**Создание демонстрационных баз данных**  
Запустим **SQL Server Management Studio (SSMS)** и при помощи скрипта создадим 3 базы данных – первые две (DemoSSIS\_SourceA и DemoSSIS\_SourceB) будут выступать в роли источников данных, а третья (DemoSSIS\_Target) в роли получателя данных:

*-- первая БД выступающая в роли источника данных*

CREATE DATABASE DemoSSIS\_SourceA

GO

ALTER DATABASE DemoSSIS\_SourceA SET RECOVERY SIMPLE

GO

*-- вторая БД выступающая в роли источника данных*

CREATE DATABASE DemoSSIS\_SourceB

GO

ALTER DATABASE DemoSSIS\_SourceB SET RECOVERY SIMPLE

GO

*-- БД выступающая в роли получателя данных*

CREATE DATABASE DemoSSIS\_Target

GO

ALTER DATABASE DemoSSIS\_Target SET RECOVERY SIMPLE

GO  
В базах источниках создадим тестовые таблицы и наполним их тестовыми данными:

USE DemoSSIS\_SourceA

GO

*-- продукты из источника A*

CREATE TABLE Products(

ID int NOT NULL IDENTITY,

Title nvarchar(50) NOT NULL,

Price money,

CONSTRAINT PK\_Products PRIMARY KEY(ID)

)

GO

*-- наполняем таблицу тестовыми данными*

SET IDENTITY\_INSERT Products ON

INSERT Products(ID,Title,Price)VALUES

(1,N'Клей',20),

(2,N'Корректор',NULL),

(3,N'Скотч',100),

(4,N'Стикеры',80),

(5,N'Скрепки',25)

SET IDENTITY\_INSERT Products OFF

GO

USE DemoSSIS\_SourceB

GO

*-- продукты из источника B*

CREATE TABLE Products(

ID int NOT NULL IDENTITY,

Title nvarchar(50) NOT NULL,

Price money,

CONSTRAINT PK\_Products PRIMARY KEY(ID)

)

GO

*-- наполняем таблицу тестовыми данными*

SET IDENTITY\_INSERT Products ON

INSERT Products(ID,Title,Price)VALUES

(1,N'Ножницы',200),

(2,N'Нож канцелярский',70),

(3,N'Дырокол',220),

(4,N'Степлер',150),

(5,N'Шариковая ручка',15)

SET IDENTITY\_INSERT Products OFF

GO  
Создадим таблицу в принимающей базе:

USE DemoSSIS\_Target

GO

*-- принимающая таблица*

CREATE TABLE Products(

ID int NOT NULL IDENTITY,

Title nvarchar(50) NOT NULL,

Price money,

SourceID char(1) NOT NULL, *-- используется для идентификации источника*

SourceProductID int NOT NULL, *-- ID в источнике*

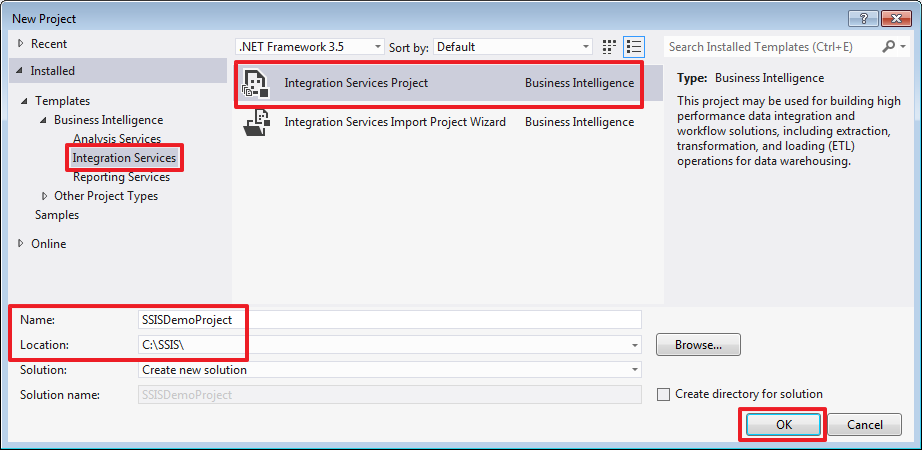
CONSTRAINT PK\_Products PRIMARY KEY(ID),

