки документа в поле html документа (из разного "веса" документа, разной скорости подключения, это время может очень сильно изменяться).

Для того, чтобы познакомиться с возможностями использования свойства "Документ" и события "Документ загружен" рассмотрим модуль основной формы обработки "Html".

```
Перем ЗагрузкаЗавершена;


Процедура ОсновныеДействияФормыДействие(Кнопка)
    Если ЗагрузкаЗавершена Тогда
        Док=ЭлементыФормы.ПолеHTML.Документ;
        Док.forms["frm"].fname.Value="Учебный центр";
        Док.forms["frm"].fio.Value="Гончаров Д.И.";
    КонецЕсли;
КонецПроцедуры


Процедура ПолеHTMLДокументСформирован(Элемент)
    Если
ЭлементыФормы.ПолеHTML.Документ.readyState="complete" Тогда
        ЗагрузкаЗавершена=Истина;
    КонецЕсли;
КонецПроцедуры


ЗагрузкаЗавершена=Ложь;
```

Процедура "ОсновныеДействияФормыДействие" является обработчиком события "Нажатие" на кнопке "Выполнить" в нижней командной панели. Для того, чтобы пример "заработал" необходимо "просмотреть" файл "anketa.html".

Также хотелось бы обратить ваше внимание на обилие (и их "детальность") событий элемента управления "поле HTML документа" (на рисунке далеко не полный перечень):



Следует отметить, что в системе "1С:Предприятие 8" существует элемент управления "ПолеТекстовогоДокумента", у которого одно из значений свойства "Расширение" может быть значение "HTML". В этом случае "подсвечиваются" синтаксические конструкции языка html (на основе данного элемента управления можно реализовать простейший HTML редактор). HTML документ может быть выбран в качестве типа макета.

Практикум № 3

Найдите некоторую логическую "некорректность", связанную с обработкой события "ДокументЗагружен".

1.4. Извлечение текста

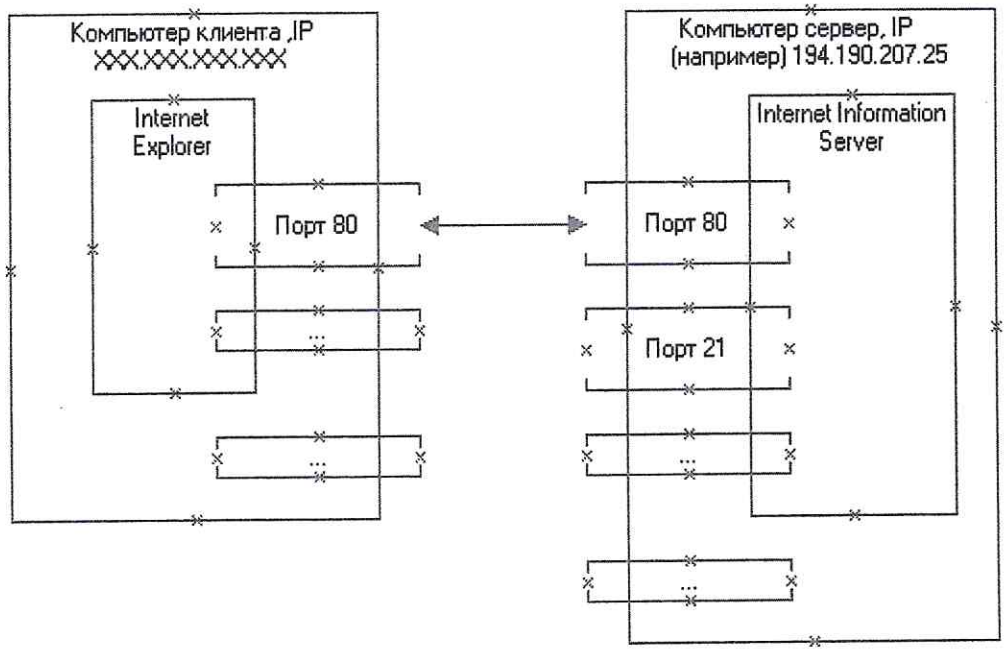
В ряде случаев возникает необходимость извлекать текст из документов разного формата, но которые имеют текстовое содержимое (при этом желательно "отсечь особенности" самого формата) В данном случае может помочь объект "Извлечение текста":

```
ИмяФайла = ПутьДоФайла;

Объект = Новый ИзвлечениеТекста(ИмяФайла);

Текст = Объект.ПолучитьТекст();

Сообщить(Текст);
```

2. Интернет технологии

Интернет уже давно проник во все сферы деятельности человека. С одной стороны это безграничный океан информации, с другой стороны это удобная "среда" ее передачи (если говорить правильно, то существуют понятия: клиент, выполняющий запрос, сервер предоставляющий данные, и протокол определяющий правила взаимодействия клиента и сервера).

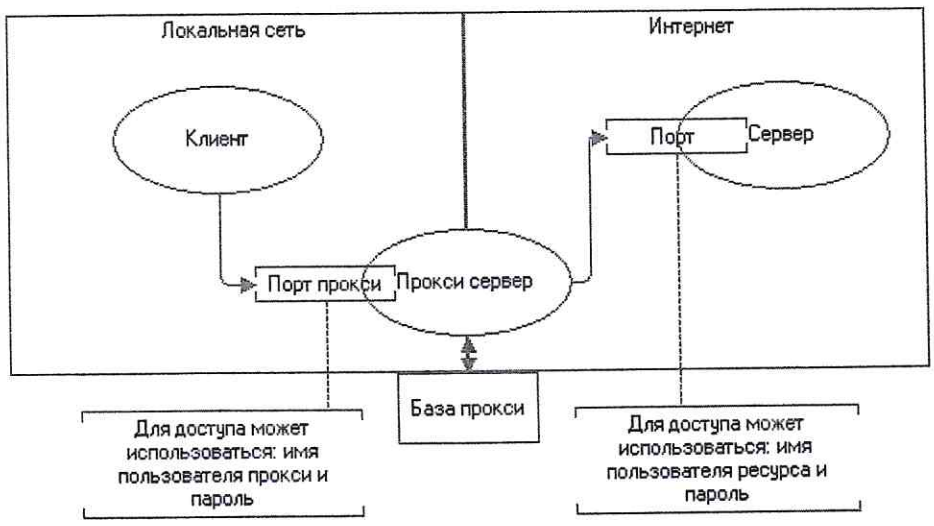
Схематично это можно представить следующим образом:



Следует отметить, что в рассмотренном варианте считалось, что клиентский компьютер был подключен к сети Интернет "напрямую". В большинстве случаев (при организации доступа из локальных сетей организаций) подключение к глобальной сети производится через специализированную ("промежуточную") службу "proxy".

Можно сказать, что proxy сервер также является специализированной программой, запущенной на компьютере и обслуживающей какой-либо его порт.

Схематически это можно представить следующим образом:



Используя средства системы "1С:Предприятие 8" выполнять http, ftp запросы, работать с электронной почтой. Именно о таких возможностях пойдет речь в данной главе.

2.1. Организация Интернет соединения

В случае если выход в Интернет производится через коммутируемые линии связи (модем), то перед тем как пользоваться возможностями системы "1С:Предприятие 8", связанными с Интернетом, это соединение необходимо установить. Для этой цели существует объект "ИнтернетСоединение". Перечень его свойств, методов приведен ниже.

Общий порядок работы довольно простой. С помощью конструктора создается экземпляр объекта "ИнтернетСоединение", далее используется метод "Установить()" и т.д.

2.2. Работа с электронной почтой

Программный комплекс "1С:Предприятие 8" позволяет работать с почтой как через установленного почтового клиента (MAPI), так и с прямым использованием протоколов SMTP, POP3 (независимо).

2.2.1. Объект "Почта"

Работа с электронной почтой при использовании объекта "Почта" не ведется "независимо". Работа ведется с почтовым клиентом "по умолчанию". В зависимости от клиента механизм может иметь те или иные особенности.

В качестве иллюстрации порядка работы с электронной почтой рассмотрим обработку "Почта"

У данной обработки нет реквизитов, в основной форме определены две процедуры:

```
Процедура ОтправкаНажатие(Элемент)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    Почта=Новый Почта;

    Сообщ=Новый ПочтовоеСообщение;

    Сообщ.Текст="Здравствуйте дорогая Катерина
Матвеевна!";

    Сообщ.Тема="Письмо из 8-ки";
```

```
    Данные = Новый ДвоичныеДанные(Путь+"result.txt");

    Сообщ.Вложения.Добавить(Данные,"first.txt");

    Сообщ.Получатели.Добавить("sawa@analit.ru");

    Почта.Подключиться("dmg","111111");

    Почта.Послать(Сообщ,Ложь);

    Почта.Отключиться();

КонецПроцедуры
```

Прием почтовых сообщений

```
Процедура Прием(Кнопка)

    Почта=Новый Почта;

    Почта.Подключиться("dmg","111111");

    // истина - только непрочитанные, истина - только
конверты

    мсвПочта=Почта.Выбрать(Истина,Ложь);

    Для Каждого элМсв Из мсвПочта Цикл

        Сообщить("Тема: "+Строка(элМсв.Тема));

        Сообщить("Текст: "+Строка(элМсв.Текст));

        Для Каждого элМсвВл Из элМсв.Вложения Цикл

            Сообщить("     Вложение:
"+Строка(элМсвВл.Наименование));

        КонецЦикла;

    КонецЦикла;

КонецПроцедуры
```

Следует иметь ввиду, что если вы будете использовать возможности данного механизма для автоматической отправки почтовых сообщений, то нужно не забывать об особенностях системы защиты почтовых клиентов.

2.2.2. Объект "ИнтернетПочта"

В отличие от объекта "Почта", который работает по технологии MAPI и требует от пользователя установленного почтового клиента, контекст "ИнтернетПочта" использует наиболее распространенные интернет протоколы SMTP и POP3. Объект "ИнтернетПочта" не требует установленного почтового клиента и, если почтовый клиент все же установлен, работает с ним(и) параллельно.

Процедуры демонстрирующие основные принципы работы данного объекта приведены далее.

Настройка профиля с которым производится подключение к почтовому серверу:

```
Процедура ОпределитьПрофиль(Профиль)

    // определить профиль

    Профиль.АдресСервераSMTP = "mail.yandex.ru";

    Профиль.АдресСервераPOP3 = "mail.yandex.ru";

    Профиль.ПортSMTP = 25;

    Профиль.ПортPOP3 = 110;

    Профиль.Пользователь = "user";

    Профиль.Пароль = "password";

КонецПроцедуры
```

Отправка сообщения:

```
Процедура ИнтернетПочтаОтправка(Кнопка)

    //создание профиля

    Профиль = Новый ИнтернетПочтовыйПрофиль;

    ОпределитьПрофиль(Профиль);


    //создание сообщения

    Сообщение=Новый ИнтернетПочтовоеСообщение;

    Сообщение.Получатели.Добавить("sawa@analit.ru");

    Сообщение.Отправитель="dmg@analit.ru";

    Сообщение.Тема="Очень важно!!!";

    Сообщение.Тексты.Добавить("Привет!");


    Почта = Новый ИнтернетПочта;


    // подключение
```

```
    Попытка
        ИнтернетПочта.Подключиться(Профиль);
    Исключение
        Сообщить(ОписаниеОшибки());
        Предупреждение("Произошли ошибки при проверке
настроек учетной записи.
        |Описание ошибки приведено в окне сообщения.");
        Возврат;
    КонецПопытки;

    Почта.Послать(Сообщение);
    Почта.Отключиться();
КонецПроцедуры
```

Прием сообщений

```
Процедура ИнтернетПочтаПрием(Кнопка)
    //создание профиля
    Профиль = Новый ИнтернетПочтовыйПрофиль;
    ОпределитьПрофиль(Профиль);

    Почта = Новый ИнтернетПочта;

    // подключение
    Попытка
        ИнтернетПочта.Подключиться(Профиль);
    Исключение
        Сообщить(ОписаниеОшибки());
        Предупреждение("Произошли ошибки при проверке
настроек учетной записи.
        |Описание ошибки приведено в окне сообщения.");
```

```
            Возврат;
      КонецПопытки;

      // истина - удалять сообщения, массив заголовков
      Сообщения = Почта.Выбрать(Истина);
      Для Каждого Сообщение Из Сообщения Цикл
            Сообщить(Сообщение.Тема);
            // и т.д.
      КонецЦикла;
      Почта.Отключиться();
КонецПроцедуры
```

Выборочный прием

```
Процедура ИнтернетПочтаВыборочныйПрием(Кнопка)
      //создание профиля
      Профиль = Новый ИнтернетПочтовыйПрофиль;
      ОпределитьПрофиль(Профиль);

      НужноПолучить=Новый Массив();

      Почта = Новый ИнтернетПочта;

      // подключение
      Попытка
            ИнтернетПочта.Подключиться(Профиль);
      Исключение
            Сообщить(ОписаниеОшибки());
            Предупреждение("Произошли ошибки при проверке
настроек учетной записи.
            |Описание ошибки приведено в окне сообщения.");
```

```
        Возврат;
    КонецПопытки;

    // получение только заголовков сообщений
    Заголовки = Почта.ПолучитьЗаголовки();
    Если Заголовки.Количество()=0 Тогда
        Сообщить("Писем нет!");
        Возврат;
    КонецЕсли;

    Для Каждого Заголовок Из Заголовки Цикл
        // критерий по которому заполняем
        // массив НужноПолучить только нужными
    КонецЦикла;

    Сообщения = Почта.Выбрать(Истина,НужноПолучить);
    Для Каждого Сообщение Из Сообщения Цикл
        Сообщить(Сообщение.Тема);
        // и т.д.
    КонецЦикла;
    Почта.Отключиться();
КонецПроцедуры
```

Единственная сложность при работе с электронной почтой, это "не запутаться" в иерархии объектов.

2.3. Использование протоколов http, ftp

Если "расшифровать" названия данных протоколов обмена, то: http- протокол передачи гипертекста, ftp- протокол передачи файлов. В интернете именно с данных аббревиатур начинается "полный адрес" ресурса.

Для работы с данными протоколами в системе "1С:Предприятие 8" существуют соответствующие объекты: "HTTPСоединение" и "FTPСоединение".

2.3.1. Http соединение

В качестве примера использования разберем порядок работы обработки "HTTP".

У обработки определяется реквизит "URL" (тип "Строка", длина 60). В диалог помимо элемента управления, связанного с реквизитом "URL" размещается поле текстового документа (имя "Поле").

В модуле основной формы размещается процедура:

```
Процедура URLПриИзменении(Элемент)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    СерверИсточник = URL;

    Адрес="price.asp";

    ИмяВходящегоФайла=ПолучитьИмяВременногоФайла("html");


    Заголовки=Новый Соответствие();

    // указываются заголовки запроса


    НЗапрос=Новый HTTPЗапрос(Адрес,Заголовки);


    HTTP = Новый HTTPСоединение(СерверИсточник);

    НОтвет=HTTP.Получить(НЗапрос,ИмяВходящегоФайла);


    Если НОтвет.КодСостояния=200 Тогда


        ВходящийФайл = Новый Файл(ИмяВходящегоФайла);


    ЭлементыФормы.Поле.Прочитать(ИмяВходящегоФайла,Кодиров
каТекста.UTF8);



    КонецЕсли;

КонецПроцедуры
```

С помощью подобного механизма можно обращаться к каким-либо ресурсам: для получения из них данных, отправления собственных данных на

обработку. Тело HTTP запроса может быть установлено из строки, может быть установлено из файла.

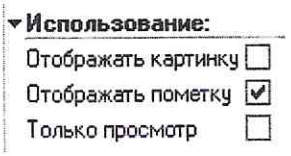
Следует отметить, что с использованием данного протокола реализован кросс-платформенный аналог механизму COM: "REST API" (позволяет обращаться к опубликованной на web сервере базе 1С:Предприятие для чтения, изменения, добавления данных). Для этой цели необходимо определенным образом настроить HTTP запрос. Выполнения запросов "GET" можно просмотреть в любом браузере, например, указав в поле URL строку следующего вида:

http://localhost/start/odata/standard.odata/Catalog_Номенклатура

2.3.2. Ftp соединение

Общие принципы работы похожи на работу с протоколом http (естественно с тем отличием, что ftp это протокол передачи файлов).

В качестве примера рассмотрим обработку "FTP". У данной обработки определен реквизит URL (тип "Строка", длина 100). В диалоге основной формы дополнительно размещен элемент управления "ПолеСписка" (имя "ПолеСписка"), при этом установлено свойство "Отображать пометку":



Кроме этого в командной панели определены две кнопки, при нажатии на которые будут вызываться следующие процедуры – обработчики событий:

Показ содержимого корневого каталога ftp сервера

```
Процедура Перейти(Кнопка)

    Сервер=Новый FTPСоединение(URL);

    МассивФайлов=Сервер.НайтиФайлы("/","*");

    Для Каждого Файл Из МассивФайлов Цикл

        Если Файл.ЭтоФайл() Тогда

            ПолеСписка.Добавить(Файл,Файл.Имя)

        КонецЕсли;

    КонецЦикла;

КонецПроцедуры
```

Получение отмеченных файлов:

```
Процедура Получить(Кнопка)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    Сервер=Новый FTPСоединение(URL);

    Для Каждого Файл Из ПолеСписка Цикл

        Если Файл.Пометка Тогда

    Сервер.Получить(Файл.Значение.ПолноеИмя,Путь+Файл.Знач
ение.ПолноеИмя);

        Файл.Пометка=Ложь;

        КонецЕсли;

    КонецЦикла;

КонецПроцедуры
```

3. Использование технологии OLE, COM

Механизмы OLE, COM позволяют обращаться из одних приложений к другим, использовать в рамках одного приложения функциональные возможности другого приложения. Например: из системы "1С:Предприятие 8" обращаться к Microsoft Excel, другому запуску системы "1С:Предприятие 8" и т.д.

3.1. Работа с Microsoft Excel

Поставим предварительно перед собой задачу: Прочитать из файла формата xls данные о ценах конкурентов.

Для этой цели определим обработку "КонкурентыExcel". Текст модуля основной формы следующий:

```
Процедура КнопкаВыполнитьНажатие(Элемент)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    Док=ПолучитьCOMОбъект(Путь+"конкуренты.xls");

    Для мНом=1 По 2 Цикл

        Контрагент=Док.Sheets(мНом).Cells(1,1).Value;

        Сообщить(Контрагент);

        Номен=Док.Sheets(мНом).Cells(2,2).Value;

        Цена=Док.Sheets(мНом).Cells(2,3).Value;

        счСтроки=3;

        Сообщить(Строка(Номен)+"-"+Строка(Цена));

        Пока СокрЛП(Номен) <> "" Цикл

    Номен=Док.Sheets(мНом).Cells(СчСтроки,2).Value;

    Цена=Док.Sheets(мНом).Cells(СчСтроки,3).Value;

    Сообщить(Строка(Номен)+"-"+Строка(Цена));

            счСтроки=счСтроки+1;

        КонецЦикла;

    КонецЦикла;

    Док.Application.Quit();

КонецПроцедуры
```

При написании (и отладке) механизмов взаимодействия по OLE (COM) (особенно при работе с Microsoft Office) можно рекомендовать использование конструкций:

```
Попытка

//тестируемый фрагмент

Исключение

     Док.Application.Quit();

КонецПопытки;
```

3.2. Назначение обработчиков событий на COM объекты

Механизм назначения обработчиков событий позволяет средствами встроенного языка назначать обработчики для событий прикладных объектов, наборов записей и COM-объектов. В качестве обработчика события может быть задан экспортируемый метод объекта или процедура/функция, находящаяся в области видимости.

```
Процедура Word(Кнопка)

     msword = Новый COMОбъект("Word.Application");

     ДобавитьОбработчик msword.DocumentBeforeClose,
ПриЗакрытииДокумента;

     msword.Visible = Истина;

КонецПроцедуры
```

Заголовок процедуры-обработчика события:

```
Процедура ПриЗакрытииДокумента(Документ,Отказ)

     // код обработчика

КонецПроцедуры
```

Подобные обработчики событий "срабатывают" после "штатных" (если они определены в самом объекте).

3.3. "1С:Предприятие 8" как OLE сервер

В качестве "внешнего" приложения может выступать и другой запуск программного комплекса "1С:Предприятие 8". Запустив внешнее "1С:Предприятие 8" можно обратиться к его данным, использовать формы, определенные во внешнем запуске.

В качестве иллюстрации рассмотрим механизм работы обработки "Ole8". Текст модуля основной формы следующий:

```
Процедура КнопкаВыполнитьНажатие(Элемент)

    V8=Новый COMОбъект("V82.Application");

    Попытка

    Открытие=V8.Connect("File=""c:\Путь…\"";Usr=""Usr1"";"
);

    Исключение

        Предупреждение("База данных не открыта!!!");

        Возврат;

    КонецПопытки;

    Сообщить(V8.Константы.Задваивать.Получить());

    Номен=V8.Справочники.Номенклатура;

    НовыйЭл=Номен.СоздатьЭлемент();

    НовыйЭл.Наименование="Новенький";

    НовыйЭл.Записать();

    Фрм=НовыйЭл.ПолучитьФорму();

    Фрм.ОткрытьМодально();

КонецПроцедуры
```

Следует отметить, что при запуске по технологии OLE Automation другого экземпляра системы "1С:Предприятие 8" в запускаемой конфигурации "срабатывают" обработчики событий модуля приложения. Из запускающего приложения можно получить доступ к написанным по определенным правилам процедурам и функциям (важно место описания, наличие ключевого слова "Экспорт"). Не доступен модуль внешнего соединения.

3.4. "1С:Предприятие 8" как COM сервер

Технология COM в общих чертах похожа на технологию Ole Automation (и то и другое относится к механизмам взаимодействия, но следует отметить, что эти механизмы существенно отличаются по архитектуре). С "пользовательской точки зрения", по сравнению с OLE технология COM намного быстрее. И в первую очередь это можно объяснить тем, что в COM из запускаемого приложения недоступны механизмы, связанные с интерфейсом (нельзя работать с формами и т.д.). При использовании COM в запускаемом "1С:Предприятии 8" задействуется модуль внешнего соединения.

Рассмотрим обработку "COM". Текст модуля формы следующий:

```
Процедура КнопкаВыполнитьНажатие(Элемент)

    V8=Новый COMОбъект("V82.ComConnector");

    Попытка

    Открытие=V8.Connect("File=""c:\.Путь..\"";Usr=""Usr1""
;");

    Исключение

        Предупреждение("База данных не открыта!!!");

        Возврат;

    КонецПопытки;

    Номен=Открытие.Справочники.Номенклатура;

    НовыйЭл=Номен.СоздатьЭлемент();

    НовыйЭл.Наименование="Старенький";

    НовыйЭл.Записать();

    Предупреждение(Открытие.ПолучитьДанные());

КонецПроцедуры
```

Для того, что бы при обращении к "1С:Предприятие" были доступны процедуры, функции общих модулей, у этих модулей должен быть отмечен флаг "Внешнее соединение".

3.5. Внешние источники данных

При эксплуатации информационной системы, построенной на базе "1С:Предприятия", могут возникать задачи, связанные с получением информации из внешних баз данных и использование этой информации в "1С:Предприятии" различным образом, например в виде отчетов или для каких-либо расчетов. Для решения такого рода задач в "1С:Предприятии" существует объект конфигурации "ВнешниеИсточникиДанных". Таких объектов может быть создано произвольное количество, каждый из этих объектов (в свою очередь) состоит из таблиц, а таблица – из полей.

Созданные, таким образом, объекты могут быть использованы следующим образом:

- в качестве источника данных для запросов;
- в качестве источника данных в системе компоновки данных;
- в качестве источника для динамических списков;
- записи таблиц могут отображаться в формах "1С:Предприятия";
- таблицы внешнего источника данных могут выступать в качестве типов реквизитов информационной базы;
- к таблицам (и полям) внешних источников данных можно применять права доступа и накладывать ограничения доступа к данным;
- доступ к таблицам и полям возможен из встроенного языка;
- таблица внешнего источника данных может входить в состав подсистем;
- таблица внешнего источника данных может входить в состав функциональных опций;
- для таблицы внешнего источника данных можно создавать характеристики.

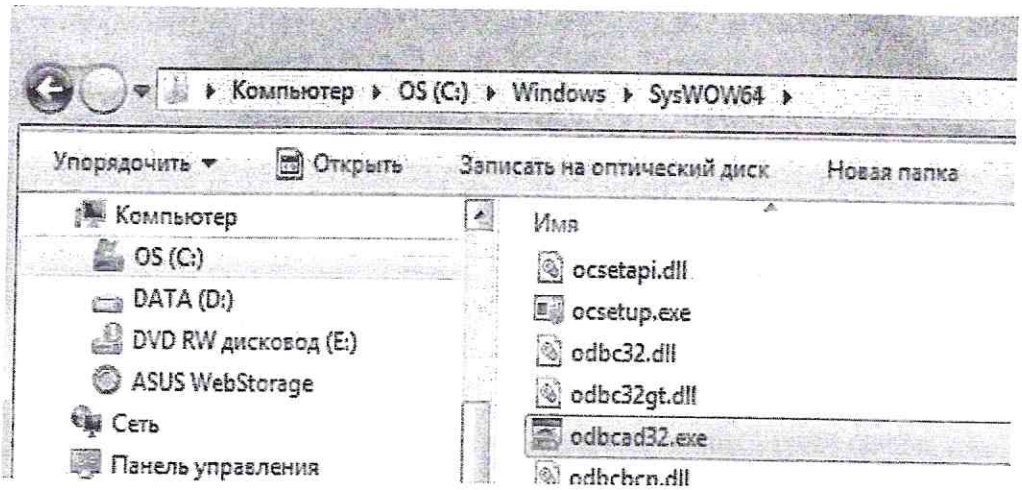
Для получения доступа к внешним источникам данных используется механизм ODBC. Данные внешних источников данных доступны как для чтения так и для записи (начиная с версии технологической платформы 8.3.5).

ВНИМАНИЕ! Механизм внешних источников данных не должен использоваться для доступа к базам данных "1С:Предприятия", так как модель данных "1С:Предприятия" не рассчитана на работу с данными на уровне физических структур хранения в СУБД

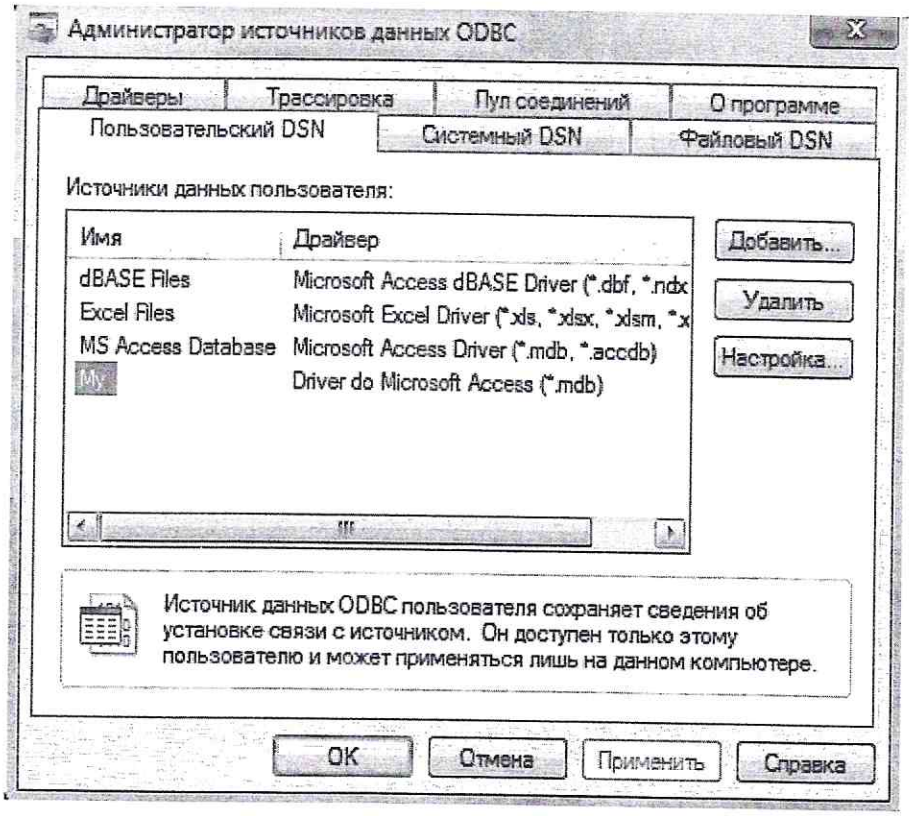
Посмотрим на эту возможность на практике.

