Занятие 4. Механизмы, лежащие в основе технологии: XML, XDTO, Универсальный формат

XML и XDTO

Если открыть любой XML- файл, можно увидеть, что это текстовый формат. Но это не простой текст, а структурированный с помощью специальных элементов разметки – тэгов. Открывающий и закрывающий тег и все, что находится между ними, представляет собой узел. В этот узел могут быть вложены другие узлы, уровней вложенности может быть много, но корневой узел должен быть один. Поэтому структура любого файла XML представляет собой дерево, в которое можно поместить любые данные.

Формат XML удобен для хранения и передачи самых разнообразных данных, и используется очень широко. Все обмены между базами 1С используют этот формат для передачи данных. Дополнительная информация для обменов – файлы правил (для обменов по правилам) и структура метаданных конфигураций – тоже переносится через XML-файлы.

В языке 1С есть несколько подходов к работе с XML-файлами:

- Работа на уровне узлов – создание, заполнение, добавление подчиненных узлов и запись в файл практически «вручную». Этот подход используется для всех обменов по правилам.
- XML-Сериализация – это автоматическая трансформация средствами платформы объектов в узлы XML и обратно – без изменения структуры этих объектов. Используется, например, для обменов в распределенных базах, где изменять объекты не нужно, поскольку конфигурации идентичны.
- Преобразование XSL. Эта технология в обменах 1С не используется
- XDTO-Сериализация, которая легла в основу новой технологии обмена через универсальный формат.

Аббревиатура **XDTO** расшифровывается как XML Data Transfer Objects и используется только в 1С.

В основе XDTO-Сериализации лежит **Объект XDTO**. Этот объект характеризуется тем, что может быть автоматически одной командой записан в XML-файл, а также прочитан из XML-файла – тоже одной командой. Объект XDTO может иметь различные свойства, в которые записаны различные значения – **Значения XDTO**, и в этом он похож на объект информационной базы. Средствами языка 1С с ним можно работать, как с объектом информационной базы – создавать, заполнять и записывать – но не в информационную базу, а в XML-поток.

Каждый Объект XDTO имеет определенную структуру, которая описана в **XDTO-Пакете**. Если Объект XDTO можно сравнить с объектом информационной базы, то XDTO-Пакет аналогичен дереву метаданных – в нем описано, какие могут быть созданы Объекты XDTO (то есть **Типы объектов XDTO**), какие у них должны быть свойства. И какие значения могут быть записаны в эти свойства – то есть **Типы значений XDTO**. XDTO-Пакеты можно самостоятельно создавать в дереве Конфигурации, редактировать, выгружать в файл и загружать из файла.

В каждой конфигурации может быть множество XDTO-пакетов. Некоторые можно увидеть, если открыть в дереве Конфигурации ветку Общие – XDTO-Пакеты. Некоторые в этой ветке не отображаются, они «зашиты» в платформу, но их тоже можно использовать. У каждой схемы есть свой уникальный идентификатор – **Пространство имен**. Пространство имен – это строка, которая обычно выглядит, как интернет-адрес, и может действительно из себя представлять ссылку на сайт, где описаны входящие в пакет типы данных. А может быть просто строкой, главное – нужно следить, чтобы она была уникальна в пределах конфигурации. Контроль средствами платформы за этим не выполняется, но если создать два пакета с одинаковым пространством имен, то при их использовании могут возникнуть неявные ошибки.

XDTO-Пакет можно выгрузить в файл, который будет иметь разрешение .xsd. XSD расшифровывается как XML Schema definition – описание Схемы XML. **Схема XML** – это не изобретение фирмы 1С, а всемирно используемый стандарт. Она представляет из себя описание структуры XML-документа. В ней описано, какие в документе XML могут быть узлы, какие узлы в них могут быть вложены, атрибуты, которые могут быть у каждого из этих узлов, и так далее.

Из этого следует, что XML-файл данных, созданный на основании конкретного XDTO-Пакета, и сам этот пакет, выгруженный в файл *.xsd, представляют из себя набор, достаточный для того, чтобы выполнить загрузку этих данных в любую программу.

В обменах по правилам на основании файла правил выгрузка данных в XML-файл производится вместе с инструкциями по загрузке этих данных в конкретную базу с конкретной конфигурацией. Если вдруг конфигурация базы-приемника изменяется, файл данных становится неактуален, нужно изменять файл правил и выгружать данные повторно.

Технология КД 3.0 и Обмен через Универсальный формат используют именно XDTO-Сериализацию, поэтому файл выгруженных данных не теряет своей актуальности. Этот файл содержит только сами данные, в нем нет ни информации ни о загрузке этих данных, ни о структуре конечной конфигурации. Вместо этого в файле с XML- схемой содержится структура этого файла. А логика загрузки, основанная на этой XML-схеме, должна быть описана в той базе, куда будет производиться загрузка.

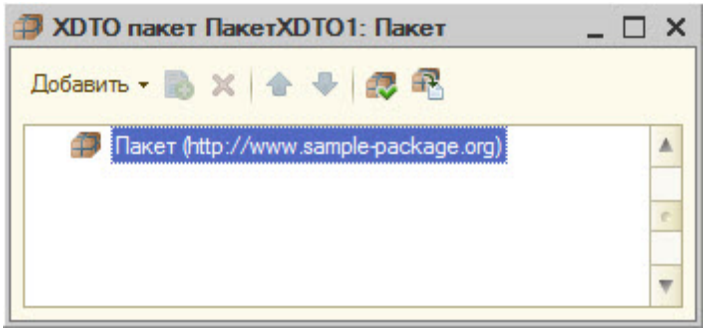
Перенос документов с помощью XDTO-Сериализации

В этом занятии перенесем из Исходной ИБ в Конечную документы *ПриемНаРаботу*. В этом документе есть реквизит *ДолжностьСотрудника* типа *СправочникСсылка.ДолжностиСотрудников*. А также есть табличная часть *Сотрудники*, и в ней реквизит «Сотрудник» типа *СправочникСсылка.Сотрудники*.

Эти объекты невозможно перенести с помощью универсального формата, так как среди объектов формата нет аналогичных справочников и документов. Поэтому напишем собственные обработки, которые будет переносить эти данные при помощи XDTO-Сериализации. Необходимо перенести все документы, а также все элементы справочников, которые в них фигурируют.

Чтобы иметь возможность выполнять XDTO-Сериализацию сначала необходимо создать XDTO-Пакет, в котором будут описаны Типы объектов XDTO, аналогичные объектам ИБ, которые необходимо перенести.

Откроем дерево Конфигурации *Исходная ИБ*, найдем ветку *Общие – XDTO-Пакеты* и создадим новый пакет с помощью команды *Добавить*.

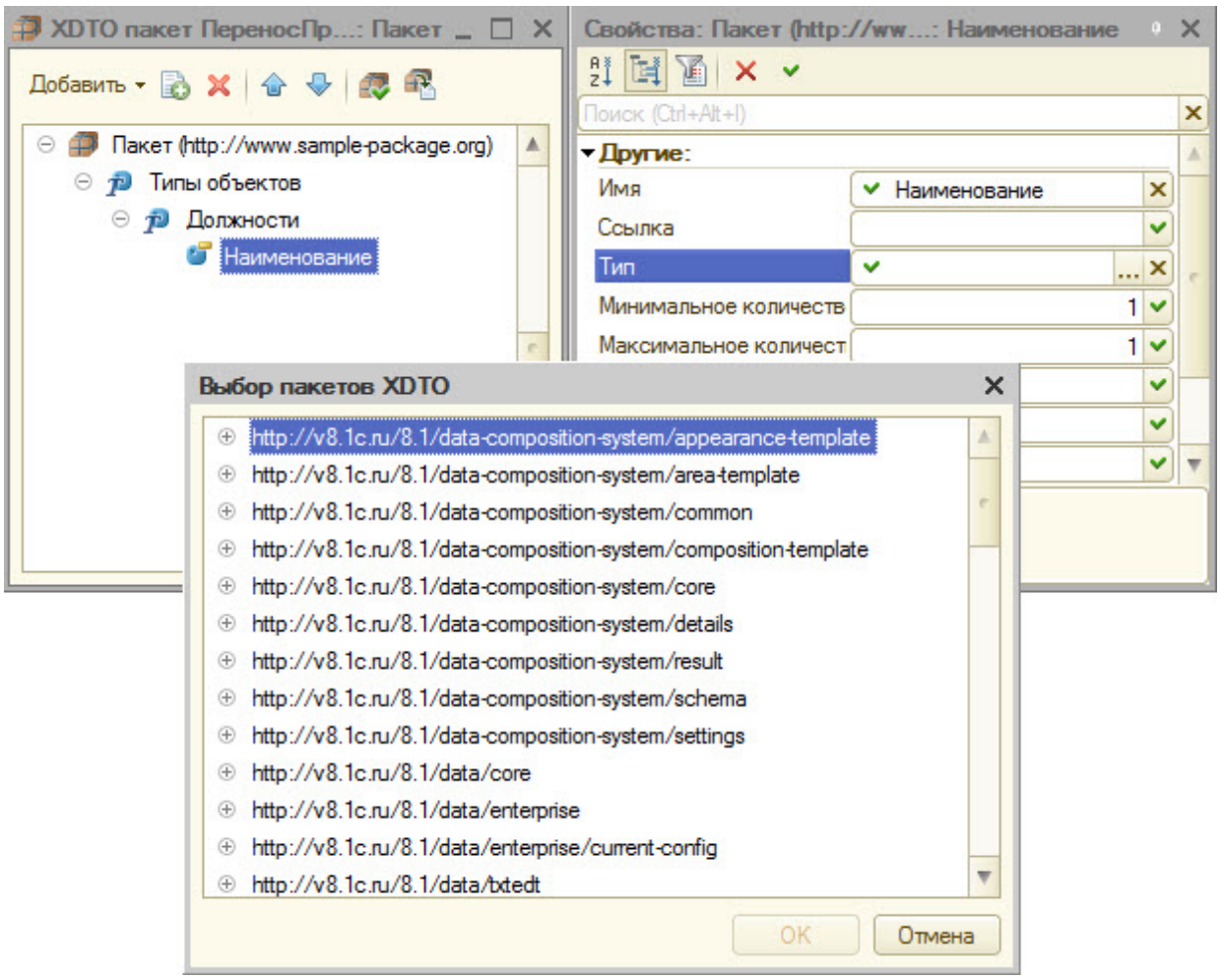


Новый XDTO пакет имеет по умолчанию пространство имен «http://www.sample-package.org». Оно не совпадает с другими пространствами имен в конфигурациях *Исходная* и *Конечная*, поэтому можем оставить его без изменений.

В дереве *Конфигурации* щелкнем правой кнопкой по этому пакету, выберем его свойства и зададим ему имя *ПереносПриемаСотрудников*.

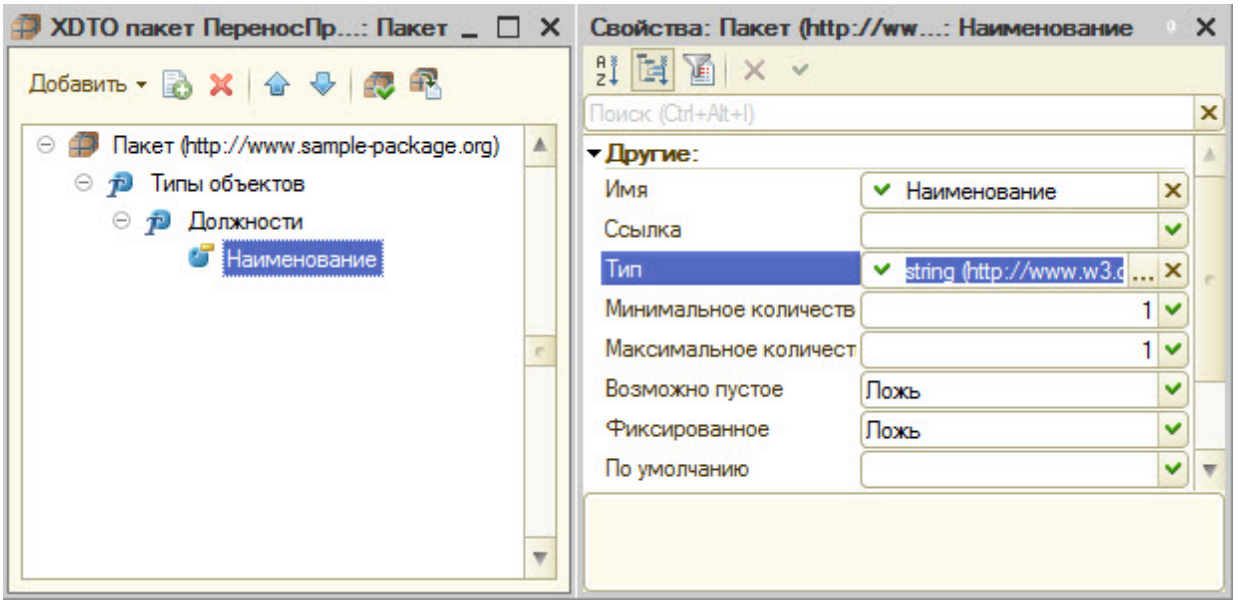
С помощью кнопки *Добавить* создадим нужные типы объектов. Типы объектов имеет смысл создавать так, чтобы они были аналогичны объектам информационной базы, тогда будет проще производить преобразование. Первым делом создадим новый тип объекта и назовем его *Должности*. Он будет использоваться для переноса справочника «Должности сотрудников». У этого справочника необходимо будет перенести только одно свойство – Наименование. Поэтому у типа объектов Должности добавим новое свойство Наименование.