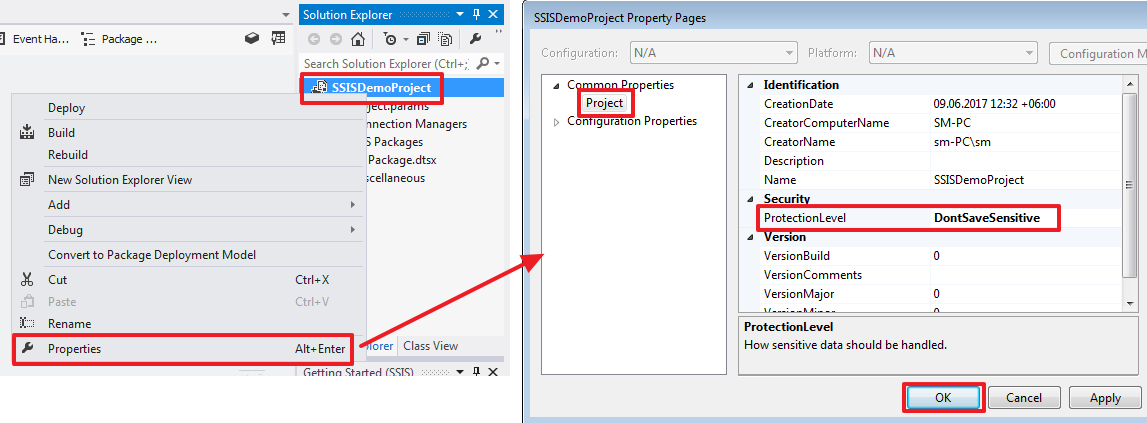
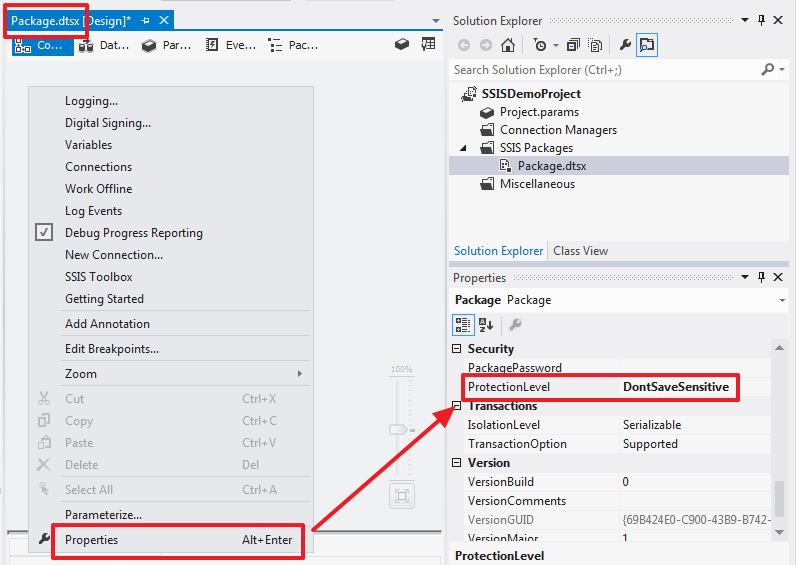
CONSTRAINT UK\_Products UNIQUE(SourceID,SourceProductID),