3.5.1. Подключение к базе Access

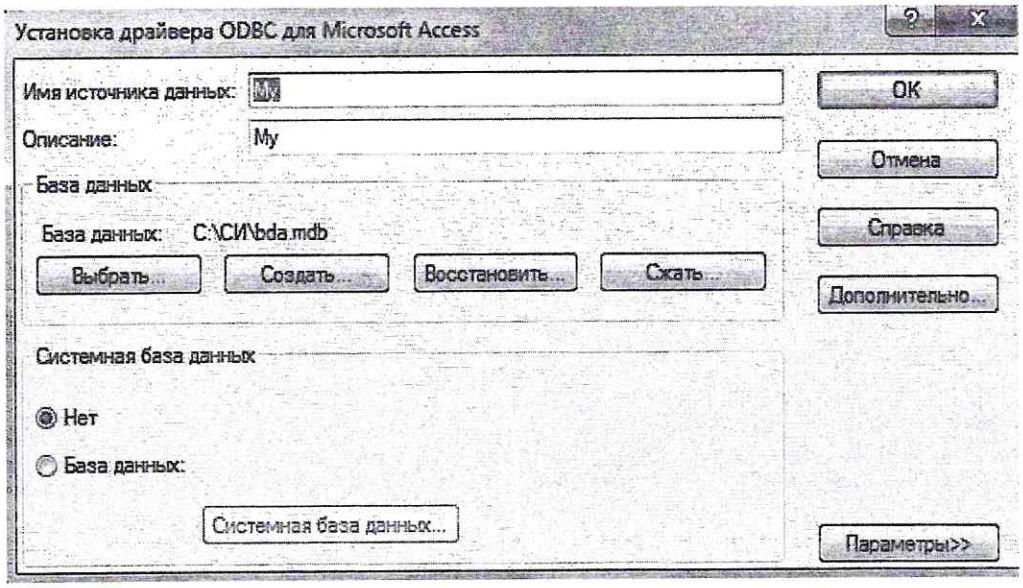
При настройке источников нужно обращать внимание на разрядность используемой операционной системы. В 32х разрядной ОС необходимо запустить оснастку по управлению источниками ODBC ("Панель управления/Администрирование"). Если используется 64х разрядная операционная система, то нужно запустить файл "odbcad32.exe".



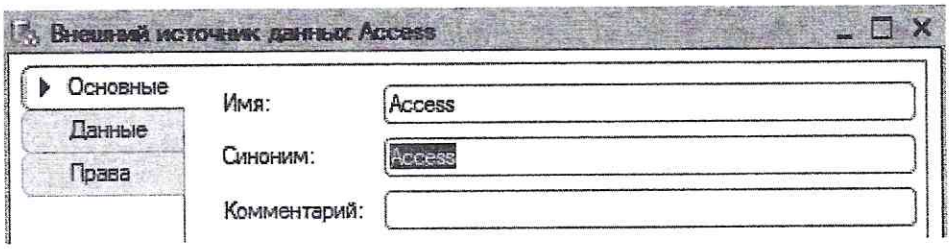
Внешний вид администратора следующий.



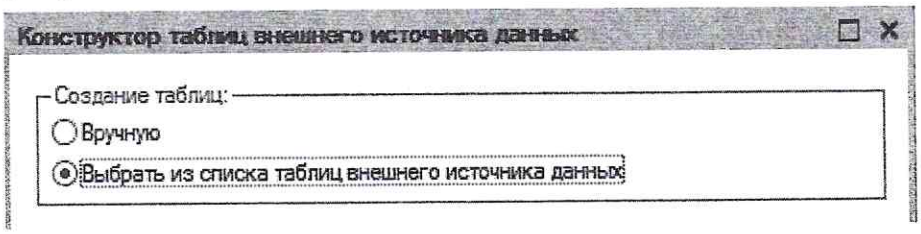
Добавим новый пользовательский источник, его настройки показаны ниже.



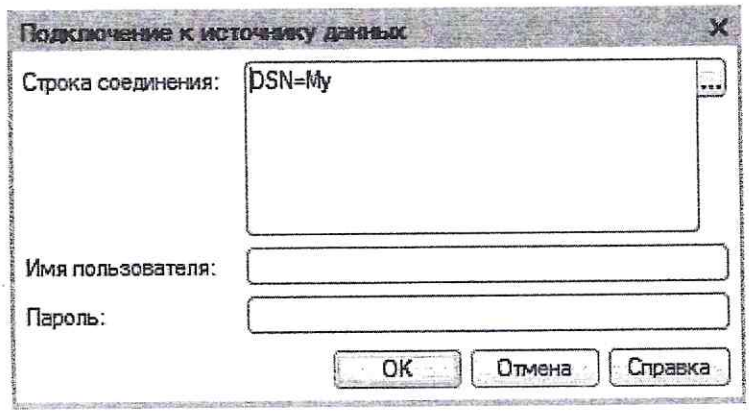
В конфигураторе определим новый внешний источник данных (объект конфигурации).



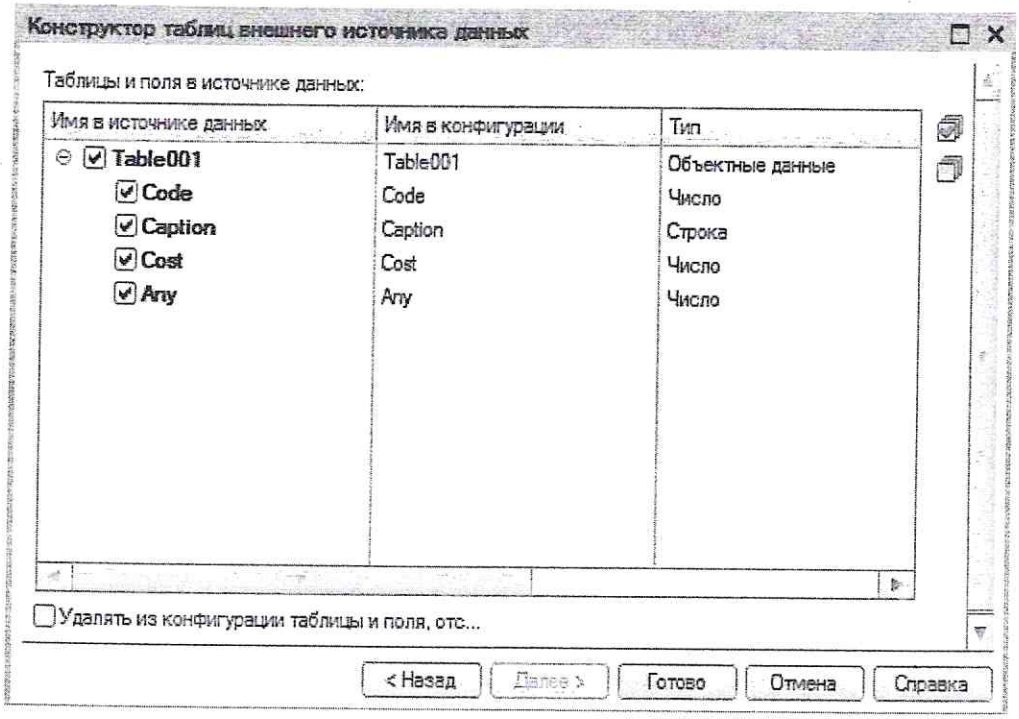
На закладке "Данные" создадим таблицу. В результате запустится конструктор таблиц внешнего источника.



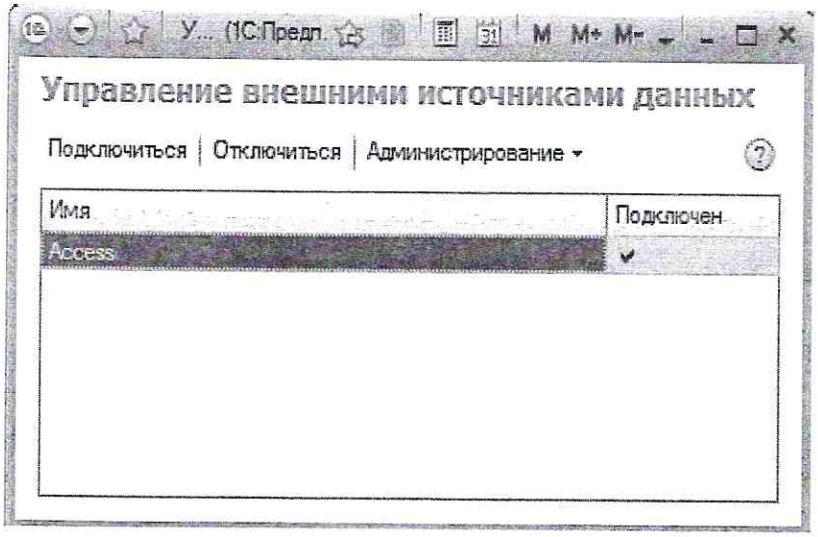
Для успешной работы конструктора необходимо задать строку подключения:



В результате можно просмотреть существующие в файле таблицы и выбрать, те которые планируем использовать (также можно определить состав используемых полей).



Для демонстрации возможности работы с внешними источниками будем использовать обработку "Внешние источники". Но перед запуском обработки необходимо установить подключение к внешнему источнику данных (Главное меню/Все функции/Стандартные).



Также следует помнить, что соединение к внешнему источнику данных следует указывать не только в Конфигураторе (если используется механизм импорта структуры таблиц из внешнего источника данных) но и в режиме 1С:Предприятия, для получения собственно данных (группа команд "Администрирование").

Создадим отчет. В схеме компоновки определим источник данных "Запрос". Текст запроса приводится ниже:

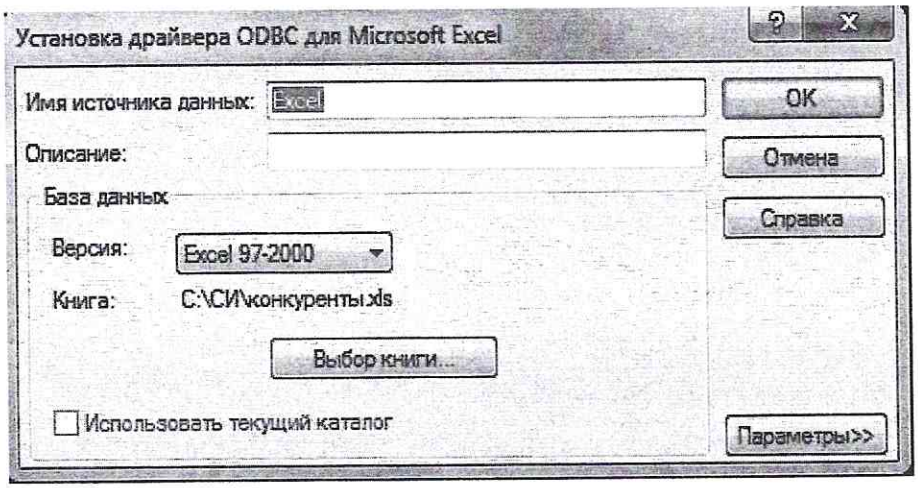
```
ВЫБРАТЬ

    Table001.Ссылка,

    Table001.Code,

    Table001.Caption,

    Table001.Cost,

    Table001.Any

ИЗ

    ВнешнийИсточникДанных.Access.Таблица.Table001 КАК
Table001
```

Остается проверить его работоспособность.

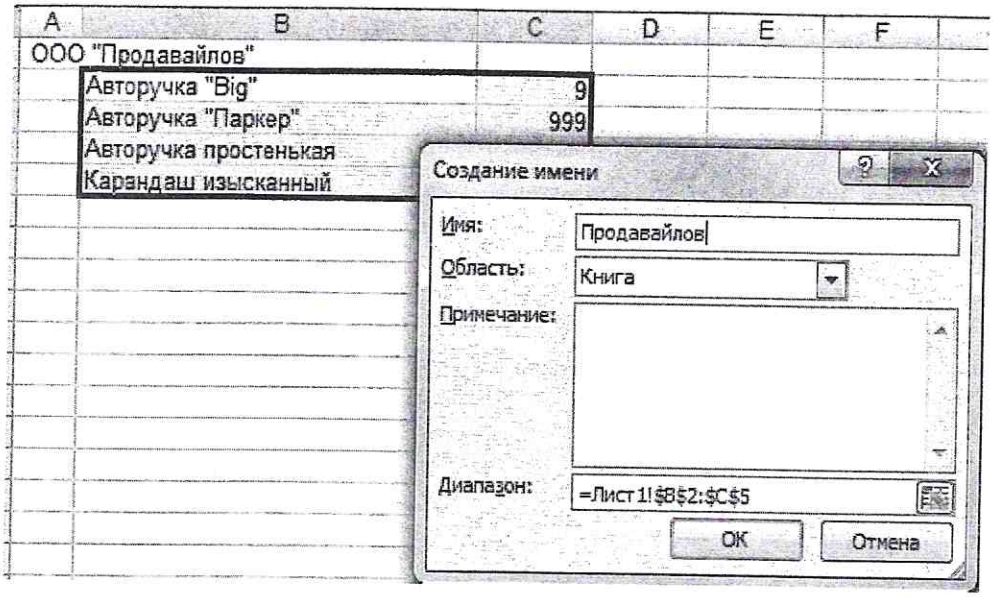
3.5.2. Подключение к Excel

Пример приведен "для разнообразия". Порядок работы с внешними источниками не поменялся.

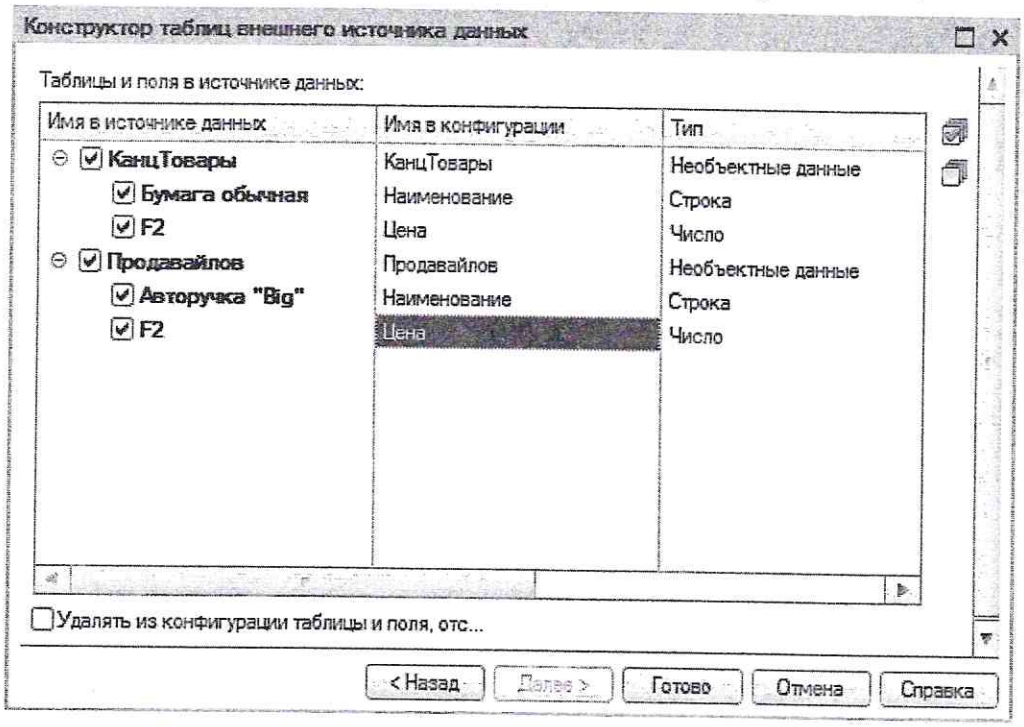
Настраиваем источник данных ODBC



В самом файле определяем имена диапазонов (они и будут выступать в качестве таблиц внешнего источника).



Создаем внешний источник (в конфигураторе), выбираем таблицы:



При желании осуществить подключение к внешнему условию программно можно использовать следующий фрагмент кода:

```
ВИД=ВнешниеИсточникиДанных.Excel.ПолучитьСостояние();

Если Вид=СостояниеВнешнегоИсточникаДанных.Отключен Тогда

    ПСВИД = Новый
ПараметрыСоединенияВнешнегоИсточникаДанных;

    ПСВИД.СтрокаСоединения= "DSN=Excel";

    ВнешниеИсточникиДанных.Excel.УстановитьОбщиеПараметрыС
оединения(ПСВИД);

    ВнешниеИсточникиДанных.Excel.УстановитьСоединение();

КонецЕсли;
```

3.5.3. Добавление данных во внешний источник

При использовании механизма внешних источников, данные в нем можно не только добавлять, но и обновлять. Приведем пример кода, добавляющего новую запись (без условно) в единственную таблицу демонстрационного источника:

```
    ВИД=ВнешниеИсточникиДанных.Access.ПолучитьСостояние();

    Если Вид=СостояниеВнешнегоИсточникаДанных.Отключен
Тогда

        Сообщить("Внешний источник Access не
подключен!");

        Возврат;

    КонецЕсли;


    МенеджерТаблицы=ВнешниеИсточникиДанных.Access.Таблицы.
Table001;

    Объект=МенеджерТаблицы.СоздатьОбъект();


    Объект.Code=5555;

    Объект.Caption="Создан из 1С:Предприятие";

    Объект.Cost=10000;

    Объект.Any=12345;

    Объект.Записать();
```

При решении реальных задач (особенно при обновлении данных во внешнем источнике) необходимо не забывать про транзакционные блокировки

4. Организация связи web приложения с информационной базой "1С:Предприятие"

Для организации подобной связи можно использовать ранее рассмотренный механизм COM. Основана эта возможность на том, что файлы, из которых "строится" web приложение, являются исполняемыми (с точки зрения web сервера). Т.е. при обращении к подобному файлу (например *.asp) web сервер его загружает и исполняется код, который прописан в загружаемом файле. В процессе исполнения формируется html, который возвращается клиенту. Так вот во время исполнения кода может производиться обращение к COM объектам (в том числе и к "1С:Предприятие").

Пример подобного обращения (файл default.asp)

```
<%@ Language=javascript %>

<%

    entConn = new ActiveXObject("v82.ComConnector");

    conn = entConn.connect("file=C:/web/base");

    Response.Write(conn.Price());

%>
```

Функция "Price()" определена в модуле внешнего соединения базы "1С:Предприятие" с ключевым словом "Экспорт". Она формирует html который содержит данные прайс-листа компании.

Следует отметить, что на диске ИТС есть более развернутый пример использования подобного механизма.

5. XML

Когда речь заходит об XML следует понимать, что это некий стандарт, язык (основа ряда технологий связанных с обменом), в составе которого можно выделить ряд взаимосвязанных частей:

- XML документ (содержит данные)
- Схема XML документа (описывает как данные лежат в документе)
- Пространство имен типов (описывает используемые типы и понятия)

Когда речь заходит об XML как об универсальном формате обмена, универсальность и заключается в наличии (их реализации) всех трех составных частей. Отсутствие какой либо части может внести неопределенность в интерпретацию данных на принимающей стороне.

Начнем разбираться с этими понятиями (попутно знакомясь с возможностями системы).

5.1. XML документ

XML документ представляет собой документ, в котором используются определенные форматирующие инструкции (сохраняется он в текстовом виде, поэтому для его просмотра в самом простом случае можно использовать "Блокнот" (Notepad)). По аналогии с HTML используемые форматирующие инструкции можно называть тэгами. Эта инструкция представляет собой выражение, заключенное в угловые скобки. К примеру: <Товар>. Обязательно существует открывающий и закрывающий тег. Закрывающий тег отличается от открывающего наличием символа "косой черты": </Товар>. Пара тегов реализует так называемый "элемент" (узел в ряде книг) XML документа.

```
<Начало>

    Содержимое

</Начало>
```

В качестве содержимого могут выступать другие элементы или строка.

При описании тегов важен регистр символов, в котором они написаны. Между открывающим и закрывающим тегами находится область содержания тега. Не допускается частичное перекрывание областей тегов (либо один полностью входит в другой, либо они не перекрываются). Если область одного элемента находится внутри области другого элемента, то считается, что первый подчинен другому.

Как было сказано выше, документ XML состоит из элементов. В правильном XML документе существует один главный элемент (единственный элемент документа), ему могут быть подчинены любое количество других элементов, которым могут быть подчинены в свою очередь другие элементы и т.д. У каждого элемента может быть свойственный только ему набор атрибутов (свойств). Атрибуты задаются с использованием пары "имя атрибута"="значение атрибута"

Пример XML документа:

```
<?xml version="1.0"?>

<Корневой Справочник="Номенклатура">

     <!--Формирование прайс листа-->

     <Элемент Код="1" Цена="100">Авторучка</Элемент>

     <Элемент Код="2" Цена="50">Карандаш</Элемент>

     <Элемент Код="3" Цена="99">Карандаш Big</Элемент>

</Корневой>
```

Можно сказать, что XML документ это дерево элементов (у каждого элемента может быть собственный набор свойств) с единственным корневым элементом.

В системе "1С:Предприятие 8" существует несколько механизмов, которые позволяют работать с XML документами.

Следует оговориться, что для работы с XML документами на самом деле существует специальный класс программного обеспечения, называемых парсерами (они выступают как "компоненты" операционной системы). Говоря о том, что "1С:Предприятие 8" может работать с XML-документами необходимо помнить, что "1С:Предприятие" работает "через" парсеры, устанавливаемые автоматически вместе с платформой.

5.2. Базовые средства работы с XML

Базовая подсистема работы с XML фактически отвечает за связь с файлом XML документа. С помощью средств этой подсистемы можно полноценно работать с документами, но делать это будет довольно сложно. Дело в том, что средства базовой подсистемы работают с XML документами в рамках модели последовательно доступа. Т.е. что при чтении, что при записи система последовательно работает с частями элемента (начало элемента, текст, конец элемента). Можно сказать что механизм производит последовательное чтение тегов.

В качестве примера использования данной подсистемы рассмотрим обработку "БазоваяПодсистемаXML".

Выгрузка данных:

```
//обработчик нажатия на кнопку "Выгрузить"

Процедура КнопкаВыполнитьНажатие(Элемент)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    Файл=Новый ЗаписьXML;

    Файл.ОткрытьФайл(Путь+"my_xml.xml");

    Файл.ЗаписатьОбъявлениеXML();

    Файл.ЗаписатьНачалоЭлемента("Корневой");

    Файл.ЗаписатьАтрибут("ВыгружаемыйСправочник","Номенкла
тура");

        Файл.ЗаписатьКомментарий("Выгрузка элементов
справочника");

    Выборка=Справочники.Номенклатура.Выбрать();

    Пока Выборка.Следующий() Цикл

        Если Не Выборка.ЭтоГруппа Тогда

            Файл.ЗаписатьНачалоЭлемента("Элемент");


    Файл.ЗаписатьАтрибут("Код",Строка(Выборка.Код));


    Файл.ЗаписатьАтрибут("Цена",Строка(Выборка.ЦенаПродажи
));

        Файл.ЗаписатьТекст(Выборка.Наименование);

            Файл.ЗаписатьКонецЭлемента();

        КонецЕсли;

    КонецЦикла;

    Файл.ЗаписатьКонецЭлемента();

    Файл.Закрыть();

КонецПроцедуры
```

Загрузка данных из ранее выгруженного файла

```
//обработчик нажатия на кнопку "Загрузить"
Процедура Загрузить(Кнопка)
    Путь= Константы.ПутьДоФайлов.Получить()+"\";
    Файл=Новый ЧтениеXML;
    Файл.ОткрытьФайл(Путь+"my_xml.xml");
    Отступ="  ";
    Пока Файл.Прочитать() Цикл
    Если Файл.ТипУзла=ТипУзлаXML.НачалоЭлемента Тогда
            Отступ=Отступ+"   ";
    Сообщить(Отступ+Строка(Файл.ЛокальноеИмя)+" тип: "
+Строка(Файл.ТипУзла));
            Пока Файл.ПрочитатьАтрибут() Цикл
Сообщить(Отступ+"атрибут:"+Файл.Имя+"="+Файл.Значение);
            КонецЦикла;
        КонецЕсли;
        Если Файл.ТипУзла=ТипУзлаXML.Текст Тогда
            Сообщить(Отступ+Строка(Файл.Значение));
        КонецЕсли;
    Если Файл.ТипУзла=ТипУзлаXML.КонецЭлемента Тогда
    Сообщить(Отступ+Строка(Файл.ЛокальноеИмя)+" тип: "
+Строка(Файл.ТипУзла));
        Отступ=Сред(Отступ,1,СтрДлина(Отступ)-3);
    КонецЕсли;
    КонецЦикла;
    Файл.Закрыть();
КонецПроцедуры
```

Фактически запись и чтение данных из XML документа производится "фрагментами" элементов. При этом наверное очевидно, что при возникновении

необходимости работы с документами, имеющими сложную структуру, реализация процедур будет также сильно усложняться.

Практикум № 4

Внесите изменения, в вышерассмотренные процедуры позволяющие выгружать и впоследствии отображать значения реквизита "ОсновнаяЕдиницаИзмерения" справочника "Номенклатура".

С другой стороны, как говорилось ранее, XML документ имеет текстовое наполнение. Т.е. при записи любого значения оно преобразуется к строке. В итоге на приемной стороне может возникнуть обратная задача. Можно пытаться решать эту задачу вручную, а можно решить воспользовавшись дополнительными средствами "1С:Предприятие 8".

5.3. XML сериализация

В "1С:Предприятие 8" реализована дополнительная подсистема: "XML сериализация".

Основная задача XML-сериализации — поддержка чтения/записи объектов данных 1С:Предприятия 8 в/из XML.

Базовые средства чтения и записи документов XML не предоставляют достаточной основы для решения данной задачи. Они не определяют форматов представления данных 1С:Предприятия 8 в XML и не предоставляют средств для чтения/записи объектов данных в/из XML в принятом формате как единого целого.

С точки зрения представления в XML типы значения можно разделить на "простые" и "сложные". К простым типам данных относятся типы, значения которых можно представить в виде элементов XML только с текстовым содержимым. Значения сложных типов представляются в виде элементов XML, содержащих вложенные элементы.

5.3.1. Простые типы

К простым типам (с точки зрения представления в XML) относятся

- Число
- Строка
- Булево
- ДвоичныеДанные
- Null
- УникальныйИдентификатор
- ХранилищеЗначения
- Все ссылки на объекты базы данных
- Ссылки на перечисления

Для работы с XML представлениями значений простых типов предназначены два метода глобального контекста:

- XMLСтрока()
- XMLЗначение()

Изменим ранее рассмотренную обработку. В строки кода, в которых производится установка значений атрибутов вместо метода "Строка()" подставим метод "XMLСтрока()".Например вместо:

```
Файл.ЗаписатьАтрибут("Цена",Строка(Выборка.ЦенаПродажи));
```

Установить

```
Файл.ЗаписатьАтрибут("Цена",XMLСтрока(Выборка.ЦенаПродажи))
;
```

При чтении значений нужно использовать следующую конструкцию:

```
Процедура Загрузить(Кнопка)

//.....

Пока Файл.ПрочитатьАтрибут() Цикл

    Если Файл.Имя="ОсновнаяЕдиницаИзмерения" Тогда

Сообщить(Отступ+"атрибут:"+Файл.Имя+"="+

XMLЗначение(Тип("СправочникСсылка.ЕдиницыИзмерения"),Файл.З
начение));

    Иначе

Сообщить(Отступ+"атрибут:"+Файл.Имя+"="+Файл.Значение);

    КонецЕсли;

КонецЦикла;
```

Сделайте изменения в ранее созданной обработке для атрибутов с ценой и основной единицей измерения. Сравните полученный результат с предыдущим.

Практикум № 5

Создайте новую обработку (скопировав полученную в результате предыдущих изменений). В данной обработке должен выгружаться справочник "Номенклатура", но при этом структура получаемого xml документа должна иметь следующий вид:

```
<?xml version="1.0" ?>
- <Корневой ВыгружаемыйСправочник="Номенклатура">
    <!-- Выгрузка элементов справочника -->
  - <Элемент Код="7" Наименование="Босоножки новые">
      <Цена>1000</Цена>
      <ОсновнаяЕдиницаИзмерения>efbfeaf3-831d-11d8-957e-00c026abdd5e</Основна
    </Элемент>
  - <Элемент Код="9" Наименование="Брюки женские">
      <Цена>1500</Цена>
      <ОсновнаяЕдиницаИзмерения>401f066f-8250-11d8-957c-00c026abdd5e</Основна
    </Элемент>
  - <Элемент Код="8" Наименование="Кросовки Reebok">
      <Цена>2500</Цена>
      <ОсновнаяЕдиницаИзмерения>efbfeaf3-831d-11d8-957e-00c026abdd5e</Основна
    </Элемент>
  - <Элемент Код="10" Наименование="Майка мужская">
      <Цена>150</Цена>
      <ОсновнаяЕдиницаИзмерения>401f066f-8250-11d8-957c-00c026abdd5e</Основна
    </Элемент>
  </Корневой>
```

При решении практикума пришлось столкнуться определенной проблемой. Когда вы позиционировались на элементе "Цена", вы знаете что "внутри" будет находиться значение цены. После исполнения метода "Прочитать()" вы позиционируетесь уже на значении и сама система забывает что находится "внутри" элемента "Цена". Т.е. нужно реализовать какие-либо механизмы "помнящие это" (чтобы знать, что мы прочитали и соответствующим образом обработать прочитанное значение).

5.4. Типы данных

При решении практикума возможно вы также задумались над вопросом: откуда на стороне выгрузки знают, что значение нужно загружать как "Число" или другой тип? В нашем случае все просто: "Мы знали…". А если получатель не знает заранее? Информацию о типе можно либо включать в сам XML документ, либо передавать в схеме XML документа (что это такое будем смотреть позже). Возникает следующий вопрос: каким образом указать название типа, чтобы не было неправильной интерпретации.

В XML (в самом стандарте) есть стандартные типы. Для удобства их классификации они разбиты на логические группы, каждая из которых называется пространством имен (имя соответствует адресу в интернете, где уже в произвольной форме, но желательно со ссылкой на международные стандарты описываются эти используемые типы). Соответственно для указания типа нужно указать имя пространства имен и имя типа в данном пространстве.

На данный момент в литературе (работа с XML в "1С:Предприятие") используются следующие описания пространств имен:

- xmlns:xsd="http://www.w3.org/2001/XMLSchema"
- xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
- xmlns:v8="http://v8.1c.ru/data"

Первые два пространства являются стандартными для XML. Последнее определено разработчиками платформы для типов, аналогов которых нет в стандартных типах XML.

