

## Практическая работа № 23

### Обмен данными (6:10)

#### Общие сведения об обмене данными

В этой работе мы познакомимся с механизмами обмена данными, которые содержит система 1С:Предприятие 8, и добавим в нашу конфигурацию возможность обмена данными с удаленными филиалами и отделениями.

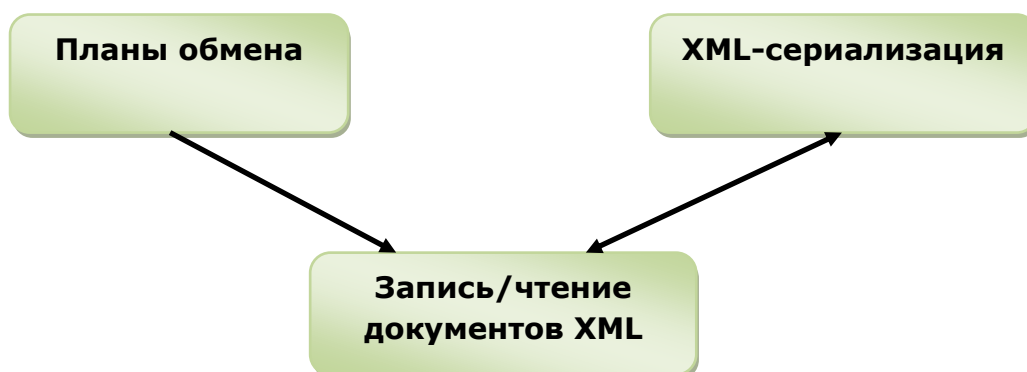
Механизмы обмена данными позволяют организовывать обмен информацией, хранимой в базе данных, с другими программными системами, основанными на 1С:Предприятие 8 или другими.

Такая гибкость обмена достигается за счет использования средств обмена данными в различных комбинациях. Кроме того, формат обмена основан на языке XML, являющимся на сегодняшний день общепринятым средством представления данных.

К механизмам обмена данными могут быть отнесены:

- Планы обмена,
- XML-сериализация,
- Средства чтения и записи документов XML.

Схема взаимодействия этих трех составляющих может быть такой:



При помощи *планов обмена* мы получаем информацию о том, какие элементы данных были изменены и в какой узел обмена их необходимо передать.

Это возможно благодаря тому, что планы обмена содержат механизм регистрации изменений. Информация об измененных данных переносится с помощью сообщений, инфраструктура которых также поддерживается планами обмена.

*XML-сериализация* позволяет преобразовать объект 1С:Предприятия 8 в последовательность данных, представленных в формате XML. Кроме этого, она выполняет и обратное преобразование – преобразует последовательность данных формата XML в объект 1С:Предприятия 8, при условии что имеется соответствующий тип данных.

*Запись и чтение документов XML* обеспечивает запись/чтение документов формата XML из встроенного языка.

При реализации алгоритма обмена данными перечисленные механизмы могут быть использованы как все вместе, так и в различной комбинации. В каждом конкретном случае разработчик решает эту задачу самостоятельно.

### **Что такое план обмена**

Чтобы существовала возможность обмена какими-либо данными с кем-либо, необходимо некоторым образом идентифицировать тех, с кем мы будем обмениваться, и для каждого из них описать перечень обмена. Обе эти задачи позволяет решать объект конфигурации *План обмена*.

Элементами данных плана обмена являются узлы плана обмена.

Каждый узел идентифицирует участника обмена по данному плану обмена. В каждом плане обмена всегда существует один предопределенный узел, идентифицирующий данную информационную базу.

В одной конфигурации может существовать несколько планов обмена. Каждый план обмена определяет набор данных, которым будет производиться обмен в рамках данного плана, и сам механизм обмена.

Наличие нескольких планов может потребоваться, если с разными узлами ведется обмен разным составом данных, или когда схема организации обмена с одними узлами отличается от схемы организации обмена с другими узлами.

В обмене данными могут участвовать:

- Объекты базы данных: элементы справочников, документы и т.д.
- Необъектные данные: наборы записей регистров, последовательностей, константы.
- Специальный объект встроенного языка – **УдалениеОбъекта**.

Для упрощения изложения в дальнейшем будем называть эти элементы информационных структур *объектами обмена*.

Разработчик имеет возможность определить состав каждого плана обмена, указав объекты конфигурации, данные которых должны участвовать в обмене по данному плану и указать для каждого типа объектов признак **Авторегистрация**. Этот признак определяет, каким образом план обмена будет отслеживать изменения данных.

Возможность отслеживать изменения данных реализована в плане обмена за счет использования *механизма регистрации изменений*.

Работа этого механизма базируется на том, что каждый из объектов обмена имеет свойство **ОбменДанными**, с помощью которого можно указать, для каких узлов необходимо производить регистрацию изменений этого объекта. Любые изменения объекта обмена сводятся к записи или удалению объекта обмена. Механизм регистрации изменений анализирует события записи и удаления объектов обмена и на основании параметров обмена данными, содержащихся в каждом из объектов обмена, формирует записи регистрации изменений. Свойство **ОбменДанными** не хранится в БД, а используется только во время записи объекта обмена.

Признак **Авторегистрация**, устанавливаемый при указании состава данных плана обмена, позволяет указать, что параметры обмена данными будут формироваться каждый раз самим механизмом регистрации изменений на основании информации, содержащейся в плане обмена.

После автоматического заполнения параметров обмена разработчик все же имеет возможность внести изменения в сформированные таким образом параметры. Для этого следует использовать обработчики событий объектов, участвующих в обмене, - **ПередЗаписью** и **ПередУдалением**, в которых можно модифицировать список узлов-получателей (т.е. тех узлов, для которых регистрируются изменения).

Как мы теперь знаем, при записи и удалении объектов обмена план обмена формирует *записи регистрации изменений*. Они хранятся в

таблицах регистрации изменений, причем для каждого объекта обмена своя таблица.

При изменении объекта обмена в таблице регистрации изменений создается столько записей (строчек), сколько узлов-получателей указано в параметрах обмена данными у объекта обмена. Каждая запись при этом будет хранить ссылку на свой узел-получатель. Таблицы регистрации изменений создаются лишь в том случае, если соответствующий объект метаданных указан в составе хотя бы одного плана обмена.

Кроме ссылки на узел обмена, каждая запись таблицы регистрации хранит номер *сообщения*, в котором изменение было передано в первый раз в этот узел. До тех пор, пока сообщение не будет передано в первый раз, это поле хранит Null.

*Сообщение с точки зрения плана обмена – это единица обмена информацией. Поэтому одной из важнейших составляющих плана обмена, помимо службы регистрации изменений, является инфраструктура сообщений.*

Поскольку сообщения передаются в рамках плана обмена от одного узла к другому, каждое сообщение точно ассоциировано с планом обмена, имеет уникальный номер и одного отправителя и получателя. За нумерацию сообщений отвечает инфраструктура сообщений. Благодаря этому записи регистрации изменений и имеют возможность хранить номера сообщений, в которых эти изменения были переданы в первый раз.

Инфраструктура сообщений позволяет также получать подтверждения от узла-получателя о приеме сообщений. Такое подтверждение содержится в каждом сообщении, приходящем от узла-получателя в виде номера последнего принятого сообщения.

Впоследствии, проанализировав номер последнего принятого сообщения и номера сообщений, содержащиеся в записях регистрации изменений, разработчик может удалить записи регистрации изменений, прием которых подтвержден получателем.

## **XML-сериализация**

Это механизм, позволяющий представить объект 1С:Предприятия в виде последовательности данных в формате XML. Кроме этого позволяет выполнить и обратное преобразование.

Дело в том, что объект обмена, являющийся в системе единым целым, на самом деле представляет собой совокупность данных различных типов, определенным образом связанных между собой.

Например, элемент справочника, кроме кода и наименования, может содержать некоторое количество реквизитов различного типа и некоторое количество табличных частей, содержащих в свою очередь, некоторое количество реквизитов различного типа.

### **Запись/чтение документов XML**

В отличие от XML-сериализации, механизмы *записи/чтения документов XML* позволяют работать с данными формата XML на базовом уровне, без привязок к объектам 1С:Предприятия.

В частности, они позволяют открывать файлы XML для чтения, читать данные из файлов, создавать новые файлы и записывать в них данные.

### **Универсальный механизм обмена данными**

Наша фирма открыла свой филиал в городе N и установило в нем такую же конфигурацию для учета работы филиала.

В результате возникла необходимость наладить обмен данными между этими двумя базами т.о., чтобы каждая из них отражала полную информацию о материалах и услугах, в то время как бухучет и расчет зарплаты велись бы в каждой базе отдельно.

Для этого мы создадим план обмена, опишем состав данных, которые будут включены в обмен, и сделаем несколько процедур, позволяющих нам формировать на жестком диске файлы обмена и соответственно загружать полученные файлы обмена с жесткого диска.

Для упрощения примера мы не будем программировать какой-либо автоматический обмен файлами между двумя базами, и запуск процедуры обмена будем осуществлять вручную.

Сначала нам нужно внести некоторые доработки. Они будут связаны с тем, что мы работали только в одной базе и использовали уникальность номеров кодов справочников и номеров документов. Теперь, когда создание новых элементов справочников и новых документов будет происходить в двух базах одновременно и независимо, нам снова требуется обеспечить их уникальность. Для этого в каждой базе к

номерам документов и кодам справочников мы будем добавлять уникальный префикс, однозначно идентифицирующий базу данных.

Для хранения префикса номеров мы используем объект конфигурации *Константа*.

### Константа для обмена данными

Объект конфигурации *Константа* предназначен для создания в базе данных таблиц, в которых будет храниться информация, не изменяющаяся во времени или изменяющаяся очень редко.

Каждый такой объект описывает таблицу для хранения одного значения.

Приступим к созданию константы, в которой мы будем хранить значение префикса номеров.

### Доработка объектов конфигурации, участвующих в обмене

#### В режиме Конфигуратор

Откроем конфигуратор под **Администратором** и добавим новый объект конфигурации *Константа* с именем **ПрефиксНумерации**. Определим типа значения константы – **Строка** с фиксированной длиной 2 символа.

Первое, что следует сделать – внести изменения в модули всех объектов, участвующих в обмене (в нашем случае это справочники, документы и планы видов характеристик).

Эти изменения будут заключаться в том, что теперь при формировании номера документа и кода справочника или плана видов характеристик будет использоваться значение константы **ПрефиксНумерации** для обеспечения уникальности номеров и кодов в каждой из наших баз.

Функцию формирования префикса мы вынесем в общий модуль, поскольку не исключена возможность. Что в будущем алгоритм формирования префикса может быть изменен.

Добавим общий модуль **Обмен**. В него поместим функцию:

```
Функция ПолучитьПрефиксНомера() Экспорт
```

```
    Возврат Константы.ПрефиксНумерации.Получить();
```

```
КонецФункции
```

Как вы видите, эта функция просто возвращает значение константы **ПрефиксНумерации**.

Теперь доработаем справочник **Клиенты**.

Откроем модуль объекта и добавим в него обработчик события **ПриУстановкеНовогоКода**:

```
Процедура ПриУстановкеНовогоКода(СтандартнаяОбработка, Префикс)
```

```
    Префикс = Обмен.ПолучитьПрефиксНомера();
```

```
КонецПроцедуры
```

Событие **ПриУстановкеНовогоКода** возникает в момент, когда выполняется установка нового кода элемента справочника. Обратите внимание, что мы пишем этот код не в модуле формы, а в модуле объекта, поскольку это событие возникает не для формы, а для объекта в целом.

Вторым параметром вызова обработчика передается префикс, который будет заполнен в данной процедуре и использован системой для генерации кода.

В обработчике события мы вызываем функцию общего модуля. Поскольку модуль неглобальный, то обращаемся к ней по имени модуля и имени функции (Обмен.ПолучитьПрефиксНомера). В этой процедуре мы устанавливаем префикс равным значению константы **ПрефиксНумерации**.

Такие же обработчики нужно будет добавить во все справочники и планы видов характеристик, участвующие в обмене.

В нашем случае это:

- Справочники:
  - **Сотрудники,**
  - **Склады,**
  - **Номенклатура,**
  - **ВариантыНоменклатуры,**
  - **ДополнительныеСвойстваНоменклатуры,**
- План видов характеристик: **СвойстваНоменклатуры.**

После этого у всех этих объектов и у справочника **Клиенты** нужно в свойствах увеличить длину кода до 11 символов.

Свойства: Клиенты

Основные:

Имя: Клиенты

Синоним: Клиенты

Комментарий:

Модуль объекта: Открыть

Модуль менеджера: Открыть

Данные:

Иерархический: ☐

Вид иерархии: Иерархия групп и элементов

Ограничивать кол-во уровней: ☐

Количество уровней: 2

Размещать группы сверху: ☒

Владельцы:

Использование подчинения: Элементам

Длина кода: 11

Длина наименования: 25

Тип кода: Строка

Допустимая длина кода: Переменная

Серии кодов: Во всем справочнике

Контроль уникальности: ☒

Автонумерация: ☒

Длина кода: 11

Теперь займемся доработкой документов.

В модуль документа **ПриходнаяНакладная** добавим обработчик события **ПриУстановкеНовогоНомера**:

```
Процедура ПриУстановкеНовогоНомера(СтандартнаяОбработка, Префикс)
```

```
    Префикс = Обмен.ПолучитьПрефиксНомера();
```

```
КонецПроцедуры
```

Такие же обработчики нужно добавить во все документы, участвующие в обмене. В нашем случае это документ **ОказаниеУслуги**.

После этого в обоих документах в свойствах увеличить длину номера до 11 символов.



**Свойства: ОказаниеУслуги**

**Основные:**

Имя: ОказаниеУслуги

Синоним: Оказание услуги

Комментарий:

Модуль объекта: Открыть

Модуль менеджера: Открыть

**Данные:**

Нумератор: ... X

Тип номера: Строка

Длина номера: 11

Допустимая длина номера: Переменная

Периодичность: Непериодический

Допустимая длина номера документа:

При этом подготовительная работа с существующими объектами конфигурации завершена, и мы можем перейти к созданию процедур обмена данными.

### Добавление плана обмена

Раскроем ветвь Общие дерева объектов конфигурации и добавим новый **ПланОбмена** с именем **Филиалы**, представление объекта – **Филиал**.

На закладке **Данные** создадим реквизит плана обмена **Главный** с типом **Булево**.

**План обмена Филиалы**

Основные

Подсистемы

Функциональные опции

Данные

Формы

Команды

Макеты

Ввод на основании

Права

Прочее

Длина кода

Длина наименования

Основное представление

☐ В виде кода

☒ В виде наименования

Реквизиты

Главный

**Свойства: Главный**

**Основные:**

Имя: Главный

Синоним: Главный

Комментарий:

Тип: Булево

**Использование:**

Индексировать: Не индексировать

Тип данных:

Этот реквизит понадобится, чтобы разрешать коллизии при обмене данными. Под коллизией понимается ситуация, когда один и тот же объект обмена данными был изменен одновременно в двух узлах.

В этом случае мы будем анализировать значение реквизита **Главный** и принимать изменения только в случае, если они сделаны в главном узле.

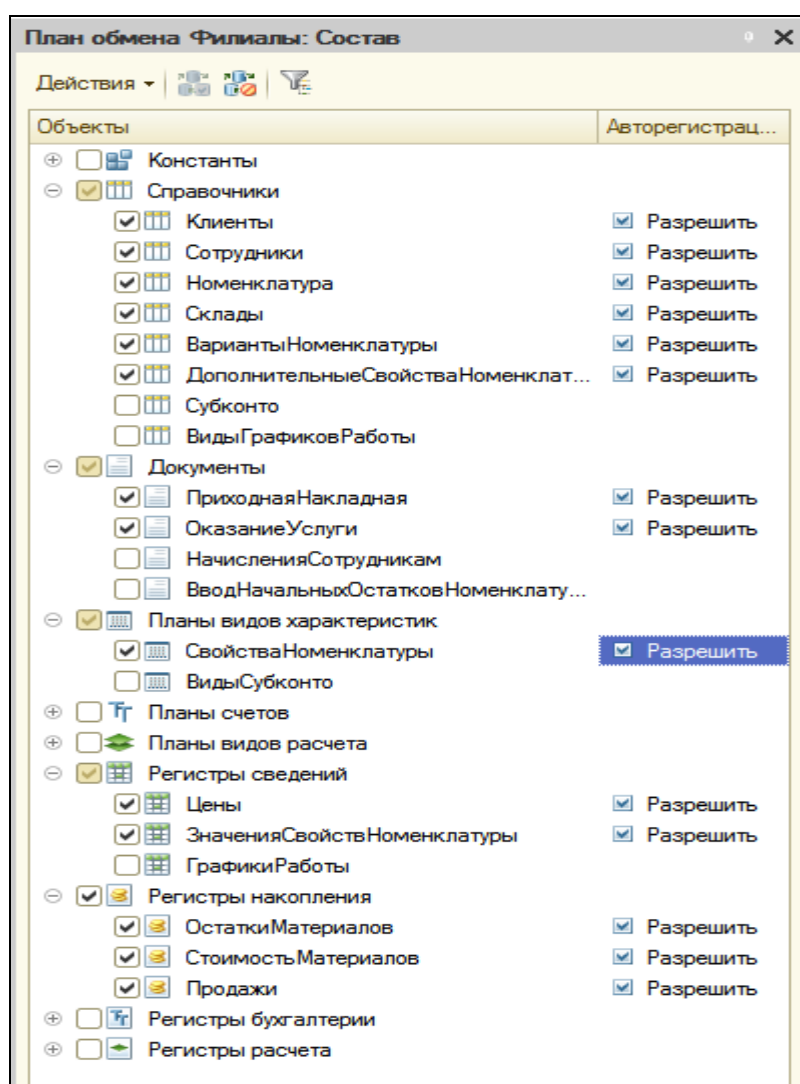
Теперь определим состав объектов, участвующих в обмене.

Для этого на закладке **Основные** нажмем кнопку **Состав**.

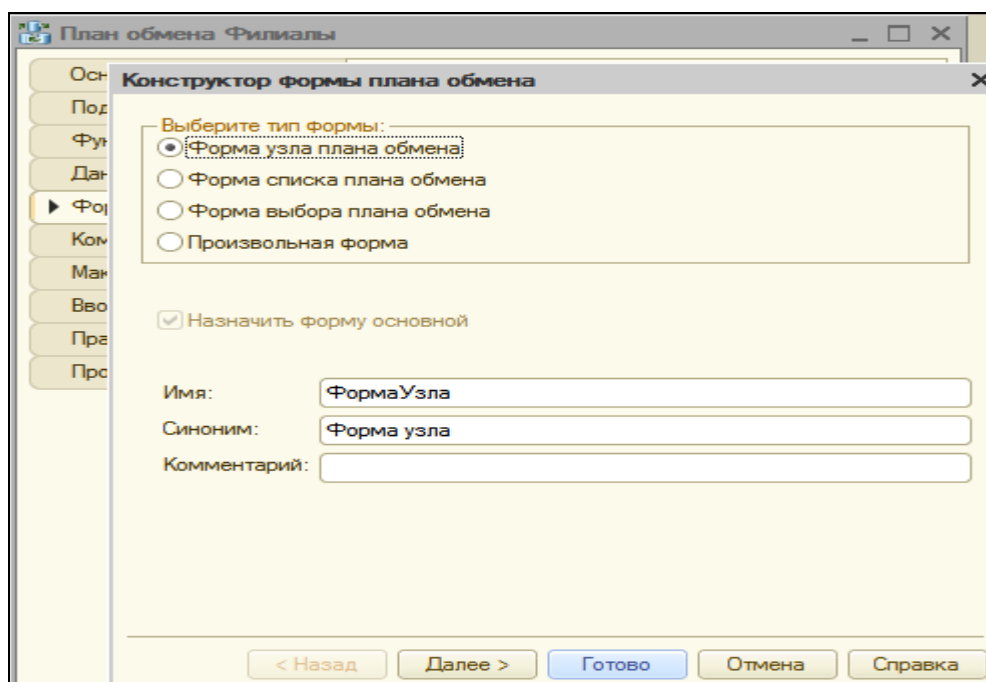
Включим в обмен все объекты, не относящиеся к ведению бухучета и расчету зарплаты.

Обратите внимание, что константа **ПрефиксНумерации** не участвует в обмене, поскольку ее значение должно быть уникальным для каждой базы, участвующей в обмене.