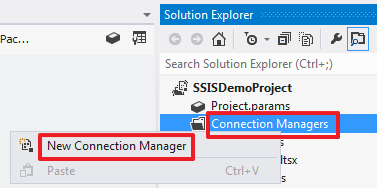
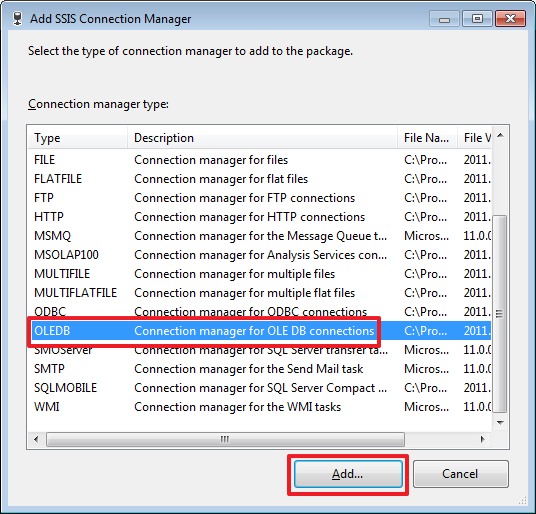
CONSTRAINT CK\_Products\_SourceID CHECK(SourceID IN('A','B'))

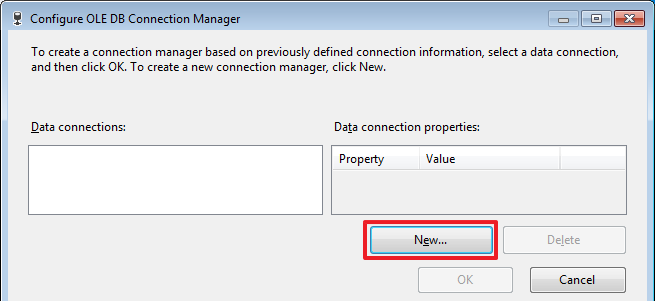
)

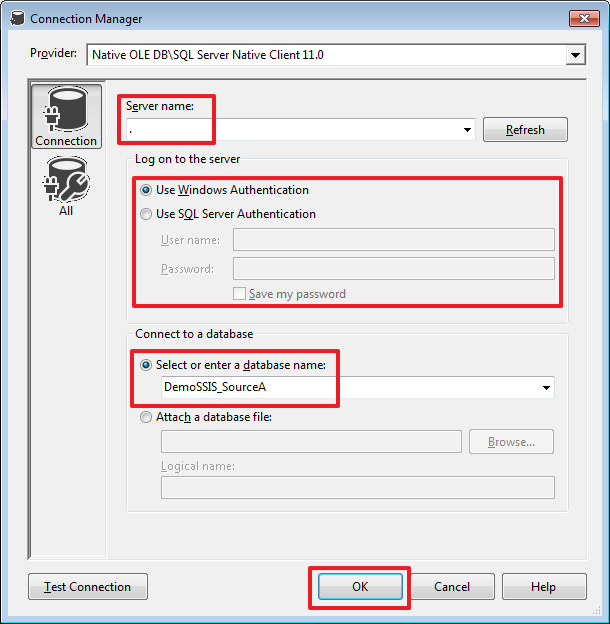
GO

Создадим новый проект (**File -> New -> Project…**):  
  


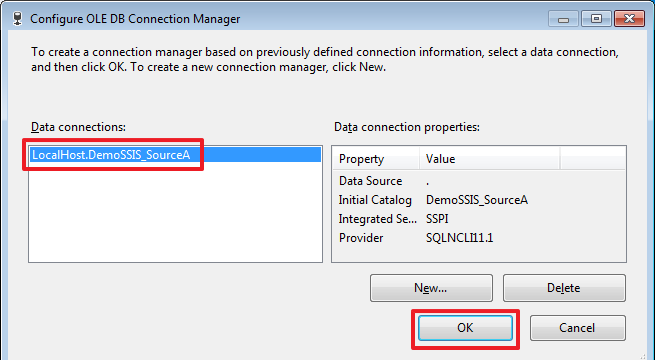
Для последующего облегчения развертывания зайдем в свойства проекта и изменим опцию **ProtectionLevel** на **DontSaveSensitive**:  
  