Для стабильной работы обменов всем свойствам рекомендуется указывать типы данных. Кликнем по полю *Тип* в панели свойств для свойства *Наименование*. При этом открывается список всех возможных пакетов XDTO.



Здесь указаны не названия этих пакетов, а пространства имен. Часть этих пакетов находится в ветке *XDTO-Пакеты* в дереве *Конфигурации*, другие «зашиты» в платформу.

Для свойства *Наименование* нужно указать тип данных *Строка*. Примитивные типы данных описаны в пакете «http://www.w3.org/2001/XMLSchema», который находится в самом низу списка. Развернем его и выберем тип *string*.



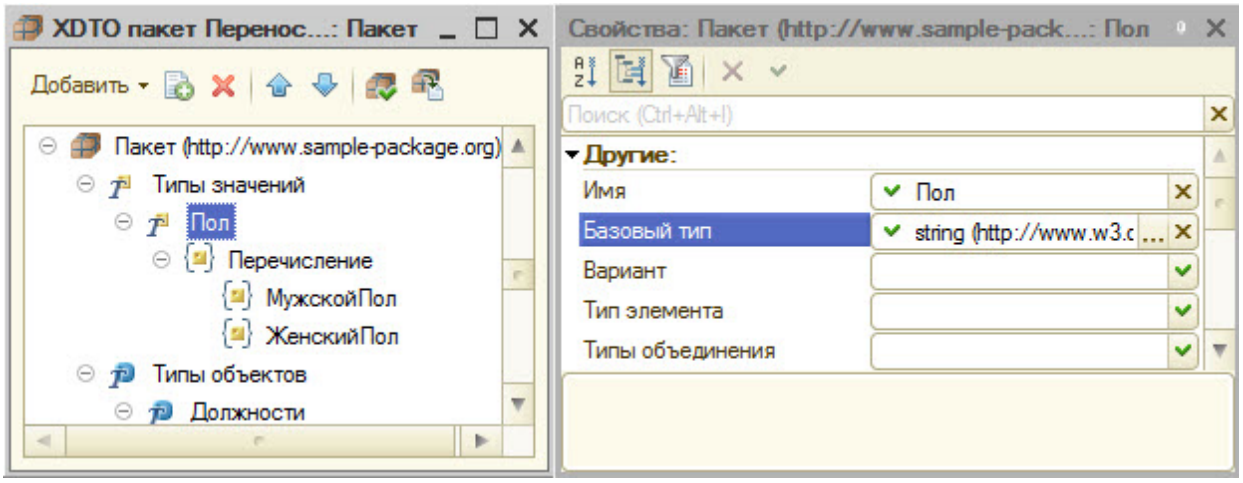
Теперь создадим новый тип объекта и назовем его Сотрудники. В Исходной конфигурации у этого справочника есть реквизиты *Дата рождения* типа *Дата* и *Пол* типа *ПеречислениеСсылка.Пол*.

Создадим у нового типа объектов соответствующие свойства. Кроме того, нужно перенести Наименование, для него создадим свойство ФИО. Укажем типы свойств: для свойства *ДатаРождения* укажем примитивный тип date, а для свойства *ФИО* – тип *string*.

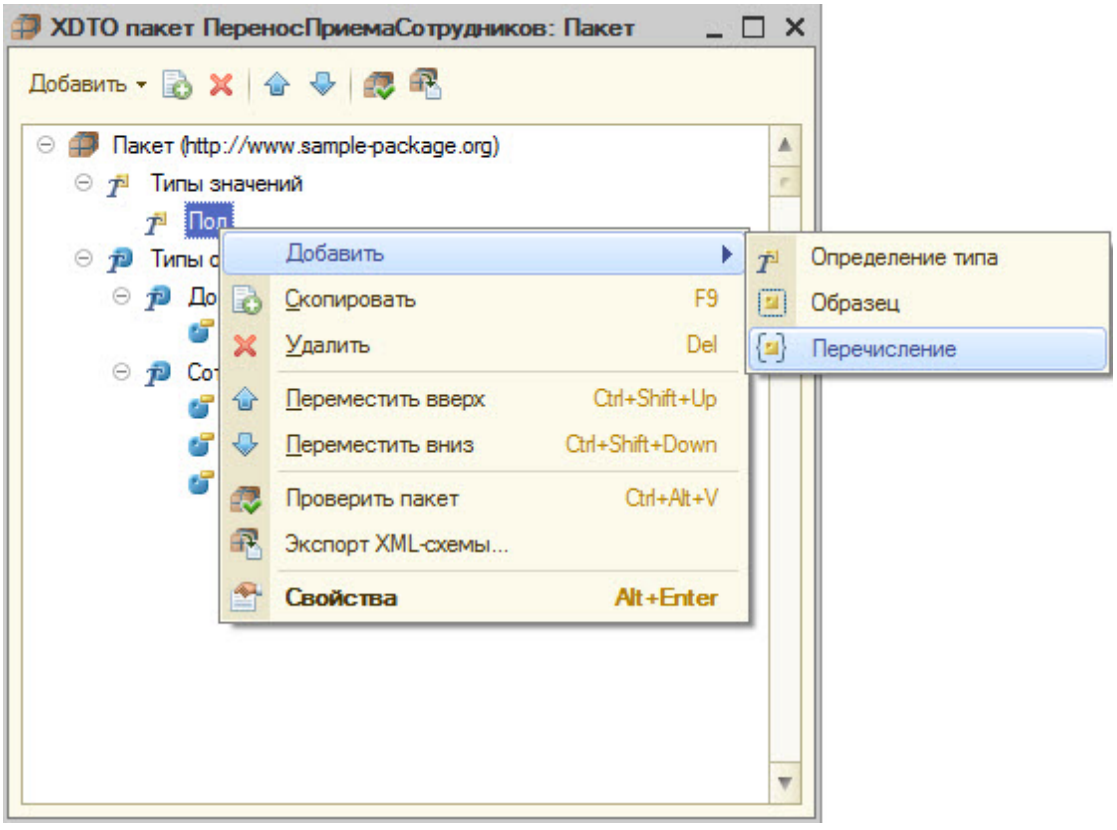
Значение *Перечисление* тоже можно перенести через свойство типа *string*. Но методологически более

грамотно создать для него *Тип значения XDTO* типа *Перечисление*. Перед записью Объекта XDTO в файл всегда производится проверка заполнения Объекта, и если в каком-то из его свойств тип значения не соответствует заявленному, выдается ошибка. Тип значения *Перечисление* гарантирует, что Объект будет записан в файл только в том случае, если в данном свойстве указано одно из заявленных значений перечисления.

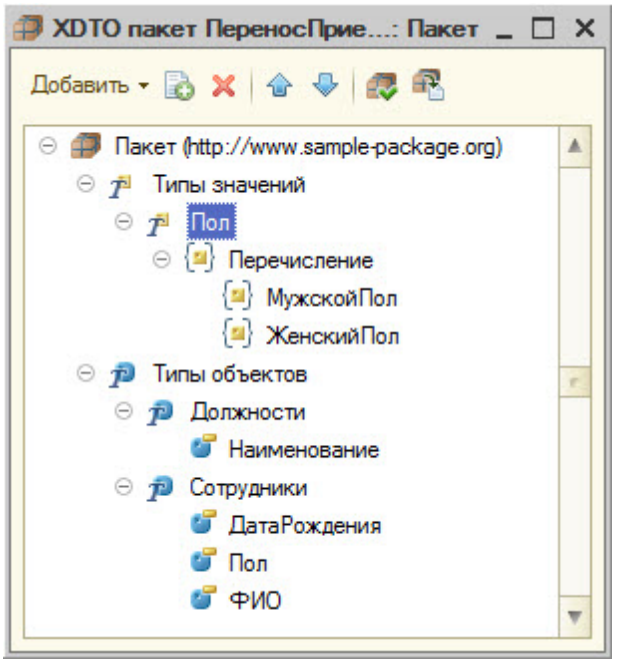
Кликнем мышкой на корневом элементе XDTO-Пакета и добавим новый тип значения *Пол*. Значения перечисления будут представлять собой текстовые строки, поэтому в панели свойств укажем ему *Базовый тип – string*, иначе будут возникать ошибки.



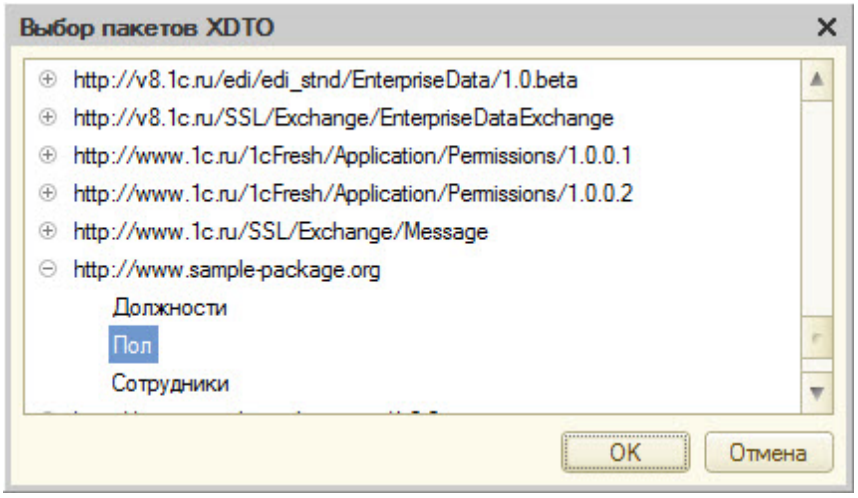
Кликнем по нему, нажмем *Добавить* и выберем пункт *Перечисление*.



В результате появится новый элемент перечисления. Назовем его *МужскойПол*. Еще раз кликнем по типу значения *Пол*, нажмем *Добавить – Перечисление*, и новый появившийся элемент назовем *ЖенскийПол*.



Теперь у свойства *Пол* можно указать созданный тип значения. Для этого в списке XDTO-пакетов нужно найти пакет «http://www.sample-package.org».



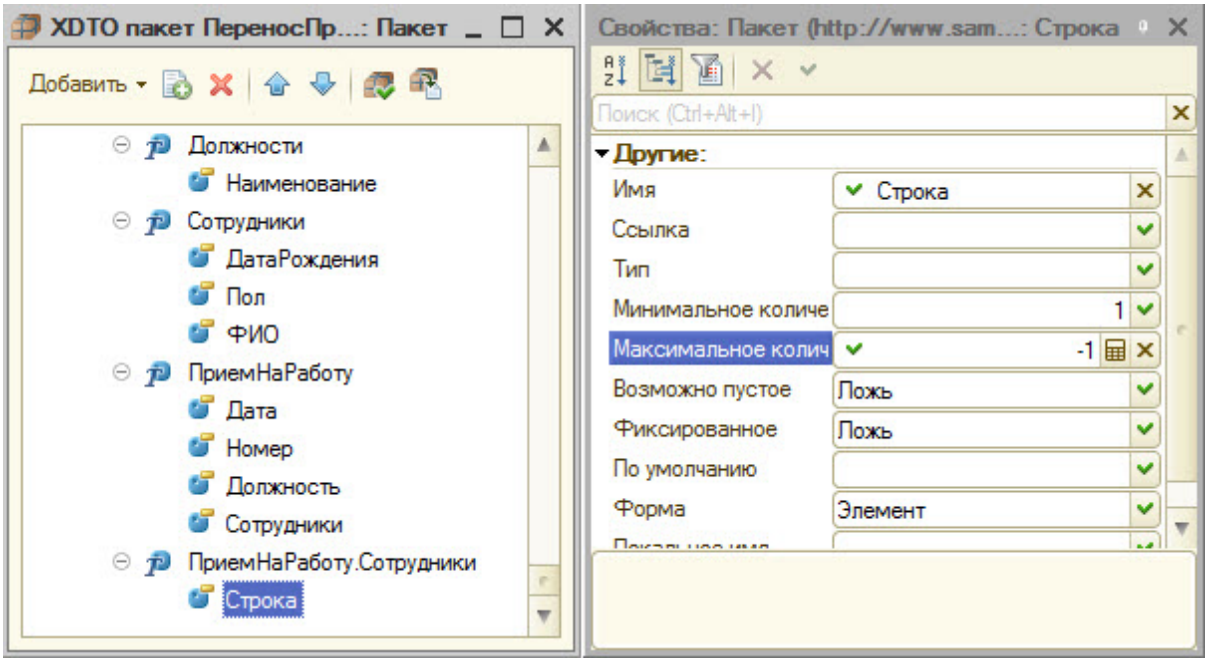
Создадим еще один *Тип* объекта *ПриемНаРаботу* для переноса документов. По аналогии с объектами информационной базы у этих объектов будут свойства *Дата*, *Номер* и *Должность*, а также должна быть табличная часть *Сотрудники*.