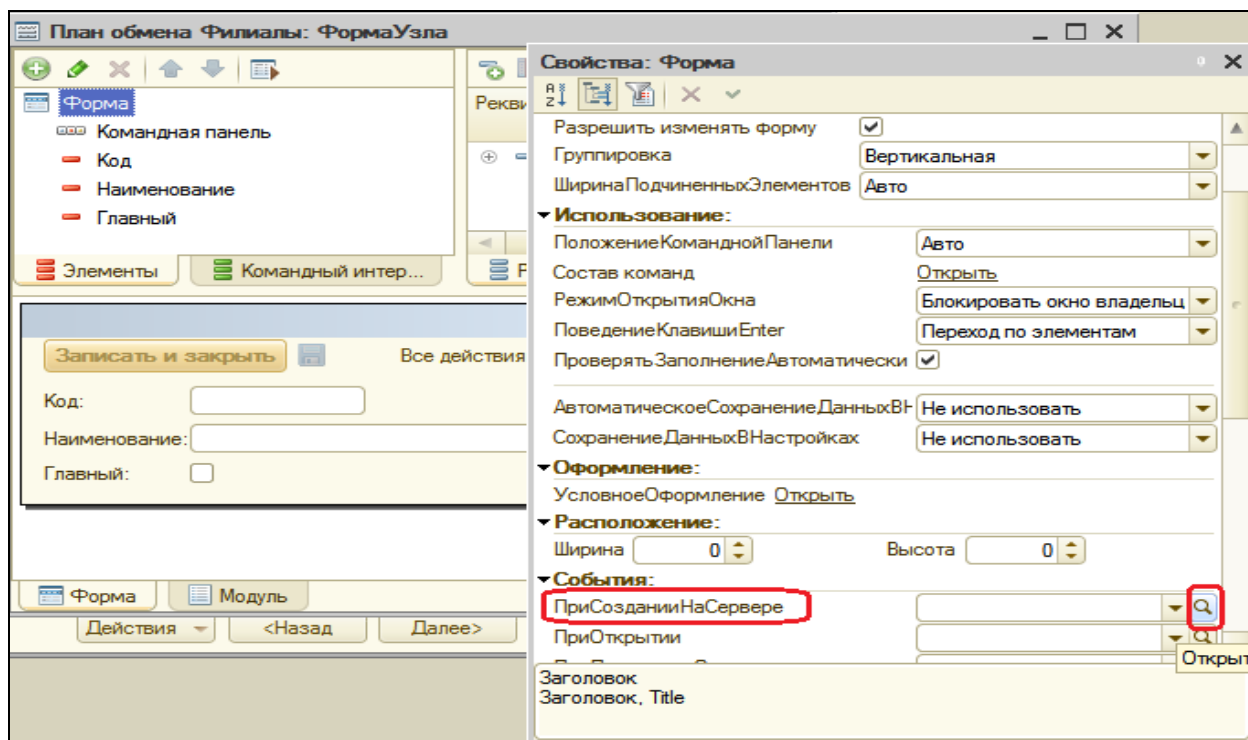
Состав данных обмена должен выглядеть следующим образом:



Теперь на закладке **Формы** нажмем кнопку открытия и с помощью конструктора создадим основную форму узла.



В окне элементов формы выделим корневой элемент **Форма**, вызовем контекстное меню - **Свойства** и создадим обработчик события формы **ПриСозданииНаСервере**. Он понадобится для того, чтобы запретить установку реквизита **Главный** для предопределенного узла, соответствующего данной информационной базе.



&НаСервере

**Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)**

**Если Объект.Ссылка = ПланыОбмена.Филиалы.ЭтотУзел() Тогда  
Элементы.Главный.Доступность = Ложь;**

**КонецЕсли;**

**КонецПроцедуры**

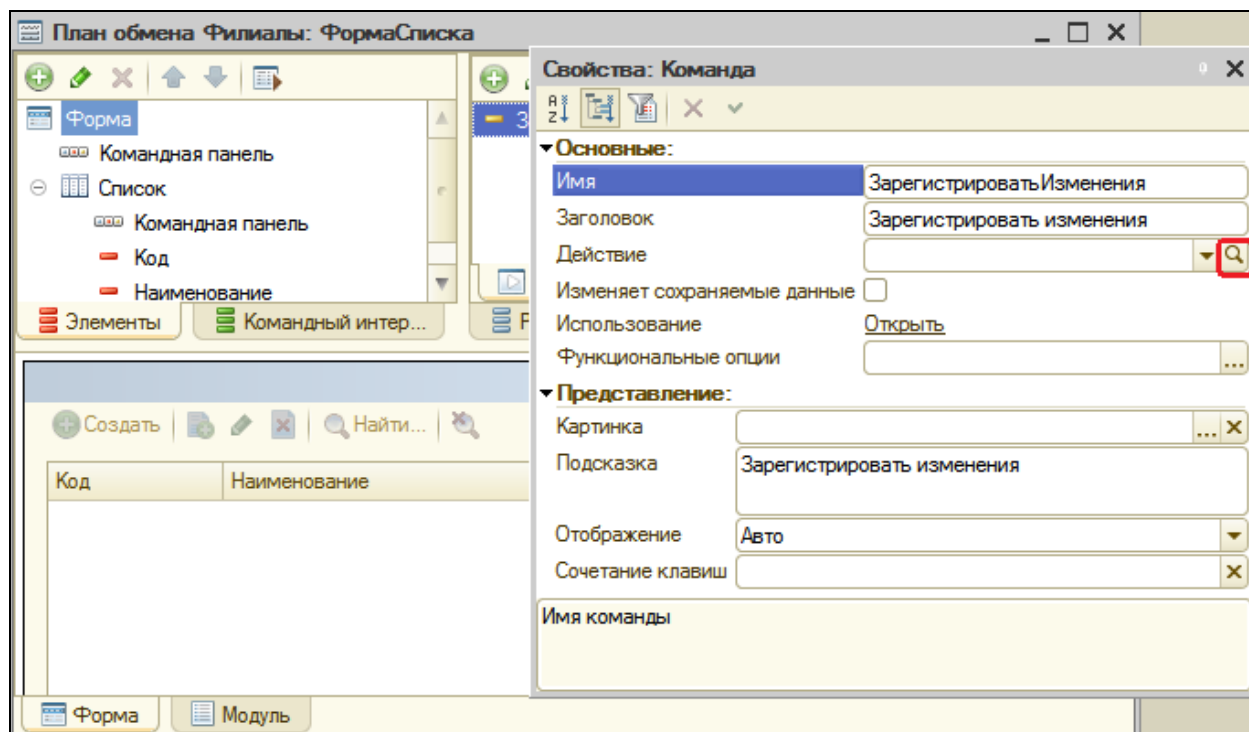
В этой процедуре мы используем метод менеджера плана обмена **ЭтотУзел()**, который возвращает ссылку на узел плана обмена, соответствующий данной информационной базе.

Затем создадим основную форму списка плана обмена, чтобы описать в ней некоторые действия по регистрации нового узла обмена.

Суть этих действий будет заключаться в том, что при регистрации нового узла обмена мы должны будем сформировать для него все необходимые записи регистрации изменений для всех объектов конфигурации, входящих в данный план обмена. Это будет своего рода начальная синхронизация узла обмена всеми данными обмена.

Для этого на закладке **Команды** создадим команду **ЗарегистрироватьИзменения**.

В окне свойств нажмем кнопку открытия  в строке **Действие**.



Иногда при нажатии на кнопку открытия ничего не происходит. В этом случае закройте окно свойств и откройте его заново – кнопка должна работать.

Шаблон обработчика события выполнения этой команды заполним следующим образом:

```
&НаКлиенте
Процедура ЗарегистрироватьИзменения(Команда)
    РегистрацияИзмененийНаСервере(Элементы.Список.ТекущаяСтрока);
КонiecПроцедуры
```

В этом обработчике мы вызываем процедуру **РегистрацияИзмененийНаСервере()**, которую мы напишем в дальнейшем. Она будет выполняться на сервере.

В параметре **Элементы.Список.ТекущаяСтрока** мы передаем в нее ссылку на объект **ПланОбмена.Филиалы**, используя свойство **ТекущаяСтрока** для таблицы **Список** (источником данных для этой таблицы является динамический список узлов плана обмена **Филиалы**).

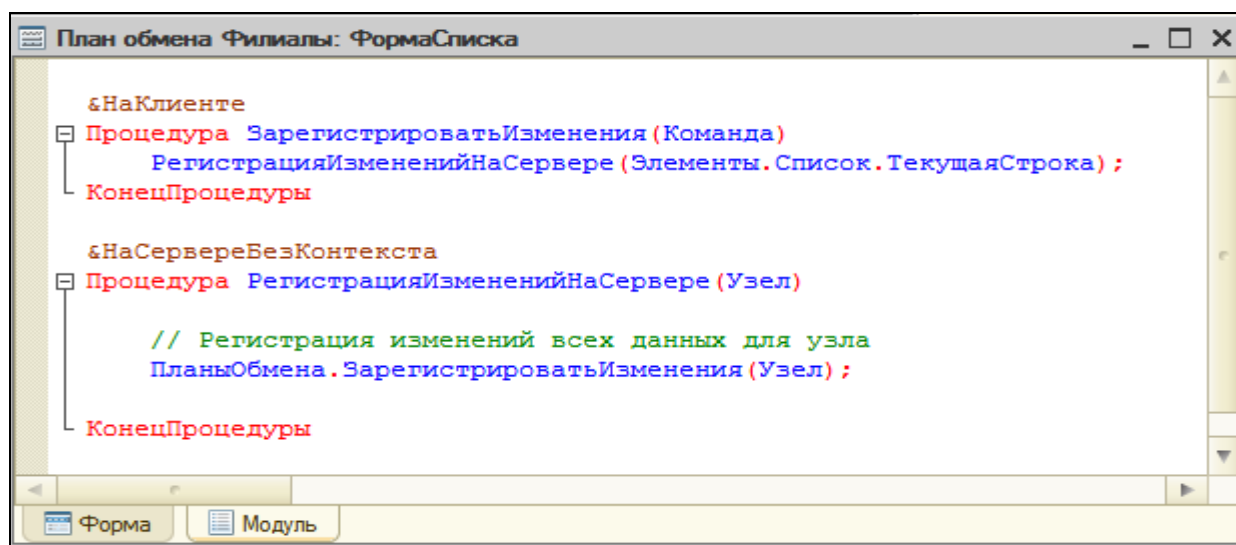
Саму процедуру **РегистрацияИзмененийНаСервере()** мы предварим директивой компиляции **&НаСервереБезКонтекста**, т.к. процедура работает быстрее, если при ее вызове не передается контекст всей формы. Запишем ее ниже предыдущей процедуры в модуле формы списка:

```
&НаСервереБезКонтекста
Процедура РегистрацияИзмененийНаСервере(Узел)

    // Регистрация изменений всех данных для узла
    ПланыОбмена.ЗарегистрироватьИзменения(Узел);

КонiecПроцедуры
```

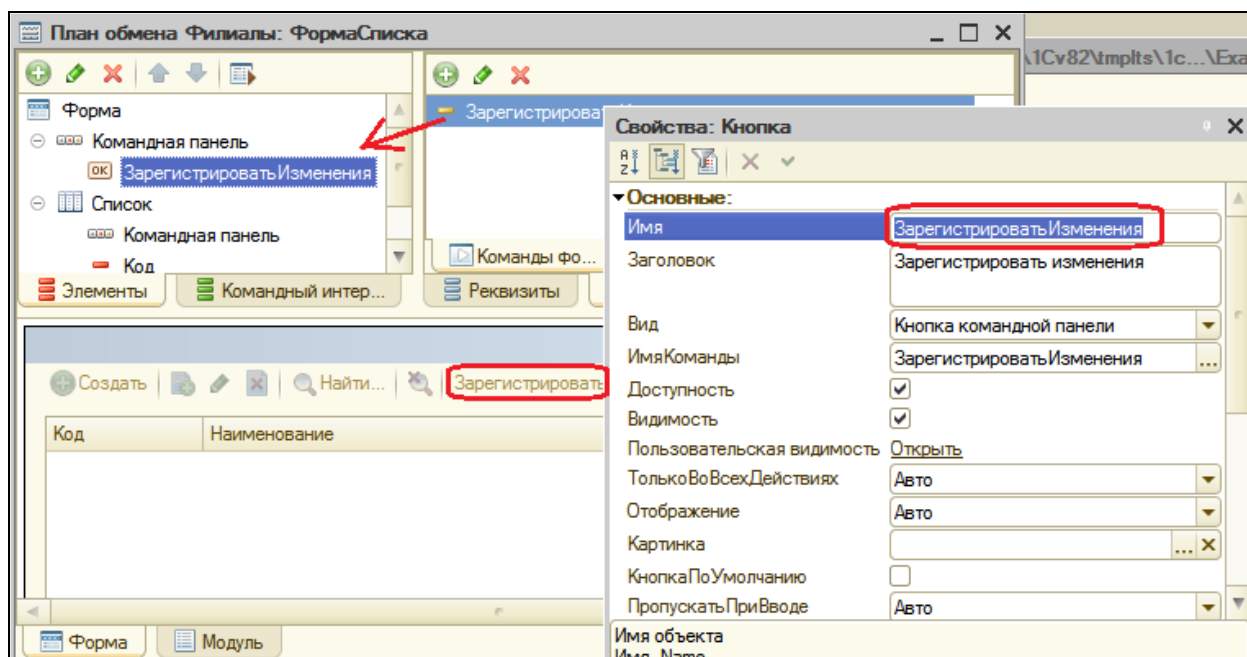
Модуль формы списка плана обмена **Филиалы** будет выглядеть так:



В этой процедуры мы обращаемся к механизму регистрации изменений, вызывая метод менеджера планов обмена – **ЗарегистрироватьИзменения()**. В этот метод передается ссылка на текущий узел плана обмена **Филиалы**.

В результате выполнения этой процедуры в информационной базе будут созданы записи регистрации изменений, предназначенные для пересылки в созданный нами узел, для всех объектов обмена, связанных в составе данного плана обмена.

Перейдем на закладку **Форма** и перетащим команду **ЗарегистрироватьИзменения** из окна команд в окно элементов формы в командную панель. Причем кнопка Зарегистрировать изменения должна быть доступна только в случае, если текущий узел является предопределенным для данной информационной базы, иначе регистрация изменений невозможна.



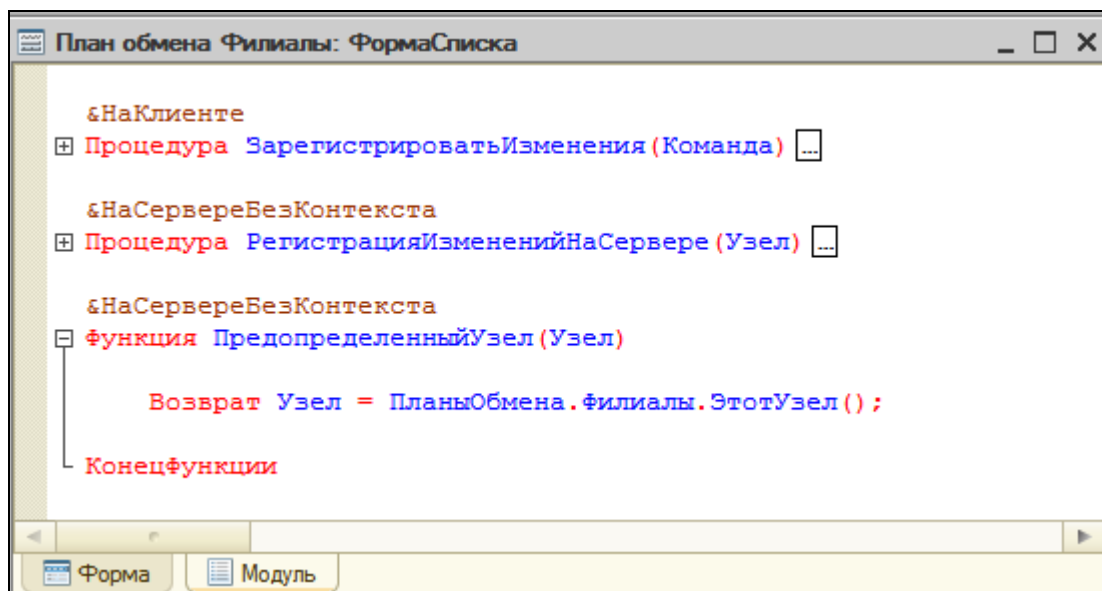
Чтобы обеспечить такое поведение кнопки, создадим в модуле формы списка функцию, выполняющуюся на сервере и возвращающую истину, если переданный в функцию узел является предопределенным.

&НаСервереБезКонтекста

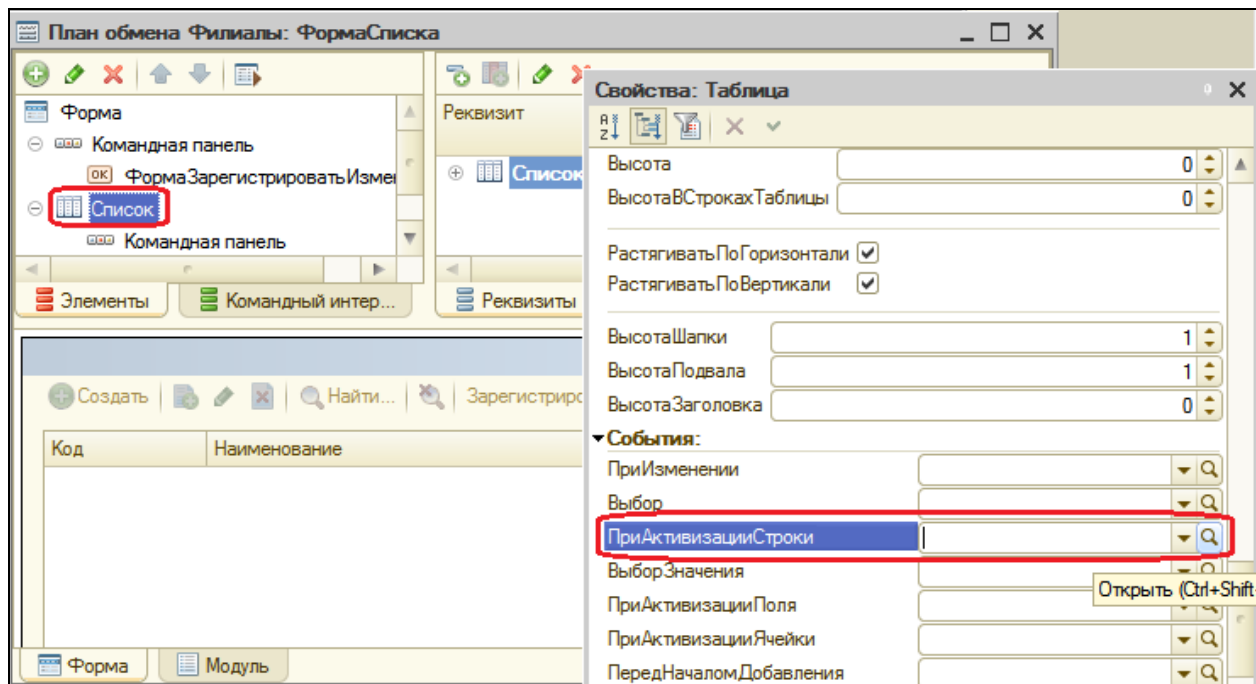
Функция ПредопределенныйУзел(Узел)

Возврат Узел = ПланыОбмена.Филиалы.ЭтотУзел();

КонецФункции



Затем в окне элементов формы выделим элемент **Список**, вызовем его свойства и создадим обработчик события **ПриАктивизацииСтроки**:



Процедура СписокПриАктивизацииСтроки(Элемент)

```

Если ПредопределенныйУзел(Элемент.ТекущаяСтрока) Тогда
    Элементы.ЗарегистрироватьИзменения.Доступность = Ложь;
Иначе
    Элементы.ЗарегистрироватьИзменения.Доступность = Истина;
КонецЕсли;

КонецПроцедуры
  
```

В этой процедуре доступность кнопки **ЗарегистрироватьИзменения** определяется в зависимости от значения функции **ПредопределенныйУзел()**, в которую передается ссылка на текущий узел (**Элемент.ТекущаяСтрока**).

На этом создание плана обмена завершено, и мы можем перейти непосредственно к созданию процедур обмена данными.

### Процедуры обмена данными

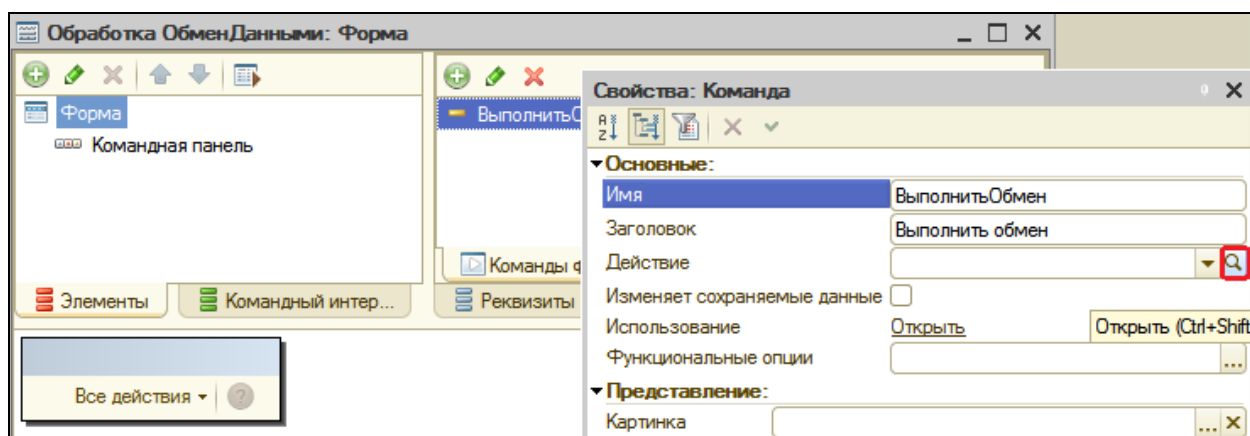
Для начала обмена данными мы используем обработку.