  
  
То же самое сделаем в свойствах пакета, который создался по умолчанию:  
  


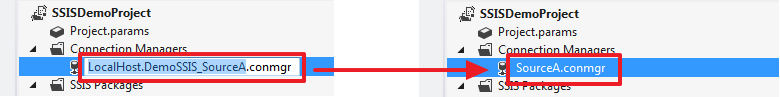
**Для всех новых пакетов данное свойство будет заполняться значением из свойства проекта.**  
  
Создадим соединения:  
  
  
  


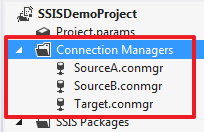
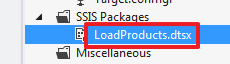


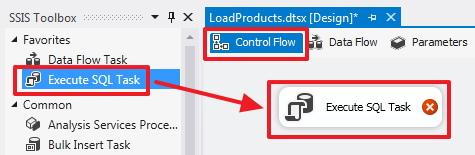
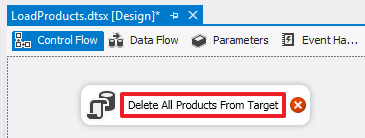
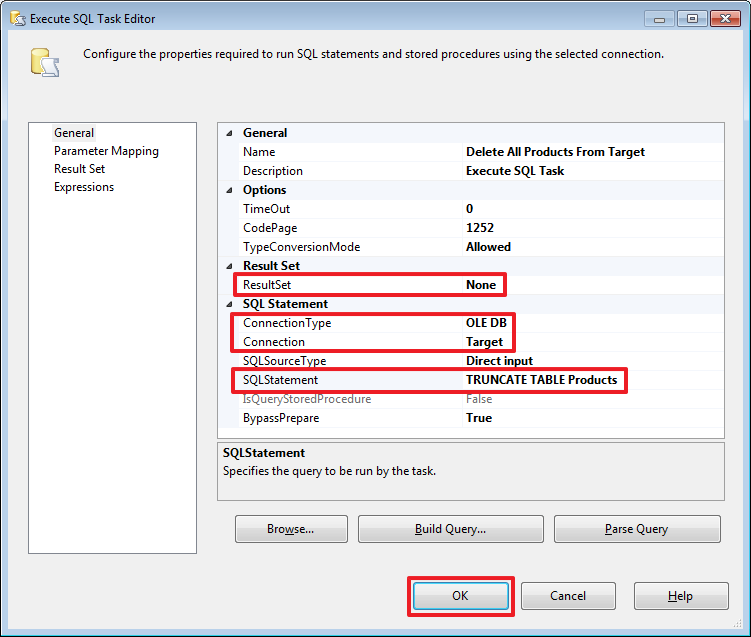
Заполняем параметры соединение с БД:  
  