Пример использования подобного описания:

```
<Data xsi:type="xsd:decimal"> 3.1415</Data>
```

Как понять какому типу "1С:Предприятие" какой тип XML соответствует и наоборот? Для получения подобной информации предназначены методы XMLТип(), XMLТипЗнч() и ИзXMLТипа()

Пример использования данных методов при выгрузке:

```
Файл.ЗаписатьНачалоЭлемента("Цена");

ТипXML = XMLТипЗнч(Выборка.ЦенаПродажи);

ПространствоИмен = ТипXML.URIПространстваИмен;

ИмяТипа = ТипXML.ИмяТипа;

Файл.ЗаписатьНачалоЭлемента(ИмяТипа,ПространствоИмен);

Файл.ЗаписатьТекст(XMLСтрока(Выборка.ЦенаПродажи));

Файл.ЗаписатьКонецЭлемента();

Файл.ЗаписатьКонецЭлемента();
```

Пример использования при чтении данных:

```
ПространствоИмен=Файл.URIПространстваИмен;

ИмяТипа=Файл.Имя;

ТипXML=Новый ТипДанныхXML(ИмяТипа,ПространствоИмен);
```

Для получения значения можно использовать следующую конструкцию:

```
XMLЗначение(ИзXMLТипа(ТипXML),Файл.Значение)
```

5.5. Сложные типы

К сложным, с точки зрения сериализации, типам относятся те типы, значения которых нельзя передать, используя один элемент xml документа.

К ним можно отнести:

- Тип
- ОписаниеТипов
- КонстантаМенеджерЗначения.Имя
- Все объекты базы данных, за исключением ПланОбменаОбъект.Имя
- Наборы записей регистров, последовательностей, перерасчетов
- УдалениеОбъекта

Для работы с такими типами данных могут использоваться следующие методы:

- ПрочитатьXML();
- ЗаписатьXML();
- ВозможностьЧтенияXML();

Следует отметить, что при чтении объекта получается именно объект, т.е. для его записи в базу необходимо воспользоваться методом Записать();

В качестве примера, показывающего возможность использования вышеуказанных методов, рассмотрим обработку "Сложные типы XML". В ней определен реквизит "Документ", тип "ДокументСсылка.Расходная" и в модуле формы определены следующие обработчики событий:

```
Процедура КнопкаВыполнитьНажатие(Элемент)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    Файл=Новый ЗаписьXML;

    Файл.ОткрытьФайл(Путь+"doc_xml.xml");

    Файл.ЗаписатьОбъявлениеXML();

    Объект=Документ.ПолучитьОбъект();

    ЗаписатьXML(Файл,Объект,НазначениеТипаXML.Явное);

    Файл.Закрыть();

КонецПроцедуры
```

Загрузка документа

```
Процедура Загрузить(Кнопка)
    Путь= Константы.ПутьДоФайлов.Получить()+"\";
    Файл=Новый ЧтениеXML;
    Файл.ОткрытьФайл(Путь+"doc_xml.xml");
    Если Файл.Прочитать() Тогда
        Если ВозможностьЧтенияXML(Файл) Тогда
            Объект=ПрочитатьXML(Файл);
            Объект.Записать();
        КонецЕсли;
    КонецЕсли;
    Файл.Закрыть();
КонецПроцедуры
```

Следует отметить, что с помощью рассматриваемых методов можно выгружать и значения простых типов.

К примеру, следующие строки кода:

```
Знач = "Строка такая";
ЗаписатьXML(Зп, Знач);
ЗаписатьXML(Зп, Знач, "Root", НазначениеТипаXML.Явное);
ЗаписатьXML(Зп, Знач, "Root", "urn:some-namespace");
```

Приведут к получению следующего результата:

```
<string>Строка такая</string>
<Root xsi:type="xsd:string">Строка такая</Root>
<d1p1:Root xmlns:d1p1="urn:some-namespace">Строка
такая</d1p1:Root>
```

Практикум № 6

Произведите изменения обработке, она должна выгружать и загружать все документы "Расходная" по выбранному контрагенту. Кроме этого в xml документ должна входить информация и о самом контрагенте.

5.5.1. Выгрузка и загрузка объектов с различающейся структурой

Методы ПрочитатьXML() и ЗаписатьXML() прекрасно справляются со своей задачей, в случае если объекты с которыми производится работа имеют одинаковую структуру метаданных. Рассмотрим пример реализации выгрузки и загрузки справочника "Номенклатура" из информационной базы источника в базу приемника

Обработка выгрузки

```
Процедура КнопкаВыполнитьНажатие(Элемент)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    Файл=Новый ЗаписьXML;

    Файл.ОткрытьФайл(Путь+"nom_xml.xml");

    Файл.ЗаписатьОбъявлениеXML();

    Файл.ЗаписатьНачалоЭлемента("Корневой");

    Выборка=Справочники.Номенклатура.Выбрать();

    Пока Выборка.Следующий() Цикл

        Если Выборка.ЭтоГруппа Тогда Продолжить;
КонецЕсли;

    Файл.ЗаписатьНачалоЭлемента("CatalogObject.Номенклатур
а");



    ЗаписатьXML(Файл,
Выборка.Ссылка.УникальныйИдентификатор(),
"Ref",НазначениеТипаXML.Явное);

    ЗаписатьXML(Файл, Выборка.Код,
"Code",НазначениеТипаXML.Явное);

    ЗаписатьXML(Файл, Выборка.Наименование,

 "Description",НазначениеТипаXML.Явное);

    ЗаписатьXML(Файл, Выборка.Услуга,

"Услуга",НазначениеТипаXML.Явное);

    ЗаписатьXML(Файл, Выборка.ОсновнаяЕдиницаИзмерения,

 "ОсновнаяЕдиницаИзмерения",НазначениеТипаXML.Явное);

    ЗаписатьXML(Файл, Выборка.ЕдиницаИзмеренияХранения,
```

```
"ЕдиницаИзмеренияХранения",НазначениеТипаXML.Явное);
        ЗаписатьXML(Файл, Выборка.ВнешнийКод,
"ВнешнийКод",НазначениеТипаXML.Явное);
        ЗаписатьXML(Файл, Выборка.ЦенаПродажи,
"ЦенаПродажи",НазначениеТипаXML.Явное);
        ЗаписатьXML(Файл, Выборка.Картинка,
"Картинка",НазначениеТипаXML.Явное);


            Файл.ЗаписатьКонецЭлемента();
    КонецЦикла;
    Файл.ЗаписатьКонецЭлемента();
    Файл.Закрыть();
КонецПроцедуры
```

Обработка загрузки:

```
Процедура Загрузить(Кнопка)
    Путь= Константы.ПутьДоФайлов.Получить()+"\";
    Файл=Новый ЧтениеXML;
    Файл.ОткрытьФайл(Путь+"nom_xml.xml");
    // читаем корневой
    Файл.Прочитать();
    // читаем элементы
    Пока Файл.Прочитать() Цикл
        Если Файл.Имя="CatalogObject.Номенклатура" Тогда
            Объект=ПрочитатьНоменклатуру(Файл);
            Объект.Записать();
        Иначе
            Если ВозможностьЧтенияXML(Файл) Тогда
                Объект=ПрочитатьXML(Файл);
```

```
                    Объект.Записать();

                КонецЕсли;

            КонецЕсли;

        КонецЦикла;

        Файл.Закрыть();

КонецПроцедуры


Функция ПрочитатьНоменклатуру(Файл)

        Файл.Прочитать();

        // работа со ссылкой документа

        ПолученнаяСсылка=ПрочитатьXML(Файл);

        Спр=Справочники.Номенклатура.ПолучитьСсылку(Новый
УникальныйИдентификатор(ПолученнаяСсылка));

        Элемент=Спр.ПолучитьОбъект();

        Если Элемент=Неопределено Тогда


        Элемент=Справочники.Номенклатура.СоздатьЭлемент();

            Элемент.УстановитьСсылкуНового(Спр);

        КонецЕсли;

        Код=ПрочитатьXML(Файл);

        Элемент.Наименование=ПрочитатьXML(Файл);

        Элемент.Услуга=ПрочитатьXML(Файл);

        Элемент.ОсновнаяЕдиницаИзмерения=ПрочитатьXML(Файл);

        Элемент.ЕдиницаИзмеренияХранения=ПрочитатьXML(Файл);

        Элемент.Артикул=ПрочитатьXML(Файл);

        Элемент.Цена=ПрочитатьXML(Файл);

        Элемент.Картинка=ПрочитатьXML(Файл);

        Возврат(Элемент);

КонецФункции
```

5.6. DOM модель работы с XML документами

Запись данных в XML документ

```
Запись=Новый ЗаписьXML;

Запись.ОткрытьФайл(СокрЛП(Константы.ПутьДоФайлов.Получить()
)+"\dom.xml");



ЗаписьДом = Новый ЗаписьDOM();

Дом=Новый
     ДокументDOM("http://v8.1c.ru/8.1/data/enterprise/curre
nt-config","Корневой");



Корневой=Дом.ЭлементДокумента;



Элемент=Дом.СоздатьЭлемент("Имя");

Элемент.ТекстовоеСодержимое="Текст";

Элемент.УстановитьАтрибут("Свойство","Значение");



Корневой.ДобавитьДочерний(Элемент);



ЗаписьДом.Записать(Дом,Запись);

Запись.Закрыть();
```

Чтение данных из документа:

```
Чтение=Новый ЧтениеXML;

Чтение.ОткрытьФайл(СокрЛП(Константы.ПутьДоФайлов.Получить()
)+"\obj.xml");



НовыйПостроительДом=Новый ПостроительDOM;

НовыйДокументДом=НовыйПостроительДом.Прочитать(Чтение);



Корневой=НовыйДокументДом.ЭлементДокумента;
```

```
Дочерние=Корневой.ДочерниеУзлы;
Для ИндексУзла=0 По Дочерние.Количество()-1 Цикл

    Элемент=Дочерние.Элемент(ИндексУзла);
    Если Элемент.ТипУзла=ТипУзлаDOM.Элемент Тогда
        ИмяЭлемента=Элемент.ИмяЭлемента;
        Текст=Элемент.ТекстовоеСодержимое;
        Сообщить(ИмяЭлемента+":"+Текст);
    КонецЕсли;
КонецЦикла;
```

5.7. Xsl преобразование (XSLT)

XSLT позволяет задавать преобразование xml документов в другие документы (например vrml, html).

Преобразование, выраженное через XSLT, описывает правила преобразования исходного дерева xml документа в конечное дерево другого документа. Преобразование строится путем сопоставления образцов и шаблонов. Образец сравнивается с элементами исходного дерева, а шаблон используется для создания частей конечного дерева. Структура конечного дерева может полностью отличаться от структуры исходного дерева. В ходе построения конечного дерева элементы исходного дерева могут подвергаться фильтрации и переупорядочению, также может быть добавлена новая структура.

В качестве примера использования преобразования рассмотрим следующий XML документ:

```
<?xml version="1.0"?>
<?xml-stylesheet type='text/xsl' href='st1.xsl'?>
<Корневой Действие="Передача товара">
    <!--Шапка документа-->
    <Шапка ДатаД="25.11.2003 18:17:38" Номер="1"
ДляКого="Продавцов"/>
    <Товары>
        <Строка Товар="Карандаш" Количество="1" Цена="10"
Сумма="10"/>
```

```
        <Строка Товар="Ручка" Количество="4" Цена="15"
Сумма="60"/>

    </Товары>

</Корневой>
```

Если данный документ определить без строки №2, то при просмотре его IE, можно будет увидеть следующую "картину":

```
<?xml version="1.0" ?>
- <Корневой Действие="Передача товара">
    <!-- Шапка документа -->
    <Шапка ДатаД="25.11.2003 18:17:38" Номер="1" ДляКого="Продавцов" />
  - <Товары>
      <Строка Товар="Карандаш" Количество="1" Цена="10" Сумма="10" />
      <Строка Товар="Ручка" Количество="4" Цена="15" Сумма="60" />
    </Товары>
  </Корневой>
```

Если определить XML документ со второй строкой, и при этом файл st1.xsl будет содержать в себе:

```
<?xml version="1.0" encoding="windows-1251"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output indent="yes" method="html"/>



<xsl:template match="/">

    <html><head><title>

        <xsl:apply-templates select="/Корневой"/>

    </title></head>



    <body>

        <xsl:apply-templates select="/Корневой/Шапка"/>

        <xsl:apply-templates select="/Корневой/Товары"/>

    </body></html>

</xsl:template>



<xsl:template match="/Корневой">

    Действие: <xsl:apply-templates select="@Действие"/>
```

```
</xsl:template>


<xsl:template match="/Корневой/Шапка">

<table cellspacing="2" bgcolor="#00FF00" width="100%">

    <tr><td>Документ №:<b><xsl:value-of
select="@Номер"/></b>от:<b><xsl:value-of
select="@ДатаД"/>"</b></td></tr>

 </table>

</xsl:template>


<xsl:template match="/Корневой/Товары">

<table cellspacing="2" bgcolor="#FCFCCD" width="100%">

        <xsl:apply-templates select="Строка"/>

 </table>

</xsl:template>


<xsl:template match="Строка">

    <tr bgcolor="#00FF00">

        <td width="120" align="center"><xsl:value-of
select="@Товар"/></td>

        <td><xsl:value-of select="@Количество"/></td>

        <td><xsl:value-of select="@Цена"/></td>

        <td><xsl:value-of select="@Сумма"/></td>

    </tr>

</xsl:template>

</xsl:stylesheet>
```

То при просмотре в IE у документа будет иметь совершенно другой вид:

В данном примере, для "задания" преобразования использовалась директива в XML документе (причем проводилось не само преобразование документа, а "преобразование" отображения документа). Преобразование в системе "1С:Предприятие 8" можно осуществлять с помощью специального объекта "ПреобразованиеXML".

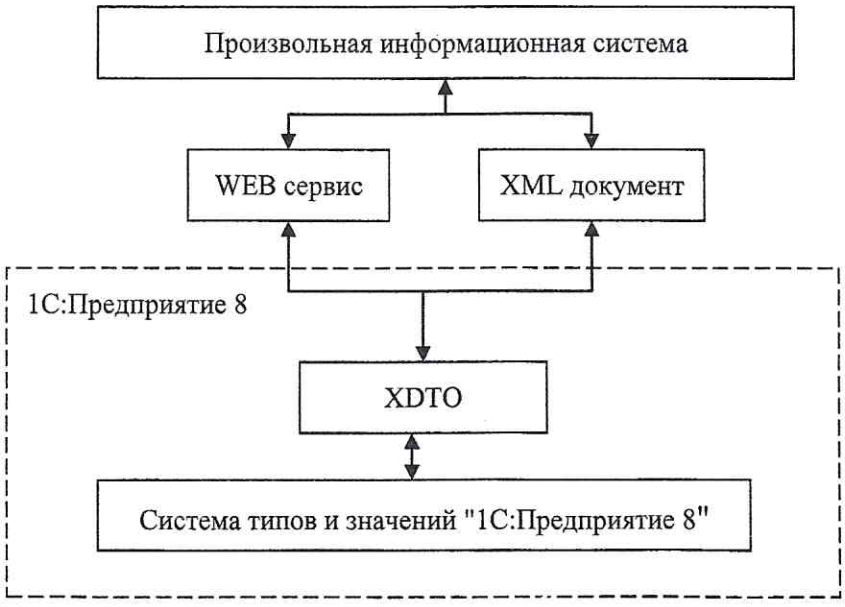
Для иллюстрации механизма использования преобразования рассмотрим обработку "Преобразование". В диалоге определен элемент управления ПолеHTMLДокумента (имя ПолеHTML) и поле текстового документа (имя "Поле"). В модуле формы определен текст процедуры:

```
Процедура Преобразование(Кнопка)

    ЭлементыФормы.ПолеHTML.Перейти("c:/ish.xml");

    Преобразование=Новый ПреобразованиеXSL;

    Преобразование.ЗагрузитьИзФайла("c:\st1.xsl");

    ЗначениеПреобразования=Преобразование.ПреобразоватьИзФ
айла("c:\apt.xml");

    ЭлементыФормы.Поле.УстановитьТекст(ЗначениеПреобразова
ния);

    ЭлементыФормы.Поле.Записать("c:\obr.html");

    Предупреждение("Продолжить");

    ЭлементыФормы.ПолеHTML.Перейти("c:/vrm.vrml");

КонецПроцедуры
```

В качестве заключения хочется отметить, что в общем случае можно производить преобразование не только в HTML документы, но и в документы других форматов...

5.8. XDTO

Механизм XDTO (XML Data Transfer Objects) является универсальным способом представления данных для взаимодействия с различными внешними системами.



Данная модель позволяет довольно просто манипулировать данными XML в "1С:Предприятие 8" и при этом хорошо приспособлена для прозрачного преобразования данных в другие форматы (в основном в XML). При работе с данной моделью используется техника работы с типами, значениями, привычная для программиста, владеющего встроенным языком "1С:Предприятие 8".

5.8.1. Фабрика XDTO

Фабрика XDTO является ключевым понятием механизма XDTO. Она содержит описание всех типов, с которыми оперирует данная некоторая система. В любой конфигурации "1С:Предприятие 8" существует глобальная фабрика XDTO, которая описывает все типы, используемые в конфигурации в терминах XDTO (эта фабрика доступно через свойство глобального контекста Фабрика XDTO).

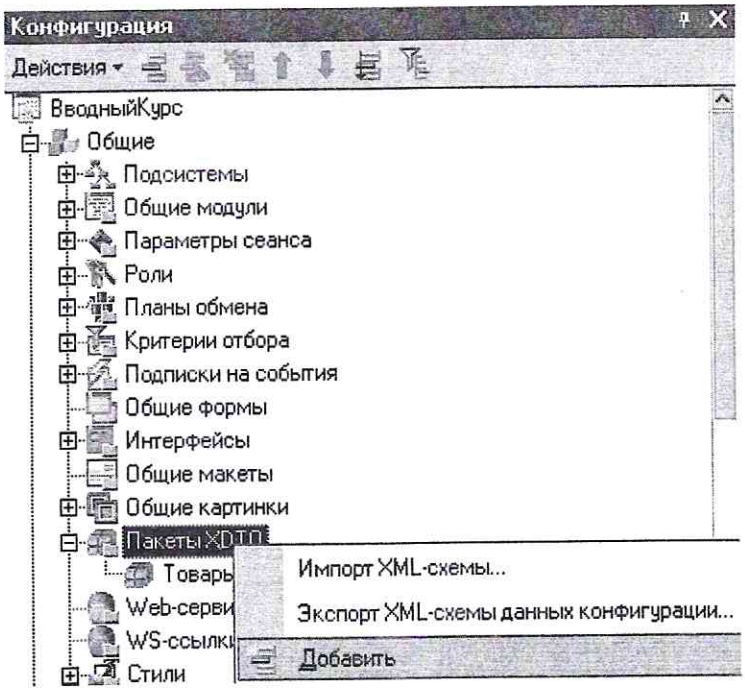
Все описания типов сгруппированы в один или несколько пакетов. Фабрика является полностью самодостаточной. Т.е. любой из типов, зарегистрированных в фабрике XDTO может ссылаться только на типы из той же самой фабрики.

Фабрика может быть создана программно:

```
НоваяФабрика=Новый ФабрикаXDTO(НаборСхем)
```

В этом случае описанные в этой фабрике типы не подлежат дополнению (создаются единовременно).

В отличии от произвольной фабрики XDTO, глобальная фабрика XDTO создается системой автоматически при создании информационной базы, и допускает добавление типов XDTO (для этого используется соответствующий объект внутри ветви "Общие" дерева конфигурации).



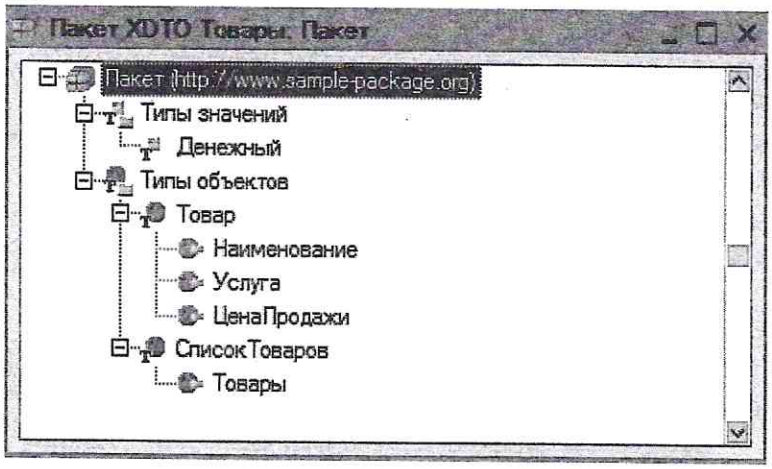
В общем, все пакеты, содержащиеся в глобальной фабрике XDTO можно разделить на три группы:

- Пакет XDTO, содержащий описания типов платформы. Он одинаков для всех конфигураций "1С:Предприятие 8.1".
- Пакет XDTO, содержащий описание типов конфигурации, созданных в результате редактирования метаданных
- Один или несколько пакетов, определенных внутри ветви "Общие/ Пакеты XDTO"

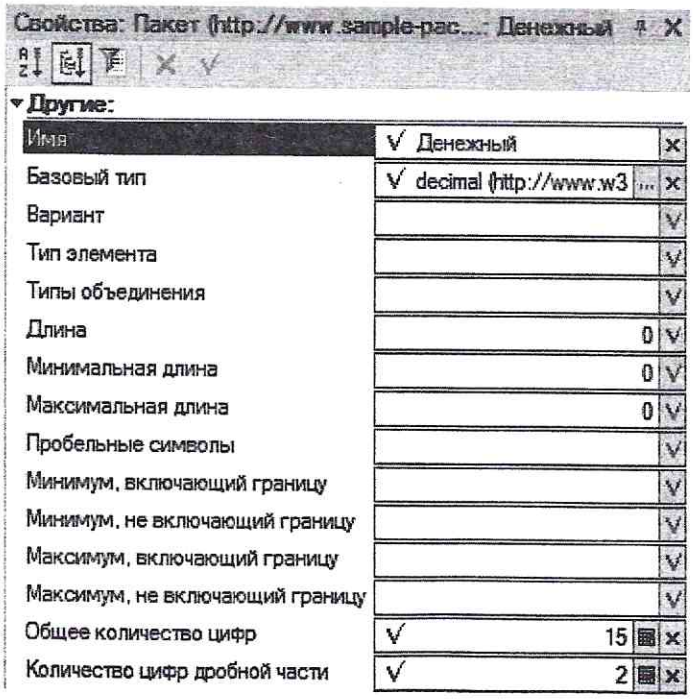
Каждый из типов данных XDTO является либо типом значения XDTO, либо типом объекта XDTO. Соответственно для описания типа значения используется объект типа "ТипЗначенияXDTO", а для описания типа объекта "ТипОбъектаXDTO".

5.8.2. Выгрузка данных посредством XDTO в xml документ

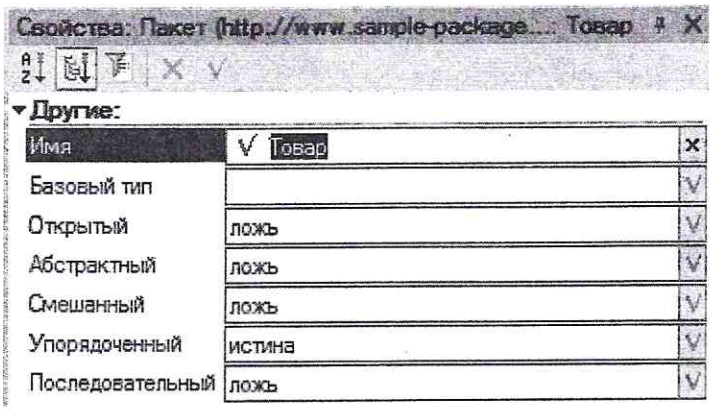
Создадим новый пакет XDTO (пространство имен "http://www.sample-package.org"). Пакет будет иметь имя "Товары".



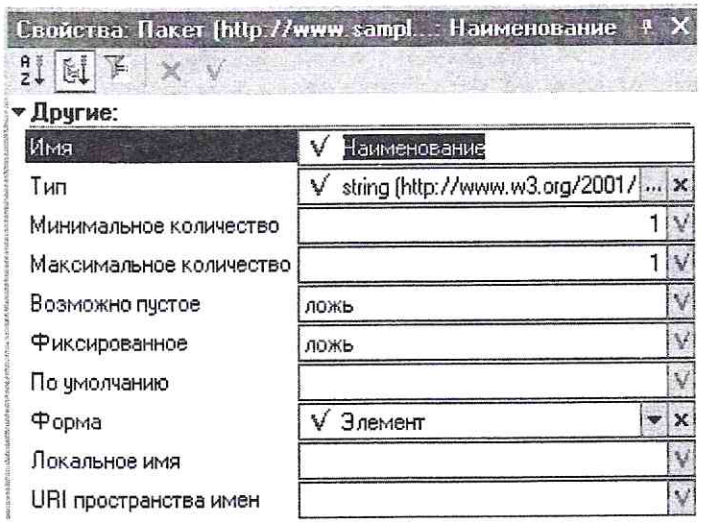
Определим тип значения "Денежный" (будет использоваться для передачи цены номенклатурной позиции)



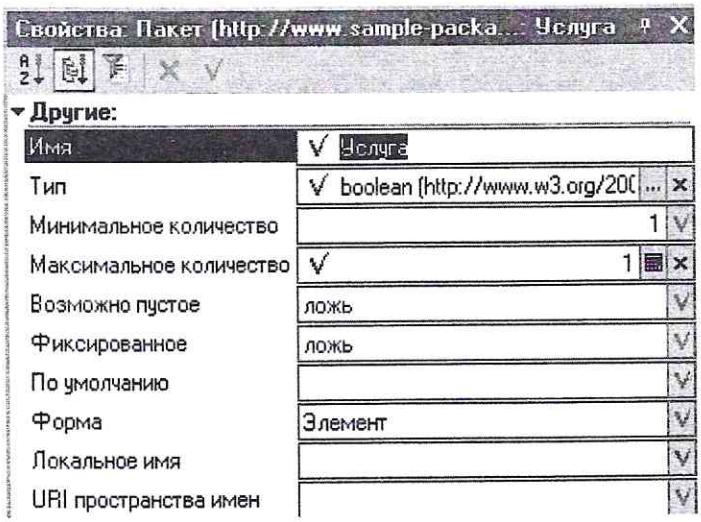
Добавим в пакет тип объекта "Позиция". Свойства приведены на рисунке.



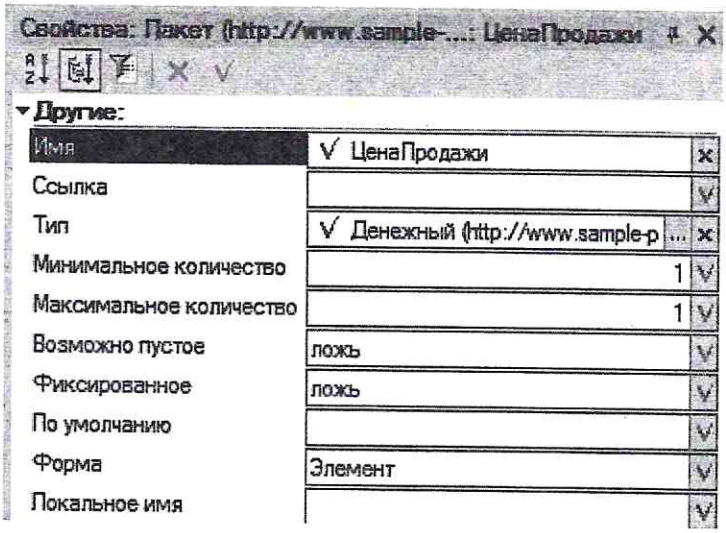
Сделав текущим добавленный тип объекта, добавим свойство объекта "Наименование".



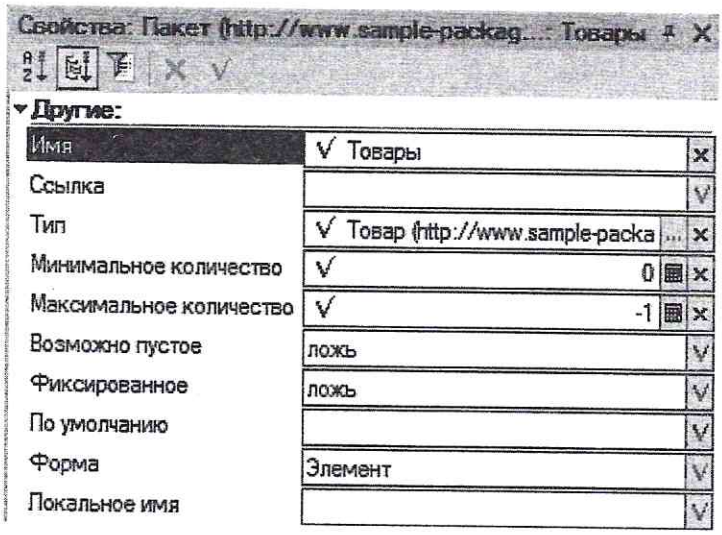
Используем тип string из пространства имен "http://www.w3.org/2001/XMLSchema". После этого таким же образом определяем еще одно свойство: "Услуга" (тип "boolean" из того же пространства имен).



Определим следующее свойство:



Добавим второй тип объекта: "СписокТоваров", у него определим новое свойство "Товары". Тип этого свойства: ранее определенный тип объекта "Товар".