Если вы не хотите детально разбираться в коде, пролистайте до конца раздела, там будет процедура полностью

Добавим новый объект **Обработка** с именем **ОбменДанными**.

На закладке **Формы** создадим основную форму обработки.

В окне редактора форм на закладке **Команды** создадим команду формы **ВыполнитьОбмен**. В строке **Действие** нажмем кнопку открытия и создадим обработчик выполнения этой команды – вызов процедуры **ОбменСФилиалами()**.



```
&НаКлиенте  
Процедура ВыполнитьОбмен(Команда)  
    ОбменСФилиалами();  
КонецПроцедуры
```

Затем в модуле формы создадим процедуру **ОбменСФилиалами**, выполняющуюся на сервере:

```
&НаСервереБезКонтекста  
Процедура ОбменСФилиалами() Экспорт
```

```
    ВыборкаУзлов = ПланыОбмена.Филиалы.Выбрать();
```



Пока ВыборкаУзлов.Следующий() Цикл

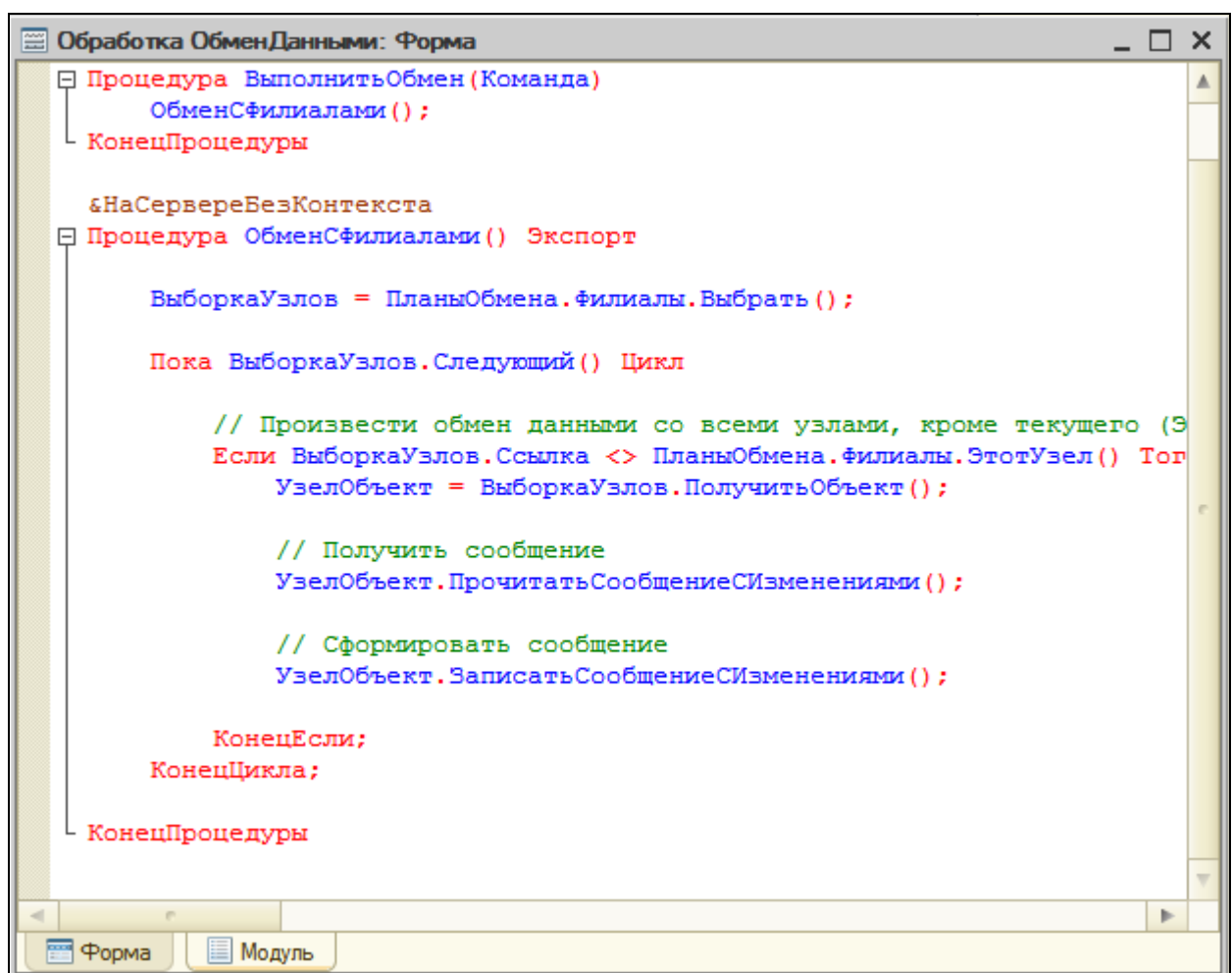
```
// Произвести обмен данными со всеми узлами, кроме текущего (ЭтотУзел)  
Если ВыборкаУзлов.Ссылка <> ПланыОбмена.Филиалы.ЭтотУзел() Тогда  
    УзелОбъект = ВыборкаУзлов.ПолучитьОбъект();
```

```
    // Получить сообщение  
    УзелОбъект.ПрочитатьСообщениеСИзменениями();
```

```
    // Сформировать сообщение  
    УзелОбъект.ЗаписатьСообщениеСИзменениями();
```

```
КонецЕсли;  
КонецЦикла;
```

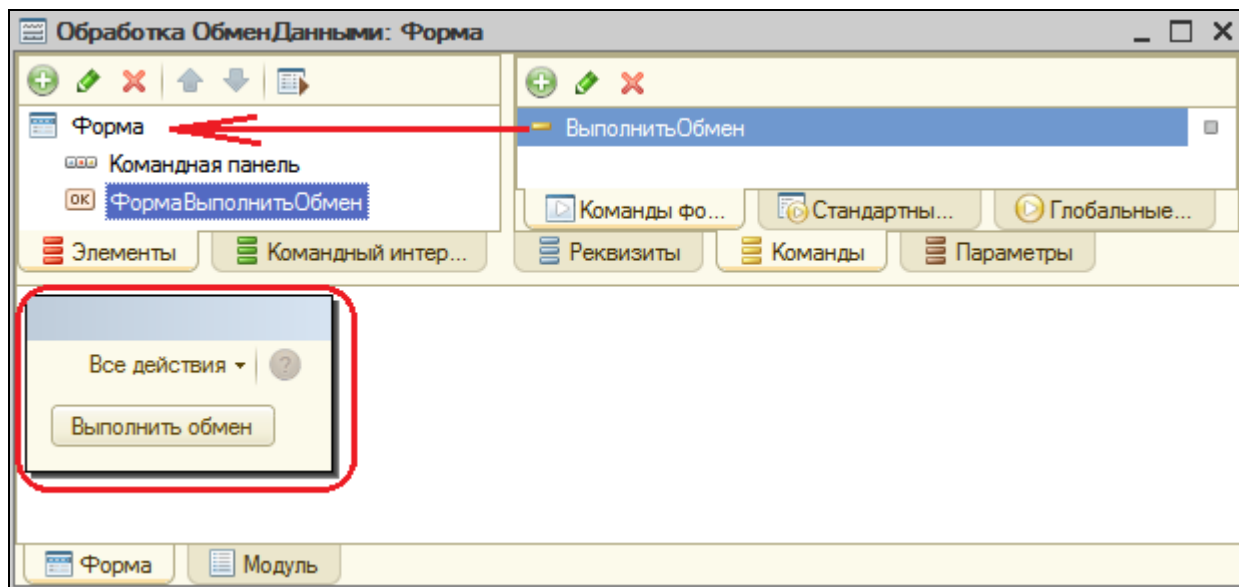
КонецПроцедуры



Алгоритм процедуры в следующем: в цикле мы перебираем узлы, которые содержатся в плане обмена **Филиалы**, и для всех узлов, кроме себя самого, производим сначала чтение сообщений, поступивших из других узлов (процедуру **ПрочитатьСообщенияСИзменениями** мы создадим позднее).

Затем мы формируем для них сообщения, предназначенные для передачи и содержащие измененные данные для этого узла (процедура **ЗаписатьСообщенияСИзменениями** также будет создана позднее).

В заключение перетащим команду **ВыполнитьОбмен** из окна команд в окно элементов формы (перетащить на **Форма**, а на командную панель).



### **Процедура записи данных**

Сами процедуры записи и чтения данных обмена мы разместим в модуле объекта План обмена **Филиалы**.

В окне редактирования этого объекта перейдем на закладку **Прочее** и откроем модуль объекта.

Сначала создадим процедуру, которая используется нами при обмене данными – **ЗаписатьСообщениеСИзменениями**. Создавать ее будем постепенно.

Сначала сформируем имя файла, который будет содержать данные для обмена, и сообщим пользователю о начале и окончании выгрузки данных в узел.

Процедура ЗаписатьСообщениеСИзменениями() Экспорт

```
Сообщение = Новый СообщениеПользователю;  
Сообщение.Текст = "----- Выгрузка в узел " + Строка(ЭтотОбъект) + " -----  
---";  
Сообщение.Сообщить();  
Каталог = КаталогВременныхФайлов();  
  
// Сформировать имя временного файла
```

```

        ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\" ) + "Message" +
        СокрЛП(ПланыОбмена.
            Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) + ".xml";
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "----- Конец выгрузки -----";
        Сообщение.Сообщить();

КонецПроцедуры

```

Для упрощения примера мы будем обмениваться сообщениями через каталог временных файлов. Имена сообщений стандартизованы и имеют вид **MessageКодУзлаОтправителя\_КодУзлаПолучателя.xml**.

После этого обратимся к механизмам записи/чтения XML-документов и создадим новый объект – **ЗаписьXML**. С его помощью откроем новый XML-файл для записи, запишем в него объявление XML. В конце процедуры завершим запись и закроем файл:

```

Процедура ЗаписатьСообщениеСИзменениями() Экспорт

    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "----- Выгрузка в узел " + Строка(ЭтотОбъект) + " -----
---";
    Сообщение.Сообщить();
    Каталог = КаталогВременныхФайлов();

    // Сформировать имя временного файла
    ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\" ) + "Message" +
    СокрЛП(ПланыОбмена.
        Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) + ".xml";

    // Создать объект записи XML
    // *** ЗаписьXML-документов
    ЗаписьXML = Новый ЗаписьXML;
    ЗаписьXML.ОткрытьФайл(ИмяФайла);
    ЗаписьXML.ЗаписатьОбъявлениеXML();
    ЗаписьXML.Закрыть();

    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "----- Конец выгрузки -----";
    Сообщение.Сообщить();

КонецПроцедуры

```

Теперь мы обратимся к механизмам инфраструктуры сообщений и создадим новый объект **ЗаписьСообщенияОбмена**, метод которого **НачатьЗапись()** позволяет кроме всего прочего создать очередной номер сообщения и записать заголовок сообщения в XML. В конце процедуры мы опять закончим запись сообщения.

Процедура ЗаписатьСообщениеСИзменениями() Экспорт

```
Сообщение = Новый СообщениеПользователю;  
Сообщение.Текст = "----- Выгрузка в узел " + Строка(ЭтотОбъект) + " -----  
---";  
Сообщение.Сообщить();  
Каталог = КаталогВременныхФайлов();  
  
// Сформировать имя временного файла  
ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\" ) + "Message" +  
СокрЛП(ПланыОбмена.  
    Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) + ".xml";  
  
// Создать объект записи XML  
// *** ЗаписьXML-документов  
ЗаписьXML = Новый ЗаписьXML;  
ЗаписьXML.ОткрытьФайл(ИмяФайла);  
ЗаписьXML.ЗаписатьОбъявлениеXML();  
  
// *** Инфраструктура сообщений  
ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();  
ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Ссылка);  
Сообщение = Новый СообщениеПользователю;  
Сообщение.Текст = " Номер сообщения: " +  
ЗаписьСообщения.НомерСообщения;  
Сообщение.Сообщить();  
ЗаписьСообщения.ЗакончитьЗапись();  
  
ЗаписьXML.Закрыть();  
  
Сообщение = Новый СообщениеПользователю;  
Сообщение.Текст = "----- Конец выгрузки -----";  
Сообщение.Сообщить();  
  
КонецПроцедуры
```

Поскольку мы находимся в модуле объекта, то мы используем стандартный реквизит **Ссылка** в качестве ссылки на объект План обмена **Филиалы**.

После этого, чтобы получить данные, которые необходимо сохранить в этом файле, мы обратимся к механизму регистрации изменений и получим выборку из записей регистрации изменений, предназначенных данному узлу. При формировании выборки мы передаем вторым параметром номер сообщения.

```
// *** Инфраструктура сообщений  
ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();  
ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Ссылка);  
Сообщение = Новый СообщениеПользователю;  
Сообщение.Текст = " Номер сообщения: " + ЗаписьСообщения.НомерСообщения;  
Сообщение.Сообщить();
```

```

// Получить выборку измененных данных
// *** Механизм регистрации изменений
ВыборкаИзменений =
ПланыОбмена.ВыбратьИзменения(ЗаписьСообщения.Получатель
,ЗаписьСообщения.НомерСообщения);

ЗаписьСообщения.ЗакончитьЗапись();

ЗаписьXML.Закрыть();

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "----- Конец выгрузки -----";
Сообщение.Сообщить();

КонецПроцедуры

```

Теперь осталось только перебрать выборку записей в цикле и сериализовать их в открытый XML-файл.

```

// Получить выборку измененных данных
// *** Механизм регистрации изменений
ВыборкаИзменений =
ПланыОбмена.ВыбратьИзменения(ЗаписьСообщения.Получатель
,ЗаписьСообщения.НомерСообщения);
Пока ВыборкаИзменений.Следующий() Цикл
// Записать данные в сообщение *** XML-сериализация
ЗаписатьXML(ЗаписьXML, ВыборкаИзменений.Получить());
КонецЦикла;

ЗаписьСообщения.ЗакончитьЗапись();

ЗаписьXML.Закрыть();

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "----- Конец выгрузки -----";
Сообщение.Сообщить();

КонецПроцедуры

```

На этом создание процедуры записи данных обмена закончено. Полностью она выглядит так:

```

Процедура ЗаписатьСообщениеСИзменениями() Экспорт

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "----- Выгрузка в узел " + Строка(ЭтотОбъект) + " -----
---";
Сообщение.Сообщить();
Каталог = КаталогВременныхФайлов();

// Сформировать имя временного файла
ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") + "Message" +
СокрЛП(ПланыОбмена.
Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) + ".xml";

```

```

// Создать объект записи XML
// *** ЗаписьXML-документов
ЗаписьXML = Новый ЗаписьXML;
ЗаписьXML.ОткрытьФайл(ИмяФайла);
ЗаписьXML.ЗаписатьОбъявлениеXML();

// *** Инфраструктура сообщений
ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Ссылка);
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = " Номер сообщения: " + ЗаписьСообщения.НомерСообщения;
Сообщение.Сообщить();

// Получить выборку измененных данных
// *** Механизм регистрации изменений
ВыборкаИзменений =
ПланыОбмена.ВыбратьИзменения(ЗаписьСообщения.Получатель
                                ,ЗаписьСообщения.НомерСообщения);
Пока ВыборкаИзменений.Следующий() Цикл
// Записать данные в сообщение *** XML-сериализация
ЗаписатьXML(ЗаписьXML, ВыборкаИзменений.Получить());
КонецЦикла;

ЗаписьСообщения.ЗакончитьЗапись();

ЗаписьXML.Закрыть();

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "----- Конец выгрузки -----";
Сообщение.Сообщить();

КонецПроцедуры

```

### ***Процедура чтения данных***

Порядок создания процедуры чтения данных будет таким же, как и ранее. Если вы не хотите детально разбираться в коде, пролистайте до конца раздела, там будет процедура полностью.

Сначала сформируем имя файла, содержащего данные обмена.

Процедура ПрочитатьСообщениеСИИзменениями() Экспорт

```

Каталог = КаталогВременныхФайлов();

// Сформировать имя файла
ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "\", "\\") +
"Message" + СокрЛП(Ссылка.Код) + "_" +
СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) + ".xml";
Файл = Новый Файл(ИмяФайла);
Если Не Файл.Существует() Тогда
    Возврат;
КонецЕсли;

```

```

УдалитьФайлы(ИмяФайла);
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "----- Конец загрузки -----";
Сообщение.Сообщить();

```

КонецПроцедуры

Мы формируем имя файла, которое надеемся найти в этом каталоге, а затем, создав новый объект **Файл** с таким именем проверяем, существует ли он. Если такого файла нет, мы завершаем работу процедуры. Если же он найден, нужно будет удалить его после того, как все данные, содержащиеся в нем, будут обработаны.

Теперь добавим в процедуру команды чтения найденного файла с данными обмена.

Процедура ПрочитатьСообщениеСИИзменениями() Экспорт

```

    Каталог = КаталогВременныхФайлов();

    // Сформировать имя файла
    ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "\", "\") +
    "Message" + СокрЛП(Ссылка.Код) + "_" +
    СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) + ".xml";
    Файл = Новый Файл(ИмяФайла);
    Если Не Файл.Существует() Тогда
        Возврат;
    КонецЕсли;
    // *** Чтение документов XML
    // Попытаться открыть файл
    ЧтениеXML = Новый ЧтениеXML;
    Попытка
        ЧтениеXML.ОткрытьФайл(ИмяФайла);
    Искключение
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "Невозможно открыть файл обмена данными.";
        Сообщение.Сообщить();
    Возврат;
    КонецПопытки;
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "----- Загрузка из " + Строка(ЭтотОбъект) + " -----";
    Сообщение.Сообщить();
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = " – Считывается файл " + ИмяФайла;
    Сообщение.Сообщить();

    ЧтениеXML.Закрыть();

    УдалитьФайлы(ИмяФайла);
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "----- Конец загрузки -----";
    Сообщение.Сообщить();

```

Именно в этот момент мы обращаемся к механизмам записи/чтения документов XML, которые работают с ними на базовом уровне.

Для этого мы создаем новый объект **ЧтениеXML**, с помощью которого открываем найденный файл для чтения. В случае успеха мы выводим сообщение о начале загрузки данных из файла. В конце процедуры мы также прекращаем чтение XML-данных из файла методом **Заккрыть()**.

Полученные таким образом данные должны являться некоторым сообщением обмена данными. Для того чтобы предоставить их в терминах сообщений, мы добавим в процедуру следующий код:

```
// *** Чтение документов XML
// Попытаться открыть файл
ЧтениеXML = Новый ЧтениеXML;
Попытка
    ЧтениеXML.ОткрытьФайл(ИмяФайла);
Исключение
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "Невозможно открыть файл обмена данными.";
    Сообщение.Сообщить();
Возврат;
КонецПопытки;
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "----- Загрузка из " + Строка(ЭтотОбъект) + " -----";
Сообщение.Сообщить();
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = " – Считывается файл " + ИмяФайла;
Сообщение.Сообщить();

// Загрузить из найденного файла
// *** Инфраструктура сообщений
ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

// Читать заголовок сообщения обмена данными – файла XML
ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

ЧтениеСообщения.ЗакончитьЧтение();

ЧтениеXML.Заккрыть();

УдалитьФайлы(ИмяФайла);
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "----- Конец загрузки -----";
Сообщение.Сообщить();

КонецПроцедуры
```

Здесь мы обращаемся к механизмам инфраструктуры сообщений планов обмена и создаем объект **ЧтениеСообщенияОбмена**. Используя метод



этого объекта **НачатьЧтение()**, мы считываем заголовок XML-сообщения, в котором содержится в том числе информация об отправителе сообщения. После того, как всё сообщение будет нами обработано, мы заканчиваем чтение.

Теперь, когда мы представили данные обмена в виде сообщения и получили его заголовок, можно произвести одну проверку перед тем, как начать собственно обрабатывать данные.

```
// Загрузить из найденного файла
// *** Инфраструктура сообщений
ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

// Читать заголовок сообщения обмена данными – файла XML
ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

// Сообщение предназначено не для этого узла
Если ЧтениеСообщения.Отправитель <> Ссылка Тогда
ВызватьИсключение "Неверный узел";
КонецЕсли;

ЧтениеСообщения.ЗакончитьЧтение();

ЧтениеXML.Закрыть();

УдалитьФайлы(ИмяФайла);
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "----- Конец загрузки -----";
Сообщение.Сообщить();

КонецПроцедуры
```

Мы проверяем, является ли отправитель сообщения тем узлом, для которого мы в данном вызове этой процедуры производим обмен данными.

Если все в порядке, то перед тем как начать чтение данных, следует удалить все записи регистрации изменений, которые были сделаны для этого узла и соответствовали номерам сообщений как номер принятого. Это делается затем, чтобы исключить дублирование данных, которые уже были ранее посланы этому узлу и им обработаны.

```
// Сообщение предназначено не для этого узла
Если ЧтениеСообщения.Отправитель <> Ссылка Тогда ВызватьИсключение
"Неверный узел";
КонецЕсли;

// Удаляем регистрацию изменений для узла отправителя сообщения.
// *** Служба регистрации изменений
ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправ
итель,ЧтениеСообщения.НомерПринятого);
```

```
ЧтениеСообщения.ЗакончитьЧтение();
```

Здесь мы обращаемся к службе регистрации изменений и используем метод **УдалитьРегистрациюИзменений()** для выполнения описанных действий.