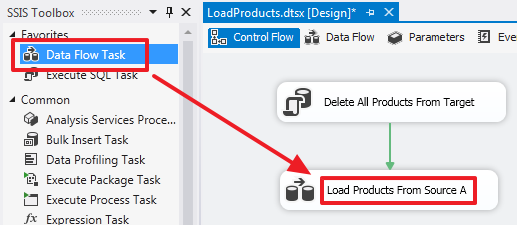
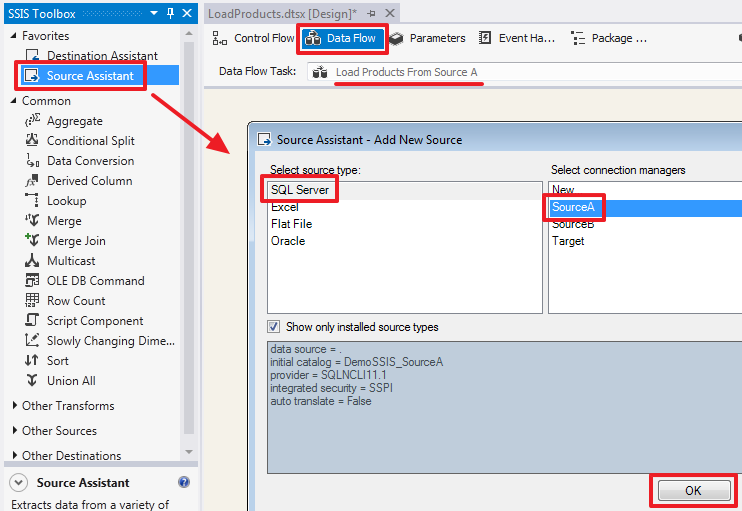
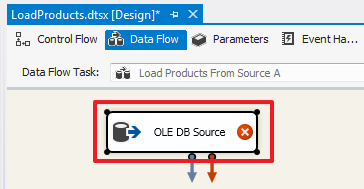
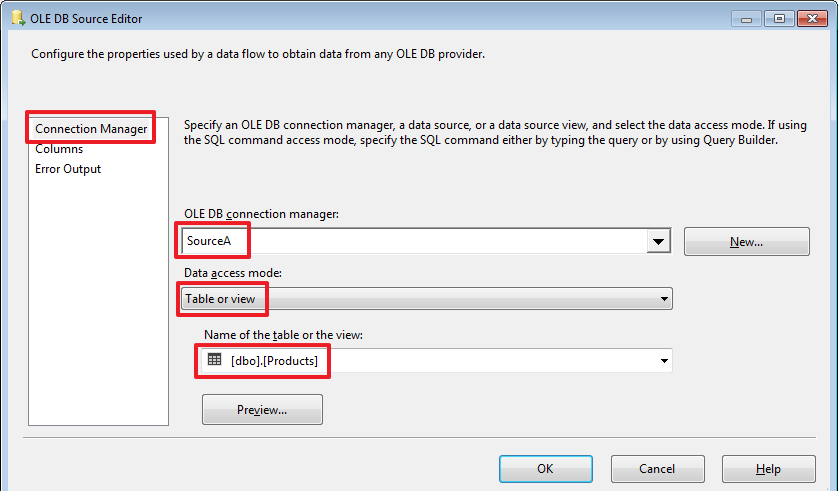
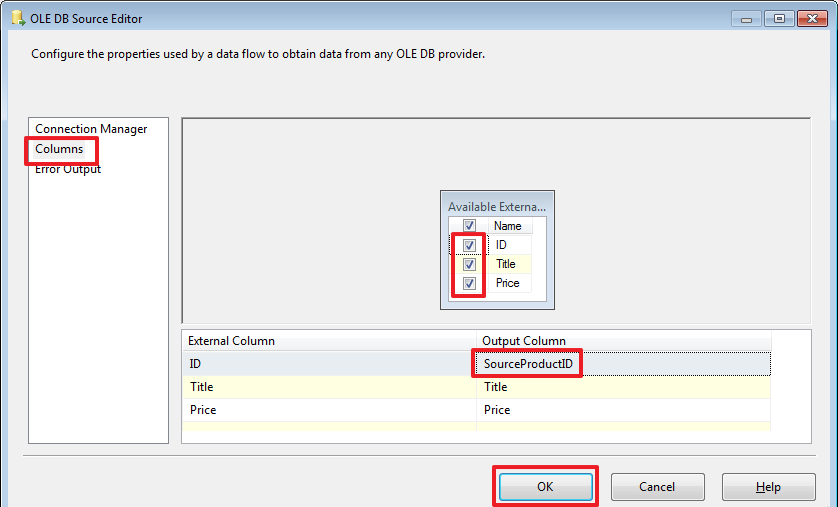
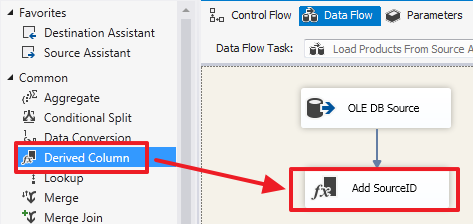