Рассмотрим обработку "XDTO" (обработчик выгрузки данных).

```
ТоварТип = ФабрикаXDTO.Тип("http://www.sample-package.org",
"Товар");

СписокТип = ФабрикаXDTO.Тип("http://www.sample-
package.org", "СписокТоваров");



Корень = ФабрикаXDTO.Создать(СписокТип);

Выборка=Справочники.Номенклатура.Выбрать();

Пока Выборка.Следующий() Цикл

      Если Выборка.ЭтоГруппа Тогда Продолжить; КонецЕсли;

      Элемент=ФабрикаXDTO.Создать(ТоварТип);
```

```
    Элемент.Наименование=Выборка.Наименование;

    Элемент.Услуга=Выборка.Услуга;

    Элемент.ЦенаПродажи=Выборка.ЦенаПродажи;


    Корень.Товары.Добавить(Элемент);

КонецЦикла;


Запись=Новый ЗаписьXML;

Запись.ОткрытьФайл(СокрЛП(Константы.ПутьДоФайлов.Получить()
)+"\new.xml");

ФабрикаXDTO.ЗаписатьXML(Запись,Корень,,,,НазначениеТипаXML.
Явное);

Запись.Закрыть();
```

Проверьте механизм на практике.

5.8.3. Чтение данных посредством XDTO из xml документа

Текст обработчика загрузки данных обработки "XDTO":

```
Чтение=Новый ЧтениеXML;

Чтение.ОткрытьФайл(СокрЛП(Константы.ПутьДоФайлов.Получить()
)+"\new.xml");

Данные=ФабрикаXDTO.ПрочитатьXML(Чтение);


Если Данные = Неопределено Тогда

    Возврат;

КонецЕсли;


Для Каждого Элемент Из Данные.Товары Цикл

    Сообщить(Элемент.Наименование);

    Сообщить(Строка(Элемент.Услуга));

    Сообщить(Строка(Элемент.ЦенаПродажи));

КонецЦикла;
```

Проверьте механизм на практике.

Практикум № 7

Проведите соответствующие изменения, как в пакете XDTO, так и в процедурах выгрузки, загрузки которые позволят выгружать/загружать данные о дополнительных свойствах товаров (данные хранятся в регистре сведений "Значения свойств номенклатуры").

5.9. Импорт, экспорт схем XML

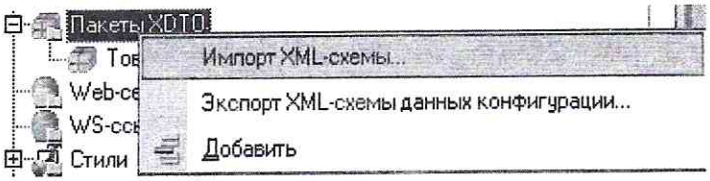
Объект конфигурации "ПакетXDTO" может быть создан "вручную", а может быть загружен из схемы XML документа (может быть выгружен в схему).

К примеру, если воспользоваться файлом схемы следующего вида:

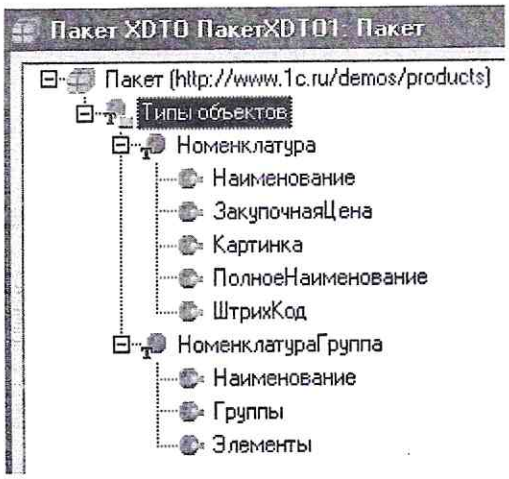
```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.1c.ru/demos/products"
attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://www.1c.ru/demos/products">

<xsd:complexType name="Номенклатура">

     <xsd:sequence>

          <xsd:element name="Наименование"
type="xsd:string"/>

          <xsd:element name="ЗакупочнаяЦена"
type="xsd:int"/>

          <xsd:element minOccurs="0" name="Картинка"
type="xsd:base64Binary"/>

          <xsd:element minOccurs="0"
name="ПолноеНаименование" type="xsd:string"/>

          <xsd:element name="ШтрихКод" type="xsd:string"/>

     </xsd:sequence>

</xsd:complexType>

<xsd:complexType name="НоменклатураГруппа">

     <xsd:sequence>

          <xsd:element name="Наименование"
type="xsd:string"/>

          <xsd:element maxOccurs="unbounded" minOccurs="0"
name="Группы" type="tns:НоменклатураГруппа"/>
```

```
        <xsd:element maxOccurs="unbounded" minOccurs="0"
name="Элементы" type="tns:Номенклатура"/>

    </xsd:sequence>

    </xsd:complexType>

</xsd:schema>
```

После выполнения загрузки:



Будет создан следующий пакет XDTO



В результате экспорта пакета "Товары" будет получена следующая схема xml документа:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.sample-package.org"
attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://www.sample-package.org">

    <xsd:complexType name="Позиция">

        <xsd:sequence>

            <xsd:element name="Наименование"
type="xsd:string"/>

            <xsd:element name="Услуга"
type="xsd:boolean"/>

        </xsd:sequence>

    </xsd:complexType>

    <xsd:complexType name="Состав">
```

```
            <xsd:sequence>

                  <xsd:element maxOccurs="unbounded"
minOccurs="0" name="Товар" type="tns:Позиция"/>

            </xsd:sequence>

      </xsd:complexType>

</xsd:schema>
```

5.10. Программное создание фабрики XDTO

Для того, чтобы работать с xml документами (посредством XDTO) совсем не обязательно для каждой схемы создавать соответствующий объект конфигурации "ПакетXDTO". Можно воспользоваться возможностью программно создать фабрику XDTO из полученной схемы.

Рассмотрим это на примере выгрузки данных по схеме "ДанныеНоменклатуры.xsd".

```
Чтение=Новый ЧтениеXML;

Чтение.ОткрытьФайл(СокрЛП(Константы.ПутьДоФайлов.Получить()
)+"\ДанныеНоменклатуры.xsd");



НовыйПостроительДом=Новый ПостроительDOM;

НовыйДокументДом=НовыйПостроительДом.Прочитать(Чтение);



НовыйПостроительСхем=Новый ПостроительСхемXML;

НоваяСхема=НовыйПостроительСхем.СоздатьСхемуXML(НовыйДокуме
нтДом);



НаборСхем=Новый НаборСхемXML;

НаборСхем.Добавить(НоваяСхема);

НоваяФабрика=Новый ФабрикаXDTO(НаборСхем);



НоменклатураТип =
НоваяФабрика.Тип("http://www.1c.ru/demos/products",
"Номенклатура");
```

```
НоменклатураГруппаТип =
НоваяФабрика.Тип("http://www.1c.ru/demos/products",
"НоменклатураГруппа");



КорневаяГруппа =
НоваяФабрика.Создать(НоменклатураГруппаТип);

КорневаяГруппа.Наименование = "Корневая";



Номенклатура = НоваяФабрика.Создать(НоменклатураТип);

Номенклатура.Наименование =
НоменклатураСсылка.Наименование;

Номенклатура.ЗакупочнаяЦена = 1000;

Номенклатура.ПолноеНаименование =
НоменклатураСсылка.Наименование;

Номенклатура.ШтрихКод = НоменклатураСсылка.Код;



КорневаяГруппа.Элементы.Добавить(Номенклатура);



Запись=Новый ЗаписьXML;

Запись.ОткрытьФайл(СокрЛП(Константы.ПутьДоФайлов.Получить()
)+"\out.xml");

НоваяФабрика.ЗаписатьXML(Запись,КорневаяГруппа,,,,Назначени
еТипаXML.Явное);

Запись.Закрыть();
```

При загрузке данных действия по загрузке схемы XML документа те же. Сама загрузка данных выполняется "обычным" способом.

5.11. "Смешанная" модель в XDTO.

Выгрузка данных:

```
ТоварТип = ФабрикаXDTO.Тип("http://www.sample-package.org",
"Позиция");

СоставТип = ФабрикаXDTO.Тип("http://www.sample-
package.org", "Состав");



Запись=Новый ЗаписьXML;

Запись.ОткрытьФайл(СокрЛП(Константы.ПутьДоФайлов.Получить()
)+"\new.xml");

Запись.ЗаписатьОбъявлениеXML();



Запись.ЗаписатьНачалоЭлемента("Контейнер");

Запись.ЗаписатьСоответствиеПространстваИмен("","http://www.
sample-package.org");

Запись.ЗаписатьСоответствиеПространстваИмен("xsd","http://w
ww.w3.org/2001/XMLSchema");

Запись.ЗаписатьСоответствиеПространстваИмен("xsi","http://w
ww.w3.org/2001/XMLSchema-instance");



Для Счетчик=1 По 100000 Цикл



Элемент=ФабрикаXDTO.Создать(ТоварТип);

Элемент.Наименование="Одинаковое наименование";

Элемент.Услуга=Ложь;



ФабрикаXDTO.ЗаписатьXML(Запись,Элемент,,,,

НазначениеТипаXML.Явное);

КонецЦикла;

Запись.ЗаписатьКонецЭлемента();

Запись.Закрыть();
```

Чтение данных:

```
Чтение=Новый ЧтениеXML;

Чтение.ОткрытьФайл(СокрЛП(Константы.ПутьДоФайлов.Получить()
)+"\new.xml");



Чтение.Прочитать();

Чтение.Прочитать();



Пока Чтение.ТипУзла<>ТипУзлаXML.КонецЭлемента Цикл

Данные=ФабрикаXDTO.ПрочитатьXML(Чтение);



Если Данные = Неопределено Тогда

     Прервать;

КонецЕсли;



//Сообщить(Элемент.Наименование);

//Сообщить(Строка(Элемент.Услуга));

КонецЦикла;
```

5.11.1. XML сериализация на основе XDTO

Значения типов конфигурации "1С:Предприятие 8.1" могут быть сериализованы непосредственно в/из файл/а XML на основе механизма XDTO.

В качестве иллюстрации рассмотрим процедуру выгрузки данных:

```
Объект=НоменклатураСсылка.ПолучитьОбъект();

НовыйСХDTO=Новый СериализаторXDTO(ФабрикаXDTO);

Запись=Новый ЗаписьXML;

Запись.ОткрытьФайл(СокрЛП(Константы.ПутьДоФайлов.Получ
ить())+"\obj.xml");

Запись.ЗаписатьОбъявлениеXML();

НовыйСХDTO.ЗаписатьXML(Запись,Объект,НазначениеТипаXML
.Явное,ФормаXML.Элемент);

Запись.Закрыть();
```

В результате получен xml документ следующего вида:

```
<?xml version="1.0" encoding="UTF-8"?>

<CatalogObject.Номенклатура
xmlns="http://v8.1c.ru/8.1/data/enterprise/current-config"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CatalogObject.Номенклатура">

    <IsFolder>false</IsFolder>

    <Ref>42a27940-c657-11d7-8e98-00c026abdd5e</Ref>

    <DeletionMark>false</DeletionMark>

    <Parent>42a2793c-c657-11d7-8e98-00c026abdd5e</Parent>

    <Code>4</Code>

    <Description>Kohinor тм</Description>

    <ОсновнаяЕдиницаИзмерения>12922f7f-3840-11db-8e4d-
00111a74afc1</ОсновнаяЕдиницаИзмерения>

    <ЕдиницаХранения>5a433250-366d-11db-8e47-
000b0d063032</ЕдиницаХранения>
```

```
    <Услуга>false</Услуга>

</CatalogObject.Номенклатура>
```

Загрузка данных из файла:

```
    НовыйСХDТО=Новый СериализаторXDTO(ФабрикаXDTO);



    Чтение=Новый ЧтениеXML;

    Чтение.ОткрытьФайл(СокрЛП(Константы.ПутьДоФайлов.Получ
ить())+"\obj.xml");



    Объект=НовыйСХDТО.ПрочитатьXML(Чтение);



    Объект.Записать();

    НоменклатураССылка=Объект.Ссылка;
```

6. Механизм Web сервисов

Механизм Web сервисов в системе "1С:Предприятие 8" является средством поддержки сервисно-ориентированной архитектуры. Такая архитектура представляет собой прикладную архитектуру, в которой все функции определены как независимые сервисы с вызываемыми интерфейсами. Обращение к таким сервисам в определенной последовательности позволяет реализовать тот или иной бизнес-процесс.

Конфигурация "1С:Предприятие 8" может экспортировать свою функциональность через Web-сервисы. Их определения задаются в дереве конфигурации, и становятся доступны произвольным информационным системам благодаря их публикации на Web-сервере.

В основе сервисной архитектуры находится менеджер сервисов. Менеджер сервисов выполняет следующие функции:

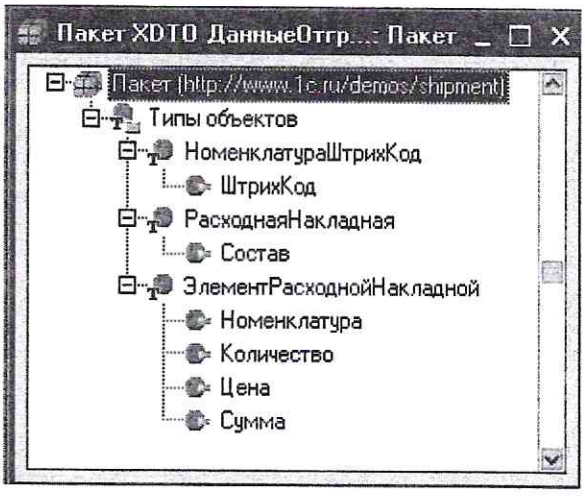
- Управление пулом соединений с информационными базами
- Поддержка WSDL описания сервиса
- Реализация протокола SOAP, сериализация сообщений, вызов соответствующего сервиса.

Менеджер сервисов выполняется в процессе сервисного хоста, который выполняет функцию приема/передачи сообщений из/в менеджер сервисов. В качестве сервисного хоста может выступать Web-сервер Microsoft IIS, Apache.

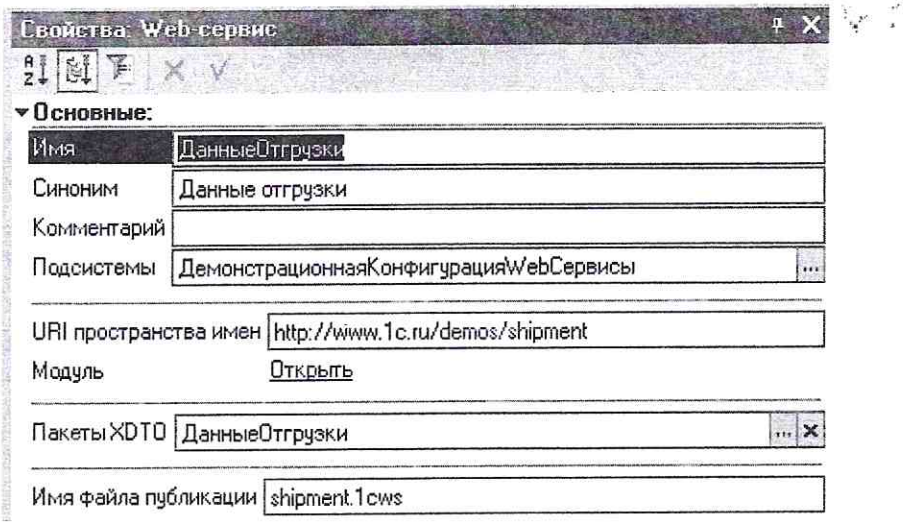
6.1. Создание WEB-сервисов "1С:Предприятие"

Рассмотрим пример организации web сервиса, использующего функциональность информационной базы "1С:Предприятие 8.1"

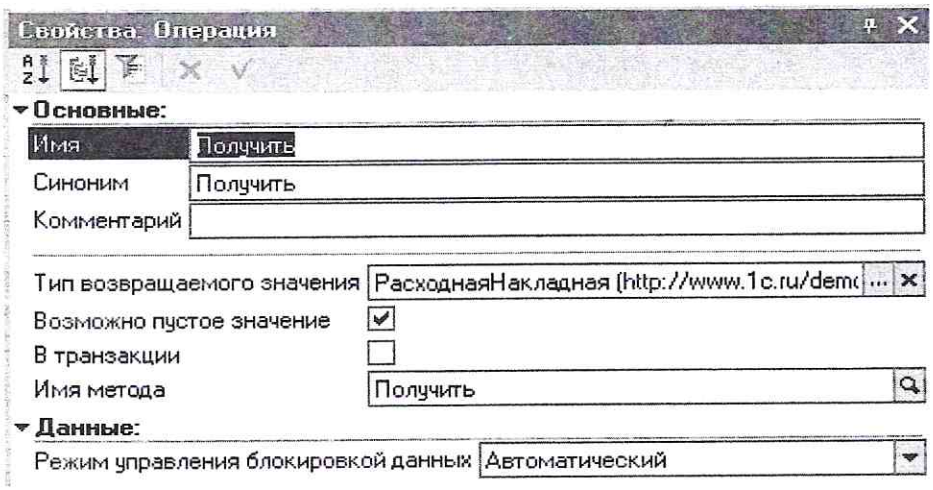
В конфигурации определен пакет XDTO:



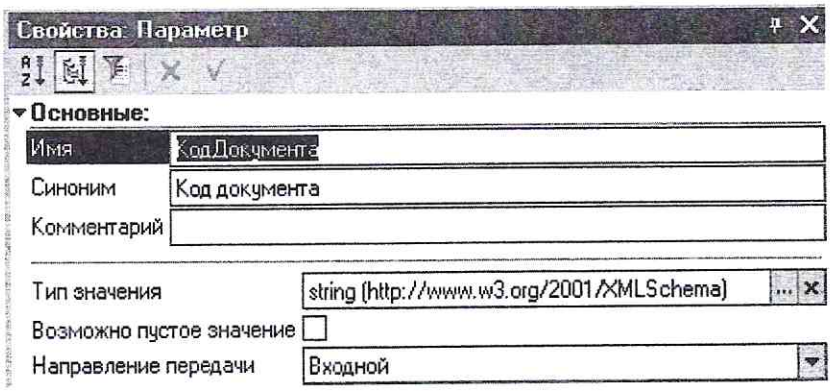
В ветви "Общие/Web-сервисы" необходимо добавить новый сервис "ДанныеОтгрузки", со следующими свойствами:



Сделав добавленный Web-сервис текущим, добавить новую операцию "Получить". Фактически этим действием определяем наличие одноименного метода у Web-сервиса. Следует отметить, что если Web-сервис создается для широкого использования, то стоит использовать имена на английском языке.



Следующим этапом определяется параметр "КодДокумента" метода "Получить".



В результате объект конфигурации выглядит следующим образом.

Остается в модуле Web-сервиса определить программную реализацию метода "Получить()":

```
Функция Получить(КодДокумента) Экспорт

    Возврат ПолучитьДанныеОтгрузки(КодДокумента);

КонецФункции
```

Реализация функции "ПоучитьДанныеОтгрузки()" находится в общем модуле.

```
Функция ПолучитьДанныеОтгрузки(КодДокумента) Экспорт

    ДокументСсылка =
Документы.РасходнаяНакладная.НайтиПоНомеру(КодДокумента,
ТекущаяДата());



    Если ДокументСсылка.Пустая() Тогда

        Возврат Неопределено;

    КонецЕсли;



    Документ = ДокументСсылка.ПолучитьОбъект();



    НоменклатураТип =
ФабрикаXDTO.Тип("http://www.1c.ru/demos/shipment",
"НоменклатураШтрихКод");

    РасходнаяНакладнаяТип =
ФабрикаXDTO.Тип("http://www.1c.ru/demos/shipment",
"РасходнаяНакладная");

    ЭлементРасходнойНакладнойТип =
ФабрикаXDTO.Тип("http://www.1c.ru/demos/shipment",
"ЭлементРасходнойНакладной");



    РасходнаяНакладная =
ФабрикаXDTO.Создать(РасходнаяНакладнаяТип);
```

```
    Для Каждого Элемент Из Документ.Состав Цикл

        ЭлементРасходнойНакладной =
ФабрикаXDTO.Создать(ЭлементРасходнойНакладнойТип);

        Номенклатура =
ФабрикаXDTO.Создать(НоменклатураТип);

        Номенклатура.ШтрихКод =
Элемент.Номенклатура.ШтрихКод;

        ЭлементРасходнойНакладной.Номенклатура =
Номенклатура;

        ЭлементРасходнойНакладной.Количество =
Элемент.Количество;

        ЭлементРасходнойНакладной.Цена = Элемент.Цена;

        ЭлементРасходнойНакладной.Сумма = Элемент.Сумма;

    РасходнаяНакладная.Состав.Добавить(ЭлементРасходнойНак
ладной);

    КонецЦикла;

    Возврат РасходнаяНакладная;

КонецФункции
```

6.2. Использование WEB-сервисов, опубликованных сторонними поставщиками

Система "1С:Предприятие 8.2" может использовать Web-сервисы предоставляемые другими поставщиками, двумя способами:

- С помощью статических ссылок, создаваемых в дереве объектов конфигурации
- С помощью динамических ссылок, создаваемыми средствами встроенного языка

Преимущество использования статических ссылок заключается в большей скорости работы, так как описание Web-сервиса поставщика получается один раз, при создании ссылки.

6.2.1. Использование динамических ссылок

В качестве примера рассмотрим использование Web-сервиса для получения данных по отгрузке товара (реализацию которого рассматривали в предыдущей главе).

Сама реализация обращения к Web-сервису выглядит следующим образом:

```
Функция ПолучитьДанныеОтгрузкиУдаленно(Знач
АдресВебСервиса, Знач КодДокумента) Экспорт

    Определения = Новый WSОпределения(АдресВебСервиса +
"?wsdl");

    Прокси = Новый WSПрокси(Определения,
"http://www.1c.ru/demos/shipment",

        "ДанныеОтгрузки", "ДанныеОтгрузкиSoap");



    Возврат Прокси.Получить(КодДокумента);

КонецФункции
```

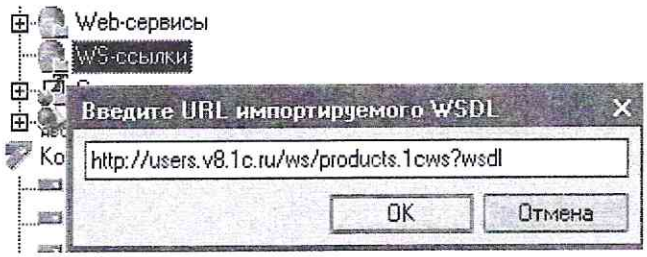
Данные используются для заполнения экземпляра документа:

```
    ДанныеОтгрузки =
ПолучитьДанныеОтгрузкиУдаленно(ДокументОбъект.Контрагент.Ад
ресВебСервиса,

        ДокументОбъект.НомерДокументаПоставщика);



    Если ДанныеОтгрузки = Неопределено Тогда

        Возврат;

    КонецЕсли;



    Для Каждого Элемент Из ДанныеОтгрузки.Состав Цикл

        Запись = ДокументОбъект.Состав.Добавить();

        Запись.Количество = Элемент.Количество;

        Запись.Цена = Элемент.Цена;

        Запись.Сумма = Элемент.Сумма;
```

```
        Запись.Номенклатура =
Справочники.Номенклатура.НайтиПоРеквизиту(
                "ШтрихКод", Элемент.Номенклатура.ШтрихКод);
    КонецЦикла;
```

6.2.2. Использование статических ссылок

Ссылка на Web-сервис задается в конфигураторе путем загрузки wsdl описания.



Функция, реализующая обращение к Web-сервису имеет следующий вид:

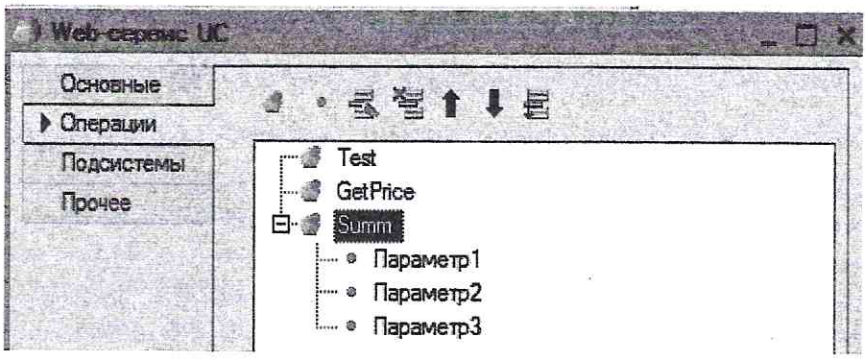
```
Функция ПолучитьДанныеНоменклатурыУдаленно() Экспорт

    Прокси =
WSCсылки.ДанныеНоменклатуры.СоздатьWSПрокси("http://www.1c.
ru/demos/products",
        "ДанныеНоменклатуры", "ДанныеНоменклатурыSoap");



    Возврат Прокси.Получить();

КонецФункции
```

Полученный объект XDTO может быть использован для получения данных таким же образом, как и в предыдущем примере

Практикум № 8

Обратитесь к web сервису (его свойства приведены ниже)

Функции сервиса:

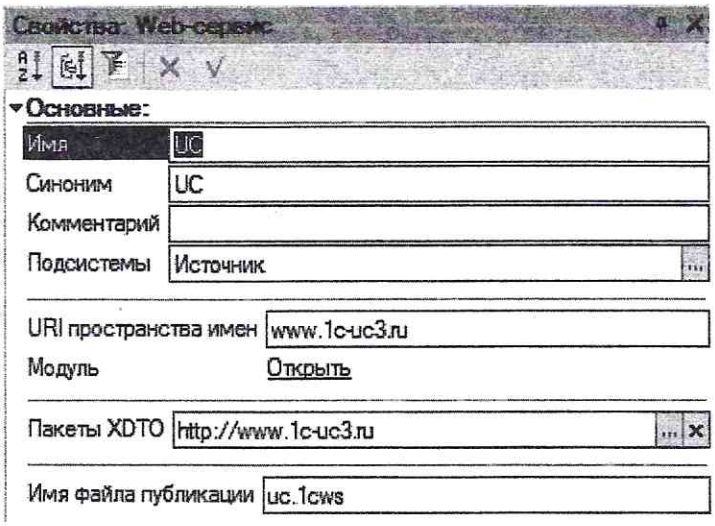


Модуль web сервиса:

```
Функция Test()

    Возврат("Ready");

КонецФункции


Функция GetPrice()

    ЦеныТип = ФабрикаXDTO.Тип("http://www.1c-uc3.ru",
"Цены");

    ТоварТип = ФабрикаXDTO.Тип("http://www.1c-uc3.ru",
"Номенклатура");

    СтрокаТип = ФабрикаXDTO.Тип("http://www.1c-uc3.ru",
"СтрокаТЧ");



    Цены = ФабрикаXDTO.Создать(ЦеныТип);

    Цены.НазваниеОрганизации="Доброе время суток!";


    Запрос=Новый Запрос;

    Запрос.Текст="ВЫБРАТЬ

                 | Цены.Номенклатура,

                 | Цены.Код,

                 | Цены.Цена,

                 | Цены.Услуга

                 |ИЗ

                 | РегистрСведений.Цены КАК Цены";
```

```
    Выборка=Запрос.Выполнить().Выбрать();
    Пока Выборка.Следующий() Цикл

        Товар=ФабрикаXDTO.Создать(ТоварТип);
        Товар.УникальныйКод=Выборка.Код;
        Товар.Наименование=Выборка.Номенклатура;
        Товар.Услуга=Выборка.Услуга;

        Строчка=ФабрикаXDTO.Создать(СтрокаТип);
        Строчка.Номенклатура=Товар;
        Строчка.Цена=Выборка.Цена;

        Цены.Строки.Добавить(Строчка);
    КонецЦикла;
    Цены.Дата=ТекущаяДата();
    Возврат(Цены);
КонецФункции

Функция Summ(Параметр1, Параметр2, Параметр3)
    Сумма=Строка(Параметр1+Параметр2);
   Параметр3=Параметр3+6;
    Возврат(Сумма);
КонецФункции
```

Свойства (как они определены в конфигураторе):



6.3. REST web сервисы

REST web-сервисы считаются более "легкими" по сравнению с SOAP web-сервисами. Их предпочтительнее использовать, когда не требуется реализовывать какую-либо сложную логику работы.

В "1С:Предприятие" возможность создания подобных сервисов появилась, начиная с версии технологической платформы 8.3.5.

Пример обращения к такому сервису приведен в демонстрационной базе "Клиент REST web-сервисов" (можно взять с диска ИТС). Приведем наиболее важные (с точки зрения идеи работы) фрагменты кода:

```
Процедура ОбновитьСписок()

    Строка = Адрес;

    Строка = СтрЗаменить(Строка, "http://", "");

    поз = Найти(Строка, "/");

    Сервер = Лев(Строка, поз);

    Ресурс = Прав(Строка, СтрДлина(Строка) - поз);


    Соединение = Новый HTTPСоединение(Сервер);

    ИмяФайла = ПолучитьИмяВременногоФайла(".rss");

    Соединение.Получить(Ресурс, ИмяФайла);


    ПараметрыЧтенияXML = Новый ПараметрыЧтенияXML(,,,,
true, true, true, true, true);
```

```
      ЧтениеXML = Новый ЧтениеXML;

      ЧтениеXML.ОткрытьФайл(ИмяФайла, ПараметрыЧтенияXML);


      ЧтениеXML.Прочитать();
      Если ЧтениеXML.Имя = "rss" Тогда
            Элементы.Очистить();
            РазборRSS(ЧтениеXML);
      Иначе
            Сообщить("Ошибка разбора RSS.");
      КонецЕсли;