Теперь можем приступить к непосредственно чтению самих данных, содержащихся в сообщении.

```
// Сообщение предназначено не для этого узла
Если ЧтениеСообщения.Отправитель <> Ссылка Тогда ВызватьИсключение
"Неверный узел";
КонецЕсли;
// Удаляем регистрацию изменений для узла отправителя сообщения.
// *** Служба регистрации изменений
ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,Чт
ениеСообщения.НомерПринятого);
// Читаем данные из сообщения *** XML-сериализация
Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл

КонецЦикла;

ЧтениеСообщения.ЗакончитьЧтение();

ЧтениеXML.Закрыть();
```

Чтение данных выполняется в цикле, причем мы снова обращаемся к механизмам XML-сериализации и методом глобального контекста **ВозможностьЧтенияXML()** получаем очередной тип данных XML из объекта **ЧтениеXML** и определяем, имеется ли соответствующий тип 1С:Предприятия. В случае успеха выполнение цикла продолжается.

Первое, что нам нужно сделать – представить данные XML в виде некоторого значения, имеющего тип 1С:Предприятия. Для этого мы используем метод глобального контекста **ПрочитатьXML()**.

```
// Читаем данные из сообщения *** XML-сериализация
Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл
// Читаем очередное значение
Данные = ПрочитатьXML(ЧтениеXML);

КонецЦикла;
```

В результате выполнения этого метода переменная **Данные** будет содержать объект 1С:Предприятия, соответствующий данным XML.

Теперь следует разрешить возможную коллизию.

```

    // Читаем данные из сообщения *** XML-сериализация
    Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл
    // Читаем очередное значение
    Данные = ПрочитатьXML(ЧтениеXML);

    // Не переносим изменение, полученное в главный из неглавного, если
    есть регистрация изменения
    Если Не ЧтениеСообщения.Отправитель.Главный И

        ПланыОбмена.ИзменениеЗарегистрировано(ЧтениеСообщения.Отправит
        ель, Данные) Тогда
            Сообщение = Новый СообщениеПользователю;
            Сообщение.Текст = " – Изменения отклонены";
            Сообщение.Сообщить();

        Продолжить;
        КонецЕсли;

    КонецЦикла;

```

Возможная коллизия решается следующим образом: мы проверяем, является ли узел-отправитель главным узлом и есть ли записи об изменении этого объекта для данного узла в нашей БД. Если объект изменялся в нашей базе и отправитель не является главным узлом, мы отклоняем запись полученного объекта. Во всех остальных случаях мы принимаем изменения полученного объекта.

Теперь единственное, что нам осталось сделать – записать полученные данные.

```

    // Не переносим изменение, полученное в главный из неглавного, если есть
    регистрация изменения
    Если Не ЧтениеСообщения.Отправитель.Главный И

        ПланыОбмена.ИзменениеЗарегистрировано(ЧтениеСообщения.Отправитель,
        Данные) Тогда
            Сообщение = Новый СообщениеПользователю;
            Сообщение.Текст = " – Изменения отклонены";
            Сообщение.Сообщить();

        Продолжить;
        КонецЕсли;

    // Записать полученные данные
    Данные.ОбменДанными.Отправитель =
    ЧтениеСообщения.Отправитель;
    Данные.ОбменДанными.Загрузка = Истина;
    Данные.Записать();

    КонецЦикла;

```

Перед записью полученного объекта мы устанавливаем у него в параметрах обмена данными узел отправителя для того, чтобы система

при записи этого объекта в нашей базе данных не формировала записи регистрации изменений этого объекта для того узла, от которого мы его только что получили.

Кроме этого, в параметрах обмена мы устанавливаем свойство Загрузка, информирующее систему о том, что запись объекта будет происходить в режиме обновления данных, полученных в результате обмена. Такое указание позволяет системе упростить процедуру записи объекта, отказавшись от ряда стандартных проверок и исключив изменения связанных данных, которые выполняются при обычной записи.

На этом создание процедуры получения и обработки данных обмена закончено. Полностью процедура будет выглядеть так:

Процедура ПрочитатьСообщениеСИИзменениями() Экспорт

```
    Каталог = КаталогВременныхФайлов();

    // Сформировать имя файла
    ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "\", "\") +
    "Message" + СокрЛП(Ссылка.Код) + "_" +
    СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) + ".xml";
    Файл = Новый Файл(ИмяФайла);
    Если Не Файл.Существует() Тогда
        Возврат;
    КонецЕсли;
    // *** Чтение документов XML
    // Попытаться открыть файл
    ЧтениеXML = Новый ЧтениеXML;
    Попытка
        ЧтениеXML.ОткрытьФайл(ИмяФайла);
    Исклучение
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "Невозможно открыть файл обмена данными.";
        Сообщение.Сообщить();
    Возврат;
    КонецПопытки;
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "----- Загрузка из " + Строка(ЭтотОбъект) + " -----";
    Сообщение.Сообщить();
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = " - Считывается файл " + ИмяФайла;
    Сообщение.Сообщить();

    // Загрузить из найденного файла
    // *** Инфраструктура сообщений
    ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

    // Читать заголовок сообщения обмена данными – файла XML
    ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

    // Сообщение предназначено не для этого узла
```

```

        Если ЧтениеСообщения.Отправитель <> Ссылка Тогда ВызватьИсключение
"Неверный узел";
        КонецЕсли;
        // Удаляем регистрацию изменений для узла отправителя сообщения.
        // *** Служба регистрации изменений
        ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,Чт
ениеСообщения.НомерПринятого);
        // Читаем данные из сообщения *** XML-сериализация
        Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл
        // Читаем очередное значение
        Данные = ПрочитатьXML(ЧтениеXML);

        // Не переносим изменение, полученное в главный из неглавного, если есть
регистрация изменения
        Если Не ЧтениеСообщения.Отправитель.Главный И

        ПланыОбмена.ИзменениеЗарегистрировано(ЧтениеСообщения.Отправитель,
Данные) Тогда
                Сообщение = Новый СообщениеПользователю;
                Сообщение.Текст = " – Изменения отклонены";
                Сообщение.Сообщить();
        Продолжить;
        КонецЕсли;

// Записать полученные данные
        Данные.ОбменДанными.Отправитель = ЧтениеСообщения.Отправитель;
        Данные.ОбменДанными.Загрузка = Истина;
        Данные.Записать();

        КонецЦикла;

        ЧтениеСообщения.ЗакончитьЧтение();

        ЧтениеXML.Закрыть();

        УдалитьФайлы(ИмяФайла);
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "----- Конец загрузки -----";
        Сообщение.Сообщить();

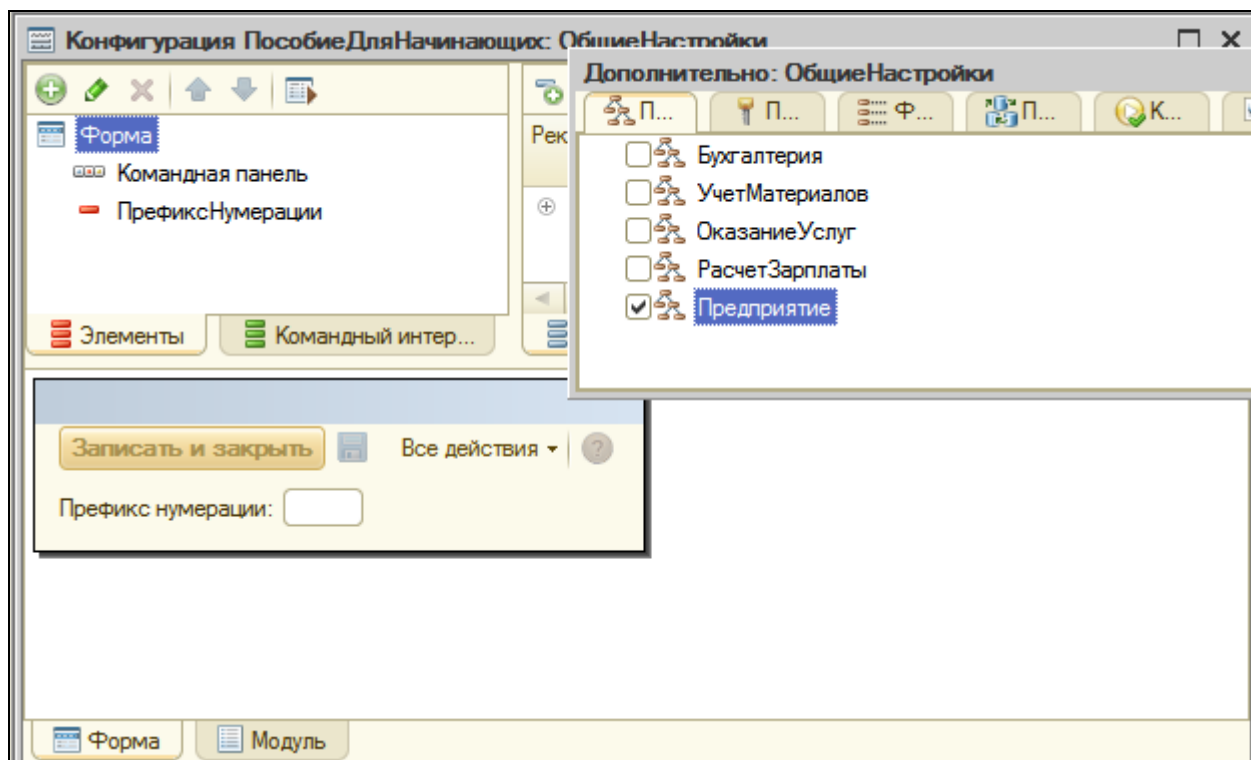
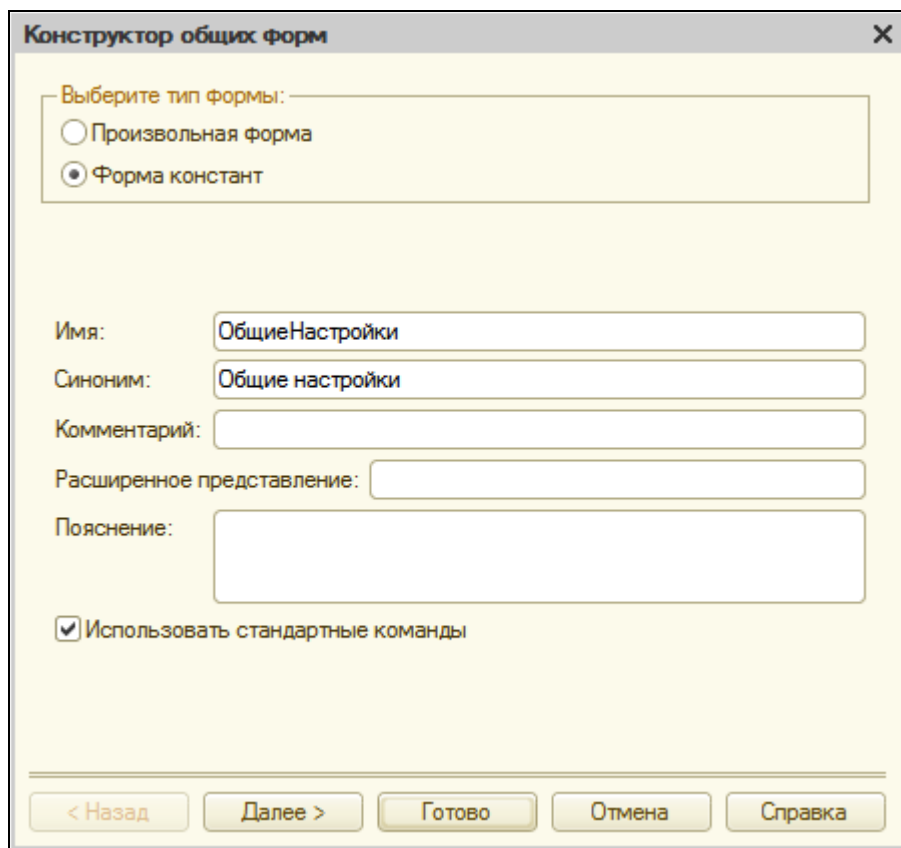
КонецПроцедуры

```

## Проверка работы обмена данными

Чтобы иметь возможность редактировать константу **ПрефиксНумерации**, раскроем ветвь **Общие**, выделим ветвь **Общие формы** и с помощью конструктора форм создадим форму констант с именем **ОбщиеНастройки**.

Откроем окно свойств **Дополнительно** для этой формы (контекстное меню – **Дополнительно**) и укажем принадлежность этой формы к подсистеме **Предприятие**.

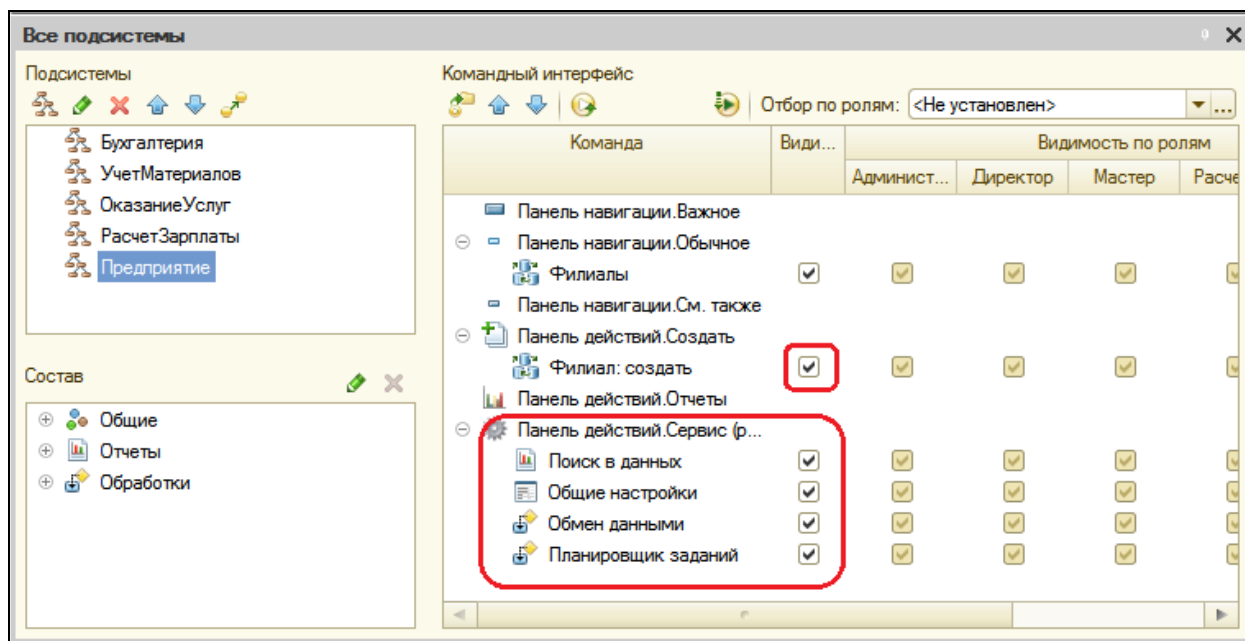


В окне редактирования плана обмена **Филиалы** и обработки **ОбменДанными** на закладке **Подсистемы** также укажем их принадлежность к подсистеме **Предприятие**.

Т.о. доступ к командам открытия плана обмена, обработки, а также формы констант будет иметь только Администратор, т.к. подсистема Предприятие будет доступна только для роли **Администратор**.

В окне редактора командного интерфейса подсистем **Все подсистемы** включим видимость у команды **Филиал: создать** в группе панели действий **Создать** подсистемы **Предприятие** и установим следующий порядок следования команд в группе панели действий **Сервис**:

- **Поиск в данных,**
- **Общие настройки,**
- **Обмен данными,**
- **Планировщик заданий.**



И в заключение создадим новый каталог на жестком диске, в котором будет размещаться база нашего филиала.

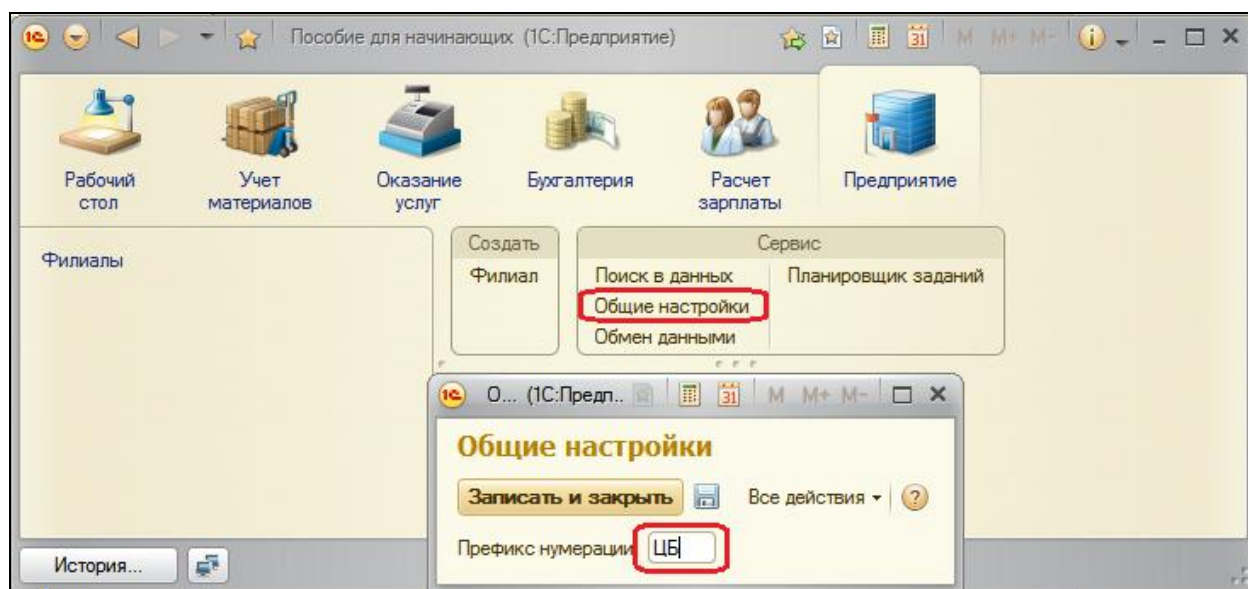
Обновим конфигурацию базы данных (F7). Затем сохраним в созданный каталог нашу конфигурацию, выполнив команду главного меню **Конфигурация – Сохранить конфигурацию в файл...**

## В режиме 1С:Предприятие

Запустим режим отладки и установим необходимые значения в нашей центральной базе.

Прежде всего, зададим значение константы **Префикс нумерации – ЦБ**.

Для этого выполним команду **Общие настройки** в панели действий раздела **Предприятие**. Нажмем **Записать и закрыть**.



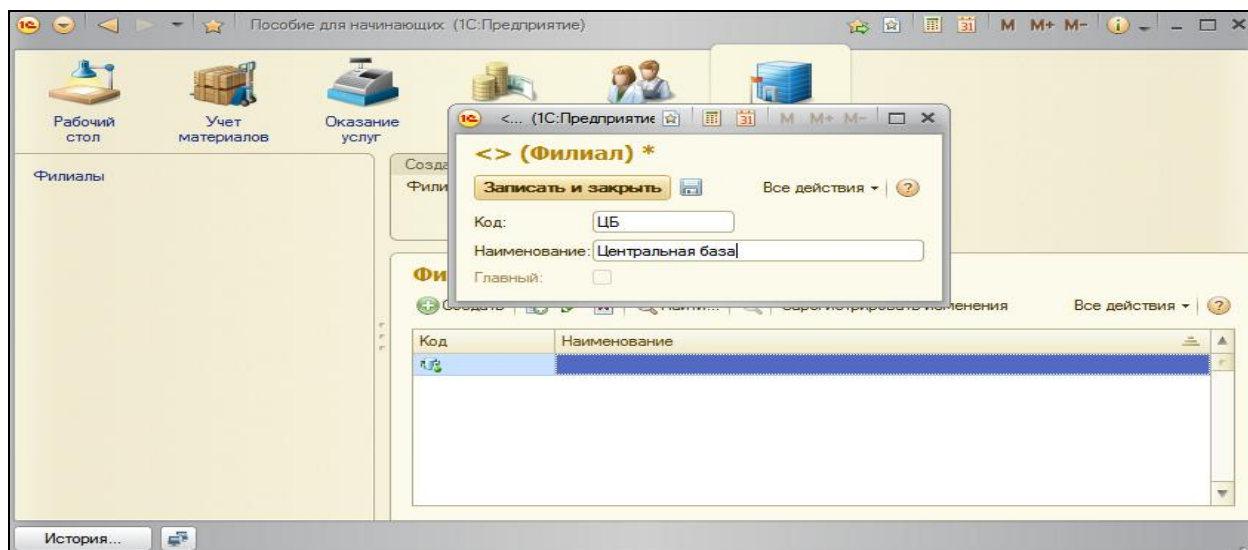
После этого откроем план обмена **Филиалы** и зададим параметры узла по умолчанию, т.е. параметры нашей базы.

Для этого выполним команду **Филиалы** в панели навигации раздела **Предприятие**. В списке планов обмена уже присутствует одна запись. Откроем и отредактируем ее.