Первым делом создадим свойства *Дата* (примитивный тип dateTime), *Номер* (примитивный тип *string*) и *Должность*. Для свойства *Должность* тоже укажем тип – найдем в списке пакет «http://www.sample-package.org» и выберем тип *Должности*. Здесь для свойства указываем не тип значения, а тип объекта. В этом случае в свойстве Объекта XDTO *ПриемНаРаботу* должно будет оказаться не отдельное значение, а целый Объект XDTO *Должности*, заполненный по всем правилам.

Теперь нужно организовать перенос табличной части. Для этого создадим свойство *Сотрудники*. Само это свойство должно быть в объекте одно, но оно должно содержать множество строк табличной части. Поэтому создадим новый *Тип* объекта, который будет представлять из себя **Список XDTO**.

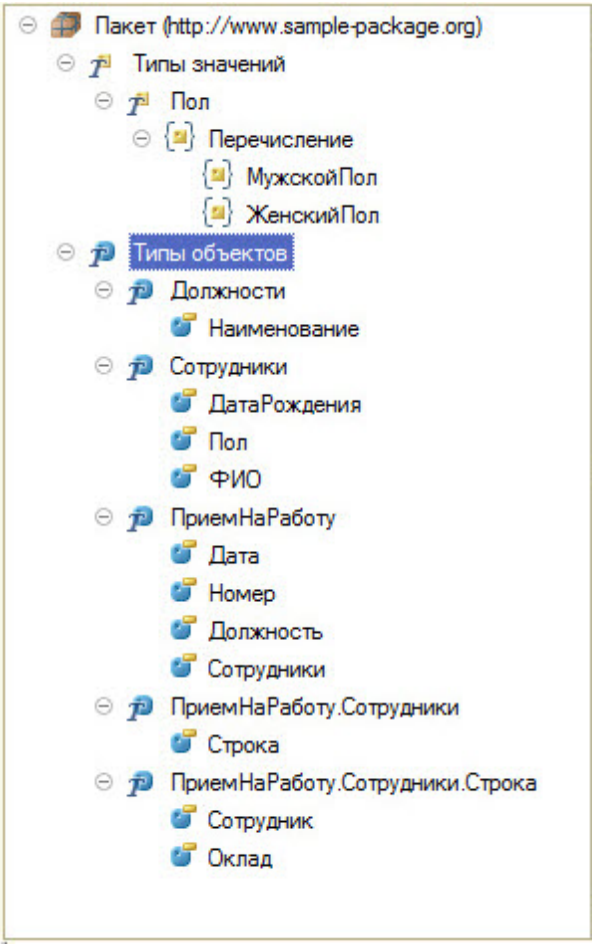
Поскольку это будет табличная часть, принадлежащая только этому документу, то назовем этот тип объекта *ПриемНаРаботу.Сотрудники*. Это должна быть коллекция однотипных элементов – строк табличной части. Поэтому здесь будет только одно свойство – *Строка*.

Откроем панель ее свойств и обратим внимание на поля *Минимальное количество* и *Максимальное количество*. Как было уже рассказано ранее, они отвечают за то, сколько раз может встречаться это свойство в каждом объекте. Чтобы в объекте *ПриемНаРаботу.Сотрудники* могло быть неограниченное количество строк табличной части, в поле *Максимальное количество* нужно установить значение « -1». Теперь каждый объект типа *ПриемНаРаботу.Сотрудники* будет представлять из себя *СписокXDTO*, и методы работы с ним будут несколько отличаться.



Наконец, нужно создать тип объекта для самой строки табличной части. Он будет называться *ПриемНаРаботу.Сотрудники.Строка* и содержать 2 свойства – *Сотрудник* (тип *Сотрудники*) и *Оклад* (примитивный тип *decimal*).

Укажем для свойства *Строка* тип *ПриемНаРаботу.Сотрудники.Строка*, а для свойства *Сотрудники* – *ПриемНаРаботу.Сотрудники. XDTO*. Пакет готов, можно приступать к разработке выгрузки.

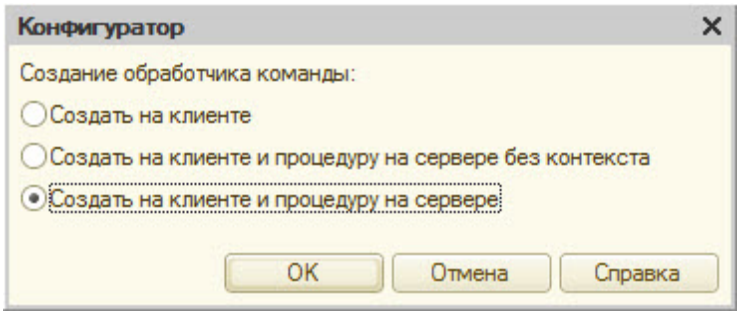


Для выгрузки создадим в конфигурации новую обработку, назовем ее *ВыгрузкаПриемаСотрудников*. Создадим форму этой обработки. В ней должно быть можно указать путь к файлу, в который будут выгружаться данные. Поэтому создадим реквизит формы *ИмяФайлаОбмена* типа *Строка* и разместим его на форме. В этом поле нужно разрешить *Кнопку выбора*, создать событие *Начало выбора* и описать его следующим образом:

```
&НаКлиенте
```

```
Процедура ИмяФайлаОбменаНачалоВыбора(Элемент, ДанныеВыбора, СтандартнаяОбработка)
      Диалог = Новый ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Сохранение);
      Диалог.Заголовок = "Файл данных: ";
      Диалог.Фильтр = "Файл xml (*.xml)|*.xml|";
      Если Диалог.Выбрать() Тогда
         ИмяФайлаОбмена = Диалог.ПолноеИмяФайла;
      КонецЕсли;
 КонецПроцедуры
```

Создадим команду формы *ВыгрузитьДокументы*, разместим ее на форме создадим обработчик команды на клиенте и процедуру на сервере.



В созданной процедуре *ВыгрузитьДокументыНаСервере()* сначала проверяем, указан ли файл с данными, и, если указан, открыть этот файл:

```
Если ИмяФайлаОбмена = "" Тогда
      Возврат;
КонецЕсли;

ФайлОбмена = Новый ЗаписьXML;
ФайлОбмена.ОткрытьФайл(ИмяФайлаОбмена);
ФайлОбмена.ЗаписатьОбъявлениеXML(); // обязательно должно быть в XML-файле
ФайлОбмена.ЗаписатьНачалоЭлемента("Message"); // корневой элемент один,
записываем
//открывающий тег
```

Далее нужно выбрать документы из базы данных и выгрузить каждый из них. Напишем цикл по выборке документов, и после него – запись закрывающего тега корневого узла *Message* и закрытие файла:

```
Выборка = Документы.ПриемНаРаботу.Выбрать();
Пока Выборка.Следующий() Цикл

КонецЦикла;
ФайлОбмена.ЗаписатьКонецЭлемента(); // Message

ФайлОбмена.Закрыть();
```

Теперь опишем тело цикла. Внутри него нужно организовать создание Объекта XDTO, его заполнение и запись. Создадим для создания и заполнения Объекта XDTO новую функцию *ВыгрузитьДокумент()*;

```
&НаСервере
Функция ВыгрузитьДокумент(ТекДокумент)

КонецФункции
```

Для создания Объекта XDTO нужно выполнить 2 действия.

- Получить нужный тип объекта из нужного XDTO-Пакета
- Создать новый Объект XDTO этого типа.

Для обоих этих действий в языке 1С используется **Фабрика XDTO**.

Фабрика XDTO – это специальный инструмент для работы с Объектами XDTO. Она необходима для

создания этих объектов, записи их в файл XML и чтения из этого файла.

Существует свойство глобального контекста *ФабрикаXDTO*, это фабрика, которая может работать со всеми пакетами XDTO, которые находятся в дереве конфигурации в ветке *XDTO-Пакеты*, а также со встроенными предопределенными пакетами (они были видны в списке при выборе типов для свойств). Пакет *ПереносПриемаСотрудников* встроен в конфигурацию, поэтому глобальная Фабрика XDTO может с ним работать.

Можно использовать, например, XDTO-Пакеты, выгруженные в файл, для этого нужно создавать отдельную фабрику, указав путь к этому файлу.

Создадим новый Объект XDTO типа *ПриемНаРаботу* из пакета http://www.sample-package.org.

```
ТипПриемНаРаботу = ФабрикаXDTO.Тип("http://www.sample-package.org",
"ПриемНаРаботу");
ОбъектПриемНаРаботу = ФабрикаXDTO.Создать(ТипПриемНаРаботу);
```

Теперь нужно корректно заполнить новый объект в соответствии с его структурой в XDTO-Пакете. Для этого потребуются Объекты XDTO следующих типов:

- Должности
- Сотрудники
- ПриемНаРаботу.Сотрудники
- ПриемНаРаботу.Сотрудники.Строка.

А также Значение XDTO типа *Пол*.

Создадим эти типы заранее, а объекты пока создавать не будем.

```
ТипДолжность = ФабрикаXDTO.Тип("http://www.sample-package.org", "Должности");
ТипСотрудник = ФабрикаXDTO.Тип("http://www.sample-package.org", "Сотрудники");
ТипТЧСотрудники = ФабрикаXDTO.Тип("http://www.sample-package.org",
"ПриемНаРаботу.Сотрудники");
ТипТЧСотрудникиСтрока = ФабрикаXDTO.Тип("http://www.sample-package.org",
"ПриемНаРаботу.Сотрудники.Строка");
ТипЗначенияПол = ФабрикаXDTO.Тип("http://www.sample-package.org", "Пол");
```

Теперь создадим и заполним Объект XDTO типа *Должности*.

```
ОбъектДолжность = ФабрикаXDTO.Создать(ТипДолжность);
ОбъектДолжность.Наименование =  ТекДокумент.ДолжностьСотрудника.Наименование;
```

В таком виде этот объект можно поместить в соответствующее свойство объекта ПриемНаРаботу.

```
ОбъектПриемНаРаботу.Должность = ОбъектДолжность;
```

Теперь нужно создать и заполнить объект для переноса табличной части. Организуем цикл по строкам табличной части документа:

```
ОбъектТЧСотрудники = ФабрикаXDTO.Создать(ТипТЧСотрудники);
Для Каждого СтрокаТЧ Из ТекДокумент.Сотрудники Цикл

КонецЦикла;
```

Внутри цикла нужно сначала создать и заполнить объект для элемента справочника *Сотрудники*.

```
ОбъектСотрудник = ФабрикаXDTO.Создать(ТипСотрудник);
ОбъектСотрудник.ФИО = СтрокаТЧ.Сотрудник.Наименование;
ОбъектСотрудник.ДатаРождения = СтрокаТЧ.Сотрудник.ДатаРождения;
```

Фабрика XDTO может также создавать Значения XDTO. Примитивные типы преобразуются в Значения XDTO автоматически средствами платформы, поэтому до сих пор мы использовали простое присваивание, например, строкового значения в свойство типа *string*. Значения перечисления, для

которого указан тип *string*, тоже могут преобразовываться автоматически.

```
ОбъектСотрудник.Пол =  "МужскойПол";
```

Но лучше сделать явное преобразование с помощью Фабрики XDTO.

```
Если СтрокаТЧ.Сотрудник.Пол = Перечисления.Пол.ПолМужской Тогда
      ОбъектСотрудник.Пол = ФабрикаXDTO.Создать(ТипЗначенияПол, "МужскойПол");
ИначеЕсли СтрокаТЧ.Сотрудник.Пол = Перечисления.Пол.ПолЖенский Тогда
      ОбъектСотрудник.Пол = ФабрикаXDTO.Создать(ТипЗначенияПол, "ЖенскийПол");
КонецЕсли;
```

Теперь создадим объект для строки и присвоим ему объект *Сотрудник*.

```
ОбъектСтрокаТЧСотрудники = ФабрикаXDTO.Создать(ТипТЧСотрудникиСтрока);
ОбъектСтрокаТЧСотрудники.Сотрудник = ОбъектСотрудник;
ОбъектСтрокаТЧСотрудники.Оклад = СтрокаТЧ.Оклад;
```

Обратим внимание, что создание объектов для сотрудника и строки табличной части обязательно должно быть внутри цикла. Если строку кода

```
ОбъектСтрокаТЧСотрудники = ФабрикаXDTO.Создать(ТипТЧСотрудникиСтрока);
```

поставить перед циклом, то в результирующем файле в каждом документе все строки табличной части будут одинаковые – соответствующие последней выгружаемой строке исходного документа.