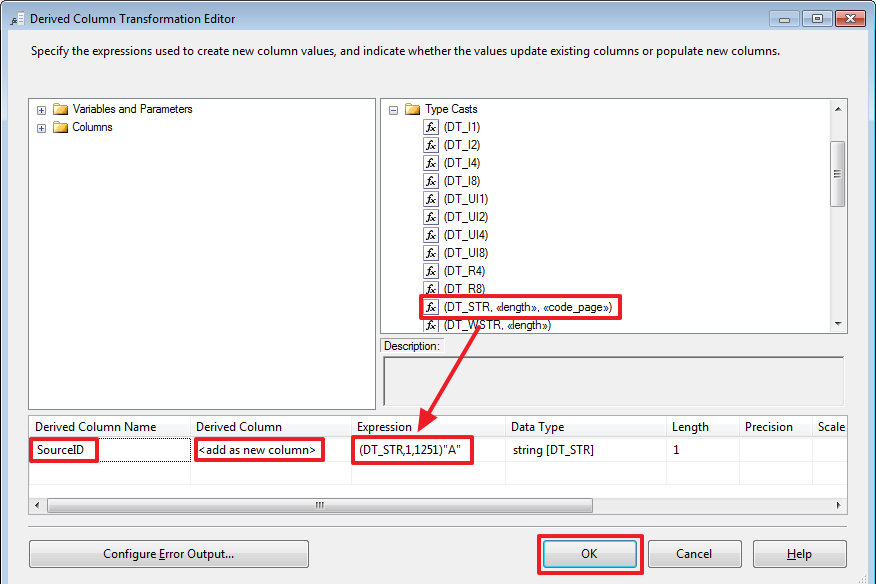
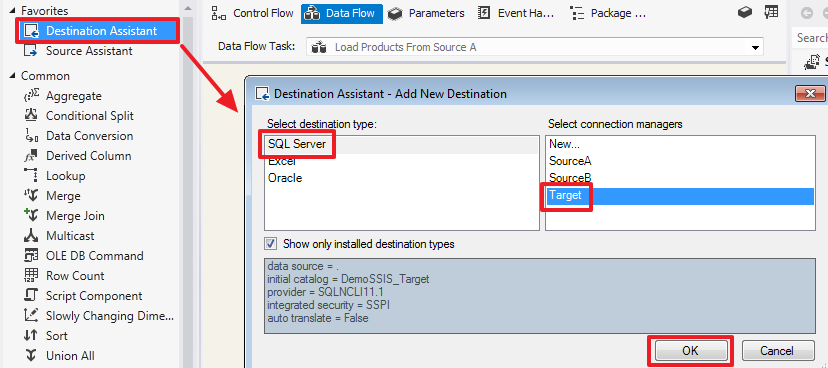
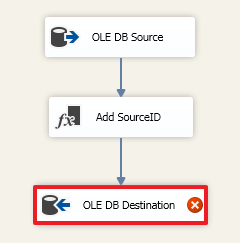
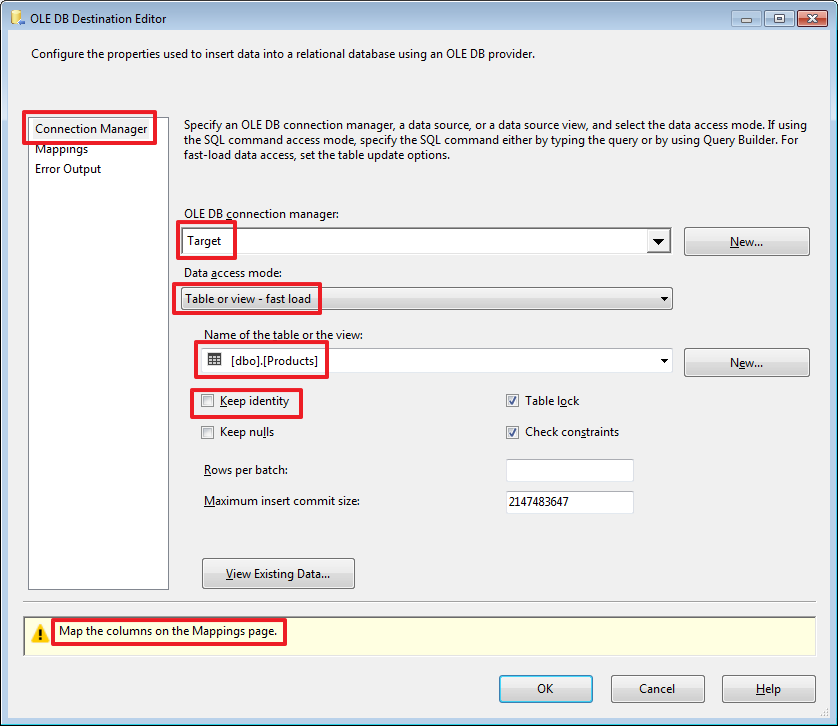
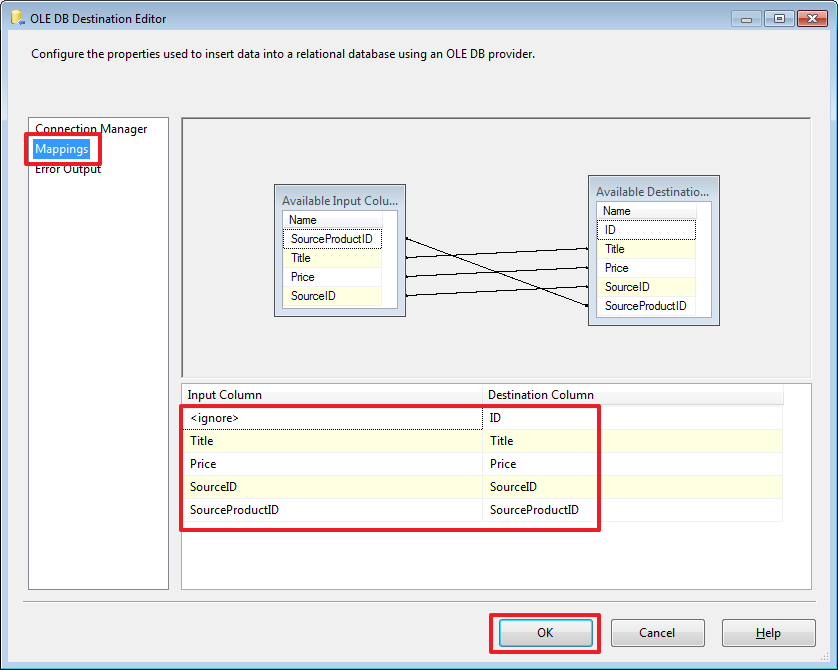
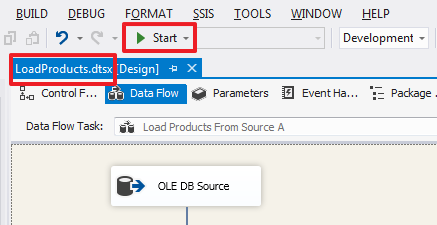