Код базы будет **ЦБ**, а наименование – **Центральная база**.

Не забудьте, что именно код идентифицирует узлы обмена в различных базах, поэтому в базе филиала мы будем создавать узлы с такими же кодами.

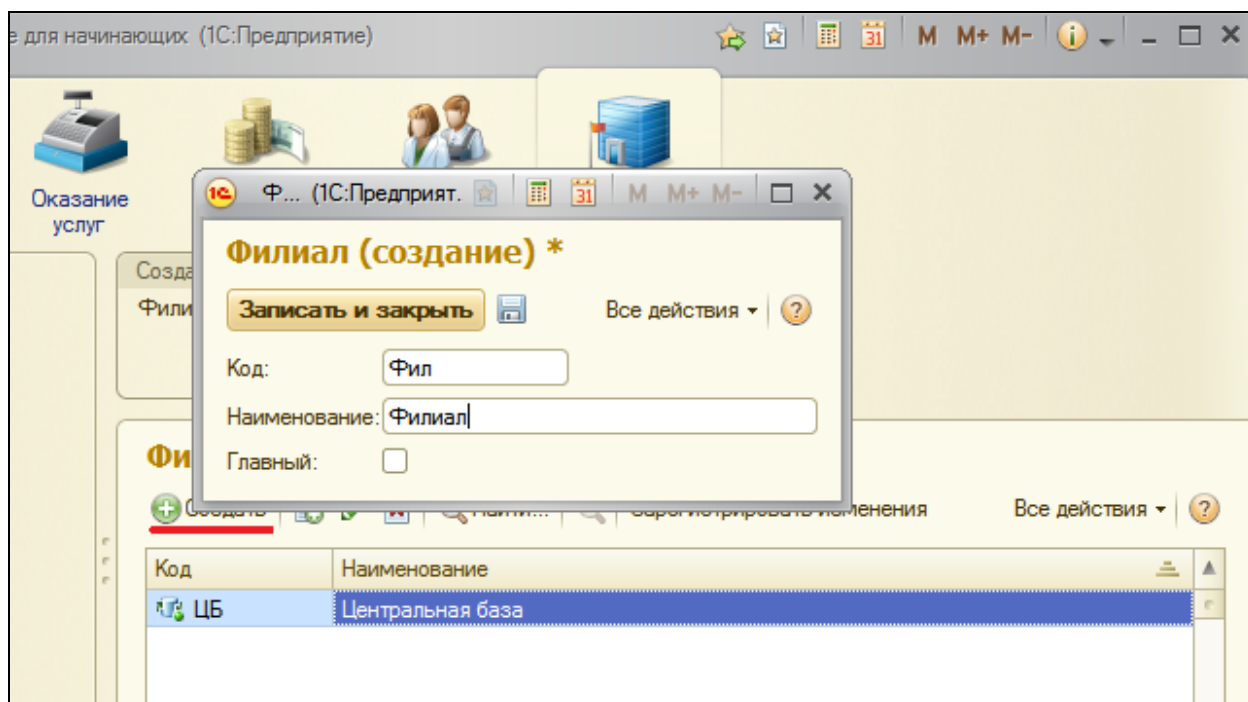




Нажмите **Записать и закрыть**.

Затем нажмем кнопку **Создать** или команду **Филиал** в панели действий.

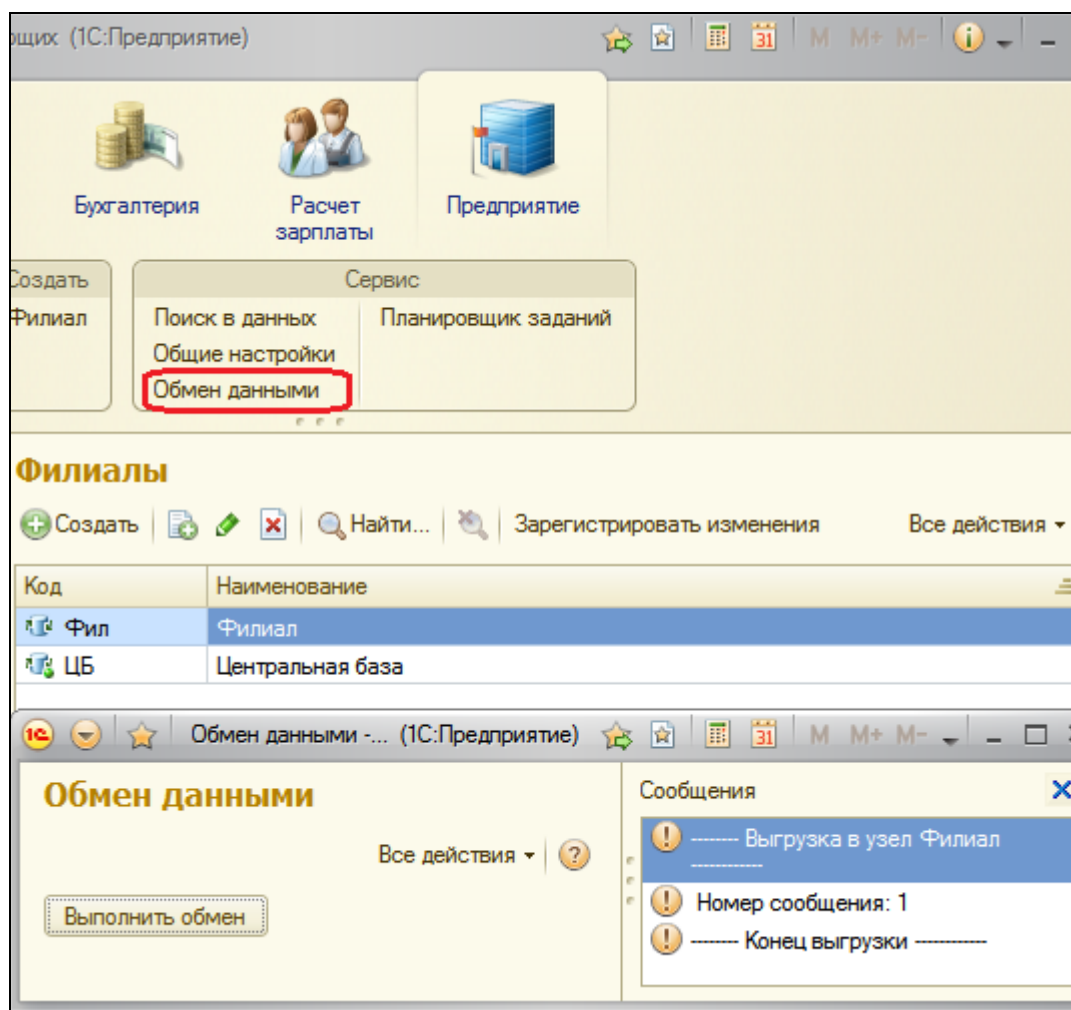
Создадим новый узел, который будет соответствовать базе филиала, присвоим ему код **Фил** и наименование **Филиал**.



Обратите внимание, что predetermined узел нашей информационной базы (**Центральная база**) выделен в списке узлов обмена специальной пиктограммой. Кнопка **Зарегистрировать изменения** недоступна для этого узла.

Выделим в списке новый узел **Филиал** и нажмем кнопку **Зарегистрировать изменения**.

Теперь вызовем обработку **ОбменДанными** и нажмем **Выполнить обмен**. В окне сообщения появится следующий текст.



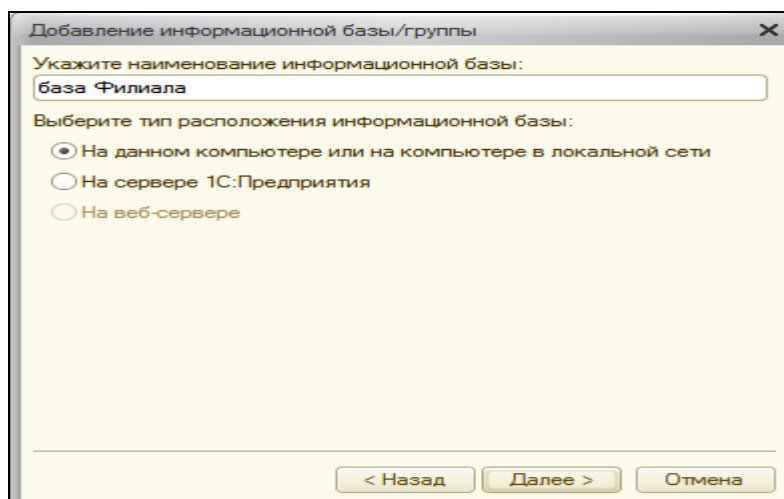
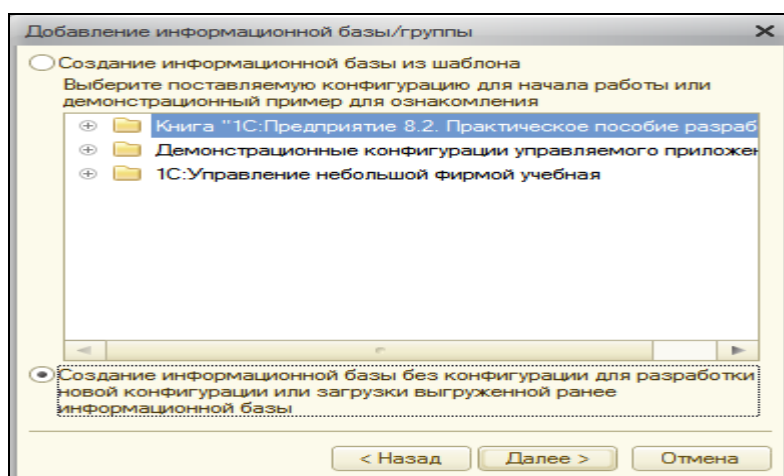
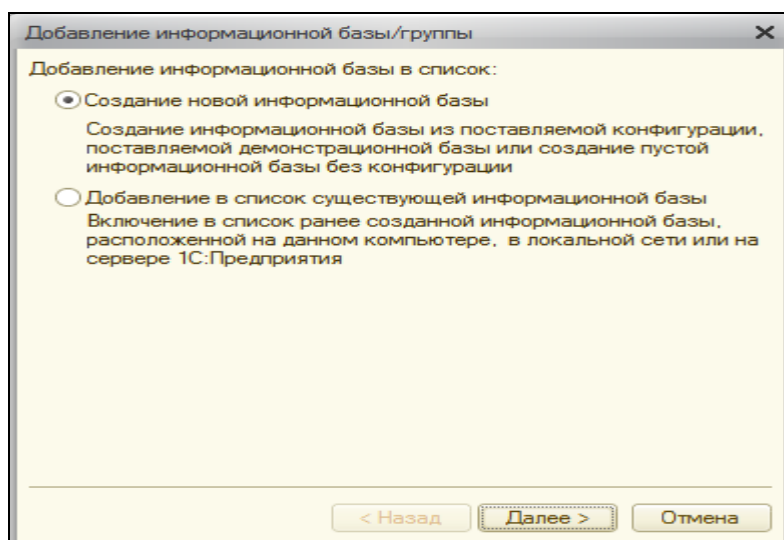
Таким образом, в результате обмена данными центральная база сформировала файл обмена, содержащий изменения всех данных, которыми она обменивается с филиалом.

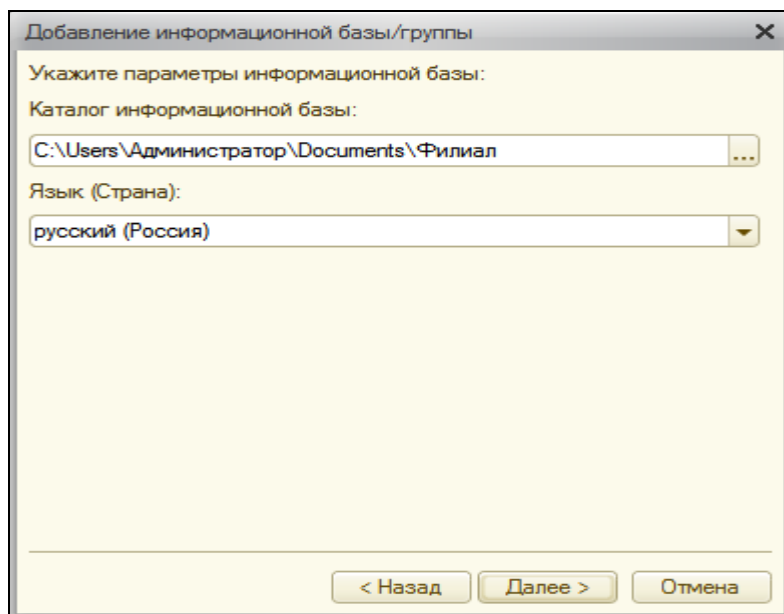
### Запуск базы филиала

Настало время перейти к базе филиала.

Запустим 1С:Предприятие и добавим в список баз новую базу с пустой конфигурацией, которая будет расположена в созданном нами каталоге базы филиала. Для этого в окне запуска 1С:Предприятия нажмем кнопку **Добавить** и выберем **Создание новой информационной базы**. Нажмем **Далее**. Выберем **Создание информационной базы без конфигурации для... загрузки выгруженной ранее информационной базы**. Нажмем **Далее** и укажем наименование информационной базы, например, **база Филиала**. Нажмем **Далее** и укажем каталог информационной базы, где находится сохраненная

конфигурация, например C:\Users\Администратор\Documents\Филиал.  
Нажмем **Далее** и **Готово**.





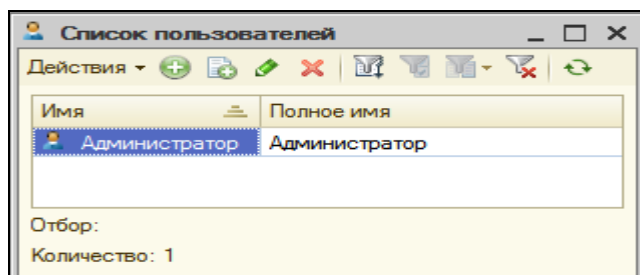
## В режиме Конфигуратор

Откроем созданную нами конфигурацию **база Филиала** в режиме Конфигуратор. Выполним команду главного меню **Конфигурация – Открыть конфигурацию**. Мы видим, что список объектов конфигурации пуст.

Теперь загрузим конфигурацию из файла (**Конфигурация – загрузить конфигурацию из файла...**).

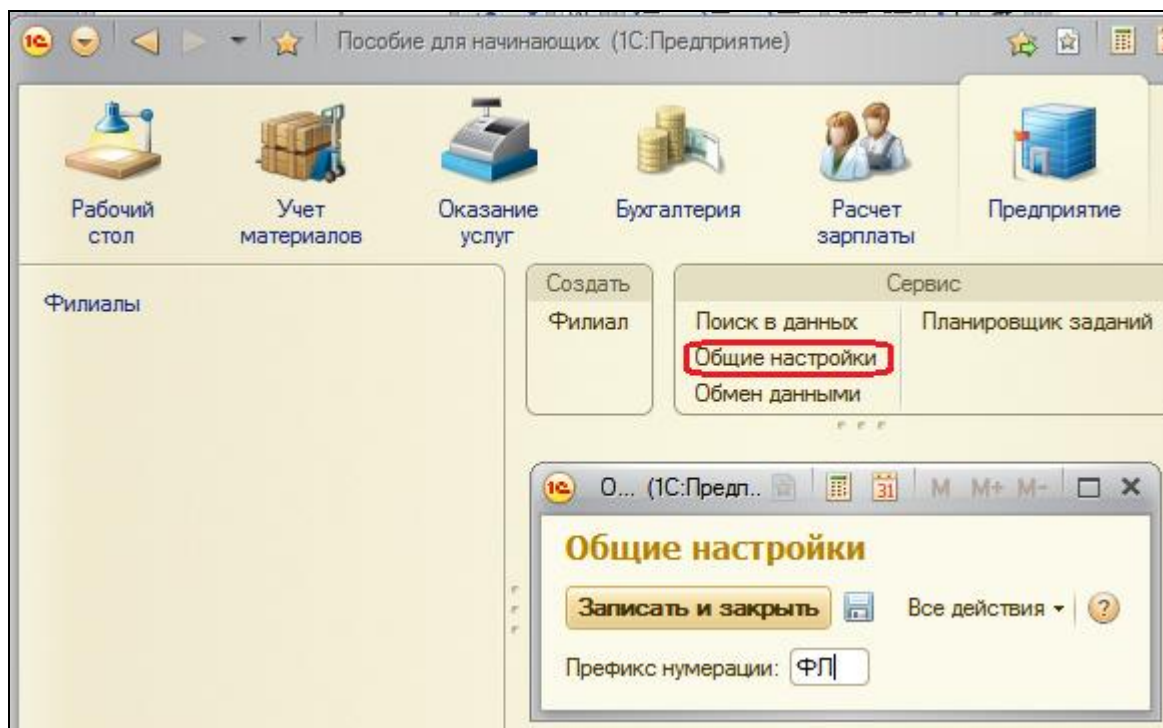
В окне выбора файла выберем каталог и имя файла, где находится сохраненная конфигурация. На вопрос обновления конфигурации ответим утвердительно и в окне изменений структуры конфигурации нажмем **Принять**.

Теперь все объекты конфигурации перенесены из нашей центральной базы. Выполним команду главного меню **Администрирование – Пользователи** и создадим в конфигурации филиала одного пользователя – **Администратор** с одноименной ролью. Дело в том, что пользователей для каждой информационной базы нужно создавать заново.



## В режиме 1С:Предприятие

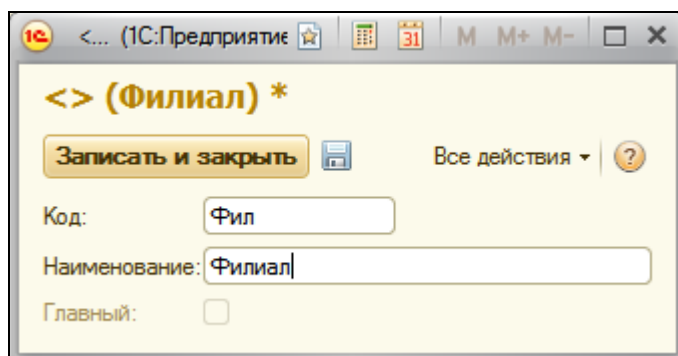
Запустим режим отладки под администратором. Первым делом зададим значение константы **ПрефиксНумерации** – **ФЛ**.



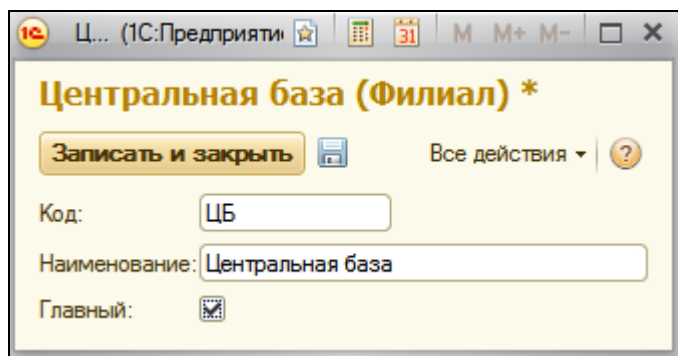
Затем откроем план обмена **Филиал** и опишем predetermined узел (узел текущей информационной базы).

Для этого выполним команду **Филиалы** в панели навигации раздела **Предприятие**.

В списке планов обмена уже присутствует запись. Откроем и отредактируем ее. Зададим код **Фил** и наименование **Филиал**.



После этого создадим новый узел плана обмена с кодом **ЦБ**, наименованием **Центральная база** и признаком **Главный**.

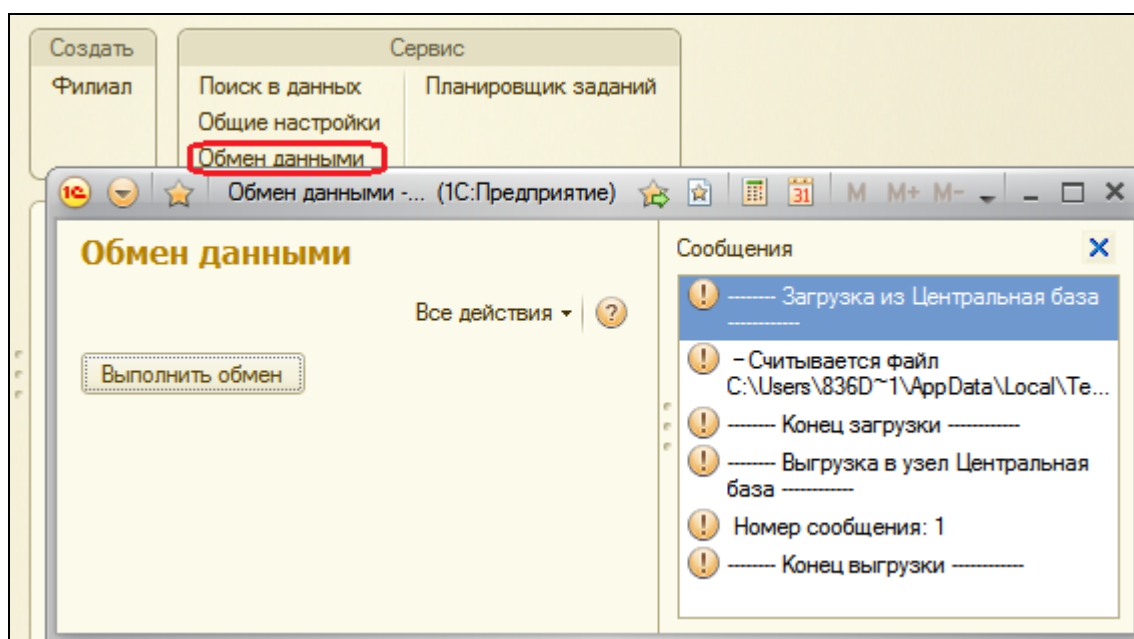


Выделим в списке узлов обмена новый узел **Центральная база** и нажмем кнопку **Зарегистрировать изменения**.