Если же перед циклом поместить строку

```
ОбъектСотрудник = ФабрикаXDTO.Создать(ТипСотрудник);
```

то свойство *Сотрудник* будет заполнено только в последней строке, а в остальных *Сотрудник* не будет вообще. Эта особенность Объектов XDTO не документирована, но ее нужно иметь в виду.

Поскольку для объекта *ПриемНаРаботу.Сотрудники.Строка* указано максимальное количество, отличное от единицы, то объекты этого типа представляют собой список. В этом случае присваивание работать не будет

```
ОбъектТЧСотрудники.Строка  = ОбъектСтрокаТЧСотрудники; // эта строка вызовет
ошибку
```

Добавление объектов в список нужно производить с помощью метода *Добавить.*

```
ОбъектТЧСотрудники.Строка.Добавить(ОбъектСтрокаТЧСотрудники);
```

Дозаполним документ и вернем получившийся Объект XDTO.

```
ОбъектПриемНаРаботу.Сотрудники = ОбъектТЧСотрудники;
ОбъектПриемНаРаботу.Дата = ТекДокумент.Дата;
ОбъектПриемНаРаботу.Номер = ТекДокумент.Номер;

Возврат ОбъектПриемНаРаботу;
```

В процедуре *ВыгрузитьДокументыНаСервере()* вставим внутрь цикла вызов функции *ВыгрузитьДокумент()* и запись полученного объекта.

```
Пока Выборка.Следующий() Цикл

      ОбъектXDTO = ВыгрузитьДокумент(Выборка);
      ФабрикаXDTO.ЗаписатьXML(ФайлОбмена, ОбъектXDTO);

КонецЦикла;
```

Если теперь произвести выгрузку и открыть файл данных, то все документы будут выгружены корректно, но в каждом узле документа будут указаны пространства имен всех пакетов, которые так или иначе использовались для формирования этого узла

Если записать эти пространства имен в начале файла, то в каждом узле они уже фигурировать не будут, это сократит размер файла. Напишем после открытия тега *Message* следующие строки:

```
ФайлОбмена.ЗаписатьСоответствиеПространстваИмен("", "http://www.sample-
package.org");
ФайлОбмена.ЗаписатьСоответствиеПространстваИмен("xs",
"http://www.w3.org/2001/XMLSchema");
ФайлОбмена.ЗаписатьСоответствиеПространстваИмен("xsi",
"http://www.w3.org/2001/XMLSchema-instance");
```

Тогда результирующий файл будет выглядеть следующим образом.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<Message xmlns="http://www.sample-package.org"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  ▼<ПриемНаРаботу>
     <Дата>2014-07-11T12:00:00</Дата>
     <Номер>000000001</Номер>
    ▼<Должность>
       <Наименование>Менеджер по продажам</Наименование>
     </Должность>
    ▼<Сотрудники>
      ▼<Строка>
        ▼<Сотрудник>
           <ДатаРождения>1965-03-15</ДатаРождения>
           <Пол>МужскойПол</Пол>
           <ФИО>Акимов О.П.</ФИО>
         </Сотрудник>
         <Оклад>30000</Оклад>
       </Строка>
      ▼<Строка>
        ▼<Сотрудник>
           <ДатаРождения>1981-09-10</ДатаРождения>
           <Пол>МужскойПол</Пол>
           <ФИО>Ефимов И.И.</ФИО>
         </Сотрудник>
         <Оклад>31000</Оклад>
       </Строка>
     </Сотрудники>
   </ПриемНаРаботу>
  ▼<ПриемНаРаботу>
     <Дата>2015-07-28T20:46:46</Дата>
     <Номер>000000002</Номер>
    ▼<Должность>
       <Наименование>Бухгалтер</Наименование>
     </Должность>
    ▼<Сотрудники>
      ▼<Строка>
```

Итоговый текст модуля обработки выгрузки:

```
&НаСервере
Функция ВыгрузитьДокумент(ТекДокумент)
	ТипПриемНаРаботу = ФабрикаXDTO.Тип("http://www.sample-package.org",
"ПриемНаРаботу");
	ОбъектПриемНаРаботу = ФабрикаXDTO.Создать(ТипПриемНаРаботу);

	ТипДолжность = ФабрикаXDTO.Тип("http://www.sample-package.org",
"Должности");
	ТипСотрудник = ФабрикаXDTO.Тип("http://www.sample-package.org",
"Сотрудники");
	ТипТЧСотрудники = ФабрикаXDTO.Тип("http://www.sample-package.org",
"ПриемНаРаботу.Сотрудники");
	ТипТЧСотрудникиСтрока = ФабрикаXDTO.Тип("http://www.sample-package.org",
"ПриемНаРаботу.Сотрудники.Строка");
	ТипЗначенияПол = ФабрикаXDTO.Тип("http://www.sample-package.org", "Пол");

	ОбъектДолжность = ФабрикаXDTO.Создать(ТипДолжность);
	ОбъектДолжность.Наименование =
ТекДокумент.ДолжностьСотрудника.Наименование;
	ОбъектПриемНаРаботу.Должность = ОбъектДолжность;

	ОбъектТЧСотрудники = ФабрикаXDTO.Создать(ТипТЧСотрудники);
	Для Каждого СтрокаТЧ Из ТекДокумент.Сотрудники Цикл
		ОбъектСотрудник = ФабрикаXDTO.Создать(ТипСотрудник);
		ОбъектСотрудник.ФИО = СтрокаТЧ.Сотрудник.Наименование;
		ОбъектСотрудник.ДатаРождения = СтрокаТЧ.Сотрудник.ДатаРождения;
		Если СтрокаТЧ.Сотрудник.Пол = Перечисления.Пол.ПолМужской Тогда
			ОбъектСотрудник.Пол = ФабрикаXDTO.Создать(ТипЗначенияПол,
```

```
"МужскойПол");
            ИначеЕсли СтрокаТЧ.Сотрудник.Пол = Перечисления.Пол.ПолЖенский Тогда
                  ОбъектСотрудник.Пол = ФабрикаXDTO.Создать(ТипЗначенияПол,
"ЖенскийПол");
            КонецЕсли;

            ОбъектСтрокаТЧСотрудники = ФабрикаXDTO.Создать(ТипТЧСотрудникиСтрока);
            ОбъектСтрокаТЧСотрудники.Сотрудник = ОбъектСотрудник;
            ОбъектСтрокаТЧСотрудники.Оклад = СтрокаТЧ.Оклад;
            ОбъектТЧСотрудники.Строка.Добавить(ОбъектСтрокаТЧСотрудники);
      КонецЦикла;

      ОбъектПриемНаРаботу.Сотрудники = ОбъектТЧСотрудники;
      ОбъектПриемНаРаботу.Дата = ТекДокумент.Дата;
      ОбъектПриемНаРаботу.Номер = ТекДокумент.Номер;

      Возврат ОбъектПриемНаРаботу;

КонецФункции

&НаСервере
Процедура ВыгрузитьДокументыНаСервере()

      Если ИмяФайлаОбмена = "" Тогда
            Возврат
      КонецЕсли;

      ФайлОбмена = Новый ЗаписьXML;
      ФайлОбмена.ОткрытьФайл(ИмяФайлаОбмена);
      ФайлОбмена.ЗаписатьОбъявлениеXML();
      ФайлОбмена.ЗаписатьНачалоЭлемента("Message");

      ФайлОбмена.ЗаписатьСоответствиеПространстваИмен("", "http://www.sample-
package.org");
      ФайлОбмена.ЗаписатьСоответствиеПространстваИмен("xs",
"http://www.w3.org/2001/XMLSchema");
      ФайлОбмена.ЗаписатьСоответствиеПространстваИмен("xsi",
"http://www.w3.org/2001/XMLSchema-instance");

      Выборка = Документы.ПриемНаРаботу.Выбрать();
      Пока Выборка.Следующий() Цикл

            ОбъектXDTO = ВыгрузитьДокумент(Выборка);

            ФабрикаXDTO.ЗаписатьXML(ФайлОбмена, ОбъектXDTO);
      КонецЦикла;
      ФайлОбмена.ЗаписатьКонецЭлемента();

КонецПроцедуры

&НаКлиенте
Процедура ВыгрузитьДокументы(Команда)

      ВыгрузитьДокументыНаСервере();

КонецПроцедуры

&НаКлиенте
Процедура ИмяФайлаОбменаНачалоВыбора(Элемент, ДанныеВыбора, СтандартнаяОбработка)
      Диалог = Новый ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Сохранение);
      Диалог.Заголовок = "Файл данных: ";
      Диалог.Фильтр = "Файл xml (*.xml)|*.xml|";
      Если Диалог.Выбрать() Тогда
      ИмяФайлаОбмена = Диалог.ПолноеИмяФайла;
      КонецЕсли;
 КонецПроцедуры
```

Включим обработку «ВыгрузкаПриемаСотрудников» в подсистему «Синхронизация данных», и она отобразится в командном интерфейсе.

Теперь займемся загрузкой. В конфигурацию *Конечная* скопируем через буфер обмена XDTO-Пакет *ПереносПриемаСотрудников*.

Создадим новую обработку *ЗагрузкаПриемаСотрудников*, создадим реквизит *ИмяФайлаОбмена типа Строка*, разместим его на форме, добавим *кнопку выбора* и определим его событие *Начало выбора* следующим образом:

```
&НаКлиенте
Процедура ИмяФайлаОбменаНачалоВыбора(Элемент, ДанныеВыбора, СтандартнаяОбработка)
      Диалог = Новый ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Открытие);
      Диалог.Заголовок = "Файл данных: ";
      Диалог.Фильтр = "Файл xml (*.xml)|*.xml|";
      Если Диалог.Выбрать() Тогда
      ИмяФайлаОбмена = Диалог.ПолноеИмяФайла;
      КонецЕсли;
КонецПроцедуры
```

Создадим команду формы *ЗагрузитьДокументы* и ее обработчик на клиенте с вызовом процедуры на сервере.

В процедуре *ЗагрузитьДокументыНаСервере()* сначала откроем файл данных:

```
ФайлОбмена = Новый ЧтениеXML;
ФайлОбмена.ОткрытьФайл(ИмяФайлаОбмена);
```

Первый тег, который встретится в этом файле – это объявление XML. Можно пропустить его с помощью стандартного чтения.

```
ФайлОбмена.Прочитать();
```

Но лучше использовать специальный метод:

```
ФайлОбмена.ПерейтиКСодержимому();
```

Следующим идет корневой тег, его тоже нужно прочитать «вручную».

А далее уже можно выполнять чтение с помощью ФабрикиXDTO. Чтобы двигаться по файлу XML, организуют цикл:

```
Пока ФайлОбмена.ТипУзла = ТипУзлаXML.НачалоЭлемента Цикл
    ОбъектXDTO = ФабрикаXDTO.ПрочитатьXML(ФайлОбмена, ТипXDTOЧитаемогоОбъекта);
     ...

КонецЦикла;
```

Чтение объекта можно выполнить по-разному. Можно сначала прочитать объект, а потом определить, какой тип он имеет.

```
ОбъектXDTO = ФабрикаXDTO.ПрочитатьXML(ФайлОбмена, ФабрикаXDTO.Тип
(ФайлОбмена.URIПространстваИмен, ФайлОбмена.ЛокальноеИмя));
Если ОбъектXDTO.Тип().Имя = "ПриемНаРаботу" Тогда
…
КонецЕсли;
```

А можно указать тип объекта явно при чтении объекта. Если читаемый объект будет иметь не тот тип, который указан, возникнет ошибка.

```
ОбъектXDTO = ФабрикаXDTO.ПрочитатьXML(ФайлОбмена,                ФабрикаXDTO.Тип
("http://www.sample-package.org", "ПриемНаРаботу"));
```

Попробуем найти загружаемый документ в базе Конечная по *Номеру* и *Дате*. Если этот документ уже есть в базе, он будет заполнен новыми данными и перезаписан, а не задублируется.