      ЧтениеXML.Закрыть();
      УдалитьФайлы(ИмяФайла);


КонецПроцедуры
```

Начало разбора структурного фрагмента:

```
Процедура РазборRSS(ЧтениеXML)
      Пока ЧтениеXML.Прочитать() Цикл
            Если ЧтениеXML.Имя = "channel" Тогда
                  Если ЧтениеXML.Имя = "rss" и
ЧтениеXML.ТипУзла = ТипУзлаXML.КонецЭлемента Тогда
                        Прервать;
                  КонецЕсли;


                  РазборChannel(ЧтениеXML);
                  ЧитатьДоКонца(ЧтениеXML, "channel");
            КонецЕсли;
      КонецЦикла;
КонецПроцедуры
```

7. Планы обмена

Одной из важных задач, возникающих при организации обмена (особенно при организации постоянного обмена) является задача определения экземпляров выгружаемых объектов. Вряд ли пользователя устроит выгрузка "всех документов за сегодняшний день" и т.п. Необходимо выгружать именно измененные объекты. С этой задачей может успешно справиться такой объект конфигурации как "План обмена".

Каждый из планов обмена определяет набор данных, которыми предполагается обмениваться в рамках данного плана обмена. Вместе с набором данных могут определяться и специфические форматы представления этих данных. Предполагается, что форматы данных основаны на XML, но благодаря гибкости языка XML и наличию развитых средств работы с XML в "1С:Предприятие 8" остается достаточно большое пространство для творчества в области способов представления данных.

В планах обмена можно выделить две значимых составляющих: инфраструктура сообщений и служба регистрации изменений.

Элементами данных плана обмена являются узлы плана обмена, подобно тому, как элементами данных справочника являются элементы справочника. Каждый из узлов плана обмена обозначает участника обмена данными по данному плану обмена. Один из узлов соответствует данной информационной базе, а остальные — другим участникам, с которыми данная информационная база может обмениваться данными.

Данные переносятся между узлами с помощью сообщений. Средства работы с сообщениями образуют инфраструктуру сообщений. Каждое сообщение относится к определенному плану обмена, имеет определенный узел-отправитель и определенный узел-получатель. Сообщение не может быть отправлено неизвестному узлу и не может быть принято от неизвестного узла. Каждое сообщение имеет свой собственный целочисленный номер.

Суть регистрации изменений состоит в том, чтобы иметь перечень измененных элементов данных которые должны быть переданы в очередном сообщении тому или иному узлу, с которым производится обмен данными. При каждом изменении данных должно быть зарегистрировано, что имеются изменения, которые предстоит передать во все узлы, с которыми поддерживается обмен этими данными. При получении подтверждения приема сообщения, в котором были отправлены изменения, записи регистрации изменений должны быть удалены.

Регистрация изменений может выполняться для следующих элементов данных:

- КонстантаМенеджерЗначения.<Имя константы>
- Объекты базы данных
- СправочникОбъект.<Имя справочника>
- ДокументОбъект.<Имя документа>
- ПланСчетовОбъект.<Имя плана счетов>
- ПланВидовХарактеристикОбъект.<Имя плана видов характеристик>
- ПланВидовРасчетаОбъект.<Имя плана видов расчета>

- БизнесПроцессОбъект.<Имя бизнес-процесса>
- ЗадачаОбъект.<Имя задачи>

Наборы записей

- РегистрСведенийНаборЗаписей.<Имя регистра сведений>
- РегистрБухгалтерииНаборЗаписей.<Имя регистра бухгалтерии>
- РегистрНакопленияНаборЗаписей.<Имя регистра накопления>
- ПоследовательностьНаборЗаписей.<Имя последовательности>
- РегистрРасчетаНаборЗаписей.<Имя регистра расчета>
- ПерерасчетНаборЗаписей.<Имя перерасчета>

Для каждого из приведенных элементов данных ведется своя таблица регистрации изменений. Таблицы имеют разную структуру, в зависимости от того, для каких элементов данных регистрируются изменения, но все-таки структуры таблиц подобны. В структуре можно выделить три составляющих:

- Ключ элемента данных, для которого регистрируются изменения
- Ссылка на узел, в который изменение должно быть передано
- Номер сообщения, в котором изменение передано в первый раз

Структуры таблиц регистрации изменений для разных данных отличаются ключом, так как ключи у разных данных разные.

Для константы ключом является идентификатор константы.

Для объектов базы данных в качестве ключа используется ссылка на объект.

Для наборов записей, для которых определен регистратор, в качестве ключа используется ссылка на объект-регистратор.

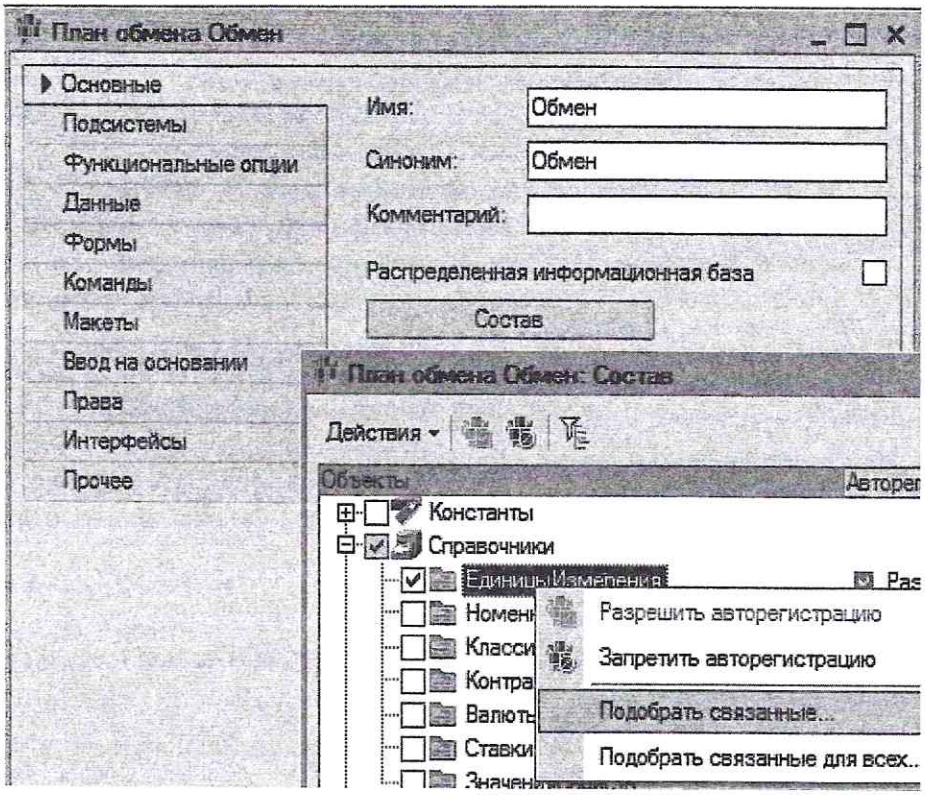
Для набора записей регистра сведений, в случае, если регистратор не определен, в качестве ключа используется совокупность измерений, входящих в основной отбор. А если регистр сведений является периодическим и включен основной отбор по периоду, то в ключ входит еще и период.

При изменении элемента данных его изменение должно быть зарегистрировано для всех узлов, в которые изменение должно быть передано. Таким образом, в результате изменения элемента данных в таблице регистрации изменений должно появиться N записей, где N — количество узлов, для которых регистрируются изменения. В каждой из этих записей указано одно и то же значение ключа элемента данных и различные значения ссылки на узел.

Непосредственно после выполнения регистрации изменения, номер сообщения имеет значение NULL. При первой отправке изменения в составе сообщения, в данное поле помещается номер сообщения, в котором изменение отправлено.

7.1. Первое знакомство

Для знакомства с планами обмена создадим соответствующий объект конфигурации. Имя создаваемого плана обмена: "Обмен". На закладке "Основные" необходимо определить состав объектов, для которых включается механизм регистрации изменений. Для этой цели необходимо нажать на кнопку "Состав", отметить справочник "Единицы измерения" и подобрать рекурсивно все связанные объекты (как показано на рисунке):



При создании нового плана обмена, в нем автоматически создается один узел — "этот узел" или узел плана обмена, соответствующий данной информационной базе. Остальные узлы, то есть узлы, с которыми данный узел может обмениваться данными, в рамках плана обмена автоматически не создаются.

Для каждого узла должен быть определен уникальный код, так как при обмене данными узел идентифицируется по коду.

Коды узлов задаются таким образом, чтобы обменивающиеся стороны "узнали" друг друга.

В пользовательском режиме для предопределенного узла обмена укажите код "Источник", создайте дополнительные узлы обмена с кодами "Приемник" и "Доп".

7.2. Универсальный обмен

Планы обмена могут участвовать в организации "двух видов обмена":

- Универсальный обмен
- Распределенная база данных

Универсальный обмен предполагает возможность организации обмена данными между базой "1С:Предприятие 8" (в которой эти механизмы создаются) и любым другим программным комплексом (не обязательно это должна быть информационная база "1С:Предприятие").

Рассмотрим простейший пример реализации процедур обмена.

Текст процедуры, осуществляющей выгрузку данных, следующий:

```
Процедура КнопкаВыполнитьНажатие(Элемент)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    ЗаписьXML=Новый ЗаписьXML();

    ЗаписьXML.ОткрытьФайл(Путь+"выгрузка.xml");

    Узел=ПланыОбмена.ПоНоменклатуре.НайтиПоКоду("Источник"
);

    ЗапСообщения=ПланыОбмена.СоздатьЗаписьСообщения();

    ЗапСообщения.НачатьЗапись(ЗаписьXML,Узел);

    Выборка=ПланыОбмена.ВыбратьИзменения(Узел,ЗапСообщения
.НомерСообщения);

    Пока Выборка.Следующий() Цикл

        Данные=Выборка.Получить();

        ЗаписатьXML(ЗаписьXML,Данные);

    КонецЦикла;

    ЗапСообщения.ЗакончитьЗапись();

    ЗаписьXML.Закрыть();

КонецПроцедуры
```

Текст процедуры, осуществляющей загрузку данных:

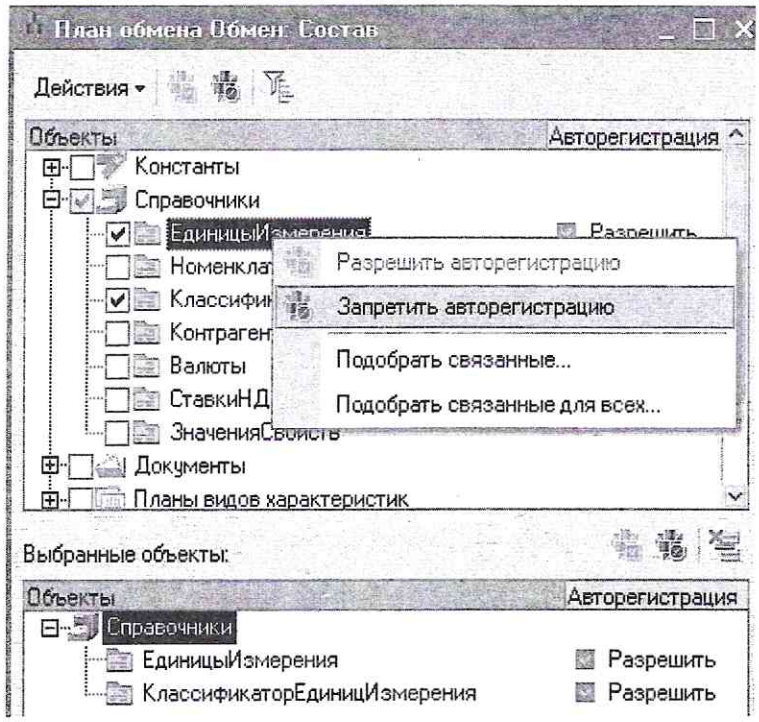
```
Процедура КнопкаВыполнитьНажатие(Элемент)

    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    ЧтениеXML=Новый ЧтениеXML();

    ЧтениеXML.ОткрытьФайл(Путь+"выгрузка.xml");

    ЧтСообщения=ПланыОбмена.СоздатьЧтениеСообщения();

    ЧтСообщения.НачатьЧтение(ЧтениеXML);

    ПланыОбмена.УдалитьРегистрациюИзменений(ЧтСообщения.От
правитель,ЧтСообщения.НомерСообщения);

    Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл

        Данные=ПрочитатьXML(ЧтениеXML);


    Данные.ОбменДанными.Отправитель=ЧтСообщения.Отправител

        Данные.ОбменДанными.Загрузка=Истина;

        Данные.Записать();

    КонецЦикла;

    ЧтСообщения.ЗакончитьЧтение();

    ЧтениеXML.Закрыть();

КонецПроцедуры
```

Можно организовывать любые схемы обмена, ограничений практически нет. Но всю "широту" спектров возможных решений разработчику необходимо полностью реализовывать самому, а именно необходимо решить следующие задачи:

- Управление регистрацией изменений
- Очистка таблиц регистрации изменений
- Определение стратегии распространения данных
- Процесс разрешения коллизий
- Создание "начального образа"
- Рассмотрим эти задачи более подробно.

7.2.1. Управление регистрацией изменений

У каждого из объектов, перечисленных в параграфе "Служба регистрации изменений, имеется свойство "ОбменДанными", имеющее тип "ПараметрыОбменаДанными". У объекта "ПараметрыОбменаДанными" есть свойство "Получатели", имеющее тип "НаборУзлов". В данном свойстве хранится перечень узлов, для которых будет выполняться регистрация изменений при записи или удалении данных. Список получателей заполнятся автоматически перед тем, как будет вызван обработчик "Перед записью()" при выполнении записи данных или "Перед удалением()" при выполнении удаления. Однако, автоматическое заполнение будет выполнено только в том случае, если свойство "Автозаполнение" объекта "НаборУзлов" имеет значение "Истина.



При автоматическом заполнении в список получателей попадают ссылки на все узлы всех планов обмена, в состав которых входит соответствующий объект метаданных, при условии, что значением свойства "Авторегистрация" является "Разрешить". При выполнении автоматического заполнения список получателей предварительно очищается.

В обработчике ПередЗаписью() (и/или ПередУдалением()) в список получателей можно внести изменения: добавить или удалить ссылки на узлы. Однако, следует помнить, что список получателей может содержать только ссылки на узлы, относящиеся к планам обмена, в состав которых входит соответствующий объект метаданных.

В приведенном ниже примере обработчик ПередЗаписью() исключает из списка получателей узел с кодом "Доп".

```
Процедура ПередЗаписью()

Узел = ПланыОбмена.Обмен.НайтиПоКоду("Доп");

ОбменДанными.Получатели.Удалить(Узел);

КонецПроцедуры
```

Присвоив свойству "Автозаполнение" значение "Ложь", можно добиться того, что автоматическое заполнение списка получателей выполняться не будет. В этом случае действия со списком получателей можно производить не только в обработчике "Перед записью()", но и в любом фрагменте кода, как показано в примере:

```
Объект = Ссылка.ПолучитьОбъект();

Узел = ПланыОбмена.Обмен.НайтиПоКоду("Доп");

Объект.ОбменДанными.Получатели.Автозаполнение = Ложь;

Объект.ОбменДанными.Получатели.Добавить(Узел);

Объект.Записать();
```

7.2.2. Очистка таблиц регистрации изменений

Существует два основных способа очистки таблиц регистрации изменений (эти способы можно комбинировать между собой).

Один из способов основан на том, что в формируемом сообщении содержится номер последнего сообщения принятого отправителем от получателя

```
ЧтениеXML = Новый ЧтениеXML();

ЧтениеXML.ОткрытьФайл(ИмяФайлаСообщения);

ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отп
равитель, ЧтениеСообщения.НомерСообщения);

Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл

      Данные = ПрочитатьXML(ЧтениеXML);

      Данные.ОбменДанными.Отправитель =
ЧтениеСообщения.Отправитель;

      Данные.ОбменДанными.Загрузка = Истина;

      Данные.Записать();

КонецЦикла;

ЧтениеСообщения.ЗакончитьЧтение();
```

В случае гарантированной доставки:

```
ЗаписьXML = Новый ЗаписьXML()

ЗаписьXML.ОткрытьФайл(ИмяФайлаСообщения);

Узел = ПланыОбмена.Обмен.НайтиПоКоду(КодУзла);

ЗаписьСообщения =
ПланыОбмена.СоздатьЗаписьСообщения();

ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Узел);

Выборка = ПланыОбмена.ВыбратьИзменения(Узел,
ЗаписьСообщения.НомерСообщения);

Пока Выборка.Следующий() Цикл

      Данные = Выборка.Получить();

      ЗаписатьXML(ЗаписьXML, Данные);

КонецЦикла;

НомерСообщения = ЗаписьСообщения.НомерСообщения;

ЗаписьСообщения.ЗакончитьЗапись();

ПланыОбмена.УдалитьРегистрациюИзменений(Узел,
НомерСообщения);
```

7.2.3. Определение стратегии распространения данных

При организации выгрузки данных на основании данных, хранящихся либо в узле плана обмена, либо в регистре сведений можно принимать решение о том, что объект не выгружается. Другим вариантом является исключение нужного узла из получателей при регистрации изменений (как было рассмотрено выше).

7.2.4. Разрешение коллизий

В приведенных выше примерах чтения и записи сообщений не учитывалось, что при обмене данными может случиться так, что один и тот же элемент данных будет изменен одновременно в двух обменивающихся данными узлах. В этом случае непонятно, какое из изменений должно быть в конечном счете принято. Такая ситуация называется коллизией.

Одним из способов разрешения коллизий может быть определение, какой из узлов является главным, а какой — подчиненным. При этом должно быть принято изменение, сделанное в главном узле, а изменение, сделанное в подчиненном узле, должно быть отвергнуто.

Для реализации этого, при приеме сообщения перед записью данных необходимо установить, зарегистрировано ли изменение этих данных, и, в зависимости от роли узла в данной паре получатель-отправитель, принять решение: записывать или не записывать данные.

Ниже приведен пример реализации стратегии "главный —подчиненный" при чтении сообщения. Предполагается, что для хранения роли узла в плане обмена был определен реквизит Главный, имеющий тип Булево.

```
Процедура КнопкаВыполнитьНажатие(Кнопка)

    ЧтениеXML = Новый ЧтениеXML()

    ЧтениеXML.ОткрытьФайл(ИмяФайлаСообщения);

    ЧтениеСообщения =
ПланыОбмена.СоздатьЧтениеСообщения();

    ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

    ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщени
я.Отправитель, ЧтениеСообщения.НомерСообщения);

    Отправитель = ЧтениеСообщения.Отправитель;

    Главный = Отправитель.Главный;

    Пока ВозможностьЧтенияXML(Чт) Цикл

        Данные = ПрочитатьXML(Чт);

        Если Главный Или

Не ПланыОбмена.ИзменениеЗарегистрировано(Отправитель,
Данные) Тогда

Данные.ОбменДанными.Отправитель =
ЧтениеСообщения.Отправитель;

            Данные.ОбменДанными.Загрузка = Истина;

            Данные.Записать();

        КонецЕсли;

    КонецЦикла;

    ЧтениеСообщения.ЗакончитьЧтение();

КонецПроцедуры
```

В реальной жизни могут использоваться более сложные критерии определения "Главный-Подчиненный".

7.2.5. Создание "начального образа"

Для создания начального образа (принудительной регистрации) может использоваться метод "ЗарегистрироватьИзменения()". Данный метод позволяет выполнять регистрацию изменений одиночных элементов данных или целых групп для одного или нескольких узлов. Первый параметр данного метода — ссылка на узел плана обмена или массив ссылок на узлы, для которых

выполняется регистрация изменений. Если первый параметр представляет собой одиночную ссылку на узел, то второй параметр может быть опущен. При этом выполняется регистрация изменений всех элементов данных, которые на данный момент присутствуют в базе данных и изменения которых могут быть зарегистрированы для данного узла.

```
Узел = ПланыОбмена.Обмен.НайтиПоКоду("Доп");

ПланыОбмена.ЗарегистрироватьИзменения(Узел);
```

Если же первый параметр представляет собой массив ссылок на узлы, то второй параметр обязательно должен быть указан. Впрочем, второй параметр может присутствовать и в том случае, если первый параметр это одиночная ссылка на узел. В зависимости от способа задания второго параметра можно зарегистрировать изменения одного элемента данных или же всех данных, относящихся к одному объекту метаданных.

```
Узлы = Новый Массив(2);

Узлы[0] = ПланыОбмена.Обмен.НайтиПоКоду("Доп");

Узлы[1] =
ПланыОбмена.УдаленныеСклады.НайтиПоКоду("Приемник");

Данные = Справочники.ЕдиницыИзмерения.НайтиПоКоду("2");

ПланыОбмена.ЗарегистрироватьИзменения(Узлы, Данные);
```

Для регистрации изменений всех данных, относящихся к объекту метаданных, в качестве второго параметра должен быть указан соответствующий объект метаданных.

```
Узлы = Новый Массив(2);

Узлы[0] = ПланыОбмена.УдаленныеСклады.НайтиПоКоду("Доп");

Узлы[1] =
ПланыОбмена.УдаленныеСклады.НайтиПоКоду("Приемник");

ПланыОбмена.ЗарегистрироватьИзменения(Узлы,
Метаданные.Справочники.Номенклатура);
```

7.3. Распределенные базы данных

Распределенная информационная база – это совокупность информационных баз 1С:Предприятия (узлов распределенной информационной базы), в которых поддерживается синхронизация конфигурации и данных. Распределенная информационная база имеет иерархическую структуру. У каждого узла распределенной информационной базы может быть один главный и произвольное число подчиненных узлов. "Самый главный узел" или узел, у которого нет главного узла, называется корневым узлом распределенной информационной базы. Каждый из узлов может обмениваться данными только со своими "соседями", то есть со своими главным и подчиненными узлами.

Изменения конфигурации допускаются только в корневом узле распределенной информационной базы с последующим ее распространением по иерархии от корневого узла к его подчиненным и т. д. Таким образом, механизм управления распределенными информационными базами обеспечивает наличие во всех узлах распределенной информационной базы одной и той же конфигурации.

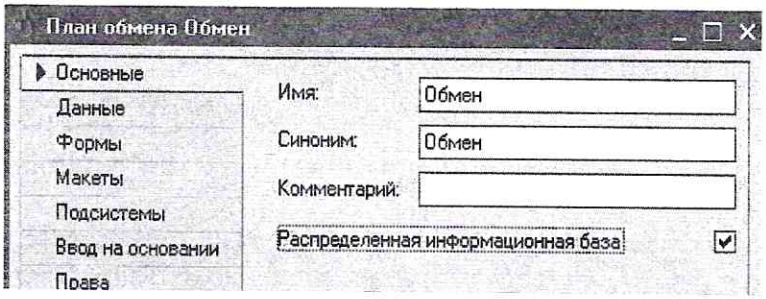
Изменение данных допускается в любом узле распределенной информационной базы. Синхронизация данных достигается путем распространения изменений данных, произведенных в одном узле, во все структуры распределенной информационной базы.

Если в рамках всей распределенной информационной базы поддерживается полная идентичность конфигурации, то полная идентичность данных не обязательна.

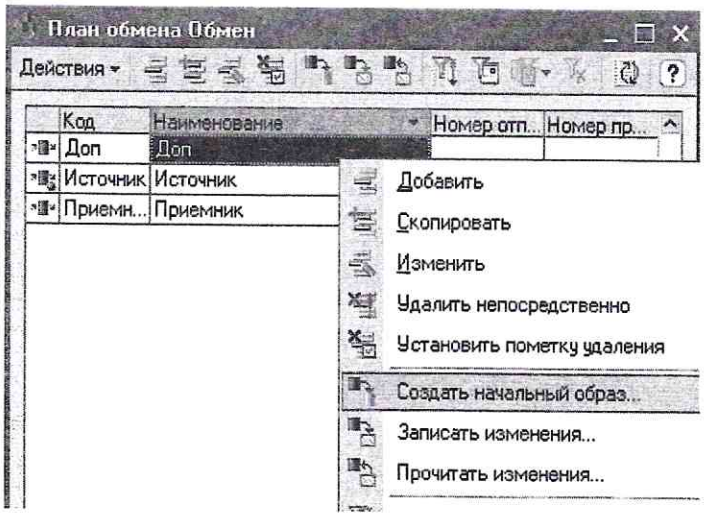
Центральное место в организации распределенных баз данных занимает объект "ПланОбмена" (с установленным флагом "распределенная база данных").

7.3.1. Создание распределенной базы

Для создания распределенной базы данных необходимо отметить у плана обмена свойство "Распределенная база данных"



После этого остается в пользовательском режиме определить состав участников обмена (мы воспользуемся уже заведенными) и создать начальный образ для периферийной базы (в общем случае их может быть любое количество).



После создания в периферийной базе уже определен главный и текущий узлы, при желании можно указать любое количество подчиненных, создать для них образы и т.п.

В интерактивном режиме обмен может производиться с помощью команд "Записать изменения", "Прочитать изменения".

Следует отметить тот факт, что возможно программное создание образа (метод "СоздатьНачальныйОбраз()"), запись и чтение изменений.

7.3.2. Порядок распространения данных

После выполнения действий, указанных в предыдущем разделе для распределенной базы данных начинает действовать порядок распространения "по умолчанию":

- Каждое изменение элемента данных, произведенное в любом из узлов распределенной информационной базы стремится распространиться по всем узлам.
- Разрешение коллизий производится на основании отношения узлов "главный – подчиненный".

При необходимости изменения данного порядка можно использовать обработчики событий объекта "План обмена".

Пример реализации "неполной" выгрузки данных:

```
Процедура ПриОтправкеДанныхПодчиненному(ЭлементДанных,
ОтправкаЭлемента)
    ТипДанных = ТипЗнч(ЭлементДанных);

    Если ТипДанных =
Тип("ДокументОбъект.РасходнаяНакладная") Тогда

        Если ЭлементДанных.Склад <> Склад Тогда

            ОтправкаЭлемента =
ОтправкаЭлементаДанных.Удалить;

        КонецЕсли;
    КонецЕсли;
КонецПроцедуры

Процедура
ПриПолученииДанныхОтПодчиненного(ЭлементДанных,ПолучениеЭле
мента, ОтправкаНазад)

    ТипДанных = ТипЗнч(ЭлементДанных);

    Если ТипДанных =
Тип("ДокументОбъект.РасходнаяНакладная") Тогда

        Если ЭлементДанных.Склад <> Склад Тогда

            ОтправкаНазад = Истина;

        КонецЕсли;
    КонецЕсли;
КонецПроцедуры
```

7.3.3. Разрешение коллизий

Для реализации "нестандартного" механизма разрешения коллизий можно использовать следующие обработчики событий (в данном случае подчиненный узел имеет более высокий приоритет по сравнению с главной базой)

```
Процедура ПриПолученииДанныхОтПодчиненного(ЭлементДанных,
ПолучениеЭлемента, ОтправкаНазад)

    ТипДанных = ТипЗнч(ЭлементДанных);

    Если ТипДанных =
Тип("ДокументОбъект.РасходнаяНакладная") Тогда

        ПолучениеЭлемента =
ПолучениеЭлементаДанных.Принять;

    КонецЕсли;
КонецПроцедуры

Процедура ПриПолученииДанныхОтГлавного(ЭлементДанных,
    ПолучениеЭлемента, ОтправкаНазад)

    ТипДанных = ТипЗнч(ЭлементДанных);

    Если ТипДанных =
Тип("ДокументОбъект.РасходнаяНакладная") Тогда

    Если ПланыОбмена.ИзменениеЗарегистрировано(Ссылка,
ЭлементДанных) Тогда

            ПолучениеЭлемента =
ПолучениеЭлементаДанных.Игнорировать;

        КонецЕсли
    КонецЕсли;
КонецПроцедуры
```

7.3.4. Работа из встроенного языка

Для записи тела сообщения обмена данными распределенной информационной базы менеджер планов обмена содержит метод "ЗаписатьИзменения()". В качестве первого параметра данному методу передается объект типа "ЗаписьСообщенияОбмена", через который осуществляется запись сообщения. У данного объекта уже должен быть вызван метод "НачатьЗапись()", но еще не должен быть вызван метод "ЗакончитьЗапись()".

Необязательный второй параметр указывает максимальное число элементов данных, которые будут помещаться в сообщение в рамках одной транзакции. Если в качестве значения параметра указано 0, то все формирование сообщения будет выполнено в одной транзакции. В таком режиме обеспечивается наилучшая согласованность данных помещаемых в сообщение, но возможны конфликты с транзакциями, выполняемыми другими пользователями. При меньшем числе элементов, обрабатываемых в одной транзакции вероятность конфликтов транзакций меньше, но больше вероятность помещения в сообщение

несогласованных данных. Поэтому, рекомендуется без острой необходимости не использовать для второго параметра значения, отличные от значения по умолчанию.

Пример обработчика:

```
    Путь= Константы.ПутьДоФайлов.Получить()+"\";

    ЗаписьXML = Новый ЗаписьXML();

    ЗаписьXML.ОткрытьФайл(Путь+"урбд.xml");

    Узел = ПланыОбмена.Обмен.НайтиПоКоду("Приемник");

    ЗаписьСообщения =
ПланыОбмена.СоздатьЗаписьСообщения();

    ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Узел);

    ПланыОбмена.ЗаписатьИзменения(ЗаписьСообщения);