Теперь для большей наглядности откроем список справочника **Клиенты**. Сейчас в нем нет ни одного элемента.

Запустим обработку **Обмен данными** и нажмем **Выполнить обмен**.

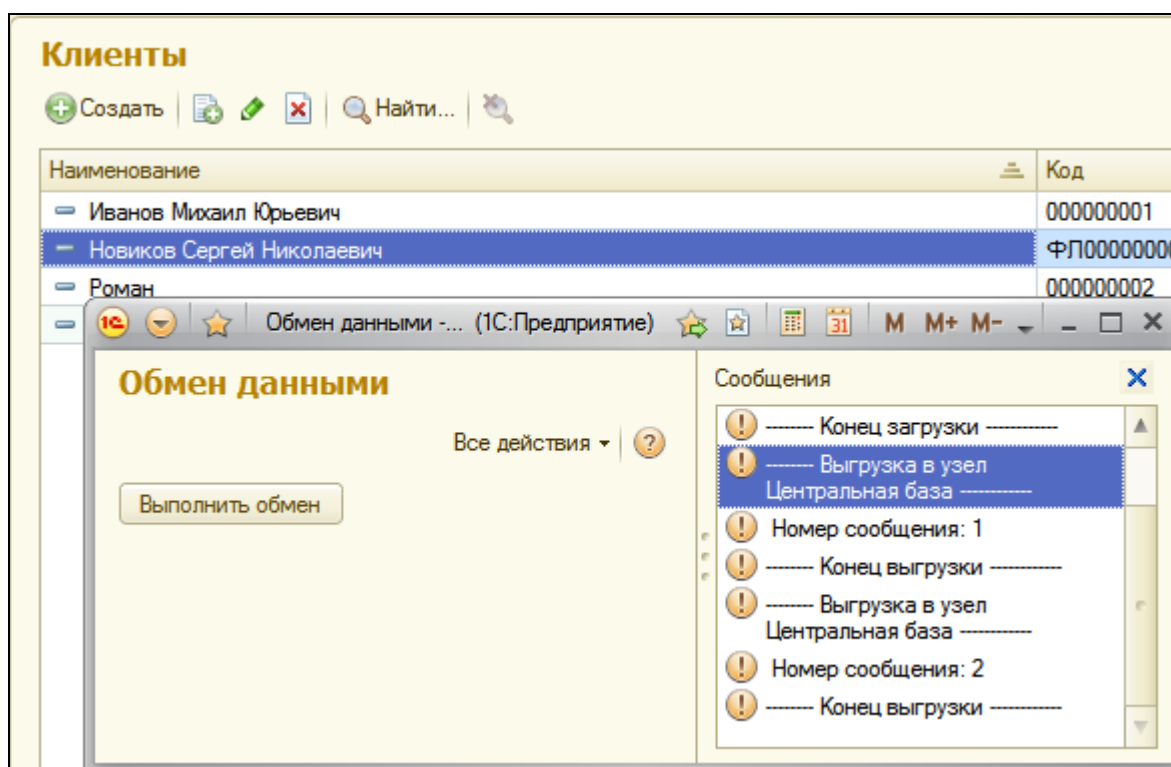
Справочник будет заполнен элементами, а в конце сообщений появится текст.



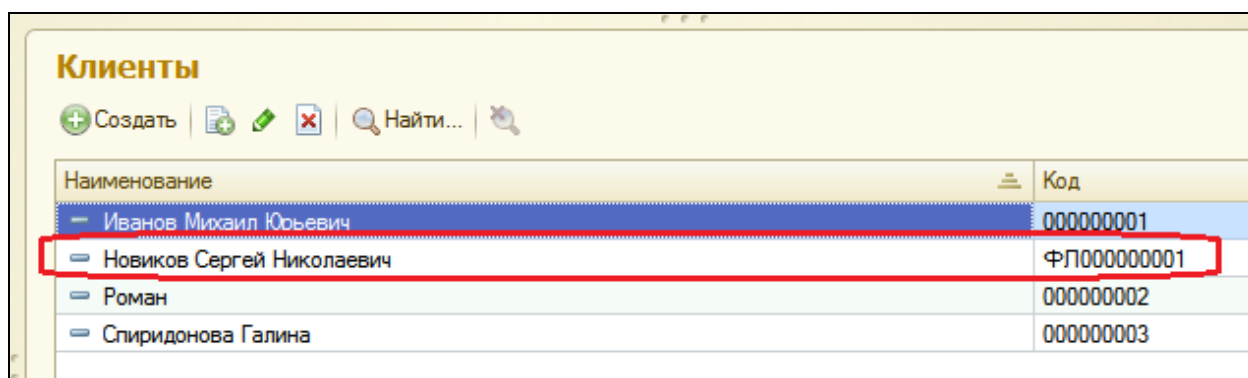
Теперь проверим, как будет происходить обмен в другую сторону.

Создадим в справочнике **Клиенты** нового клиента с произвольным наименованием. Обратите внимание, что нумерация кода нового клиента начинается с единицы и имеет префикс ФЛ.

После этого снова нажмем **Выполнить обмен** в открытой форме обработки **ОбменДанными**.



Затем перейдем в центральную базу, также выполним обмен данными и убедимся, что клиент, созданный в базе филиала, перенесен в центральную базу.



## Механизм распределенных информационных баз

Является развитием универсального механизма обмена данными.

Он реализует модель распределенной информационной базы, которая подразумевает наличие идентичных конфигураций во всех узлах, имеет древовидную структуру и позволяет выполнять обмен как измененными данными, так и изменениями, внесенными в конфигурацию.

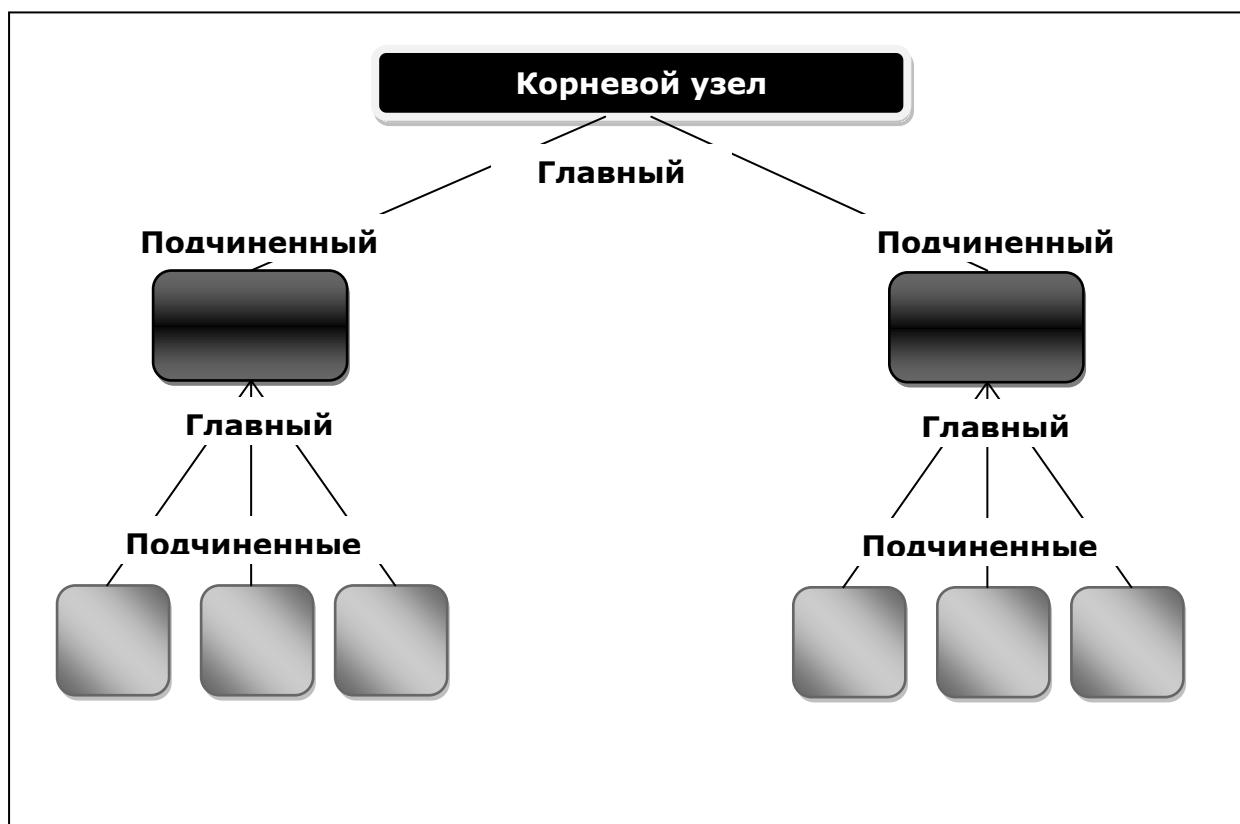
Механизм распределенных информационных баз реализуется планами обмена. Для этого план обмена содержит свойство **Распределенная информационная база**.

Если это свойство установлено, для данного плана обмена включается механизм распределенных информационных баз и разработчик получает возможность создать распределенную базу исключительно интерактивными средствами, без написания кода.

Такая возможность не исключает программного управления обменом, которое также доступно при работе с распределенными информационными базами. В ходе создания примера мы рассмотрим оба варианта организации обмена в распределенных информационных базах.

### Основные сведения

Распределенная база должна иметь четко определенную древовидную структуру. Количество уровней в такой структуре не ограничено, главное – между двумя связанными узлами всегда должно быть определено отношение «главный-подчиненный».





Все узлы, кроме одного, должны иметь по одному главному узлу, и один узел не будет иметь главного узла – это корневой узел. Такое жесткое задание структуры узлов необходимо для определения порядка миграции изменений данных и изменений конфигурации.

Конфигурация может быть изменена только в узле, не имеющем главного узла (в корневом). Изменения данных могут выполняться в любом узле.

Изменения конфигурации будут передаваться от главного к подчиненным узлам. Изменения данных могут передаваться между любыми связанными узлами.

Разрешение коллизий также будет производиться исходя из отношения «главный-подчиненный». Если изменения выполнены одновременно и в главном, и в подчиненном узле, при обмене данными будут приняты только изменения главного узла, а изменения подчиненного отвергнуты.

Для любого подчиненного узла возможно создание *начального образа* – информационной базы, созданной на основании конфигурации и данных главного узла в соответствии с правилами, определяемыми планом обмена. Процедура создания начального образа узла может выполняться неоднократно, при этом удаляются все записи изменений в базе главного узла для подчиненного узла. Сразу после создания начальный образ готов к обмену с главным узлом.

### **Постановка задачи**

В качестве примера мы покажем использование механизма распределенных баз в создании нескольких отделений нашей фирмы.

В отличие от филиалов, которые расположены в других городах, являются отдельными юридическими лицами и довольно самостоятельны в плане организации учета деятельности, отделения нашей фирмы расположены в этом же городе, никакой юридической самостоятельностью не обладают и ведут учет в точности так, как это организовано в главном офисе.

Поэтому все они используют ту же конфигурацию, что и главный офис, причем если главный офис вносит какие-либо изменения в свою конфигурацию, они должны быть своевременно внесены в конфигурацию отделения.

## Интерактивный обмен

Для построения распределенной информационной базы нам понадобится создать еще один план обмена с именем **Отделения**, представлением объекта – **Отделение**.

Для этого плана установим свойство **Распределенная информационная база**.

The screenshot shows a software window titled 'План обмена Отделения'. On the left is a sidebar with a tree view containing: 'Основные' (selected), 'Подсистемы', 'Функциональные опции', 'Данные', 'Формы', 'Команды', 'Макеты', 'Ввод на основании', 'Права', and 'Прочее'. The main area contains the following fields and controls:

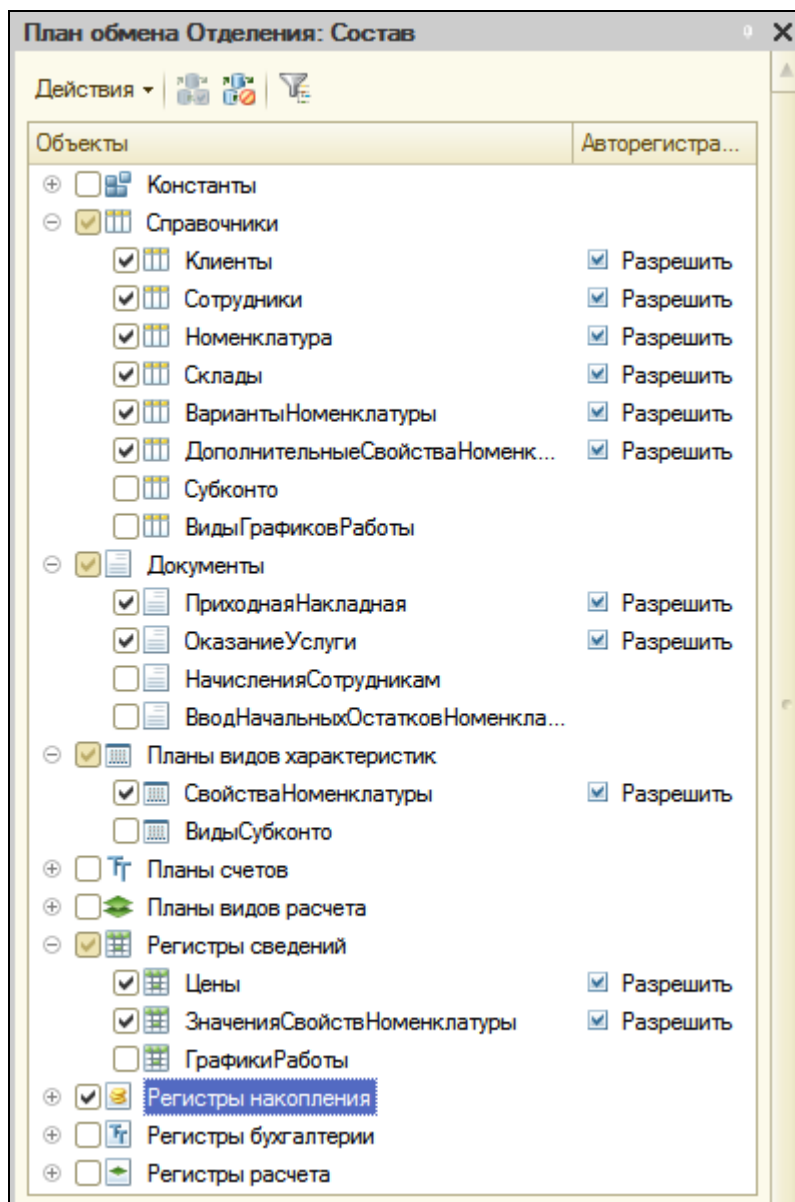
- 'Имя:' text box with 'Отделения' entered.
- 'Синоним:' text box with 'Отделения' entered.
- 'Комментарий:' empty text box.
- 'Распределенная информационная база' checkbox, which is checked and highlighted with a red square.
- 'Состав' button, highlighted with a red rectangle.
- 'Представление объекта:' text box with 'Отделение' entered, highlighted with a red rectangle.
- 'Расширенное представление объекта:' empty text box.
- 'Представление списка:' empty text box.
- 'Расширенное представление списка:' empty text box.
- 'Пояснение:' empty text area.

At the bottom are five buttons: 'Действия' (with a dropdown arrow), '<Назад', 'Далее>', 'Заккрыть', and 'Справка'.

Нажмем кнопку **Состав** и определим тот же состав данных для обмена, что и в плане обмена **Филиалы**. Включим в обмен все объекты, не относящиеся к ведению бухгалтерии и расчету зарплаты.

На закладке **Формы** включим флажок **Быстрый выбор**, чтобы иметь возможность выбора узлов плана обмена из выпадающего списка.

На закладке **Подсистемы** укажем принадлежность плана обмена к подсистеме **Предприятие**. Т.о. команда открытия плана обмена будет доступна только для роли **Администратор**.



А также в окне редактора командного интерфейса подсистем **Все подсистемы** включим видимость у команды **Отделение: создать**.

В режиме 1С:Предприятие

Запустим режим отладки. Откроем план обмена Отделения и зададим параметры центрального узла (предопределенный элемент плана обмена).

Для этого выполним команду **Отделения** в панели навигации раздела **Предприятие**.

В списке планов обмена, как и раньше, присутствует одна запись. Это предопределенный узел нашей информационной базы. Откроем и отредактируем ее.

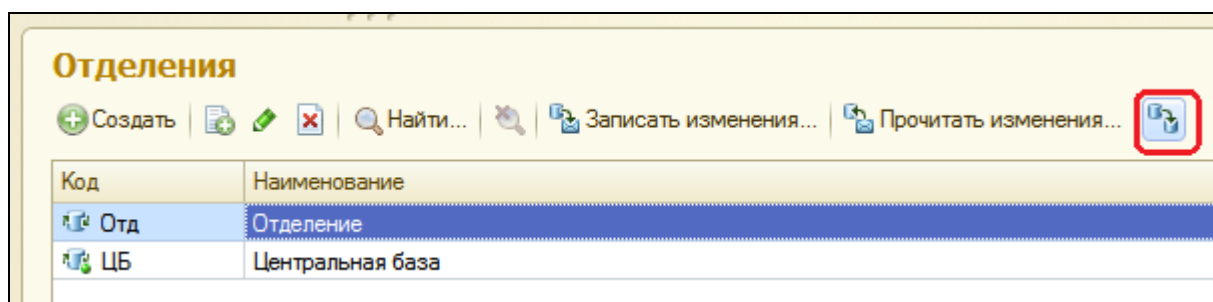
Внесем код **ЦБ** и наименование **Центральная база**.

После этого создадим новый узел с кодом **Отд** и наименованием **Отделение**.

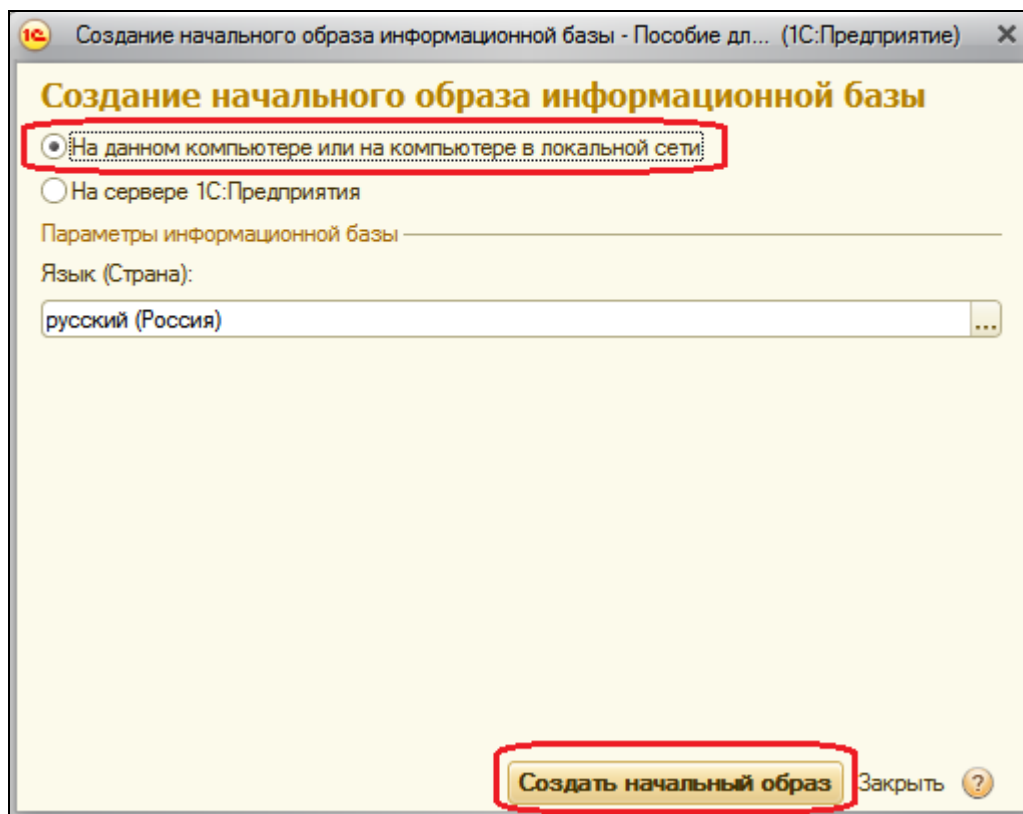
Для созданного узла доступны три кнопки в командной панели формы плана обмена: **Записать изменения**, **Прочитать изменения** и **Создать начальный образ**.