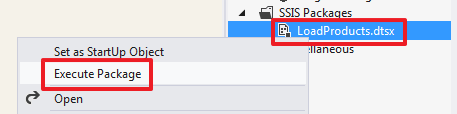
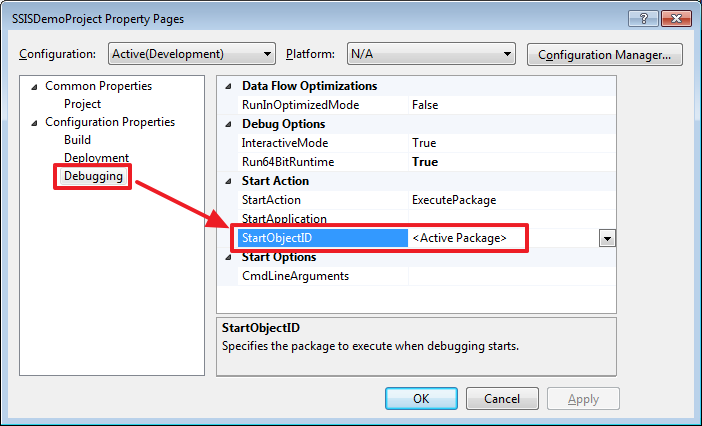
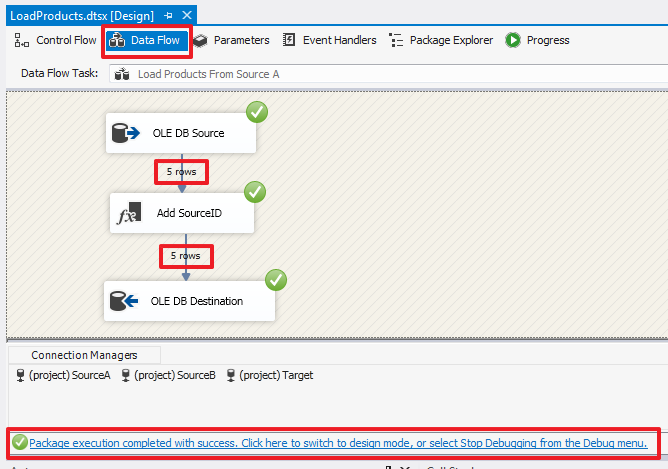
**Боевые параметры соединения в дальнейшем можно будет настроить при создании задачи SQL Server Agent.**