```
ДатаДок = ОбъектXDTO.Дата;
НомерДок = ОбъектXDTO.Номер;
```

```
НайденныйДокумент = Документы.ПриемНаРаботу.НайтиПоНомеру(НомерДок,ДатаДок);
Если ЗначениеЗаполнено(НайденныйДокумент) Тогда
	ДокументОбъект = НайденныйДокумент.ПолучитьОбъект();
	ДокументОбъект.Сотрудники.Очистить();
Иначе
	ДокументОбъект = Документы.ПриемНаРаботу.СоздатьДокумент();
КонецЕсли;
ДокументОбъект.Дата = ДатаДок;
ДокументОбъект.Номер = НомерДок;
```

Должность попробуем найти по *Наименованию*, если не найдем, то создадим:

```
ДолжностьСсылка =
Справочники.ДолжностьСотрудника.НайтиПоНаименованию(ОбъектXDTO.Должность.Наименование);
Если НЕ ЗначениеЗаполнено(ДолжностьСсылка) Тогда
	ДолжностьОбъект = Справочники.ДолжностьСотрудника.СоздатьЭлемент();
	ДолжностьОбъект.Наименование = ОбъектXDTO.Должность.Наименование;
	ДолжностьОбъект.Записать();
	ДолжностьСсылка = ДолжностьОбъект.Ссылка;
КонецЕсли;

ДокументОбъект.ДолжностьСотрудника = ДолжностьСсылка;
```

Далее организуем цикл по списку XDTO *ОбъектXDTO.Сотрудники.Строка*, и загрузим каждый из элементов, как строку табличной части. Сотрудника будем искать по *Наименованию*.

Итоговый текст модуля обработки загрузки:

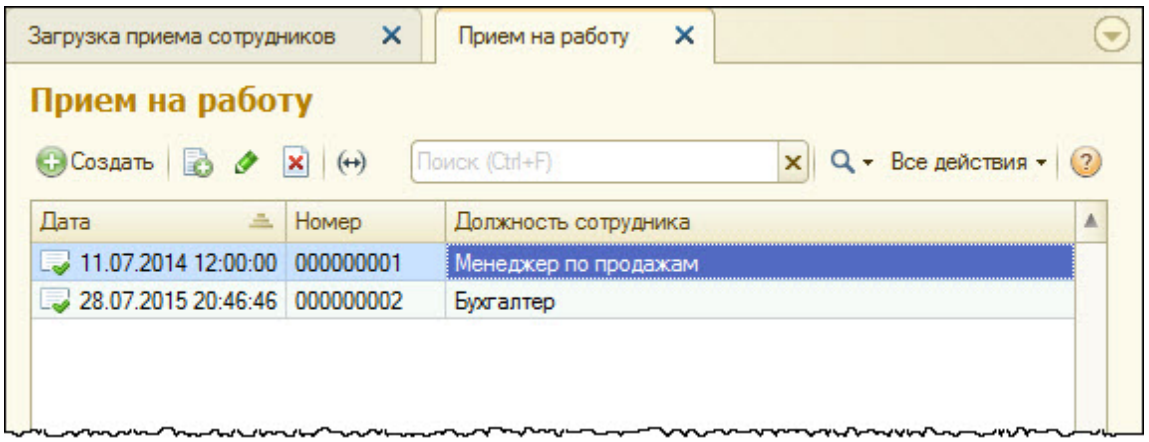
```
&НаСервере
Процедура ЗагрузитьДокументыНаСервере()
	ФайлОбмена = Новый ЧтениеXML;
	ФайлОбмена.ОткрытьФайл(ИмяФайлаОбмена);

	ФайлОбмена.ПерейтиКСодержимому();
	ФайлОбмена.Прочитать(); // тэг message

	Пока ФайлОбмена.ТипУзла = ТипУзлаXML.НачалоЭлемента Цикл
		ОбъектXDTO = ФабрикаXDTO.ПрочитатьXML(
			ФайлОбмена,
			ФабрикаXDTO.Тип("http://www.sample-package.org", "ПриемНаРаботу"));

		ДатаДок = ОбъектXDTO.Дата;
		НомерДок = ОбъектXDTO.Номер;
		НайденныйДокумент =
Документы.ПриемНаРаботу.НайтиПоНомеру(НомерДок,ДатаДок);
		Если ЗначениеЗаполнено(НайденныйДокумент) Тогда
			ДокументОбъект = НайденныйДокумент.ПолучитьОбъект();
			ДокументОбъект.Сотрудники.Очистить();
		Иначе
			ДокументОбъект = Документы.ПриемНаРаботу.СоздатьДокумент();
		КонецЕсли;

		ДокументОбъект.Дата = ДатаДок;
		ДокументОбъект.Номер = НомерДок;

		ДолжностьСсылка =
Справочники.ДолжностьСотрудника.НайтиПоНаименованию(ОбъектXDTO.Должность.Наименование);
		Если НЕ ЗначениеЗаполнено(ДолжностьСсылка) Тогда
			ДолжностьОбъект = Справочники.ДолжностьСотрудника.СоздатьЭлемент();
			ДолжностьОбъект.Наименование = ОбъектXDTO.Должность.Наименование;
			ДолжностьОбъект.Записать();
			ДолжностьСсылка = ДолжностьОбъект.Ссылка;
		КонецЕсли;

		ДокументОбъект.ДолжностьСотрудника = ДолжностьСсылка;

		Для Каждого СтрокаСотрудникиXDTO Из ОбъектXDTO.Сотрудники.Строка Цикл
```

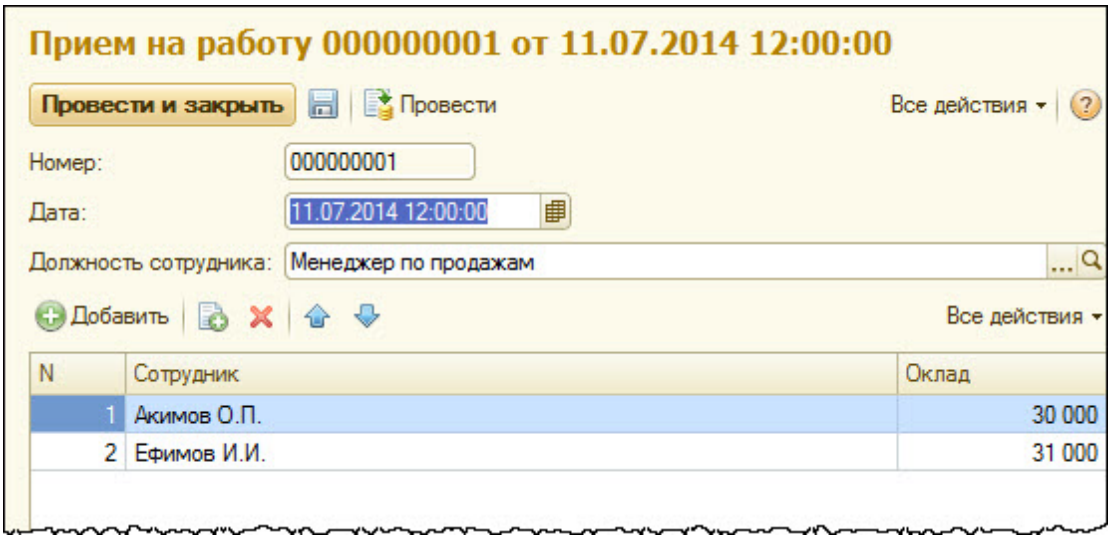
```
                    СтрокаТЧСотрудники = ДокументОбъект.Сотрудники.Добавить();
                    СотрудникСсылка =
Справочники.Сотрудники.НайтиПоНаименованию(СтрокаСотрудникиXDTO.Сотрудник.ФИО);
                    Если НЕ ЗначениеЗаполнено(СотрудникСсылка) Тогда
                          СотрудникОбъект =  Справочники.Сотрудники.СоздатьЭлемент();
                          СотрудникОбъект.Наименование =
СтрокаСотрудникиXDTO.Сотрудник.ФИО;
                          СотрудникОбъект.ДатаРождения =
СтрокаСотрудникиXDTO.Сотрудник.ДатаРождения;
                          СотрудникОбъект.ДатаРождения =
СтрокаСотрудникиXDTO.Сотрудник.ДатаРождения;

                          Если СтрокаСотрудникиXDTO.Сотрудник.Пол = "МужскойПол" Тогда
                                СотрудникОбъект.Пол = Перечисления.Пол.ПолМужской;
                          ИначеЕсли СтрокаСотрудникиXDTO.Сотрудник.Пол = "ЖенскийПол"
Тогда
                                СотрудникОбъект.Пол = Перечисления.Пол.ПолЖенский;
                          КонецЕсли;
                          СотрудникОбъект.Записать();
                          СотрудникСсылка = СотрудникОбъект.Ссылка;
                    КонецЕсли;
                    СтрокаТЧСотрудники.Сотрудник = СотрудникСсылка;
                    СтрокаТЧСотрудники.Оклад = СтрокаСотрудникиXDTO.Оклад;
              КонецЦикла;
              ДокументОбъект.Записать(РежимЗаписиДокумента.Проведение);
        КонецЦикла;
КонецПроцедуры

&НаКлиенте
Процедура ЗагрузитьДокументы(Команда)
        ЗагрузитьДокументыНаСервере();
КонецПроцедуры