Создайте на диске новый каталог, в котором будет располагаться база отделения (Отделение).

Воспользуемся кнопкой, чтобы создать начальный образ информационной базы нашего отделения.



Укажите, что информационная база будет расположена на данном компьютере. Нажмите **Создать начальный образ**.



На следующем шаге укажите каталог информационной базы и нажмите **Готово**.

Система создаст в указанном каталоге начальный образ информационной базы нашего отделения.

### **Запуск базы отделения**

Теперь перейдем к базе отделения. Запустим 1С:Предприятие и добавим в список баз созданную нами базу, расположенную в каталоге, в который мы поместили начальный образ информационной базы нашего отделения.

Для этого в окне запуска 1С:Предприятия нажмем кнопку **Добавить** и выберем **Добавление существующей информационной базы**.

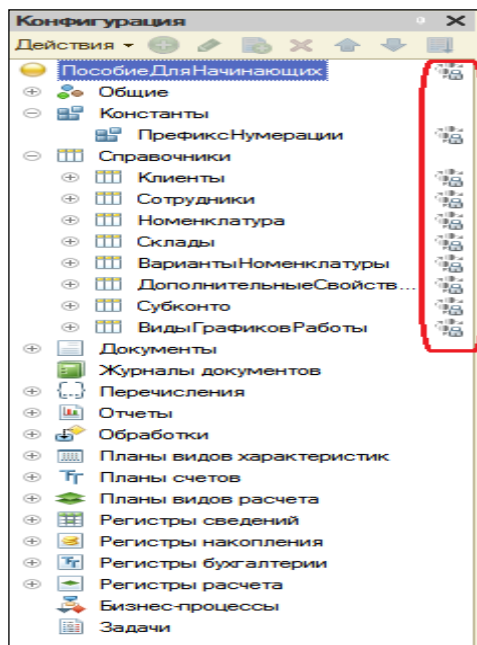
Нажмем **Далее**, затем укажем наименование информационной базы (**база Отделения**).

Нажмем **Далее**, укажем каталог информационной базы отделения, нажмем **Готово**.

### **В режиме Конфигуратор**

Откроем созданную нами конфигурацию **база Отделения** в режиме Конфигуратор.

Откройте конфигурацию. Мы видим, что конфигурация нашего отделения стала защищенной от изменений средствами управления распределенной информационной базой.



Выполним команду главного меню **Администрирование – Пользователи** и создадим нового пользователя **Администратор** с одноименной ролью.

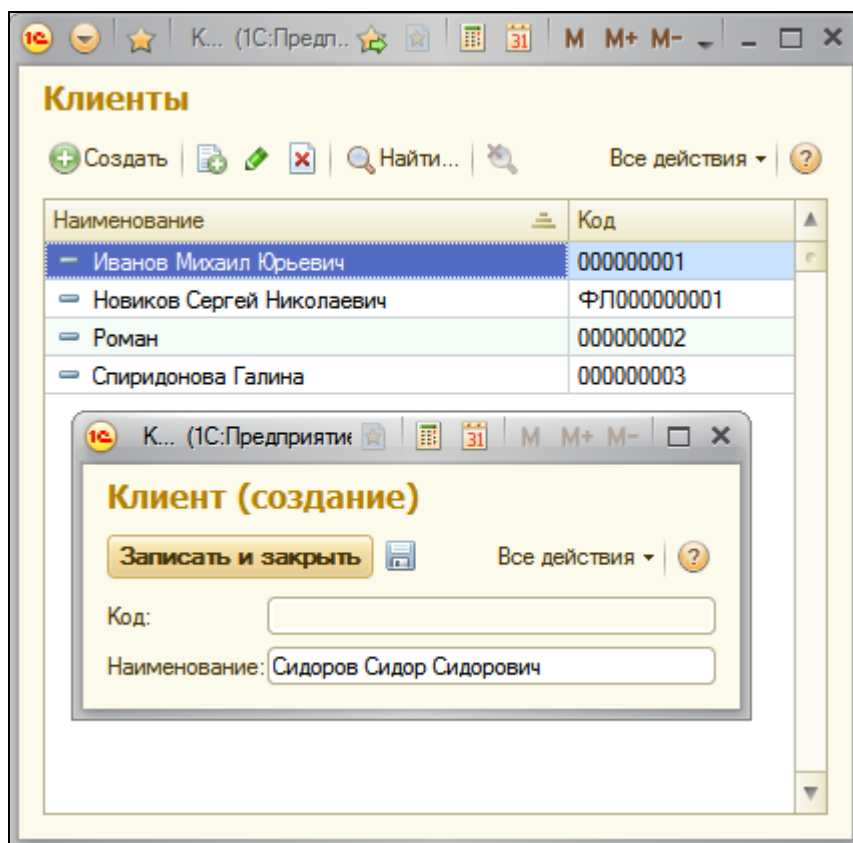
В режиме 1С:Предприятие

Запустим базу отделения в режиме отладки и откроем план обмена **Отделения**.

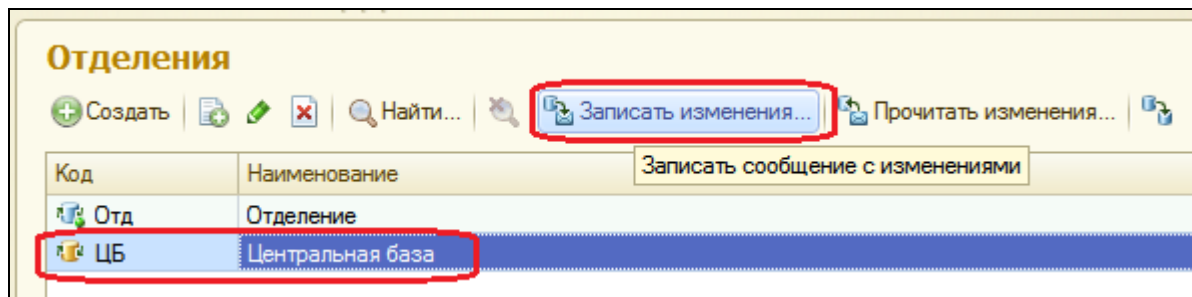
Теперь проверим работу обмена данными.

Откроем список констант (**Все функции – Константы**) и зададим значение константы **ПрефиксНумерации – ОТ**.

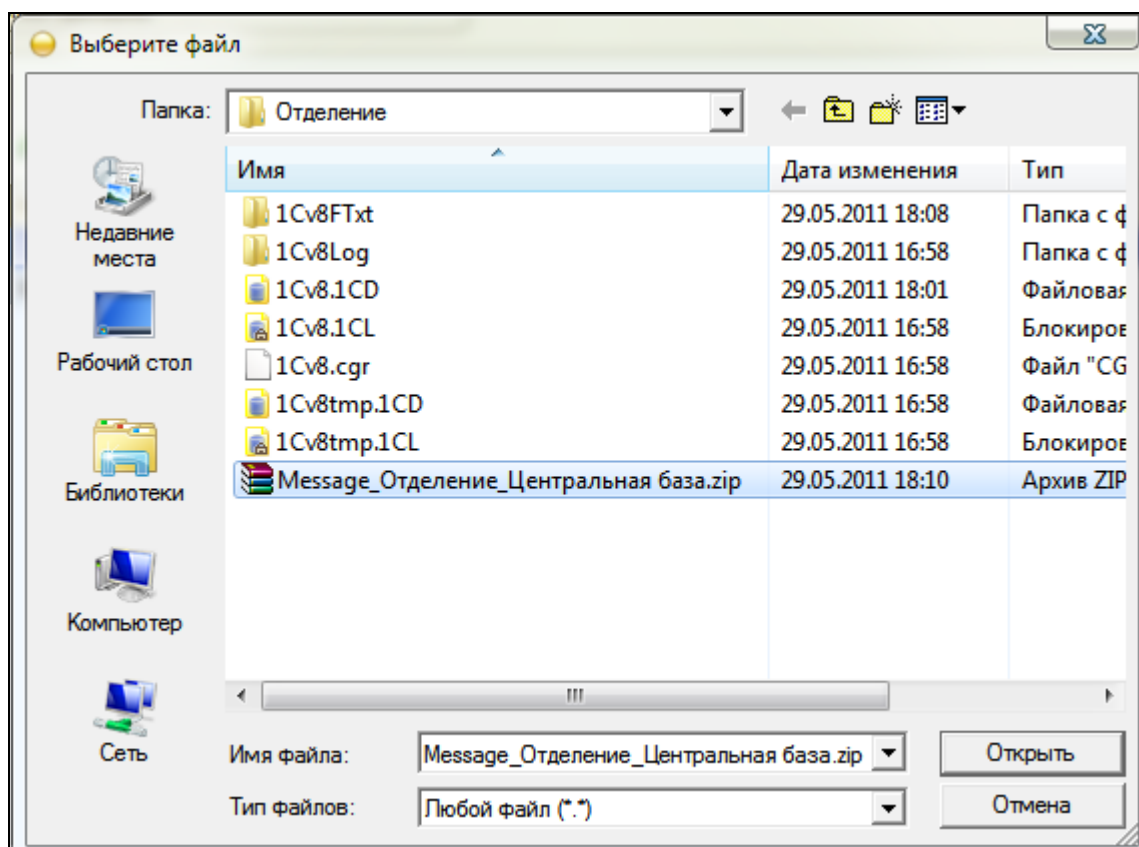
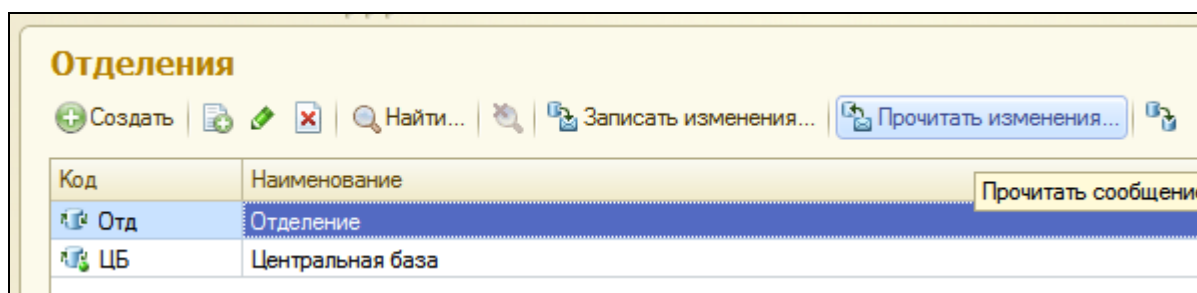
После этого откроем справочник клиентов и добавим в него нового клиента.



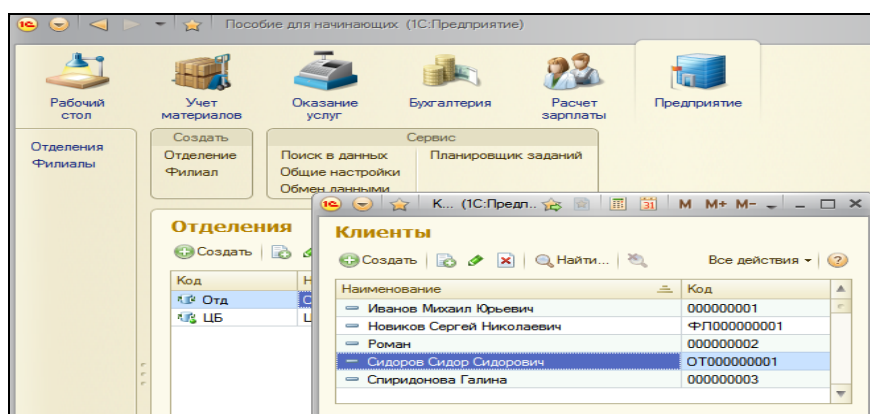
Затем выполним запись изменений для центральной базы (указав имя файла сообщения).



Перейдем в центральную базу и выполним чтение изменений для центральной базы (выбрав имя файла сообщения).



Убедимся, что новый клиент, созданный в базе отделения, присутствует и в центральной базе.

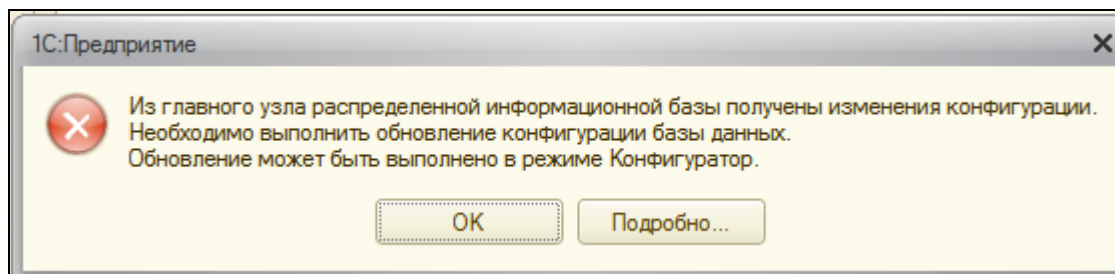


Теперь посмотрим, как будут переноситься изменения конфигурации между главным и подчиненными узлами.

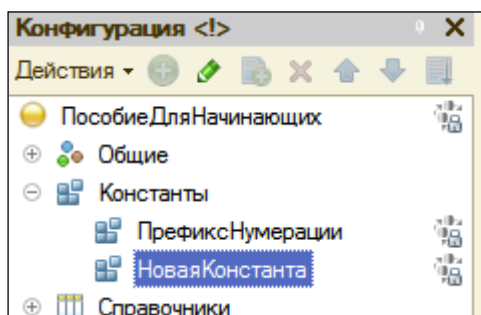
В конфигураторе центральной базы создадим новую константу с именем **НоваяКонстанта**.

Выполним обновление конфигурации и запустим режим отладки (проигнорируем сообщение о возможной ошибке про подсистемы).

Откроем план обмена **Отделения** и выполним запись изменений для подчиненного узла **Отделение**. После этого закроем конфигуратор информационной базы отделения, запустим эту базу в режиме 1С:Предприятие и выполним чтение изменений в базе подчиненного узла (**Центральная база**). По окончании чтения система выдаст следующее сообщение:



Нажмем ОК. Откроем конфигуратор базы отделения и увидим, что в основной конфигурации появилась новая константа **НоваяКонстанта**, т.е. изменения, внесенные в конфигурацию центральной базы, были автоматически перенесены в конфигурацию подчиненного узла.



Остается выполнить обновление конфигурации базы данных в подчиненном узле.

О порядке принятия изменений, когда в одном сообщении получены как изменения конфигурации, так и изменения данных.

В этом случае сначала будет изменена основная конфигурация и выдано сообщение о необходимости выполнения сохранения конфигурации. После объединения конфигураций следует выполнить повторное получение данных, при котором будут приняты уже изменения данных, содержащиеся в сообщении. Такой порядок принятия изменений не зависит от того, относятся измененные данные к существующим объектам конфигурации или к новым.



В заключение удалим объект **НоваяКонстанта** из дерева объектов нашей главной конфигурации (все равно не сможете удалить из подчиненной).

## Программный обмен

Все описанные выше действия по обмену данными в распределенной информационной базе можно выполнить программно.

Мы создадим обработку, которая будет программно выполнять для выбранного узла все те действия, которые были рассмотрены в предыдущем разделе.

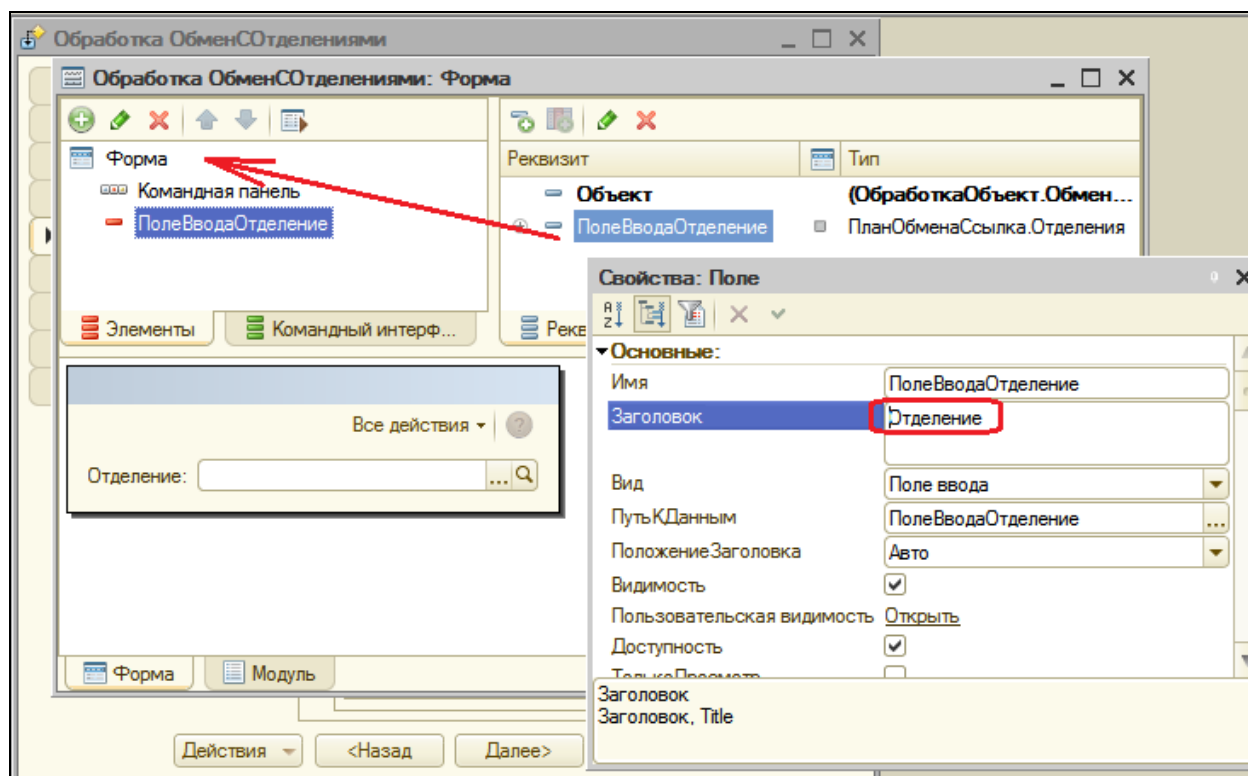
## В режиме Конфигуратор

В конфигураторе центральной базы создадим новую обработку с именем **ОбменСОтделениями**.

На закладке **Формы** создадим основную форму обработки.

В окне редактора форм на закладке **Реквизиты** добавим реквизит формы **ПолеВводаОтделение** с типом **ПланОбменаСсылка.Отделения** и перетащим его в окно элементов формы.

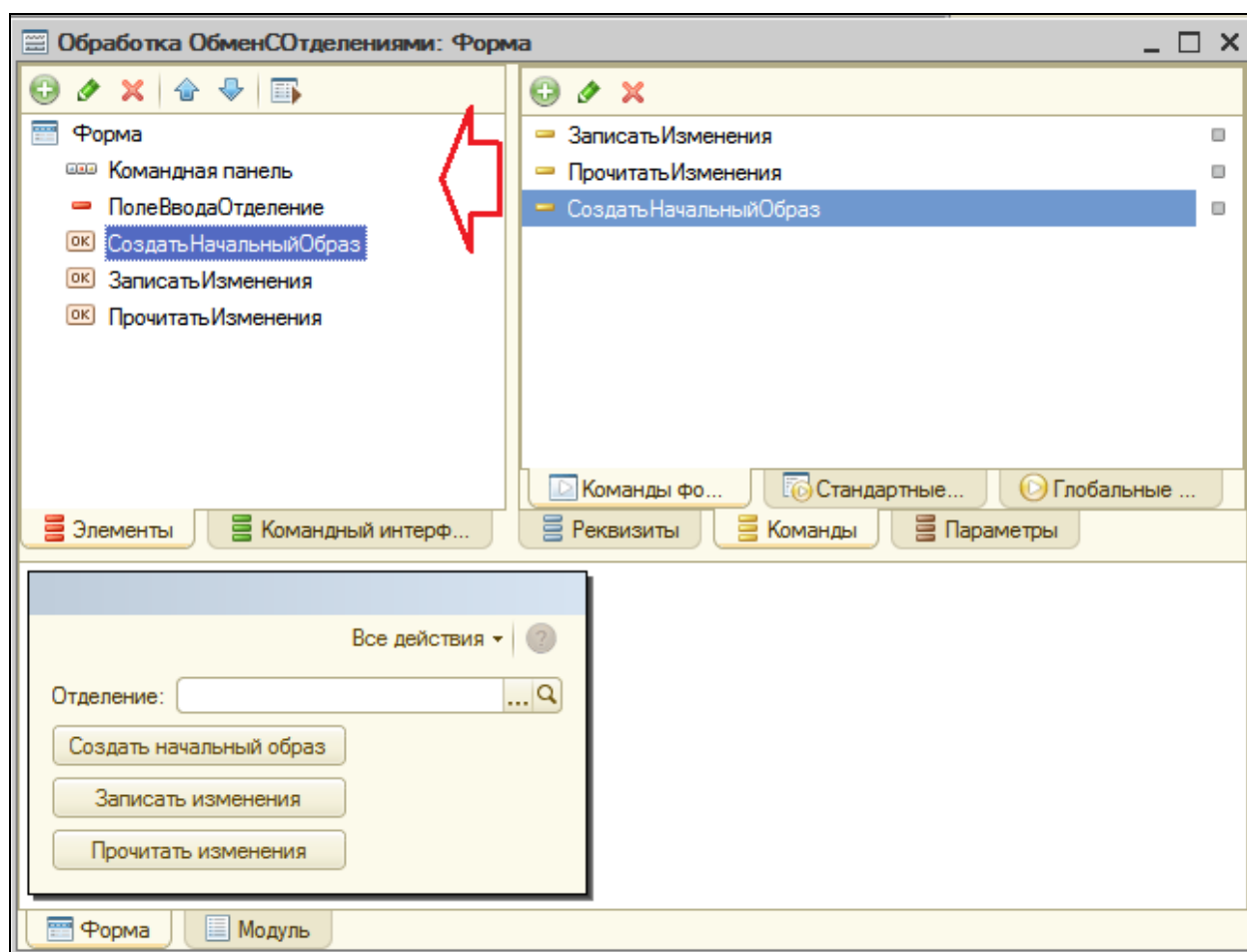
В открывшемся окне свойств этого поля зададим заголовок – **Отделение**.