    ЗаписьСообщения.ЗакончитьЗапись();
```

Для чтения тела сообщения обмена данными распределенной информационной базы менеджер планов обмена содержит метод "ПрочитатьИзменения()".

В качестве первого параметра методу передается объект типа "ЧтениеСообщенияОбмена", через который осуществляется чтение сообщения в целом. У этого объекта уже должен быть вызван метод "НачатьЧтение()", но еще не вызван метод "ЗакончитьЧтение()". Настоятельно рекомендуется, чтобы при обращении к методу "НачатьЧтение()" объекта "ЧтениеСообщенияОбмена" значение второго параметра не указывалось или же указывалось значение по умолчанию – "ДопустимыйНомерСообщения.Больший". Это связано с тем, что вся логика обмена данными в распределенной информационной базе построена с учетом того, что отдельные сообщения обмена данными могут быть утеряны, но при этом не допускается повторный прием одного и того же сообщения.

В качестве значения необязательного второго параметра может быть указано максимальное число элементов данных считываемых из сообщения и помещаемых в базу данных в рамках одной транзакции. Значение параметра по умолчанию – 0, означает, что все чтение сообщения будет выполняться в одной транзакции. Если все чтение сообщения выполняется в одной транзакции, то в случае возникновения ошибок не может оказаться так, что часть элементов данных была считана из сообщения и помещена в базу данных, а часть – нет. Но при таком режиме может случиться, что число изменений базы данных, которое надо выполнить в рамках одной транзакции окажется слишком большим. Кроме того, повышается вероятность конфликтов между транзакцией, в которой происходит чтение сообщения и транзакциями, выполняемыми другими пользователями. Для того, чтобы избежать неприятностей такого рода введена возможность ограничения числа элементов данных, обрабатываемых в одной транзакции. Но, если нет острой нужды, рекомендуется использовать режим по умолчанию, то есть производить считывание всех элементов данных в одной транзакции.

Последовательность действий, выполняемая методом ПрочитатьИзменения() выглядит примерно следующим образом:

- Считывается и проверяется "подпись".плана обмена, чтобы с максимальной достоверностью убедиться, что сообщение прибыло от ожидаемого плана обмена ожидаемой конфигурации.
- Из сообщения считываются изменения конфигурации. При считывании проводится проверка цифровых подписей конфигурации, чтобы исключить возможность того, что в узле-отправителе и текущем узлах распределенной информационной базы находятся несовместимые конфигурации. Для каждого из считанных измененных объектов конфигурации проверяется: является ли этот объект конфигурации фактически измененным. Напомним, что измененные объекты конфигурации могут содержаться только в сообщениях, передаваемых от главного узла подчиненному.
- Удаляются записи регистрации изменений объектов конфигурации и элементов данных, отправленных в сообщениях, для которых получено подтверждение приема. Напомним, что максимальный из номеров принятых узлом-отправителем сообщений содержится в заголовке сообщения и доступен через свойство НомерПринятого объекта ЧтениеСообщенияОбмена.
- Если в результате проверки цифровых подписей конфигурации удалось установить, что конфигурация в текущем узле распределенной информационной базы отличается от конфигурации узла-отправителя, то возможны два варианта дальнейших действий. Если текущий узел является главным по отношению к узлу-отправителю, то дальнейший прием данного сообщения невозможен ни при каких условиях. Если же текущий узел является подчиненным по отношению к узлу-отправителю, то перед продолжением приема сообщения необходимо обновить конфигурацию базы данных, приведя ее в соответствие с конфигурацией узла-отправителя. При обновлении конфигурации базы данных в нее будут перенесены ранее принятые и сохраненные измененные объекты конфигурации. Далее в обоих случаях вызывается исключение с соответствующим текстом сообщения об ошибке. И, если в первом случае сообщение не может быть прочитано, то во втором случае после обновления конфигурации базы данных, которое может быть выполнено в режиме Конфигуратор, то же самое сообщение может быть успешно прочитано и принято.
- Если производится прием сообщения обмена данными, отправленного из главного узла распределенной информационной базы, то данные текущего узла и узла-отправителя приводятся в соответствие с данными об этих узлах, содержащимися в сообщении обмена данными. Если сообщение обмена данными получено от подчиненного узла, то там таких данных быть не должно (подробнее см. "Сообщение обмена данными в распределенной информационной базе").
- Производится считывание из сообщения и запись в базу данных элементов данных. При этом в сообщении могут содержаться только элементы данных, соответствующие объектам метаданных, входящих в состав плана обмена, к которому относится данное сообщение. Для каждого прочитанного из сообщения элемента данных, у объекта типа

"ПланыОбменаОбъект.<Имя плана обмена>" вызывается обработчик события "ПриПолученииДанныхОтГлавного" ("ПриПолученииДанныхОтПодчиненного"). Дальнейшие действие по каждому из элементов данных определяются результатами действий обработчика (подробнее см. выше "Обработчики событий плана обмена").

Пример обработчика:

```
Путь= Константы.ПутьДоФайлов.Получить()+"\";

ЧтениеXML=Новый ЧтениеXML();

ЧтениеXML.ОткрытьФайл(Путь+"урбд.xml");

ЧтСообщения=ПланыОбмена.СоздатьЧтениеСообщения();

ЧтСообщения.НачатьЧтение(ЧтениеXML);

ПланыОбмена.ПрочитатьИзменения(ЧтСообщения,0);

ЧтСообщения.ЗакончитьЧтение();