&НаКлиенте
Процедура ИмяФайлаОбменаНачалоВыбора(Элемент, ДанныеВыбора, СтандартнаяОбработка)
        Диалог = Новый ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Открытие);
        Диалог.Заголовок = "Файл данных: ";
        Диалог.Фильтр = "Файл xml (*.xml)|*.xml|";
        Если Диалог.Выбрать() Тогда
            ИмяФайлаОбмена = Диалог.ПолноеИмяФайла;
        КонецЕсли;
КонецПроцедуры
```

Включим обработку «ЗагрузкаПриемаСотрудников» в подсистему «Синхронизация данных», перезапустим информационную базу и выполним загрузку.

Проверим, как загрузились данные:

Наименование	Код	Дата рождения	Пол
Акимов О.П.	000000001	15.03.1965	Пол мужской
Аничкова С.В.	000000003	06.07.1986	Пол женский
Ефимов И.И.	000000002	10.09.1981	Пол мужской





Универсальный формат представляет из себя XDTO-Пакет, специально разработанный фирмой 1С для обменов между базами на основе типовых конфигураций. Он содержит ряд объектов, аналогичных объектам типовых конфигураций, которыми, по мнению разработчиков из «1С», должны обмениваться эти базы.

Этот пакет ничем особенным не отличается от других XDTO-пакетов, кроме того, что имеет определенную структуру, благодаря чему с ним может работать механизм обмена через XDTO.

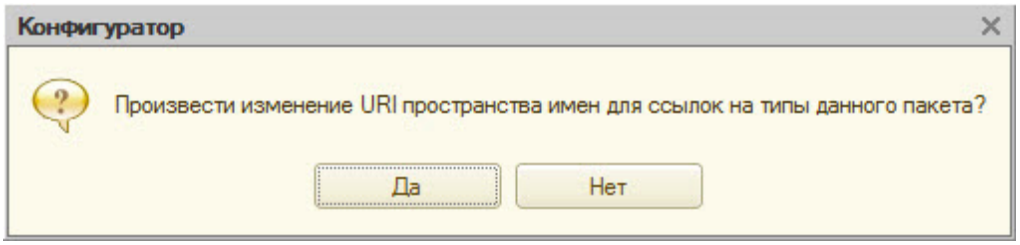
Разработчики 1С предполагают, что если необходимо будет обмениваться какой-то дополнительной информацией, кроме той, что описана в формате, разработчик может добавить эти типы объектов в формат, соблюдая его структуру. Правда, при этом будет потеряна универсальность формата, но не во всех задачах это принципиально.

В этом уроке мы изменим разработанную ранее схему для переноса документов Прием на работу так, чтобы с ней работал обмен через XDTO. Таким образом, будет разработан собственный формат обмена, на базе которого будет организован отдельный обмен данными.

Договоримся, что новый формат будет называться sample-package, и все объекты, имеющие к нему отношение, будут иметь префикс sp.

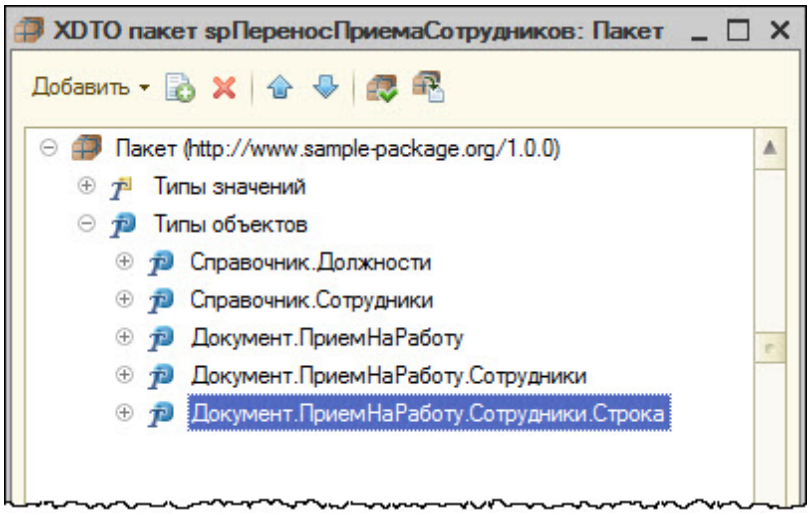
Создадим копию XDTO-Пакета *ПереносПриемаСотрудников*, назовем копию *spПереносПриемаСотрудников* и сразу изменим пространство имен. Для работы пакета в качестве формата необходимо, чтобы в его пространстве имен была указана версия. В качестве версии воспринимается набор символов после последнего слэша, версия должна состоять из нескольких цифр, разделенных точкой. Поэтому изменим пространство имен следующим образом: «http://www.sample-package.org/1.0.0».

После изменения система задает вопрос:

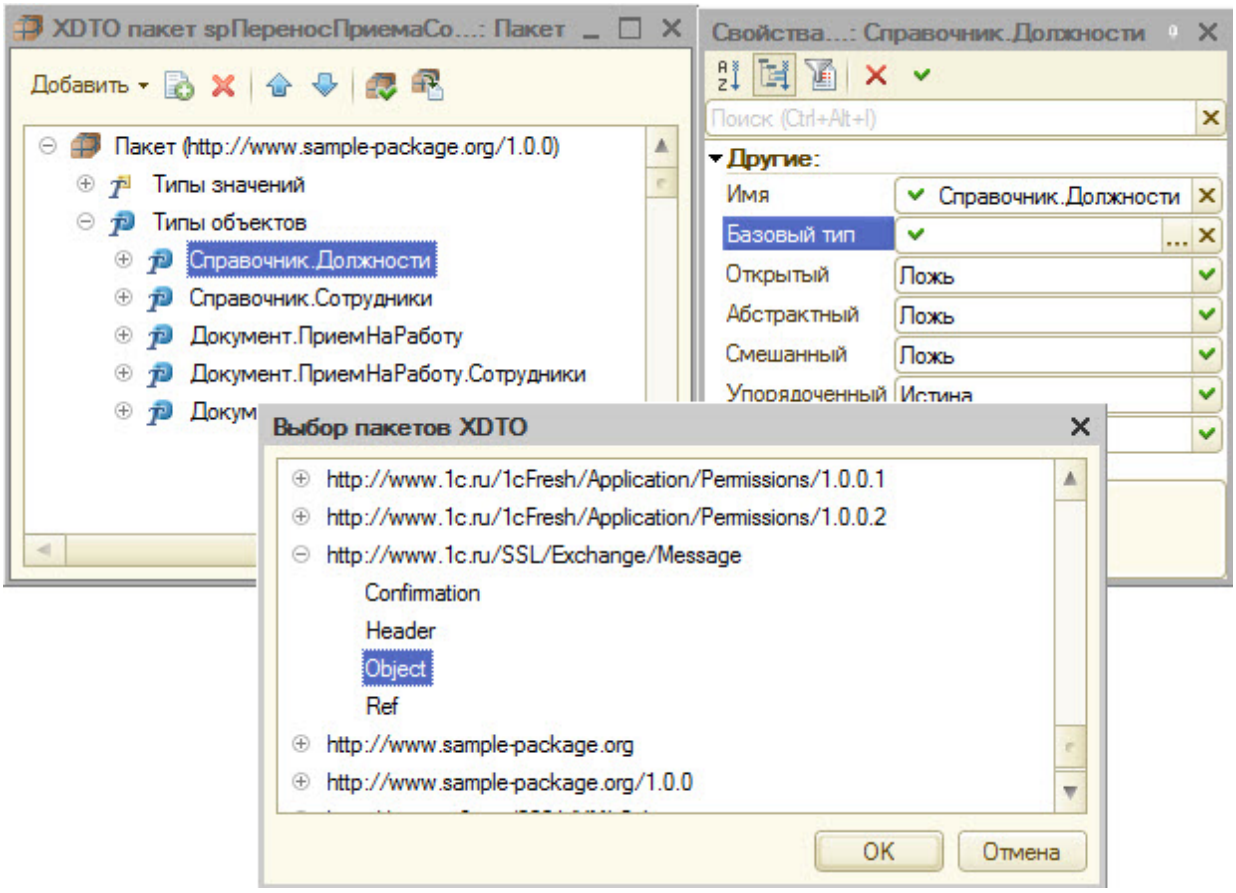


Нажмем *Нет*, поскольку в процессе работы над пакетом все равно придется менять ссылки объектов друг на друга.

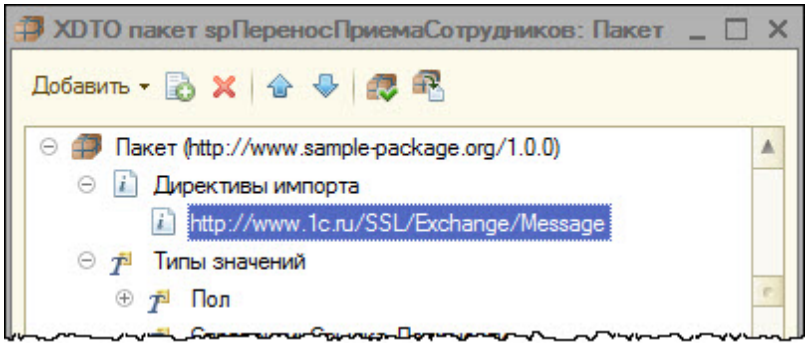
Теперь нужно изменить имена типов объектов в пакете. По стандартам они должны называться *Справочник.Должности*, *Справочник.Сотрудники*, *Документ.ПриемНаРаботу*, *Документ.ПриемНаРаботу.Сотрудники* и *Документ.ПриемНаРаботу.Сотрудники.Строка*.



А также для типов объектов *Справочник.Должности*, *Справочник.Сотрудники* и *Документ.ПриемНаРаботу* нужно указать базовый тип *Object*. Дело в том, что типы объектов могут наследоваться друг от друга. Тип-потомок будет наследовать свойства типа-родителя, а также иметь свои дополнительные свойства. Для всех типов, с помощью которых должны переноситься объекты информационной базы, должен быть указан родительский тип. Он находится в пространстве имен «http://www.1c.ru/SSL/Exchange/Message».



Чтобы обеспечить наследование типов объектов одного пространства имен от типов объектов другого, добавим в XDTO-Пакет *spПереносПриемаСотрудников Директивы импорта* и укажем пространство имен «http://www.1c.ru/SSL/Exchange/Message».



Теперь восстановим некоторые нарушенные ссылки.

- Тип объекта Сотрудники.
  - Свойство *Пол*, укажем ему другой тип. Найдем пространство имен «http://www.sample-package.org/1.0.0» и выберем из него тип значения *Пол*.
- Тип объекта *Документ.ПриемНаРаботу*.
  - Свойство *Сотрудники*, укажем тип *Документ.ПриемНаРаботу.Сотрудники* из простанства имен «http://www.sample-package.org/1.0.0».
- Тип объекта *Документ.ПриемНаРаботу.Сотрудники*
  - Свойство *Строка*, укажем тип *Документ.ПриемНаРаботу.Сотрудники.Строка* из пространства имен «http://www.sample-package.org/1.0.0».

Следующим шагом нужно создать типы значений для ссылок всех этих типов. Создадим типы значений *СправочникСсылка.Должности*, *СправочникСсылка.Сотрудники*, *ДокументСсылка.ПриемНаРаботу*, у каждого из них нужно установить базовый тип *Ref* из того же пространства имен «http://www.1c.ru/SSL/Exchange/Message».

Теперь нужно создать типы объектов *КлючевыеСвойства* для двух справочников и документа. *КлючевыеСвойства* – это узел, содержащий информацию, которая может использоваться для поиска объекта в информационной базе-приемнике при загрузке. Этот узел обязательно используется при выгрузке справочников и документов, и он же вставляется в узлы свойств, которые ссылаются на эти

справочники и документы.

Создадим типы объектов:

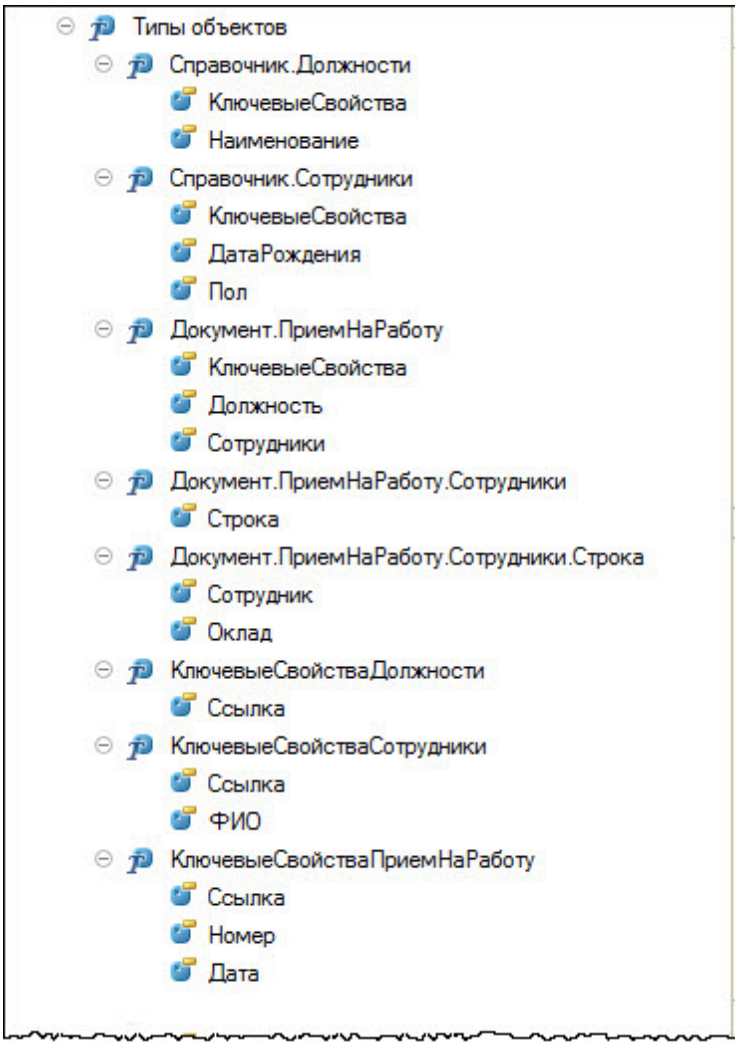
- *КлючевыеСвойстваДолжности*. Свойство:
  - *Ссылка*, тип значения *СправочникСсылка.Должности*.
- *КлючевыеСвойстваСотрудники*. Свойства:
  - *Ссылка*, тип *СправочникСсылка.Сотрудники*
  - *ФИО,* тип *string*.
- *КлючевыеСвойстваПриемНаРаботу*. Свойства:
  - *Ссылка*, тип *ДокументСсылка.ПриемНаРаботу*
  - *Номер*, тип *string*
  - *Дата*, тип *dateTime*.

Кликнем мышкой по типу объекта *Справочник.Должности*, создадим новое свойство *КлючевыеСвойства* и укажем тип – *КлючевыеСвойстваДолжности*. Свойство *КлючевыеСвойства* должно быть выше всех остальных свойств в типе объекта.

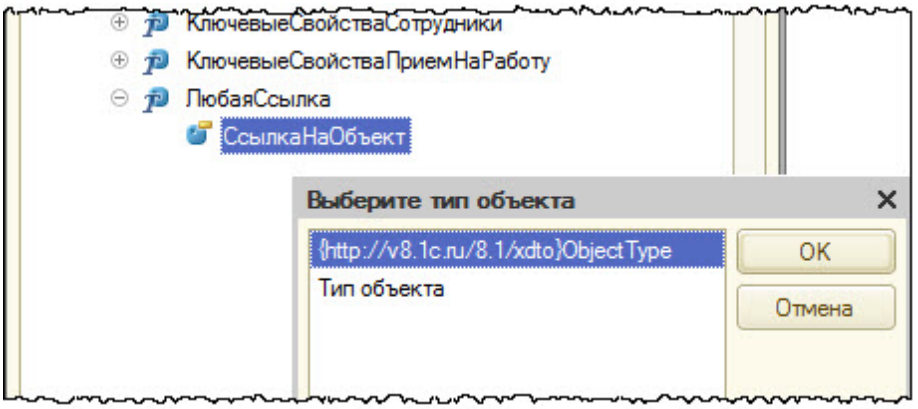
Создадим аналогичные свойства для типов *Справочник.Сотрудники* и *Документ.ПриемНаРаботу*. Свойство Сотрудников *ФИО* и свойства документа *Номер* и *Дата* теперь входят в ключевые свойства, и из основного объекта их нужно удалить.

Перенастроим еще две ссылки.

- Тип объекта *Документ.ПриемНаРаботу.*
  - Свойство *Должность* должно иметь тип *КлючевыеСвойства Должности*
- Тип объекта *Документ.ПриемНаРаботу.Сотрудники.Строка*
  - Свойство *Сотрудник* должно иметь тип *КлючевыеСвойстваСотрудники*.



Осталось добавить 2 служебных объекта для переноса удаления объекта. Добавим новый тип объекта *ЛюбаяСсылка*. Добавим свойство – *ССылкаНаОбъект*. Добавим этому свойству определение типа. При этом программа предложит выбрать, какое определение типа необходимо.



Вопреки написанному, пункт *Тип объекта* на самом деле означает, что в свойство будет добавлен Тип значения. Нужно выбрать вариант *{http://v8.1c.ru/8.1/xdto}ObjectType*, чтобы определением типа был именно тип объекта. В это определение типа нужно добавить свойства

- *ДолжностиСсылка* (тип *СправочникСсылка.Должности*, минимальное количество 0)
- *СотрудникиСсылка* (тип *СправочникСсылка.Сотрудники*, минимальное количество 0)
- *ПриемНаРаботуСсылка* (тип *ДокументСсылка.ПриемНаРаботу*, минимальное количество 0)

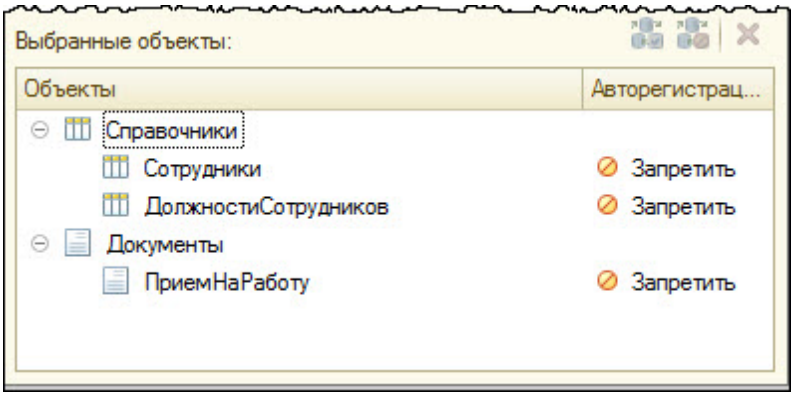
Теперь добавим тип объекта *УдалениеОбъекта*, у него одно свойство *СсылкаНаОбъект*, тип – *ЛюбаяСсылка*.

Пакет, который может выступать в качестве формата, готов.

Для обмена через этот новый формат мы не будем изменять механизм обмена через Универсальный формат. Создадим ряд новых объектов с префиксом «sp».

Создадим общий модуль *spМенеджерОбменаЧерезУниверсальныйФормат*, пока он будет пустой. Заполним его после настройки правил.

Скопируем план обмена *СинхронизацияДанныхЧерезУниверсальныйФормат* и назовем копию *spСинхронизацияДанныхЧерезУниверсальныйФормат*. Настроим его состав: на нем должны регистрироваться только два справочника и документ, авторегистрация для всех запрещена.



Откроем его модуль менеджера для дополнительной настройки.

- Функция *НастройкаОтборовНаУзле()* Указываем другую версию формата – «1.0.0».
- Функция *ЗаголовокКомандыДляСозданияНовогоОбменаДанными()* должна возвращать строку «Синхронизация с КБ через формат sample-package»;
- Процедура *ОпределитьНастройки()*. В элемент структуры *НаименованиеКонфигурацииКорреспондента* должна быть помещена строка «Через формат SP»
- Функция *ИмяФайлаНастроекДляПриемника()* должна возвращать строку «Синхронизация данных через формат sample-package»
- Функция *ФорматОбмена()* должна возвращать строку «http://www.sample-package.org»
- Процедура *ПолучитьВерсииФорматаОбмена()* должна содержать одну строку