На закладке **Команды** поочередно создадим команды **СоздатьНачальныйОбраз**, **ЗаписатьИзменения** и **ПрочитатьИзменения**.

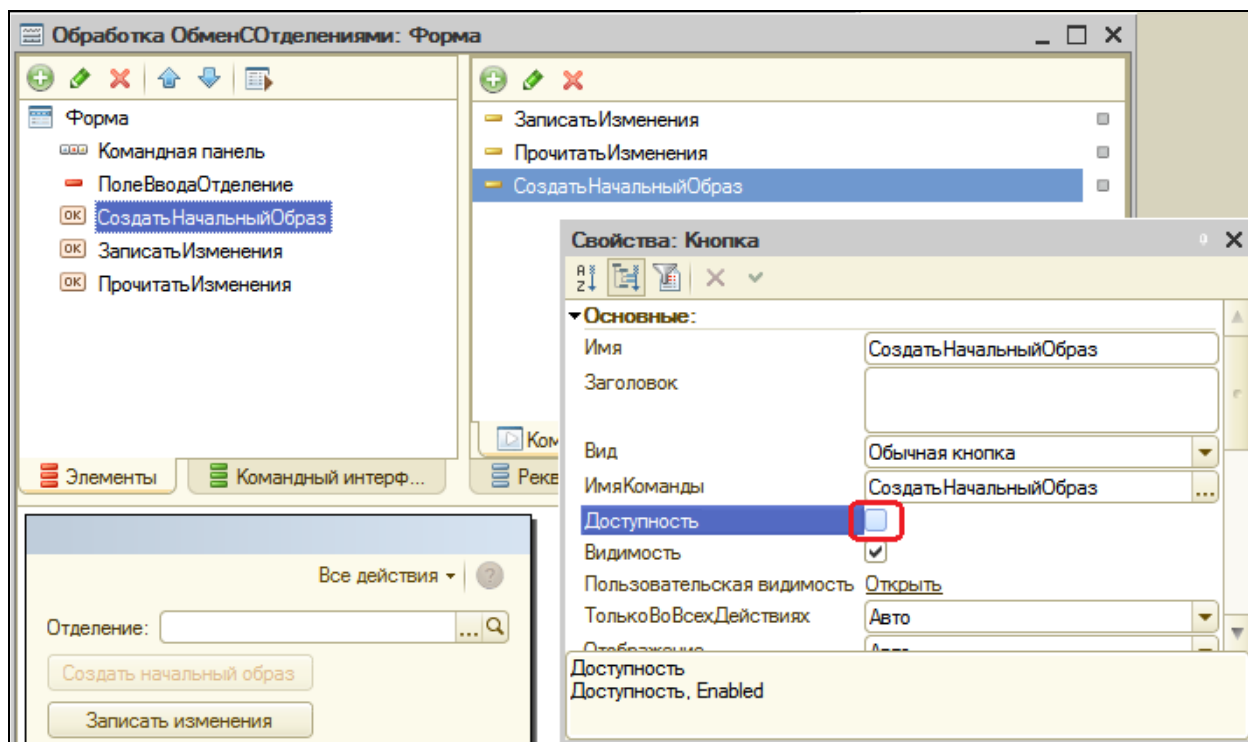
Нажмем кнопку открытия в строке **Действие** для каждой команды.

Шаблоны обработчиков для каждой команды пока заполнять не будем, а перейдем на закладку **Форма** и поочередно перетащим эти команды в окно элементов формы.



Откроем свойства кнопки **СоздатьНачальныйОбраз** и снимем флажок у свойства **Доступность**.

Т.о., при открытии обработки кнопка будет недоступной, пока не выбран узел плана обмена в поле **ПолеВводаОтделение**. Эта кнопка также будет недоступна в случае выбора predetermined узла базы, т.е. создание начального образа невозможно, если выбранный узел является predetermined.



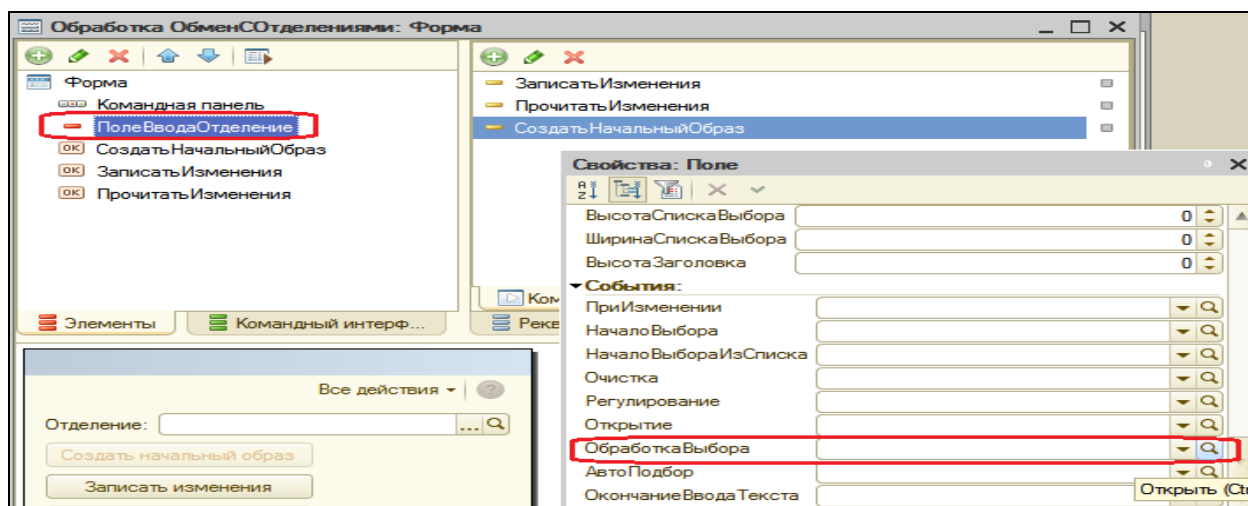
Чтобы обеспечить такое поведение кнопки, создадим в модуле формы обработки функцию, выполняющуюся на сервере и возвращающую истину, если переданный в функцию узел является предопределенным.

&НаСервереБезКонтекста  
Функция ПредопределенныйУзел(Узел)

Возврат Узел = ПланыОбмена.Отделения.ЭтотУзел();

КонецФункции

Затем в окне элементов формы выделим элемент **ПолеВводаОтделение**, вызовем его свойства и создадим обработчик события **ОбработкаВыбора**.



Заполним обработчик:

```
&НаКлиенте
Процедура ПолеВводаОтделениеОбработкаВыбора(Элемент, ВыбранноеЗначение,
СтандартнаяОбработка)
    Если ПредопределенныйУзел(ВыбранноеЗначение) Тогда
        Элементы.СоздатьНачальныйОбраз.Доступность = Ложь;
    Иначе
        Элементы.СоздатьНачальныйОбраз.Доступность = Истина;
    КонецЕсли;
КонецПроцедуры
```

В этой процедуре доступность кнопки **СоздатьНачальныйОбраз** определяется в зависимости от значения функции **ПредопределенныйУзел()**, в которую передается ссылка на выбранный узел (**ВыбранноеЗначение**).

Теперь заполним обработчик команды **СоздатьНачальныйОбраз**:

```
&НаКлиенте
Процедура СоздатьНачальныйОбраз(Команда)
    Диалог = Новый
    ДиалогВыбораФайла(РежимДиалогаВыбораФайла.ВыборКаталога);
    Диалог.Заголовок = "Укажите каталог информационной базы:";
    Если Диалог.Выбрать() Тогда
        СоздатьНачальныйОбразНаСервере(ПолеВводаОтделение,
        Диалог.Каталог);
        Предупреждение("Создание начального образа узла завершено.");
    КонецЕсли;
КонецПроцедуры
```

В начале процедуры мы вызываем диалог выбора каталога, в который будет помещен образ информационной базы, и затем вызываем процедуру **СоздатьНачальныйОбразНаСервере()**, исполняющуюся на сервере, в которой вызывается метод **СоздатьНачальныйОбраз()** объекта **ПланыОбменаМенеджер**.

Именно этот метод и позволяет нам создать образ подчиненного узла распределенной базы. В первом параметре метода передается ссылка на узел (реквизит формы ПолеВводаОтделение), для которого мы хотим создать начальный образ, а во втором – строка соединения, указывающая информационную базу. Поместите эту процедуру после только что написанной:

```

&НаСервереБезКонтекста
Процедура СоздатьНачальныйОбразНаСервере(Узел, КаталогСоединения)

    ПланыОбмена.СоздатьНачальныйОбраз(Узел, "File =" + КаталогСоединения);

КонецПроцедуры

```

Теперь создадим обработчик команды **Записать изменения**.

```

&НаКлиенте
Процедура ЗаписатьИзменения(Команда)
    Диалог = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Сохранение);
    Диалог.Заголовок = "Укажите файл обмена:";
    Если Диалог.Выбрать() Тогда
        ЗаписатьИзмененияНаСервере(ПолеВводаОтделение,
Диалог.ПолноеИмяФайла);
        Предупреждение("Запись изменений завершена.");
    КонецЕсли;

КонецПроцедуры

```

В начале процедуры мы вызываем диалог ввода имени файла, в который будут записаны изменения, и затем вызываем процедуру **ЗаписатьИзмененияНаСервере()**, исполняющуюся на сервере. В первом параметре метода передается ссылка на узел (реквизит формы **ПолеВводаОтделение**), для которого будет производиться запись изменений.

В этой процедуре мы создаем объект **ЗаписьXML** для работы с этим файлом.

Затем создаем объект **ЗаписьСообщенияОбмена**, с помощью которого будем делать сообщение обмена. В метода **НачатьЗапись()**, во втором параметре мы указываем, для какого узла обмена будет создаваться это сообщение.

После этого мы выполняем метод **ЗаписатьИзменения()** объекта **ПланыОбменаМенеджер**, который и записывает изменения, предназначенные для передачи в выбранный узел, в указанное сообщение обмена.

В заключение заканчиваем запись сообщения обмена и закрываем файл:

```

&НаСервереБезКонтекста
Процедура ЗаписатьИзмененияНаСервере(Узел, ИмяФайла)

    // Создать и проинициализировать объект ЗаписьXML
    ЗаписьXML = Новый ЗаписьXML;

```

```

ЗаписьXML.ОткрытьФайл(ИмяФайла);

// Создать объект ЗаписьСообщенияОбмена и начать запись сообщения
ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Узел);

// Записать содержимое тела сообщения обмена данными распределенной ИБ
ПланыОбмена.ЗаписатьИзменения(ЗаписьСообщения);

// Закончить запись сообщения и запись XML
ЗаписьСообщения.ЗакончитьЗапись();
ЗаписьXML.Закрыть();

```

КонецПроцедуры

Последним мы создадим обработчик команды **Прочитать изменения**.

```

&НаКлиенте
Процедура ПрочитатьИзменения(Команда)
    Диалог = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Открытие);
    Диалог.Заголовок = "Укажите файл обмена:";
    Если Диалог.Выбрать() Тогда
        ПрочитатьИзмененияНаСервере(Диалог.ПолноеИмяФайла);
        Предупреждение("Чтение изменений завершено.");
    КонецЕсли;

```

КонецПроцедуры

В начале процедуры мы снова вызываем диалог ввода имени файла, который будет прочитан, и затем вызываем процедуру **ПрочитатьИзмененияНаСервере()**, исполняющуюся на сервере.

В этой процедуре создается объект **ЧтениеXML** для работы с этим файлом. Затем создаем объект **ЧтениеСообщенияОбмена** для чтения сообщения, содержащегося в указанном файле.

Затем методом **ПрочитатьИзменения()** объекта **ПланыОбменаМенеджер** мы читаем полученное сообщение.

В заключение процедуры мы завершаем чтение сообщения обмена и закрываем файл:

```

&НаСервереБезКонтекста
Процедура ПрочитатьИзмененияНаСервере(ИмяФайла)

    // Создать и проинициализировать объект ЧтениеXML
    ЧтениеXML = Новый ЧтениеXML;
    ЧтениеXML.ОткрытьФайл(ИмяФайла);

    // Создать объект ЧтениеСообщенияОбмена и начать чтение сообщения
    ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

```

```

ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

// Прочитать содержимое тела сообщения
ПланыОбмена.ПрочитатьИзменения(ЧтениеСообщения);

// Закончить чтение сообщения и чтение XML
ЧтениеСообщения.ЗакончитьЧтение();
ЧтениеXML.Закрыть();

```

КонецПроцедуры

Полностью обработка **ОбменСОтделениями** в модуле формы будет такой:

```

&НаКлиенте
Процедура ЗаписатьИзменения(Команда)
    Диалог = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Сохранение);
    Диалог.Заголовок = "Укажите файл обмена:";
    Если Диалог.Выбрать() Тогда
        ЗаписатьИзмененияНаСервере(ПолеВводаОтделение,
Диалог.ПолноеИмяФайла);
        Предупреждение("Запись изменений завершена.");
    КонецЕсли;

```

КонецПроцедуры

&НаСервереБезКонтекста

Процедура ЗаписатьИзмененияНаСервере(Узел, ИмяФайла)

```

// Создать и проинициализировать объект ЗаписьXML
ЗаписьXML = Новый ЗаписьXML;
ЗаписьXML.ОткрытьФайл(ИмяФайла);

// Создать объект ЗаписьСообщенияОбмена и начать запись сообщения
ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Узел);

// Записать содержимое тела сообщения обмена данными распределенной ИБ
ПланыОбмена.ЗаписатьИзменения(ЗаписьСообщения);

// Закончить запись сообщения и запись XML
ЗаписьСообщения.ЗакончитьЗапись();
ЗаписьXML.Закрыть();

```

КонецПроцедуры

&НаКлиенте

Процедура ПрочитатьИзменения(Команда)

```

    Диалог = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Открытие);
    Диалог.Заголовок = "Укажите файл обмена:";
    Если Диалог.Выбрать() Тогда
        ПрочитатьИзмененияНаСервере(Диалог.ПолноеИмяФайла);

```

```

        Предупреждение("Чтение изменений завершено.");
    КонецЕсли;

КонецПроцедуры

&НаСервереБезКонтекста
Процедура ПрочитатьИзмененияНаСервере(ИмяФайла)

    // Создать и проинициализировать объект ЧтениеXML
    ЧтениеXML = Новый ЧтениеXML;
    ЧтениеXML.ОткрытьФайл(ИмяФайла);

    // Создать объект ЧтениеСообщенияОбмена и начать чтение сообщения
    ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();
    ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

    // Прочитать содержимое тела сообщения
    ПланыОбмена.ПрочитатьИзменения(ЧтениеСообщения);

    // Закончить чтение сообщения и чтение XML
    ЧтениеСообщения.ЗакончитьЧтение();
    ЧтениеXML.Заккрыть();

КонецПроцедуры

&НаКлиенте
Процедура СоздатьНачальныйОбраз(Команда)

    Диалог = Новый
        ДиалогВыбораФайла(РежимДиалогаВыбораФайла.ВыборКаталога);
    Диалог.Заголовок = "Укажите каталог информационной базы:";
    Если Диалог.Выбрать() Тогда
        СоздатьНачальныйОбразНаСервере(ПолеВводаОтделение,
        Диалог.Каталог);
        Предупреждение("Создание начального образа узла завершено.");
    КонецЕсли;

КонецПроцедуры

&НаСервереБезКонтекста
Процедура СоздатьНачальныйОбразНаСервере(Узел, КаталогСоединения)

    ПланыОбмена.СоздатьНачальныйОбраз(Узел, "File =" + КаталогСоединения);

КонецПроцедуры

&НаСервереБезКонтекста
Функция ПредопределенныйУзел(Узел)

    Возврат Узел = ПланыОбмена.Отделения.ЭтотУзел();

КонецФункции

&НаКлиенте

```



```

Процедура ПолеВводаОтделениеОбработкаВыбора(Элемент, ВыбранноеЗначение,
СтандартнаяОбработка)
    Если ПредопределенныйУзел(ВыбранноеЗначение)Тогда
        Элементы.СоздатьНачальныйОбраз.Доступность = Ложь;
    Иначе
        Элементы.СоздатьНачальныйОбраз.Доступность = Истина;
    КонецЕсли;
КонецПроцедуры

```

В заключение на закладке **Подсистемы** укажем принадлежность обработки **ОбменСОтделениями** к подсистеме **Предприятие**.

В окне редактирования командного интерфейса этой подсистемы (**Все подсистемы**) установим следующий порядок следования команд в группе панели действий **Сервис**:

- Поиск в данных,
- Общие настройки,
- Обмен данными,
- Обмен с отделениями,
- Планировщик заданий.

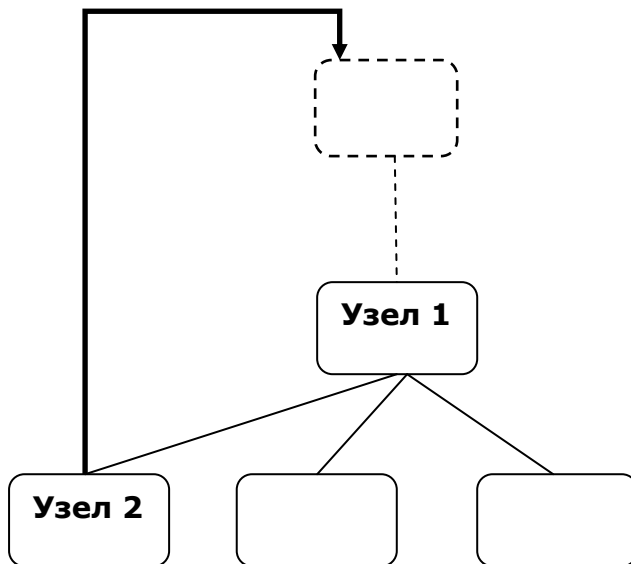
Проверить нашу обработку можно на примере, аналогичном приведенному в разделе универсального обмена данными (но не обязательно).

### Изменение структуры узлов

В заключение следует сказать о том, что механизм распределенных информационных баз содержит программное средство реконфигурирования структуры узлов распределенной базы.

Для этого следует использовать метод **УстановитьГлавныйУзел()** объекта **ПланыОбменаМенеджер**. В параметре этого метода передается ссылка на узел плана обмена распределенной базы, который устанавливается главным для текущей базы. Также в этом параметре может быть передано значение **Неопределено**, и это приведет к тому, что у текущей информационной базы будет отсутствовать главный узел.

Допустим, необходимо переместить один из подчиненных узлов в корень дерева.



Для этого следует выполнить следующие действия в плане обмена:

```
// В информационной базе Узла2.  
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Неопределено);  
  
// В информационной базе Узла1.  
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Узел2);
```

При этом будут удалены все записи регистрации изменений конфигурации Узла1, относящиеся к Узлу2, т.к. передача изменений конфигурации будет возможна теперь только от Узла2 к Узлу1. Записи регистрации изменения данных удалены не будут, т.к. передача изменений данных по-прежнему возможна между этими узлами.

Таким же образом, используя значение параметра метода Неопределено, мы можем отключать от дерева отдельную информационную базу или целое поддерево

```
// В информационной базе Узла1.  
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Неопределено);
```

Кроме этого, мы можем создавать распределенную базу из отдельных информационных баз с идентичной конфигурацией.

```
// В информационных базах Узла2, Узла3 и Узла4.  
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Узел1);
```

### Контрольные вопросы

- ✓ Какие средства входят в состав механизма универсального обмена данными.

- ✓ Для чего предназначен объект конфигурации План обмена.
- ✓ Каковы основные составляющие плана обмена.
- ✓ Что такое узлы плана обмена.
- ✓ Что такое состав плана обмена и для каких элементов данных возможен обмен данными.
- ✓ Что такое авторегистрация.
- ✓ Для чего предназначен механизм регистрации изменений.
- ✓ Как работает инфраструктура сообщений.
- ✓ Каково назначение XML-сериализации.
- ✓ Для чего используется запись/чтение документов XML.
- ✓ Как создать план обмена.
- ✓ Как настроить конфигурацию для обмена данными.
- ✓ Как реализовать обмен данными в общем виде.