ЧтениеXML.Закрыть();
```

Практикум № 9

Распределите базу "Источник". Попробуйте обменяться данными между ними. Обменяйтесь изменениями в конфигурации.

8. Конфигурация "Конвертация данных"

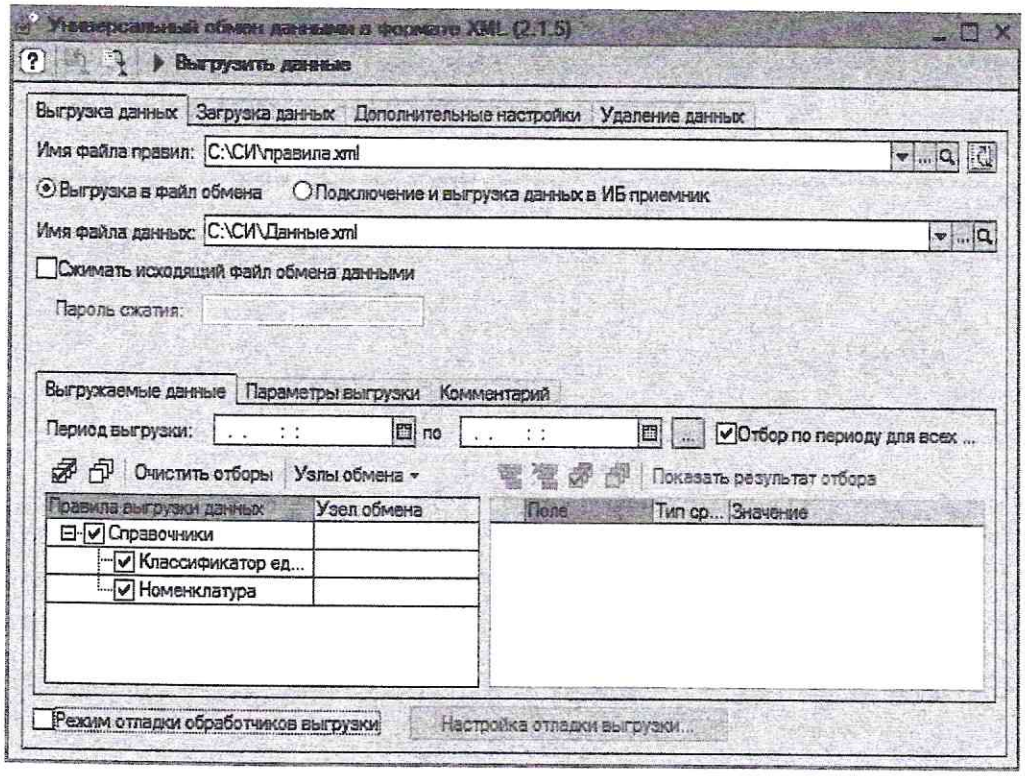
В ряде случаев может возникать задача организации переноса данных между конфигурациями "1С:Предприятие" (для разового или постоянного использования). Причем решение задачи должно быть произведено в кратчайшие сроки ("вчера") и к скорости самого обмена жестких требований не предъявляется. В таком случае можно воспользоваться предлагаемым разработчиками фирмы 1С механизмом "Конвертация данных".

Данный механизм позволяет очень просто настроить и произвести обмен между конфигурациями "1С:Предприятие" (с разной структурой конфигураций) версий 7.7, 8.0, 8.1, 8.2.

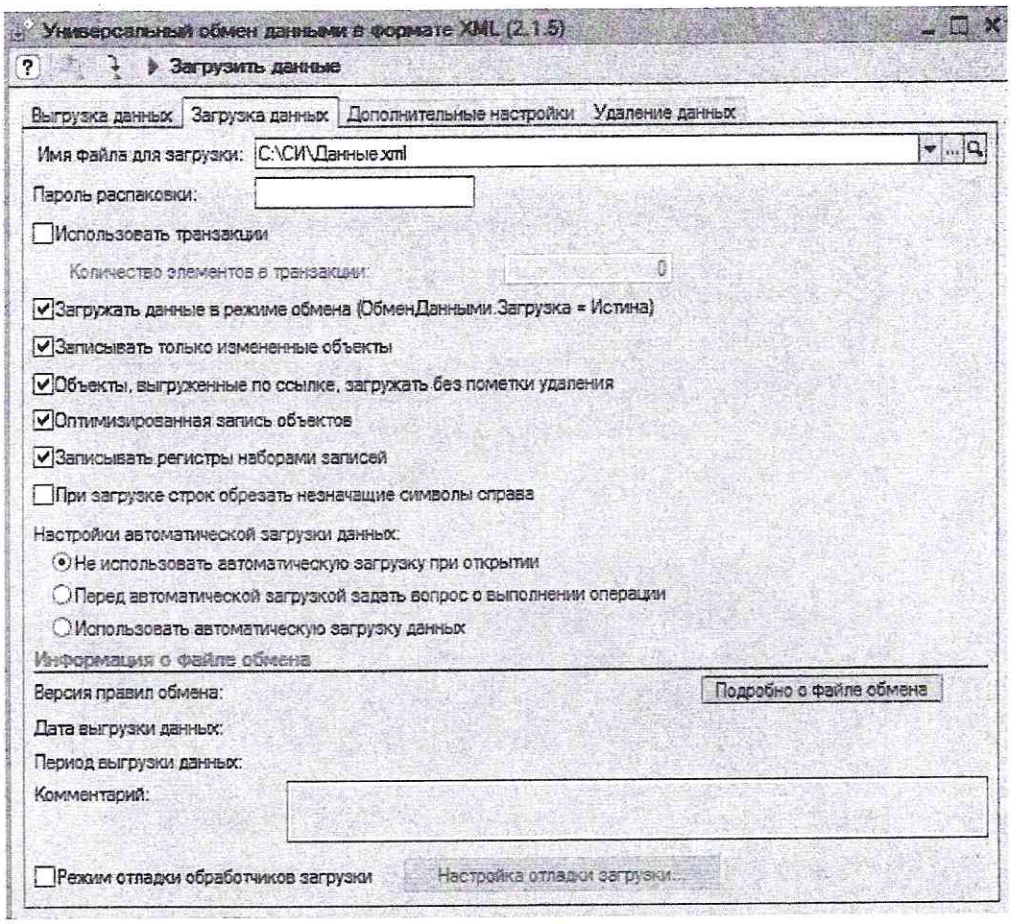
8.1. Общие принципы

Общие принципы работы механизма следующие:

- В конфигурации – источнике данных запускается специализированная обработка (она различается для разных версий "1С:Предприятие"). В обработке указываются пути до файла правил, файла с данными. После этого производится сама выгрузка.



- В конфигурации – приемнике данных также открывается специализированная обработка (другая закладка обработки), указывается путь до файла данных и производится сама загрузка.

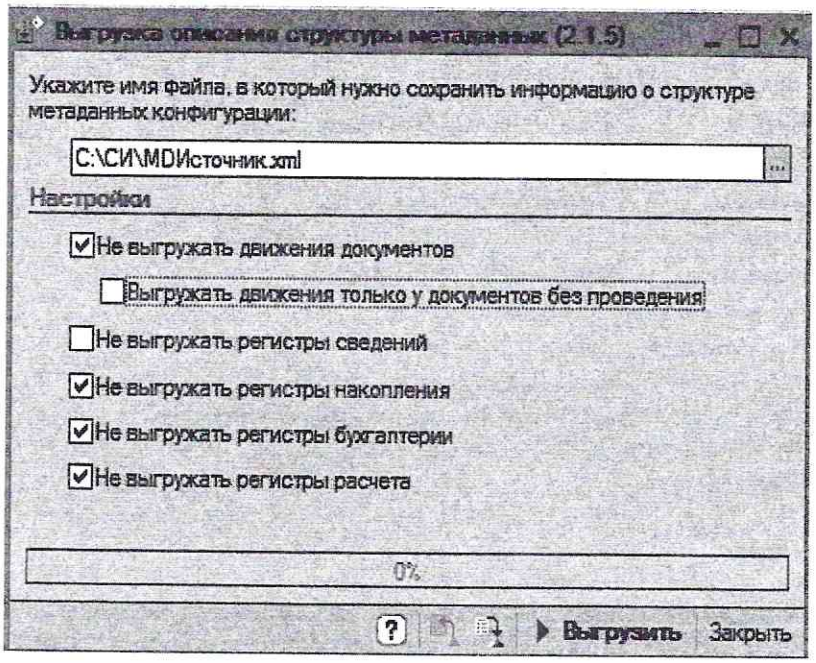


- Если файла правил нет, для его получения используется специализированная конфигурация "Конвертация данных" (и ряд вспомогательных внешних обработок, поставляемых вместе с конфигурацией).

8.2. Настройка правил обмена

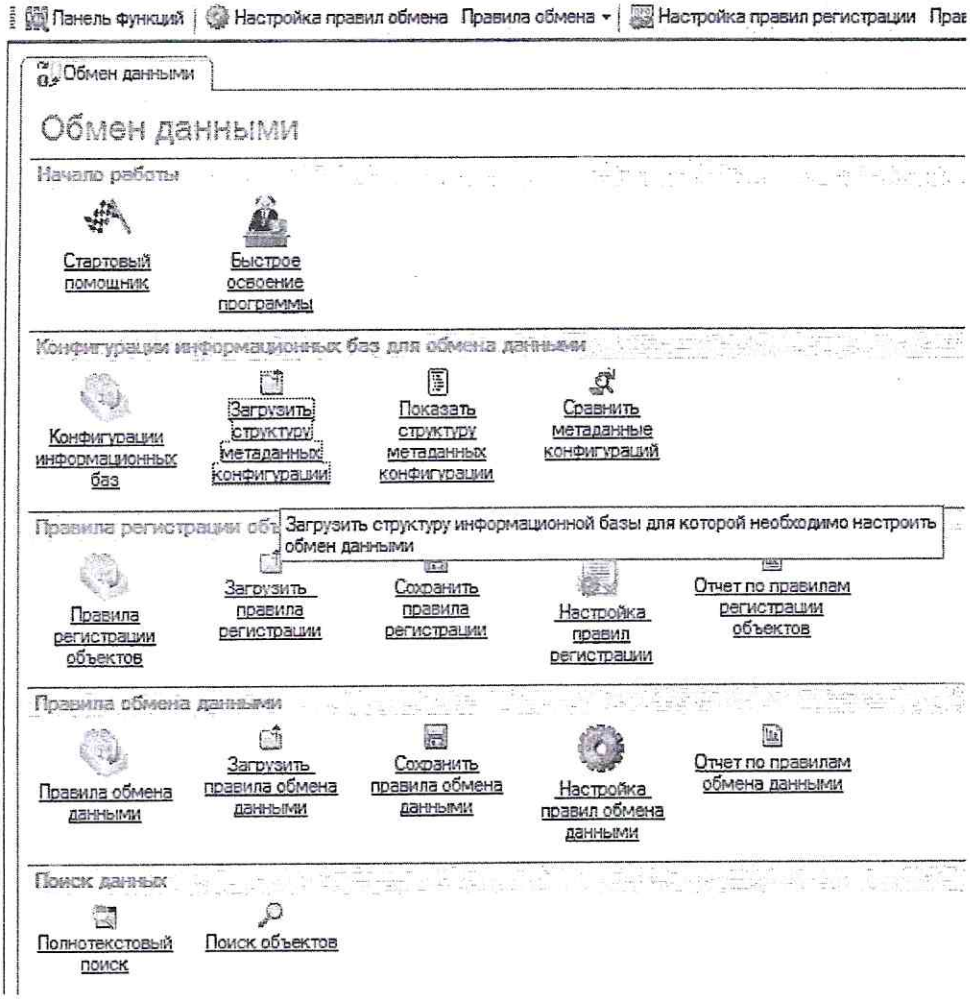
Настроим обмен между учебными конфигурациями "Источник" и "Приемник".

Перед тем как приступить к созданию правил обмена необходимо сформировать описание структур конфигураций источника и приемника. Для этой цели можно воспользоваться обработкой "MD82Exp.epf".

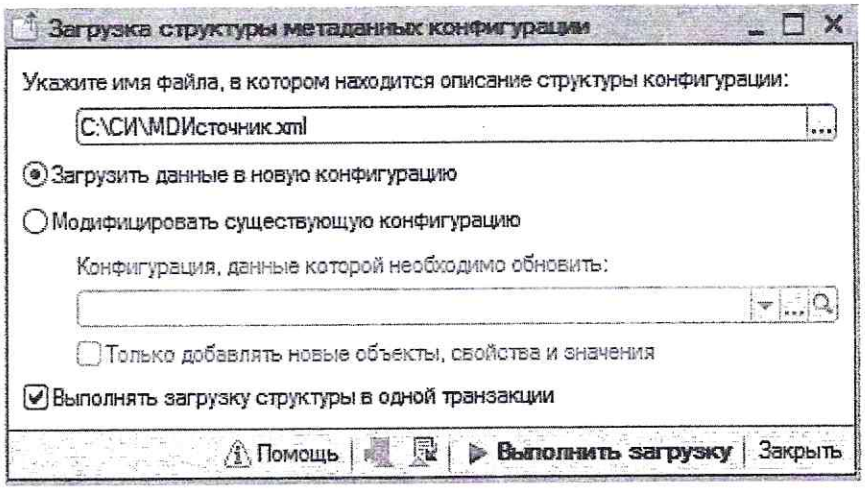


Получите метаданные конфигураций источника и приемника.

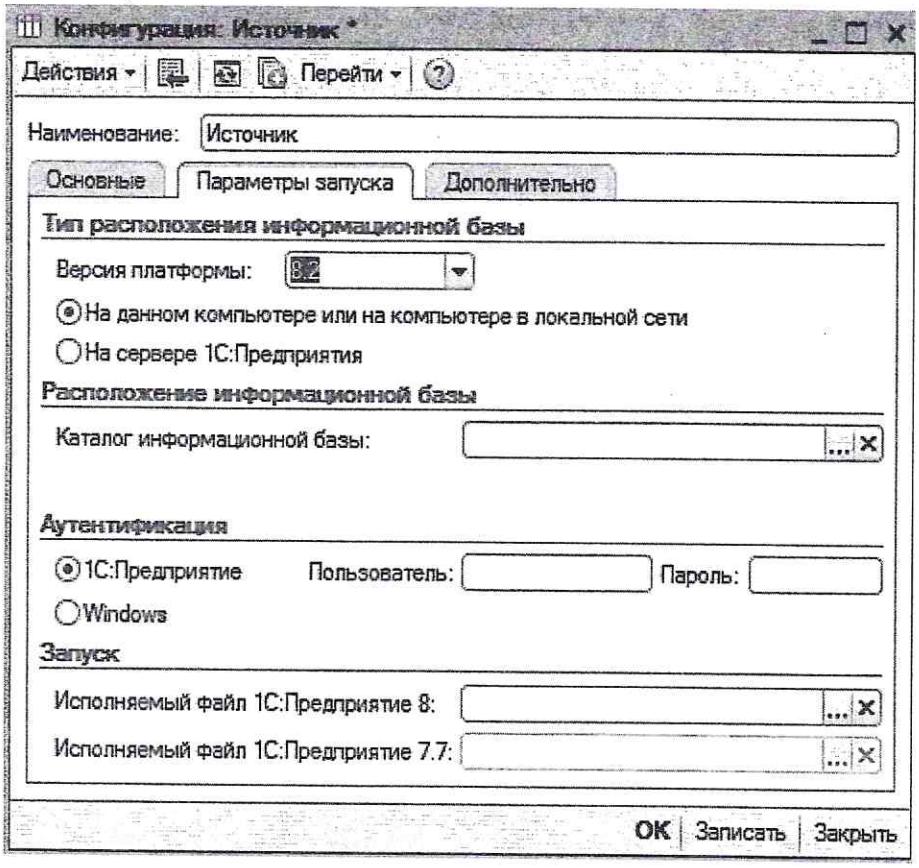
После этого откройте конфигурацию "Конвертация данных, редакция 2" (далее КД). В панели функций выберите "Загрузить структуру метаданных конфигурации"



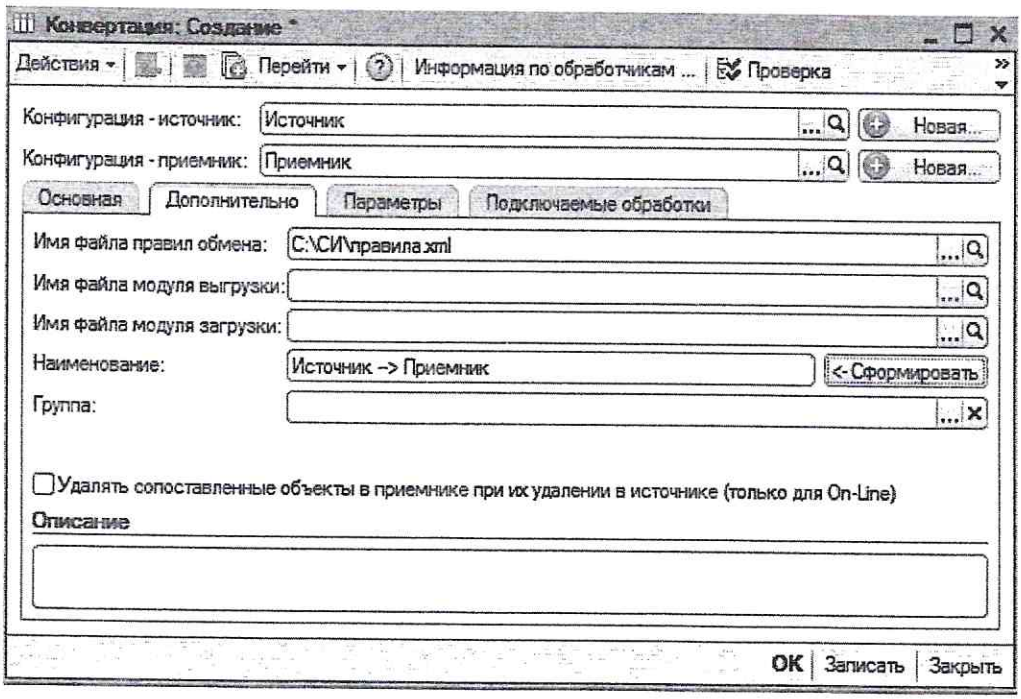
В открывшемся окне, указывая поочередно путь до файла метаданных источника и приемника, произведите загрузку данных из них.



После каждой загрузки метаданных можно по необходимости откорректировать информацию о загруженной конфигурации. Сделать это можно в окне следующего вида:

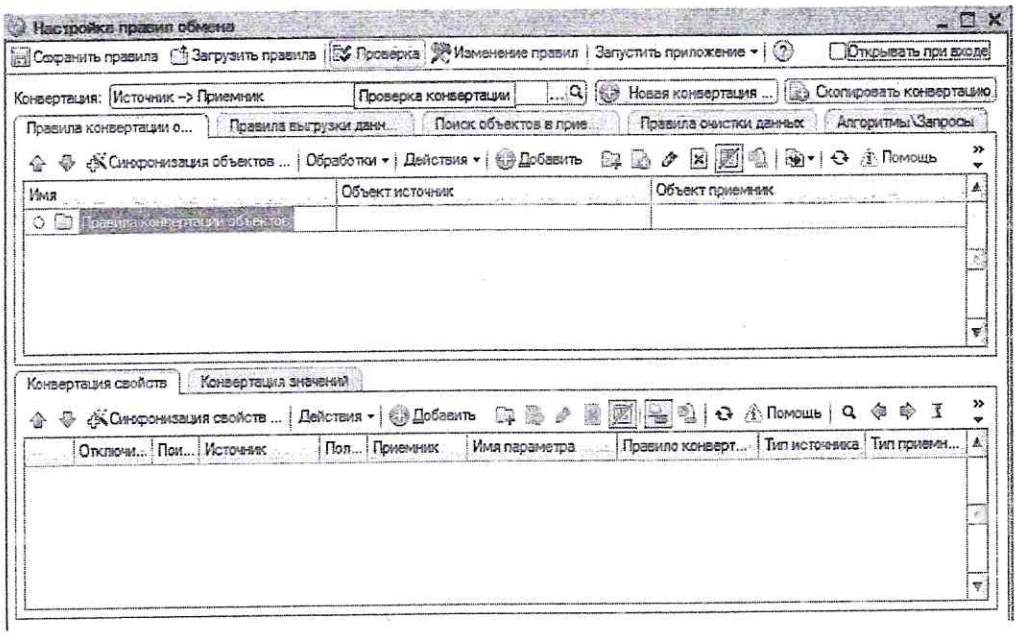


В панели функций выберите пункт "Правила обмена данными". Создайте новую конвертацию (укажите конфигурацию источник, приемник, путь до формируемого файла правил).



Откройте окно настройки правил обмена (одноименная кнопка командной панели или пункт панели функций).

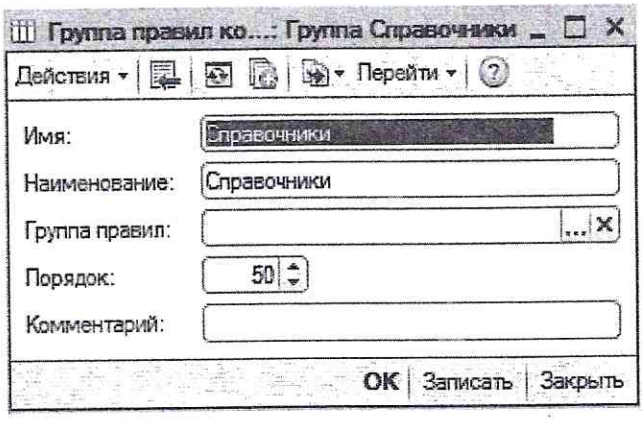
В открывшейся форме выберите вновь созданную конвертацию.



Все готово к настройке правил.

8.3. Перенос данных идентичных объектов

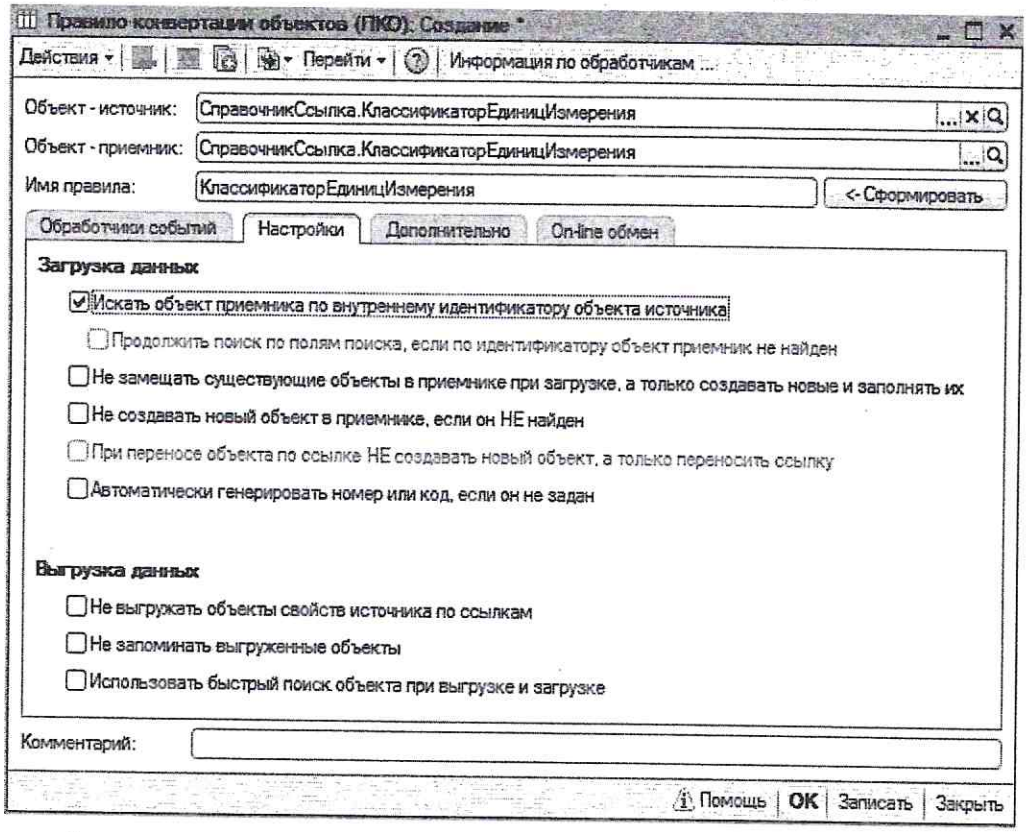
В форме "Настройка правил обмена" создайте первоначально группу правил "Справочники" (кнопка "Добавить группу" верхней панели).



Внутри группы создадим новое правило (кнопка "Добавить" той же панели). В качестве объекта источника и объекта приемника выберите справочники "КлассификаторЕдиницИзмерения" (справочники имеют одинаковые имена).

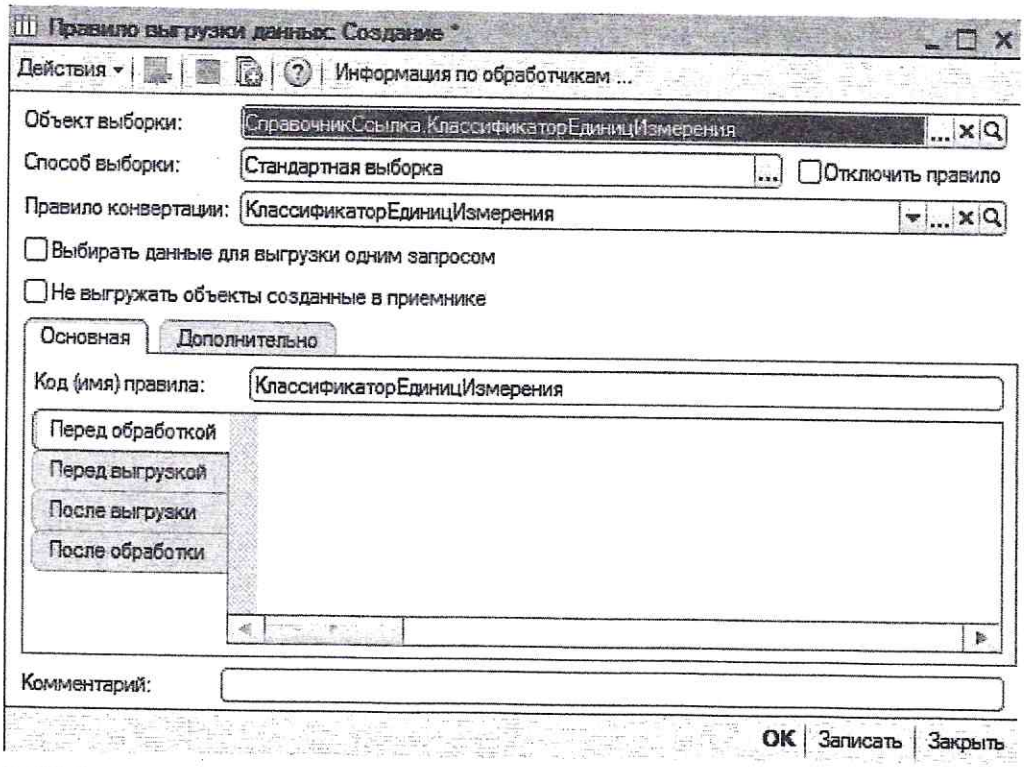
Имя правила сформируется автоматически.

При желании получить справочную информацию по обработчикам событий, представленным на первой странице формы можно нажать кнопку "Информация по обработчикам" верхней командной панели формы.



Справочники идентичны, поэтому можно "смело" соглашаться со всеми предложениями системы. Будет предложено автоматически синхронизировать свойства объектов и создать правило выгрузки данных.

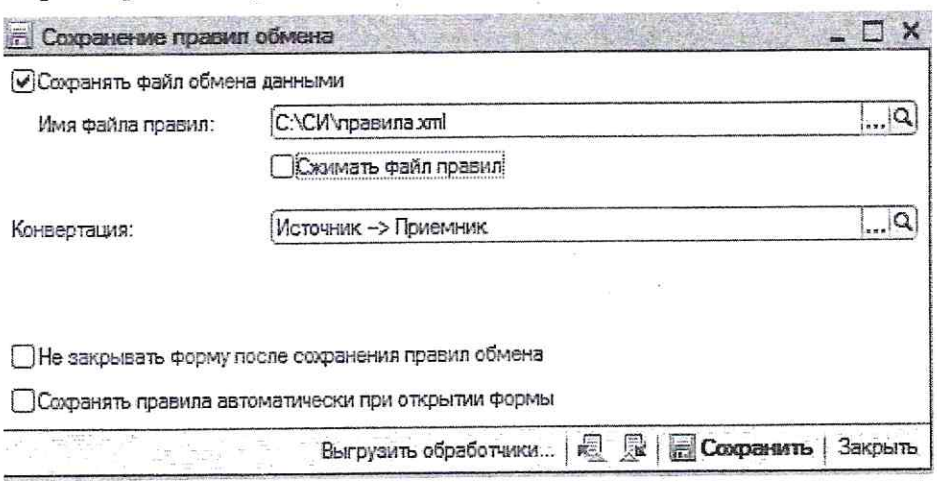
Созданное правило выгрузки данных можно посмотреть на третьей закладке обработки настройки правил обмена.



Следует отметить, что правило конвертации объектов (с подчиненными правилами конвертации свойств или значений) определяет способ конвертации данных (каким образом данные/значения одного объекта преобразуются в данные/значения другого объекта), а правило выгрузки данных определяет способ получения коллекции объектов, предназначенных к выгрузке (к которым будет применяться какое-либо правило конвертации объекта).

После выполнения указанных действий остается сохранить правила конвертации (используется соответствующая кнопка в форме настройки правил обмена).

При сохранении правил обмена откроется диалог следующего вида:



После сохранения файла правил произведите обмен данными между конфигурациями "Источник" и "Приемник". Проверьте полученный результат.

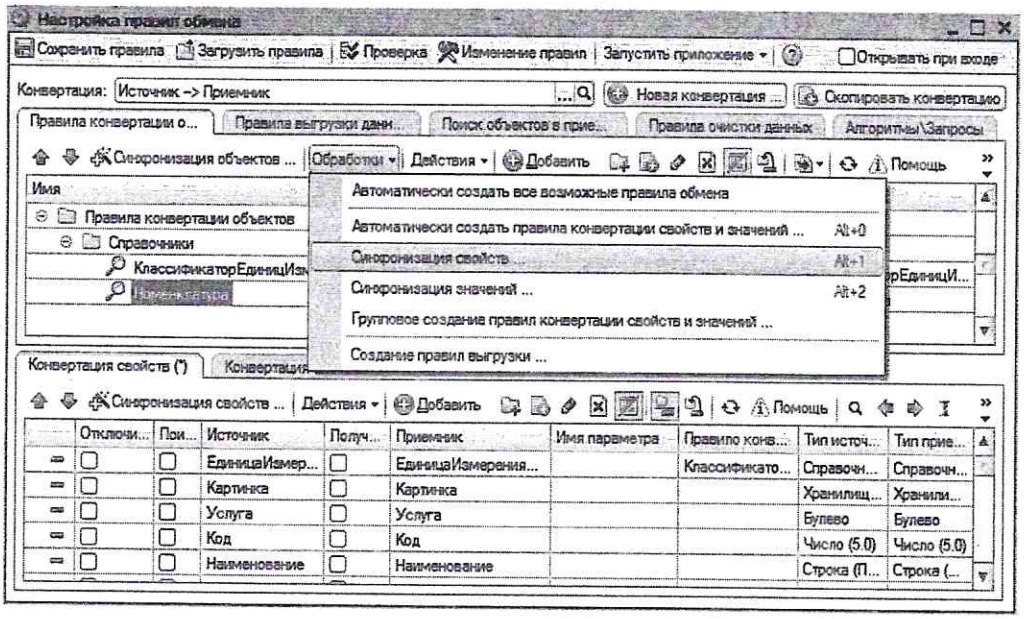
8.4. Перенос данных объектов с различной структурой

Поставим перед собой задачу перегрузить справочник "Номенклатура", но при этом не выгружать иерархию и значения свойств "ОсновнаяЕдиницаИзмерения".

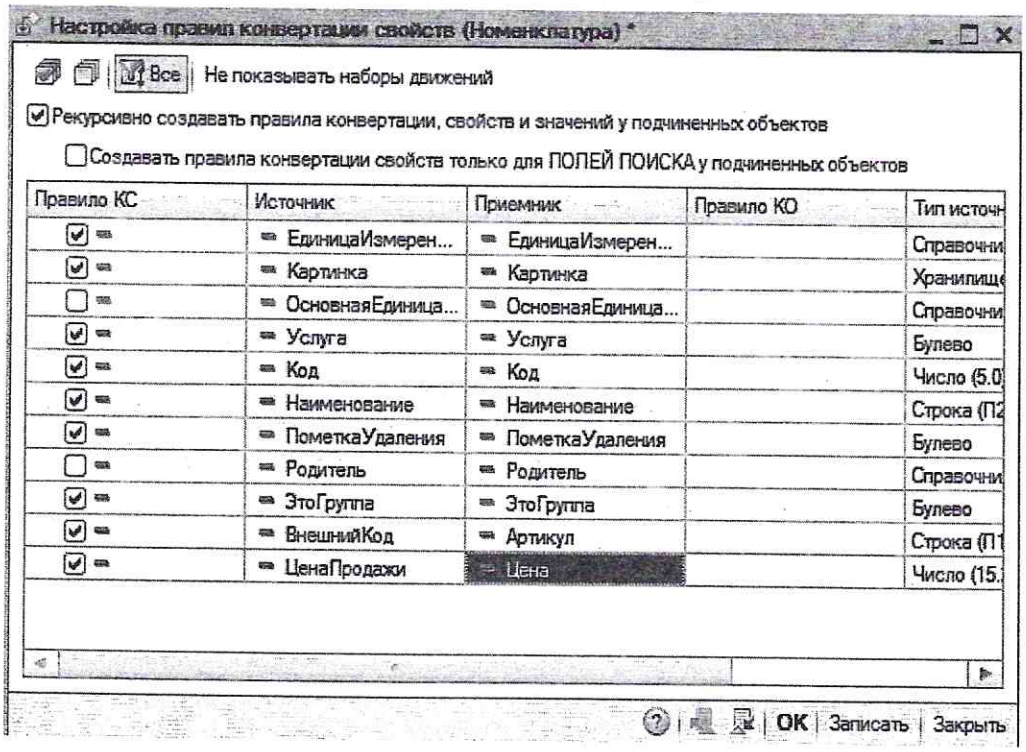
Дело в том, что в приемнике ограничено количество уровней иерархии (и этих уровней не хватит данным, которые находятся в источнике). Кроме этого справочник "ЕдиницыИзмерения" (реквизит "ОсновнаяЕдиницаИзмерения" имеет тип "СправочникСсылка.ЕдиницыИзмерения") является в приемнике подчиненным справочнику "Номенклатура". Тогда когда в источнике этот справочник "не зависимый".

8.4.1. Сопоставление реквизитов с разными именами

Создайте правило конвертации объектов для справочника "Номенклатура". Откажитесь от автоматической синхронизации и создания правила выгрузки данных. Синхронизацию свойств запустим "самостоятельно". Для этого в обработке "Настройка правил обмена" выберем пункт "Обработки/ Синхронизация свойств"



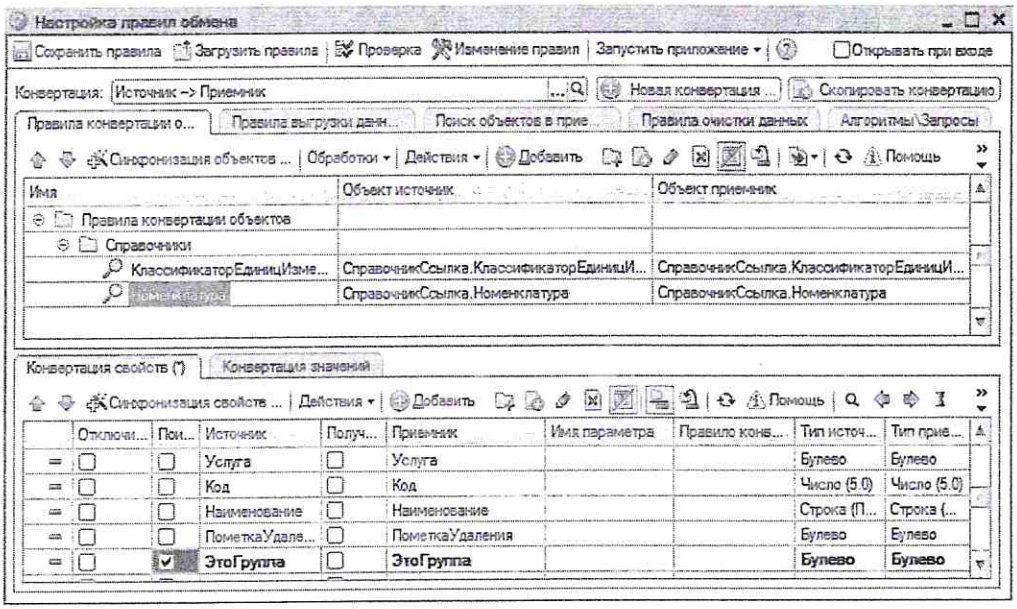
В открывшейся форме сопоставим реквизиты справочников:



Напротив строк с основной единицей измерения и родителем снимем флаги создания правил конвертации свойств.

Для корректной конвертации данных потребуется дополнительная настройка параметров синхронизации. Потребуется произвести отметку флага в колонке "Поиск" строки со свойством "ЭтоГруппа". Дело в том, что при создании объекта система оперирует только теми данными, по которым осуществляется

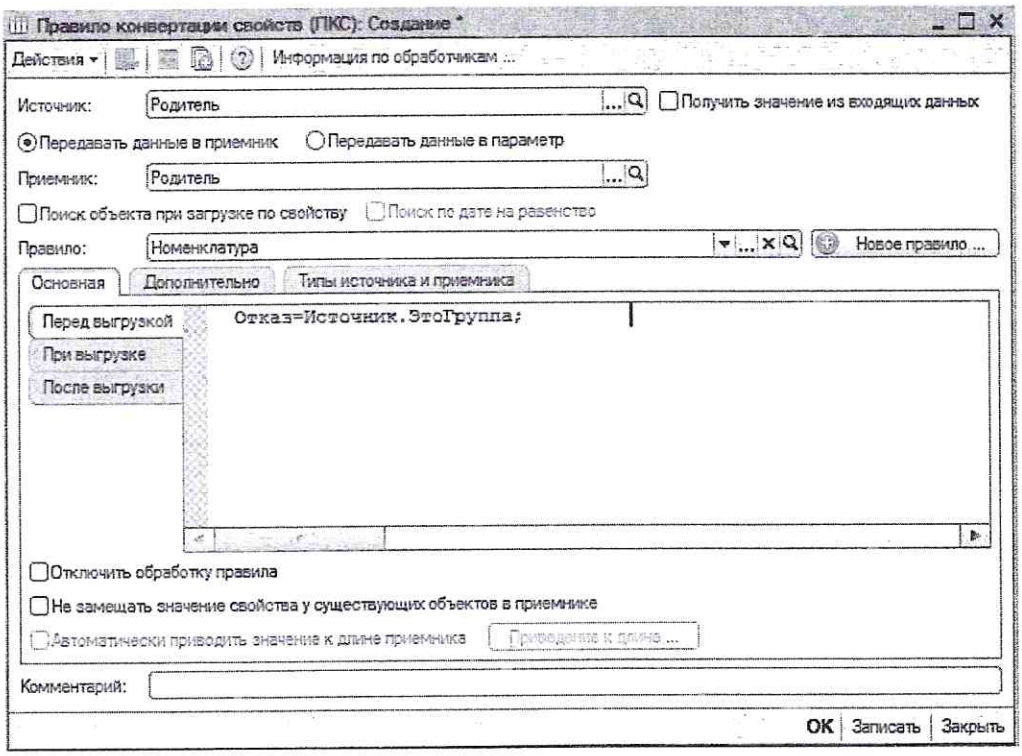
синхронизация (в данном случае уникальный идентификатор объекта и признак является ли запись справочника группой или элементом). Если флаг напротив свойства "ЭтоГруппа" оставить неотмеченным, то группы базы источника будут загружены в приемник как элементы справочника.



Сохраните файла правил и произведите обмен данными между конфигурациями "Источник" и "Приемник". Проверьте полученный результат (не забудьте о создании правила выгрузки данных).

8.4.2. Перенос данных с различающейся иерархией

Иерархия определяется свойством "Родитель". Добавим вручную правило конвертации свойств "Родитель->Родитель".



Так как в конфигурации "Приемник" количество уровней иерархии ограничено, то при выгрузке будем "отсекать" лишнюю иерархию. С этой целью

опишем обработчик события "Перед выгрузкой". Текст приведен на рисунке выше.

После сохранения файла правил произведите обмен данными между конфигурациями "Источник" и "Приемник". Проверьте полученный результат.

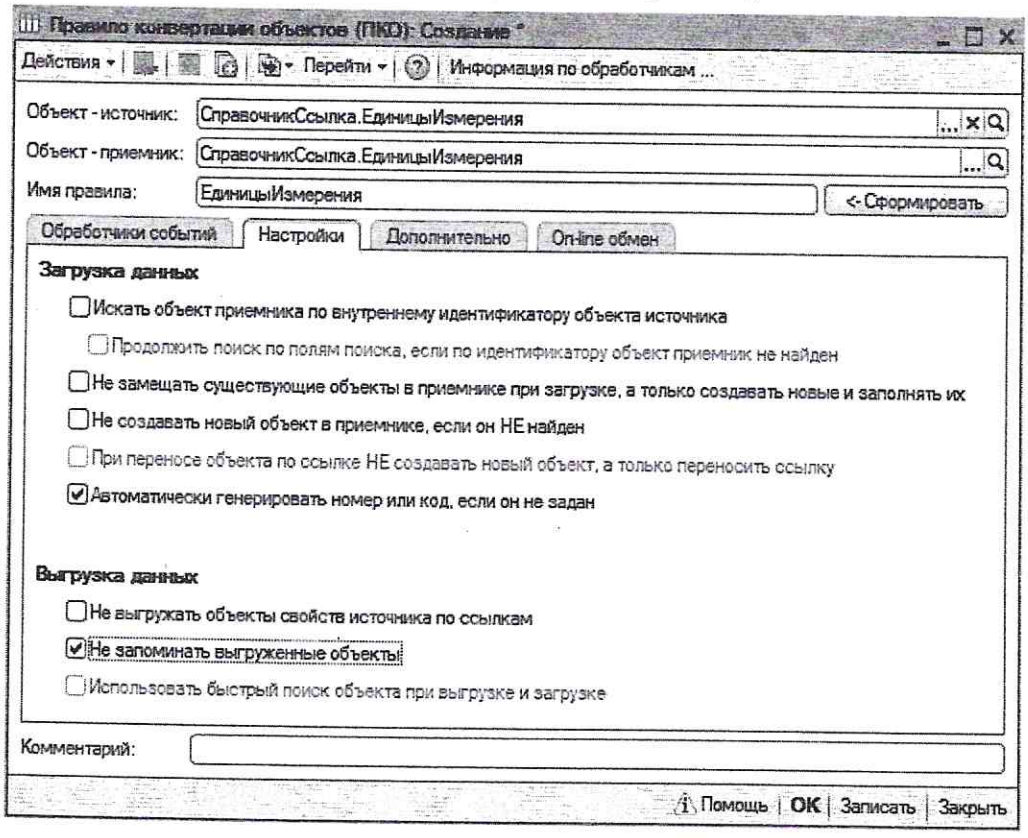
8.4.3. Перенос из обычного справочника в подчиненный

"Сложность" перегрузки данного свойства состоит в особенности организации справочника "ЕдиницыИзмерения" в обеих конфигурациях. В конфигурации "Приемник" данный справочник является подчиненным, в конфигурации "Источник" он не подчинен.

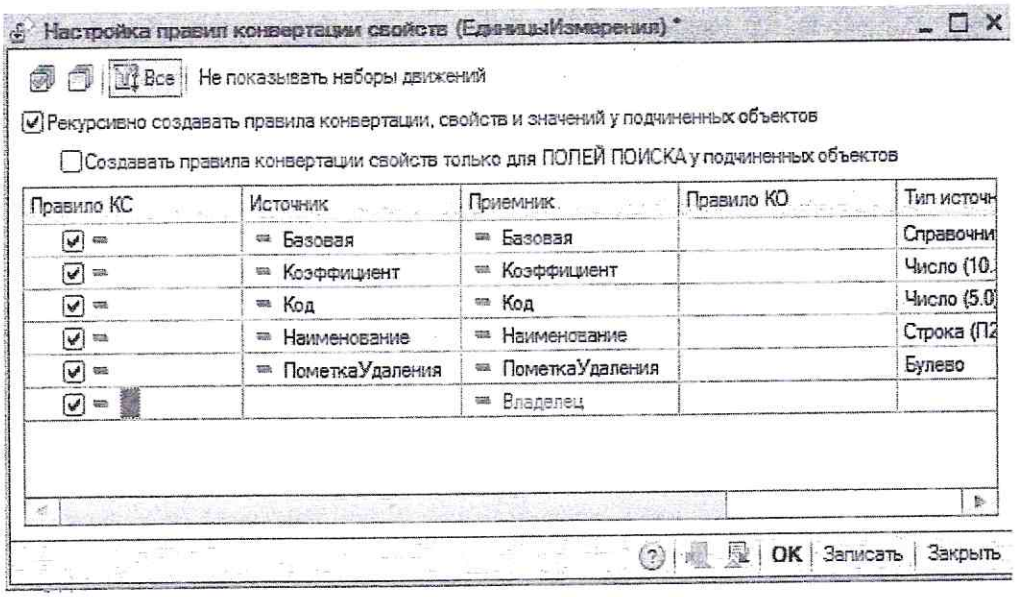
В конфигурации "Источник" на один элемент данного справочника может содержаться ссылка в разных элементах справочника "Номенклатура", в конфигурации "Приемник" в каждом элементе справочника "Номенклатура" может содержаться ссылка только на подчиненный ему элемент справочника "ЕдиницаИзмерения".

Таким образом возникает задача (в общем случае) выгрузки одного элемента справочника "ЕдиницаИзмерения" источника в несколько элементов справочника приемника.

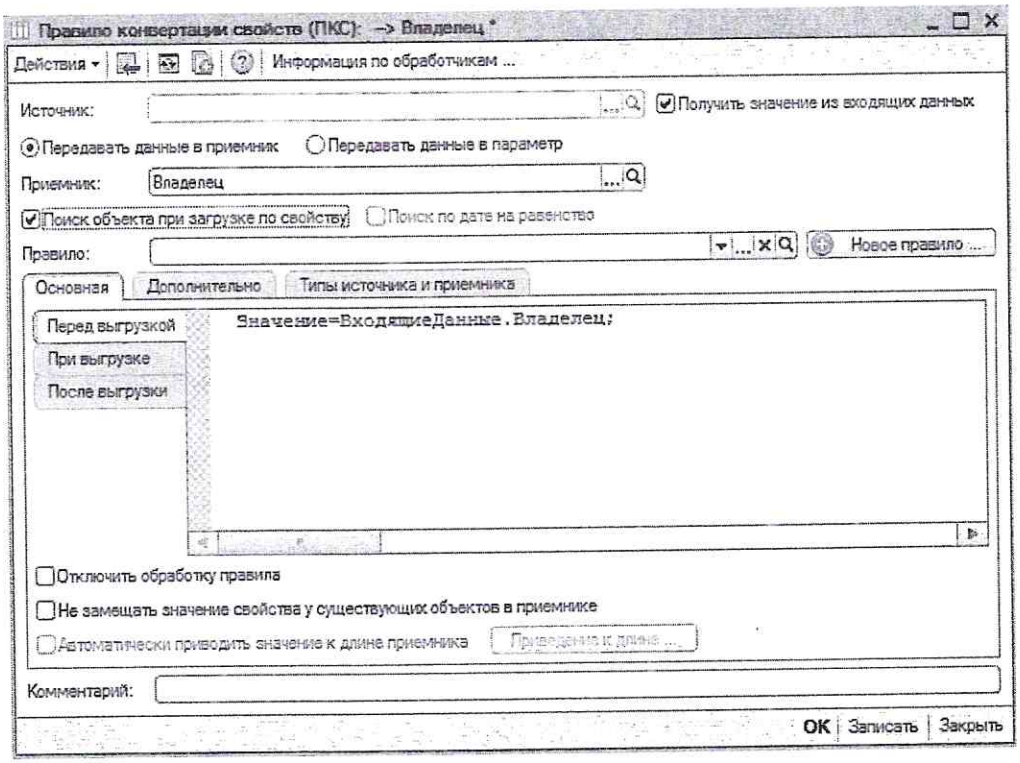
Создадим правило конвертации объекта для справочника "ЕдиницыИзмерения". Определим следующие настройки правила:



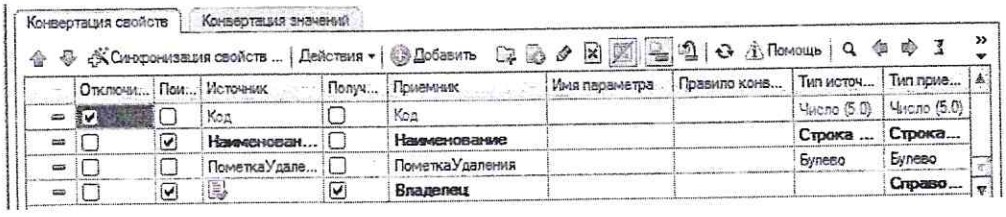
В настройке правил конвертации свойств отметим флаг напротив строки "Владелец"



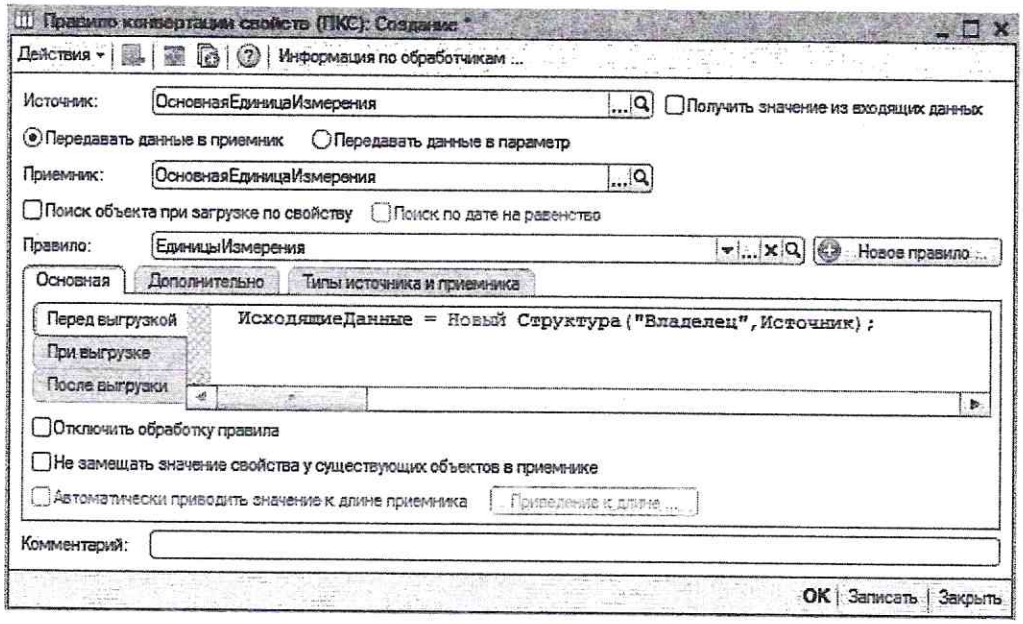
Доопределим правило конвертации свойства "Владелец". Помимо определения обработчика события потребуется отметить флаг "Получить значение из входящих данных".



Настроим синхронизацию:



Вернемся к справочнику "Номенклатура". Создадим правило конвертации свойств "ОсновнаяЕдиницаИзмерения" (у правила конвертации объектов). Текст обработчика события определим как на рисунке.

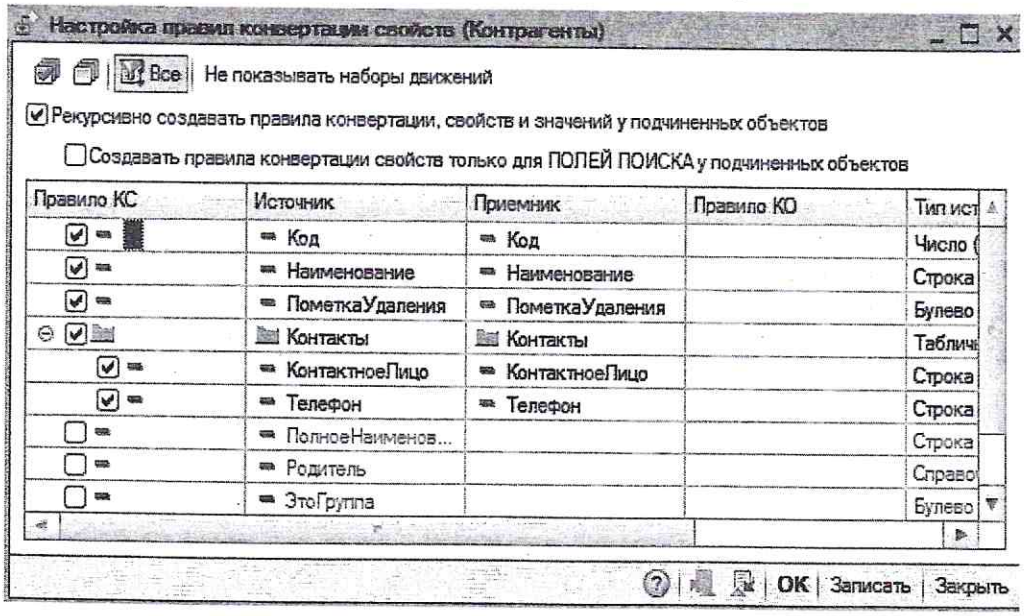


Следует отметить, что в данном случае попытка создания правила выгрузки данных для справочника "ЕдиницыИзмерения" приведет к возникновению ошибки.

После сохранения файла правил произведите обмен данными между конфигурациями "Источник" и "Приемник". Проверьте полученный результат.

8.4.4. Сопоставление табличных частей

Создайте правило конвертации для справочника "Контрагенты". Окно синхронизации свойств объектов выглядит следующим образом:



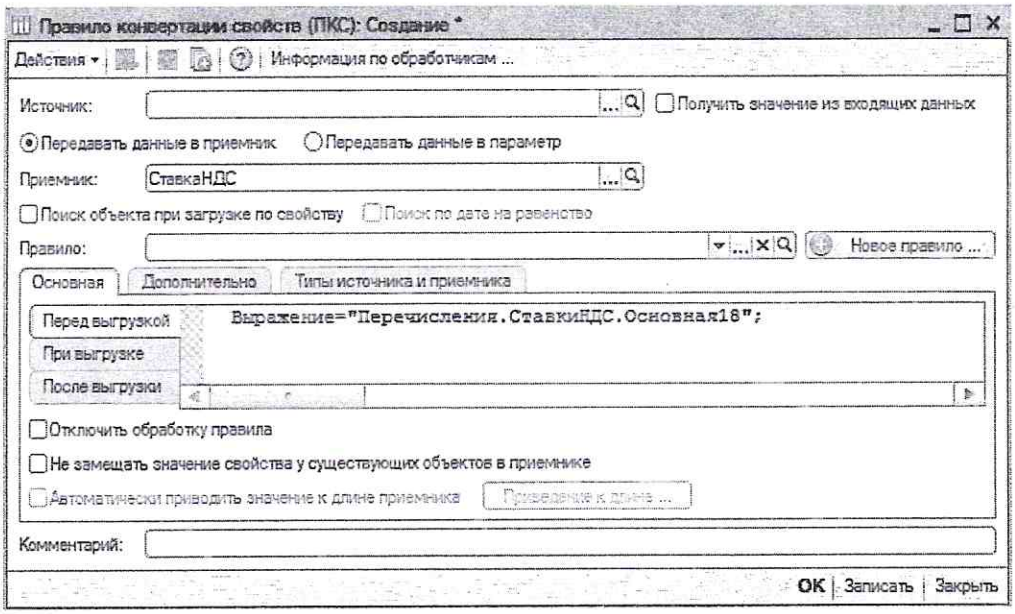
После сохранения файла правил произведите обмен данными между конфигурациями "Источник" и "Приемник". Проверьте полученный результат.

Практикум № 10

Создайте правило конвертации и соответствующее правило выгрузки для переноса данных из документа "Продажа товаров" источника в документ "Поступление товаров" приемника. При настройке правила конвертации для табличной части не определяйте правило свойства для ставок НДС.

8.4.5. Синхронизация элемента справочника со значением перечисления

В правиле конвертации объектов "Продажа товаров-> Поступление товаров" определите правило конвертации свойства "СтавкаНДС". Опишите обработчик события "Перед выгрузкой".



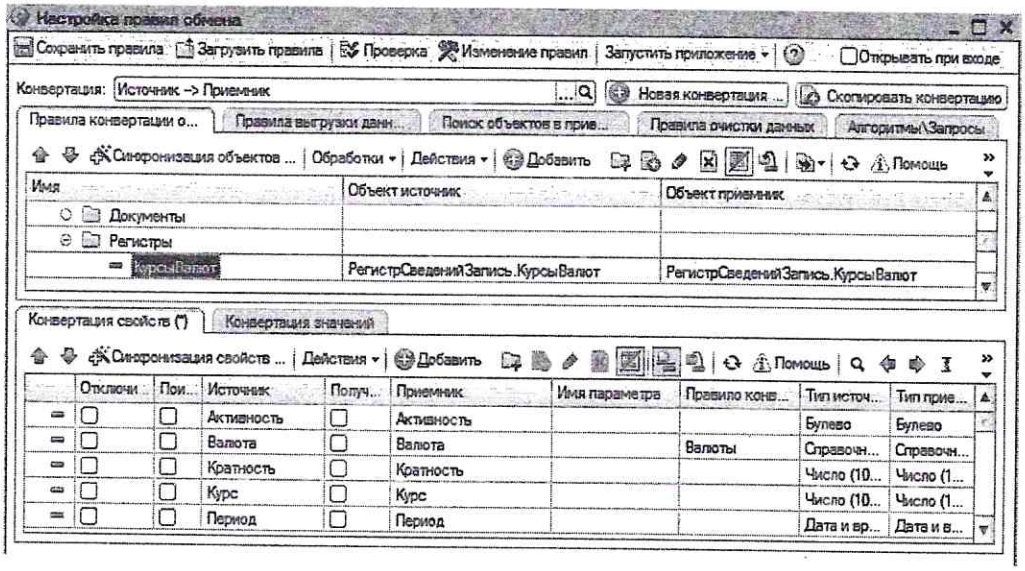
После сохранения файла правил произведите обмен данными между конфигурациями "Источник" и "Приемник". Проверьте полученный результат.

Практикум № 11

Используя параметр "ОбъектКоллекции" обработчика события "Перед выгрузкой" добейтесь "корректной" перегрузки ставки НДС. Ставка должна перегрузиться именно такой, какой она была в исходном документе.

Практикум № 12

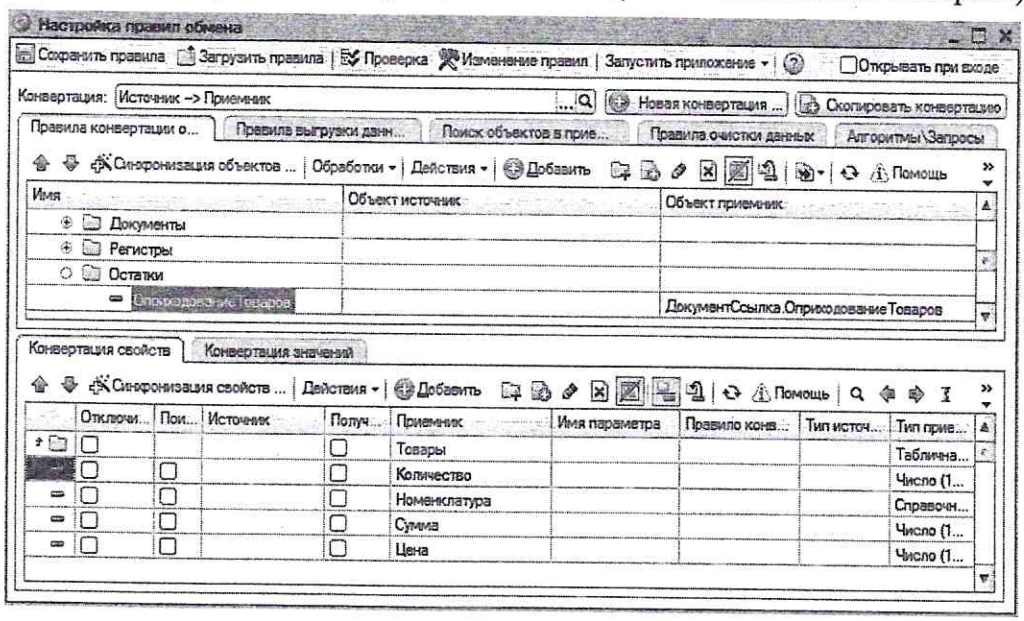
Создайте правило для переноса справочника "Валюты", регистра сведений "КурсыВалют"



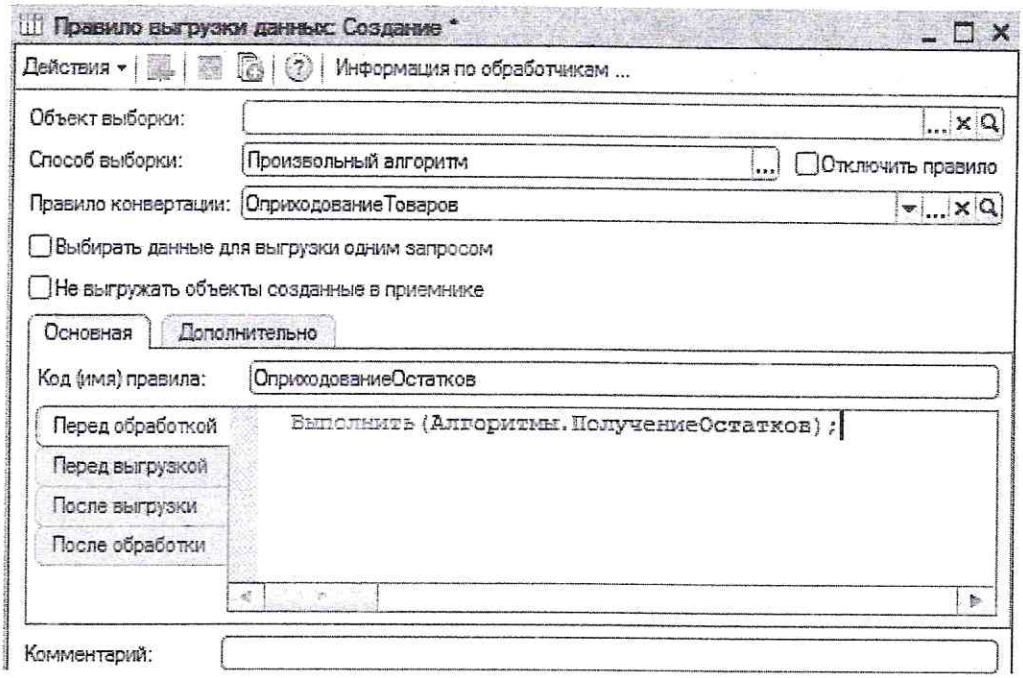
8.5. Перенос остатков

Поставим перед собой задачу перегрузить из источника в приемник остатки номенклатуры. Данные в источнике находятся в регистре накопления, загружать их будем в документ "Оприходование товаров" конфигурации приемника.

Создадим правило конвертации объектов (объект источник не выбираем).

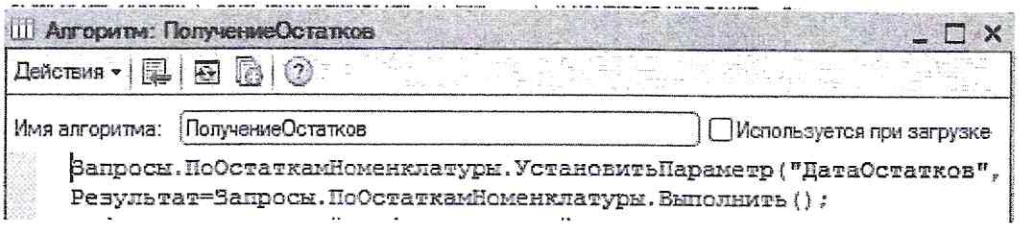


После этого необходимо создать правило выгрузки "Остатки". У него необходимо определить способ выборки "Произвольный алгоритм", и сопоставить ему правило конвертации "ОприходованиеТоваров". В обработчике события "Перед обработкой" прописать текст, представленный на рисунке.



Произвольный алгоритм подразумевает, то, что этот алгоритм необходимо дополнительно определить. Как раз это мы и делаем в обработчике события "ПередОбработкой". Но не будем в самом обработчике описывать весь алгоритм. Воспользуемся другой функциональностью конфигурации конвертации данных.

На закладке "Алгоритмы/Запросы" формы "Настройка правил обмена" создадим новый алгоритм (имя "ПолучениеОстатков"):

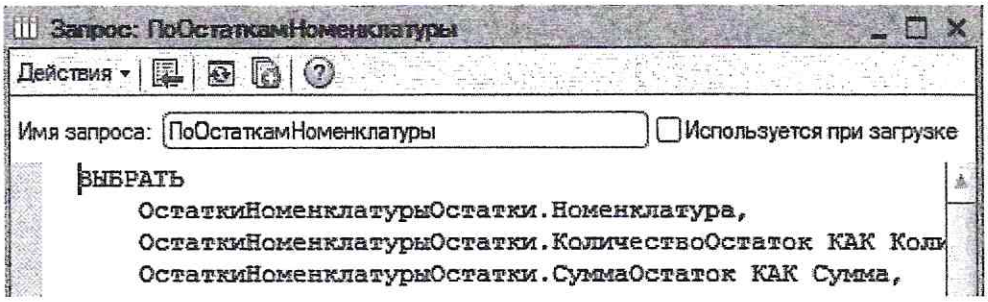


Текст алгоритма следующий:

```
Запросы.ПоОстаткамНоменклатуры.УстановитьПараметр("ДатаОста
тков", ДатаОкончания);

Результат=Запросы.ПоОстаткамНоменклатуры.Выполнить();

ВыборкаДанных=Новый ТаблицаЗначений;

ВыборкаДанных.Колонки.Добавить("Товары");

Стр=ВыборкаДанных.Добавить();

Стр.Товары=Результат.Выгрузить();
```

Таким образом, описанный алгоритм в свою очередь использует возможность работы с общими (для правила конвертации) запросами.

На той же закладке где определяли алгоритм, определите новый запрос (имя "ПоОстаткамНоменклатуры"):



Текст запроса следующий:

```
ВЫБРАТЬ

	ОстаткиНоменклатурыОстатки.Номенклатура,

	ОстаткиНоменклатурыОстатки.КоличествоОстаток КАК
Количество,

	ОстаткиНоменклатурыОстатки.СуммаОстаток КАК Сумма,

	ВЫБОР

		КОГДА
ОстаткиНоменклатурыОстатки.КоличествоОстаток ЕСТЬ NULL

			ТОГДА 0

		ИНАЧЕ ОстаткиНоменклатурыОстатки.СуммаОстаток /
ОстаткиНоменклатурыОстатки.КоличествоОстаток

	КОНЕЦ КАК Цена

ИЗ

	РегистрНакопления.ОстаткиНоменклатуры.Остатки(&ДатаОст
атков) КАК ОстаткиНоменклатурыОстатки
```

Запрос производится по виртуальной таблице остатков регистра накопления "ОстаткиНоменклатуры". При обращении к виртуальной таблице используется параметр "ДатаОстатков".

Проверьте созданный механизм на практике.

Практикум № 13

Настройте перенос документов "Продажа товаров" из конфигурации DB3 в документ "Поступление товаров" конфигурации DB4. При этом должны переноситься и "сопутствующие" справочники. При перегрузке значение реквизита "Организация" переносится в реквизит "Контрагент", реквизит "Организация" заполняется из соответствующей константы конфигурации приемника.

Выгружаются только измененные (с момента прошлой выгрузки) расходные накладные.

9. Мобильная платформа

9.1. Введение (выдержки с "http://v8.1c.ru/overview/Term_000000818.htm")

Мобильная платформа "1С:Предприятия 8" - это общее название технологии, позволяющей создавать приложения, работающие на мобильных устройствах под управлением операционных систем Android или iOS. Такими устройствами, как правило, являются различные смартфоны и планшетные ПК.

Мобильное приложение, установленное на устройстве, представляет собой совокупность мобильной платформы и информационной базы. Информационная база на мобильном устройстве содержит аналог файловой базы данных (для хранения данных, с которыми работает пользователь) и мобильное приложение (программный код, исполняющийся на мобильном устройстве).

Основным назначением мобильных приложений является организация удаленных рабочих мест для прикладных решений, функционирующих на стационарных компьютерах. С такими приложениями стандартными средствами платформы организуется постоянный обмен данными в режиме off-line.

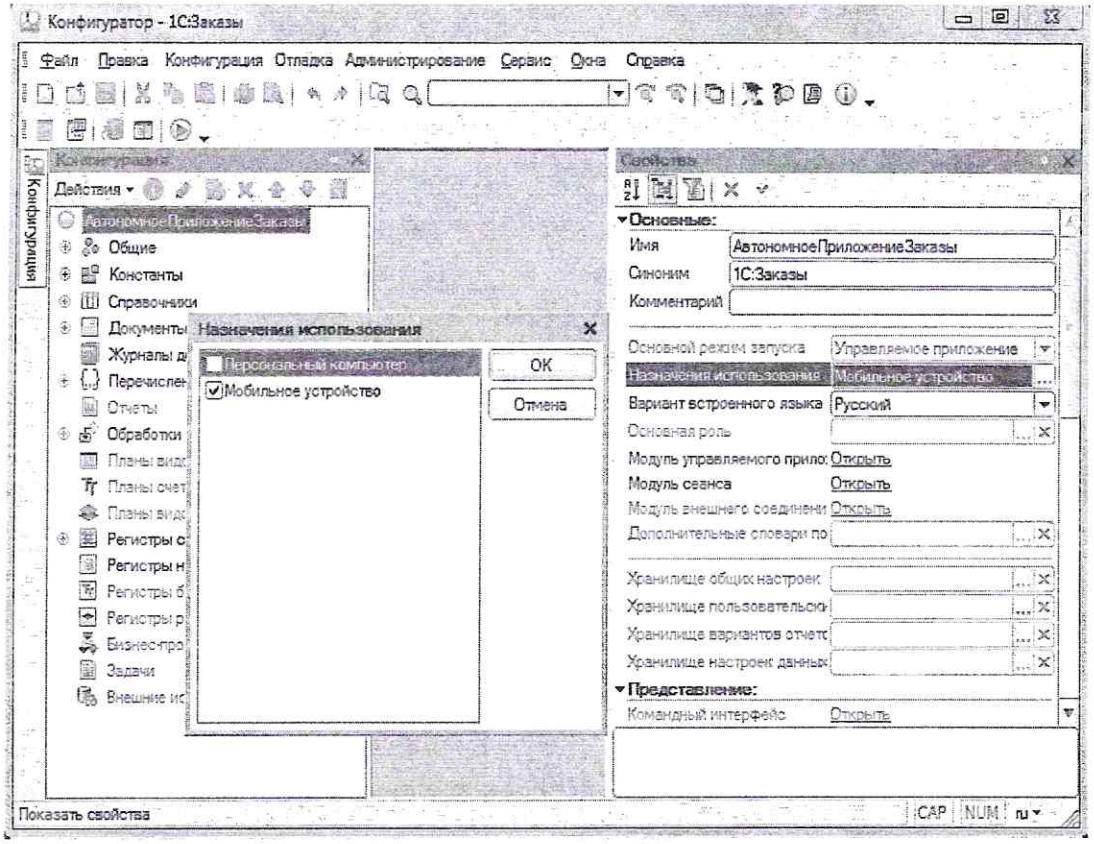
Разработка мобильных приложений ведется так же, как и разработка "обычных" приложений, с той лишь разницей, что необходимо учитывать ограничения, которые накладывает мобильная платформа:

- используются не все классы объектов конфигурации;
- не используется язык запросов и система компоновки данных;
- не используется механизм распределенных информационных баз;
- используется ограниченный набор элементов формы;
- рабочий стол содержит только одну форму;
- не поддерживается пошаговая отладка;
- и др.

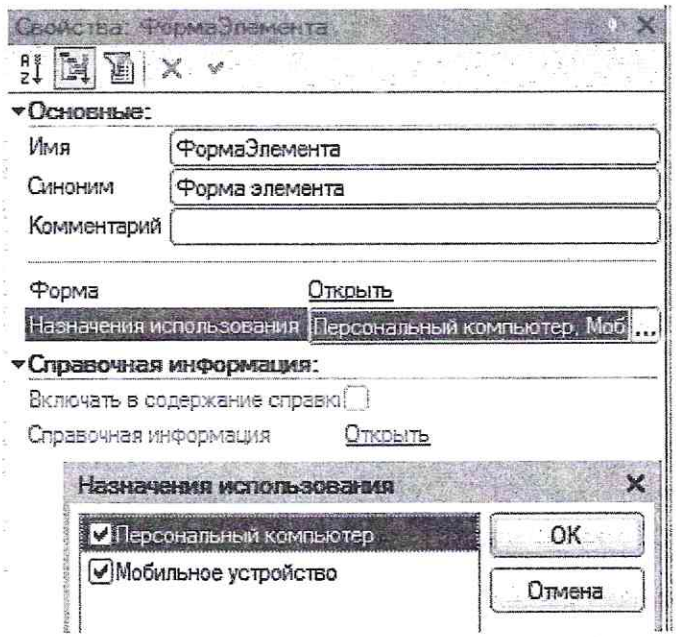
Поэтому для конфигурации, разрабатываемой как мобильное приложение, следует устанавливать свойство НазначениеИспользования в значение МобильноеУстройство. В этом случае система автоматически скроет возможности, недоступные для мобильной платформы, а рабочие инструменты (проверка синтаксиса, проверка конфигурации и пр.) будут настроены на работу именно с тем контекстом встроенного языка, который доступен в мобильной платформе.

9.2. Разработка базы данных

Как было сказано выше, разработка такого приложения ведется привычным образом с использованием конфигуратора. Необходимо только поставить свойство "Назначение использования" в нужное значение:



Следует обратить внимание на то, что подобное свойство появилось и в свойствах форм.



9.3. Предварительная настройка

Для того, чтобы собрать мобильное приложение требуется предварительно установить на рабочей машине Android SDK и Java SDK.

Android SDK можно получить по адресу: http://developer.android.com/sdk/index.html

Минимальные требования к Android SDK следующие:

- Версия Android SDK Tools - не ниже 20.0.3;
- Версия Android SDK Platform-tools - не ниже 14;
- Версия SDK Platform - API 8 (не ниже версии 8.3).

Java SDK можно получить по адресу: http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u25-download-346242.htm

Нужно установить именно JDK6, а не JDK7..

С сайта фирмы 1С необходимо получить мобильную платформу (Методическая поддержка/ Интернет поддержка пользователей). Полученный архив (mobile.zip) распаковать на жесткий диск компьютера.

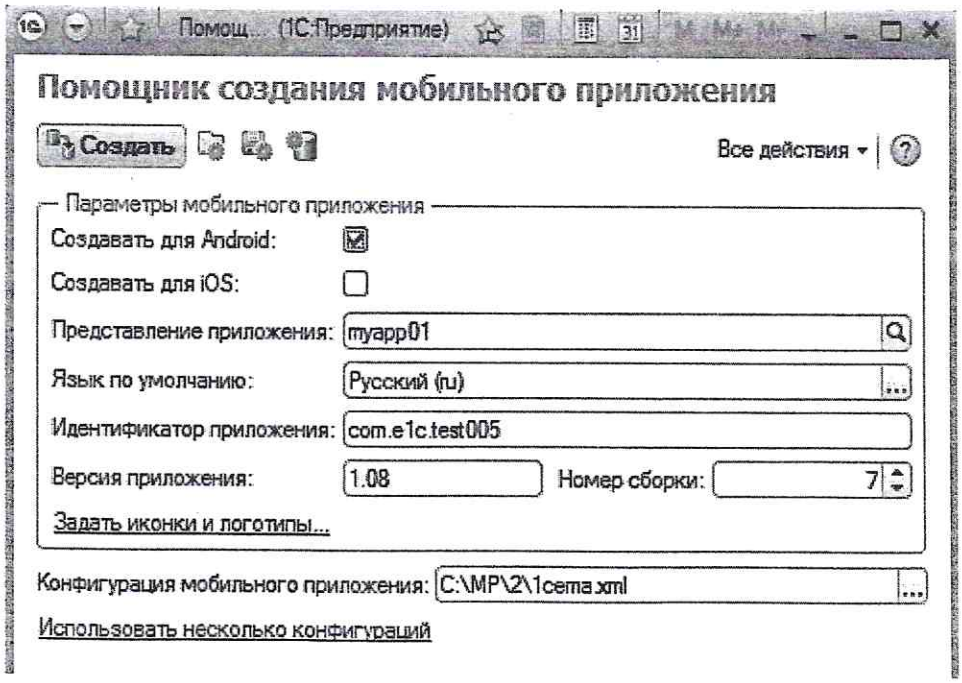
9.4. Сборка мобильного приложения

После подготовки базы данных необходимо выгрузить описание мобильного приложения в xml документ. Делается это из конфигуратора с помощью команды "Конфигурация/Мобильное приложение/Выгрузить в файл".

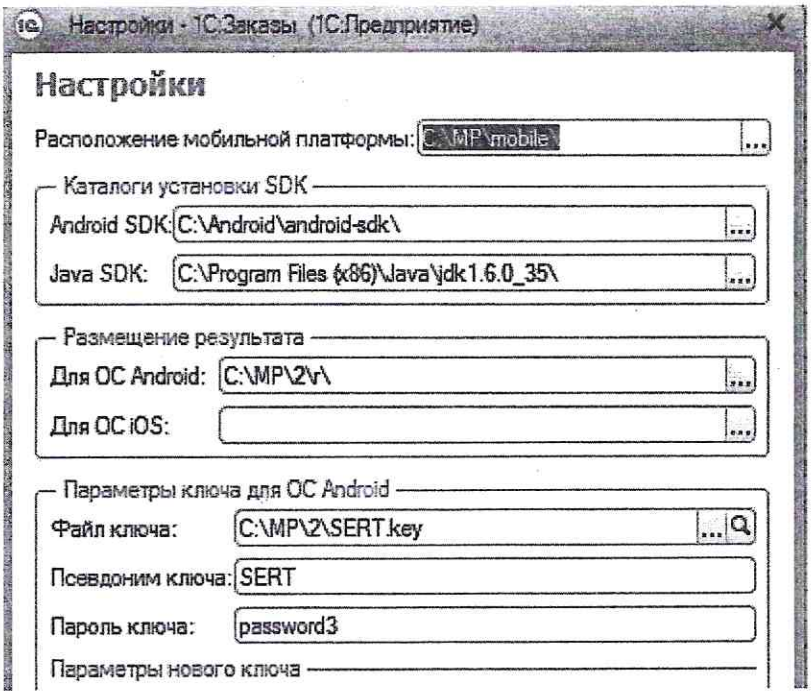
Для сборки мобильного приложения потребуется обработка "MobileAppWizard.epf" (входящая в состав мобильной платформы).

Назначение полей описано в справочной системе самой обработки.

Пример заполнения:



Необходимо также заполнить окно с настройками:

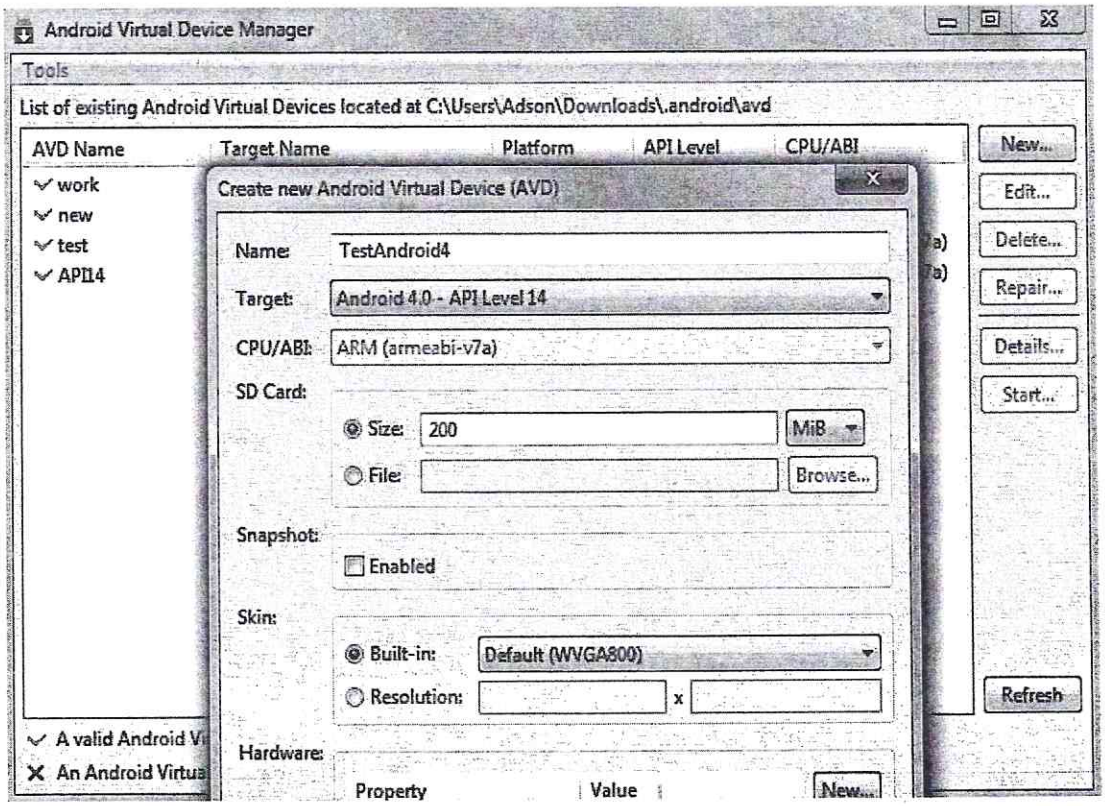


9.5. Тестирование приложения

После создания мобильного приложения его можно скопировать и установить на мобильное устройство.

Приложение под ОС Android можно протестировать с помощью менеджера AVD (входящий в Android SDK).

В данном менеджере можно создать виртуальное устройство с выбранной версией ОС.



После создания необходимо запустить виртуальное устройство.

Для того, чтобы переместить на виртуальную машину созданный apk файл потребуется в командной строке выполнить следующую команду:

C:\Android\android-sdk\platform-tools\adb install C:\Путь до файла\имя.apk

Если у вас Android SDK находится по другому пути, требуется откорректировать команду.

После установки приложения можно проверить его работу