```
ВерсииФормата.Вставить("1.0.0", spМенеджерОбменаЧерезУниверсальныйФормат);
```

Все остальные процедуры и функции оставим без изменения.

Добавим новый план обмена и общий модуль в подсистему *Синхронизация данных*, но запретим отображение в командном интерфейсе.

Далее в те общие команды, в которых параметром уже выступает план обмена *СинхронизацияДанныхЧерезУниверсальныйФормат*, нужно добавить также новый план обмена, указав, что тип данных будет составным. Это команды:

- ЗагрузитьПравилаРегистрацииОбъектов
- НастроитьПараметрыТранспортаСообщенийОбмена
- ПолучитьНастройкиСинхронизацииДляДругойПрограммы
- Синхронизировать
- СинхронизироватьСДополнительнымиПараметрами
- СобытияОтправки
- СобытияПолучения
- СоставОтправляемыхДанных
- СценарииСинхронизации
- УдалитьНастройкуСинхронизации

Также добавим новые подписки на события, которые будут регистрировать изменения на узлах нового плана обмена. Скопируем существующие подписки и изменим их наименование

- Подписка *spСинхронизацияДанныхЧерезУниверсальныйФорматЗарегистрироватьИзменение*
  - Источник: *СправочникОбъект.ДолжностиСотрудников, СправочникОбъект.Сотрудники*
  - Событие: *ПередЗаписью*
- Подписка *spСинхронизацияДанныхЧерезУниверсальныйФорматЗарегистрироватьИзменениеДокумента*
  - Источник: *ДокументОбъект.ПриемНаРаботу*
  - Событие: *ПередЗаписью*
- Подписка: *spСинхронизацияДанныхЧерезУниверсальныйФорматЗарегистрироватьУдаление*
  - Источник: *ДокументОбъект.ПриемНаРаботу, СправочникОбъект.ДолжностиСотрудников, СправочникОбъект.Сотрудники*
  - Событие: *ПередУдалением*

Обработчики событий для них нужно создать отдельные, но в том же модуле *ОбработчикиПодписокНаСобытия*. Текст обработчиков можно скопировать их уже имеющихся, но название плана обмена в них нужно изменить на новое.

Следующим шагом откроем общий модуль *ОбменДаннымиПереопределяемый*, процедуру *ПолучитьПланыОбмена()*. В ней должны быть 2 строки:

```
ПланыОбменаПодсистемы.Добавить(
Метаданные.ПланыОбмена.СинхронизацияДанныхЧерезУниверсальныйФормат);
ПланыОбменаПодсистемы.Добавить(
Метаданные.ПланыОбмена.spСинхронизацияДанныхЧерезУниверсальныйФормат);
```

Теперь можно запускать пользовательский режим, но перед этим увеличим версию конфигурации. Сделаем это для того, чтобы добавленные объекты автоматически попали в справочник *Идентификаторы объектов метаданных*. Иначе в процессе работы могут возникать ошибки. Откроем свойства конфигурации и укажем номер версии 1.0.0.1. Можно сохранить конфигурацию и временно оставить, она пока не понадобится.

Выполним аналогичную настройку в конфигурации Конечная. Скопируем в нее XDTO-Пакет *spПереносПриемаСотрудников*, план обмена *spСинхронизацияДанныхЧерезУниверсальныйФормат* и общий модуль *spМенеджерОбменаЧерезУниверсальныйФормат*.

Аналогичным образом нужно настроить параметры *Общих команд*. Подписки на события настраивать не требуется. Допишем процедуру *ПолучитьПланыОбмена()* в общем модуле *ОбменДаннымиПереопределяемый*.

Добавим план обмена и общий модуль в подсистему «Синхронизация данных», но запретим их отображение в командном интерфейсе. Увеличим номер конфигурации до версии 1.0.0.1

Далее нужно выгрузить XDTO-Пакет *spПереносПриемаСотрудников*. Он понадобится для настройки правил в КД 3.0. Для этого найдем его в дереве конфигурации, кликнем по нему правой кнопкой мыши и из контекстного меню выберем пункт *Экспорт XML-схемы*. Точно так же в тот же каталог нужно

экспортировать пакет *ExchangeMessage* – они понадобятся оба.

Откроем *Конвертацию данных 3.0*, в *Главном меню* выберем *Формат данных – Версии формата*. Вверху окна нажмем на кнопку *Загрузка структуры формата*. В открывшемся окне в поле *Файлы со структурой формата* нажмем *кнопку выбора* и найдем каталог, в который экспортировали XDTO-Пакеты. С помощью множественного выбора отметим оба выгруженных пакета и нажмем *Открыть*. Изменим поле *Имя основного пакета XDTO*, запишем туда строку «http://www.sample-package.org/1.0.0». Нажмем на кнопку *Выполнить загрузку*.

В *Главном меню* выберем *Конвертации – Конвертации*, и нажмем на кнопку *Создать*. В новой конвертации укажем:

- Наименование: spИсходная
- Конфигурация: Исходная
- Версии формата: 1.0.0

Добавим новую группу правил *Сотрудники* в группу *Справочники*, для этого выберем в *Главном меню* пункт *Конвертации – Группы правил*, создадим группу, укажем наименование и запишем.

Теперь найдем в *Главном меню* пункт *Конвертации – Настройка правил конвертации* и выберем конвертацию *spИсходная*. В группе *Сотрудники* создадим ПКО для справочника *Должности*

- Идентификатор правила: Справочники_Должности_Отправка
- Объект конфигурации: СправочникСсылка.ДолжностиСотрудников
- Объект формата: Справочник.Должности
- Область применения: Для отправки
- ПКС:
    - Наименование – Наименование

Создадим обычное ПОД для этого ПКО. Создадим ПКО для справочника *Сотрудники*

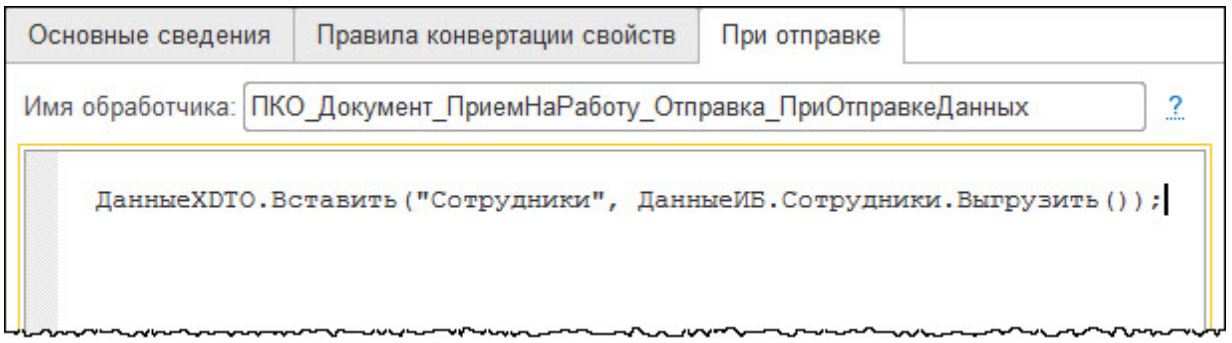
- Идентификатор правила: Справочник_Сотрудники_Отправка
- Объект конфигурации: СправочникСсылка.Сотрудники
- Объект формата: Справочник.Сотрудники
- Область применения: Для отправки
- ПКС:
    - Наименование – ФИО
    - Пол – Пол
    - ДатаРождения – ДатаРождения

Создадим обычное ПКПД для перечисления Пол и укажем в ПКС Пол. Создадим обычное ПОД для этого ПКО Создадим в группе *Документы* ПКО для документа *ПриемНаРаботу*:

- Идентификатор правила: Документ_ПриемНаРаботу_Отправка
- Объект конфигурации: ДокументСсылка.ПриемНаРаботу
- Объект формата: Документ.ПриемНаРаботу
- Область применения: Для отправки
- ПКС:
    - Дата – Дата
    - Номер – Номер
    - ДолжностьСотрудника – Должность (укажем правило)
    - Сотрудники.Сотрудник – Сотрудники.Сотрудник (укажем правило)
    - Сотрудники.Оклад – Сотрудники.Оклад

В событии *При отправке* напишем:

```
ДанныеXDTO.Вставить("Сотрудники", ДанныеИБ.Сотрудники.Выгрузить());
```

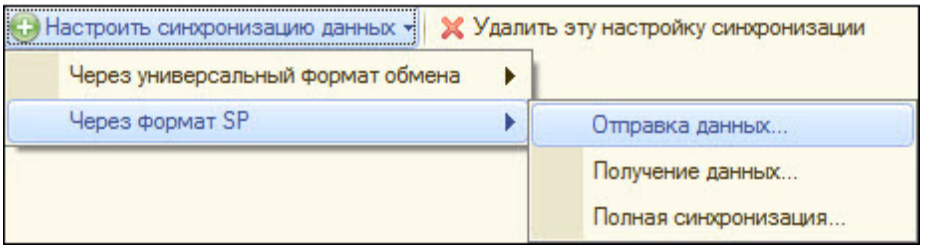


Настроим обычное ПОД для этого ПКО. Теперь создадим новую конвертацию *spКонечная*, в ней создадим аналогичные ПКО, ПОД и ПКПД с областью применения *Для получения*. В ПКО для документа *ПриемНаРаботу* ПКС *Сотрудники.Сотрудник* и *Сотрудники.Оклад* должны быть отмечены флагом *Используется алгоритм конвертации*. В событии *При конвертации данных XDTO* напишем код:

```
МассивСтрокСотрудники = Новый Массив;
Если ДанныеXDTO.Свойство("Сотрудники")
      И ЗначениеЗаполнено(ДанныеXDTO.Сотрудники) Тогда

      Для Каждого Строка Из ДанныеXDTO.Сотрудники Цикл
      СтруктураДанныхСтроки = Новый Структура;
      СтруктураДанныхСтроки.Вставить("Сотрудник", Строка.Сотрудник);
      СтруктураДанныхСтроки.Вставить("Оклад", Строка.Оклад);
      МассивСтрокСотрудники.Добавить(СтруктураДанныхСтроки);

      КонецЦикла;
КонецЕсли;
Если МассивСтрокСотрудники.Количество() > 0 Тогда
      ПолученныеДанные.ДополнительныеСвойства.Вставить("Сотрудники",
МассивСтрокСотрудники);
КонецЕсли;
```

Сохраним модули, вставим их в конфигурации и запустим базы в пользовательском режиме. Начнем настройку нового обмена в базе Исходная ИБ.

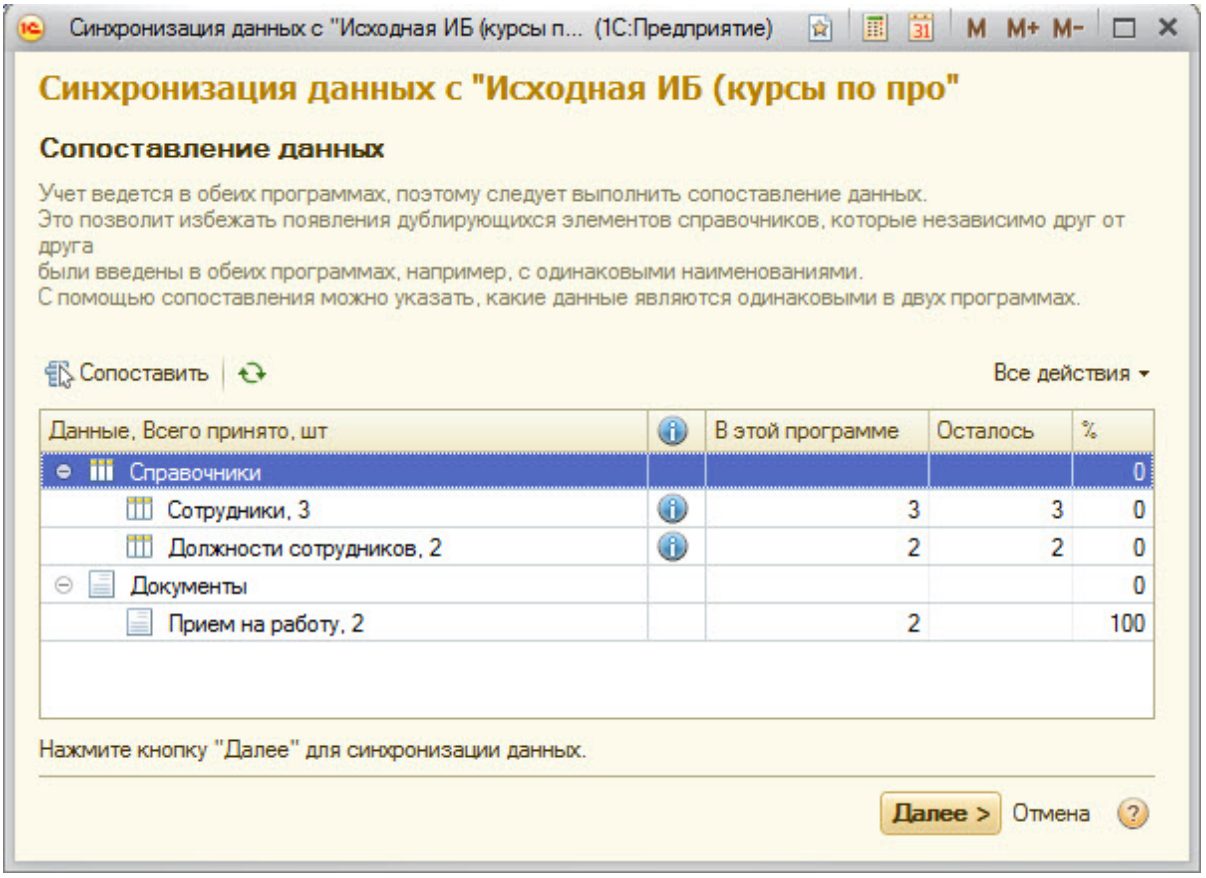
Зайдем в подсистему *Синхронизация данных*, выберем пункт *Синхронизация данных*. Внизу окна нажмем на кнопку *Настроить синхронизацию данных*, в выпадающем меню выберем *Через формат SP – Отправка данных*.



В открывшемся окне сразу нажмем *Далее*, чтобы перейти к настройке транспорта сообщений обмена. Настроим транспорт через сетевой каталог, укажем путь к нему. Остальные варианты транспорта пропустим, нажимая кнопку *Далее*. В поле *Наименование другой базы* укажем *Конечная ИБ*, префикс *КБ*. Нажмем *Далее*, чтобы создалась новая настройка синхронизации, и еще раз нажмем *Далее*, чтобы произошла выгрузка.

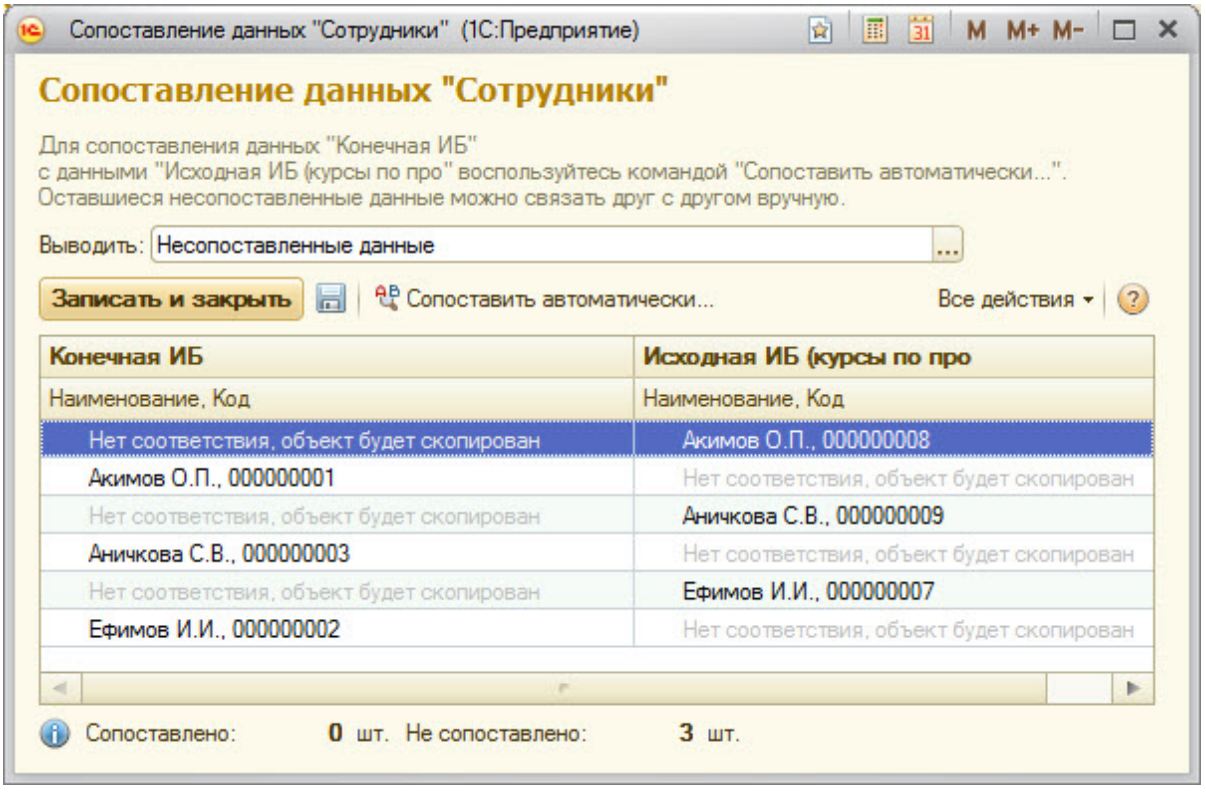
На стороне Конечной базы также в подсистеме *Синхронизация данных* выберем команду *Синхронизация данных*. Нажмем *Настроить синхронизацию данных – Через формат SP – Получение данных*. выберем пункт *Загрузить файл с настройками, созданный в другой программе*, нажмем в поле кнопку выбора и найдем в каталоге обмена этот файл, он должен называться *Синхронизация данных через формат sample-package.xml*. Больше ничего настраивать не потребуется, нажмем *Далее* несколько раз, так, чтобы началась загрузка данных.

Система предлагает установить соответствие загружаемых элементов тем, которые уже есть в Конечной ИБ.

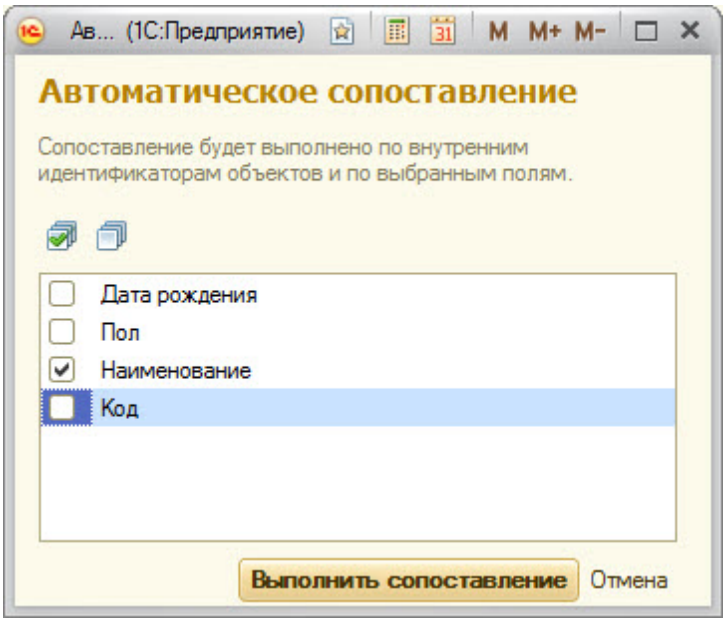


Дело в том, что при переносе документов и элементов справочников через XDTO-сериализацию, перенос уникальных идентификаторов не выполнялся, в базе-Приемнике создавались новый идентификаторы. А в КД 3.0 по умолчанию синхронизация выполняется именно по Уникальному идентификатору. Поэтому соответствие между ними автоматически установлено не будет. Поэтому установим соответствие вручную в диалоге *Сопоставление данных.*

Этот диалог предлагает по умолчанию выполнять сопоставление элементов справочников по *Коду* и *Наименованию*, а документов по *Номеру* и *Дате*. Поэтому для документов указано 100% сопоставление объектов, а справочники сопоставлены не были, так как коды в базах Исходной и Конечной у них отличаются. Кликнем дважды на строке *Сотрудники*. При этом откроется окно сопоставления *Сотрудников*.



Здесь можно настроить соответствие вручную, а можно нажать *Сопоставить автоматически*, и в списке полей оставить флажок только напротив пункта *Наименование*.



Потом нажать *Выполнить сопоставление*, и после того, как объекты будут сопоставлены, нажать *Применить*. Также можно поступить и со справочником *Должности*. Теперь нажмем кнопку *Далее*, и данные будут успешно загружены в базу.

Для проверки того, как переносятся документы и удаления объектов, можно добавить новый документ, выполнить обмен сначала в Исходной, а потом в Конечной базах так, чтобы он появился в списке документов в Конечной базе. А потом удалить его из Исходной и выполнить еще раз синхронизацию в Исходной, а затем в Конечной ИБ. Документ не будет удален непосредственно, в Конечной базе он будет помечен на удаление.