  
  
Для удобства я переименую название соединения на **SourceA**:

  
  
Таким же образом создадим и переименуем соединения для баз **DemoSSIS\_SourceB** и **DemoSSIS\_Target**:

  
  
Переименуем пакет, созданный по умолчанию, в «**LoadProducts.dtsx**»:  
  


**Сначала напишем простую логику, которая будет полностью очищать таблицу Products в базе DemoSSIS\_Target и снова загружать в нее данные из двух баз данных DemoSSIS\_SourceA и DemoSSIS\_SourceB.**  
  
Для очистки воспользуемся компонентом «Execute SQL Task», который мы при помощи мыши создадим в области «Control Flow»:  
  
  
  
Для наглядности можно переименовать название компонент. Зададим ему имя «Delete All Products From Target»:  
  
  
  
Для этой цели используется свойство Name.  
  
Дважды щелкнем на этом элементе и пропишем следующие свойства:  
  
  
  
Т.к. TSQL команда «TRUNCATE TABLE Products» ничего не возвращает оставим свойства ResultSet равным None.

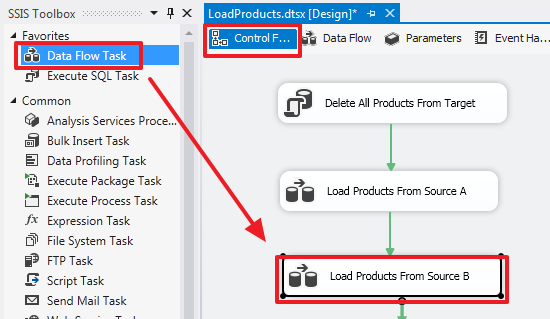
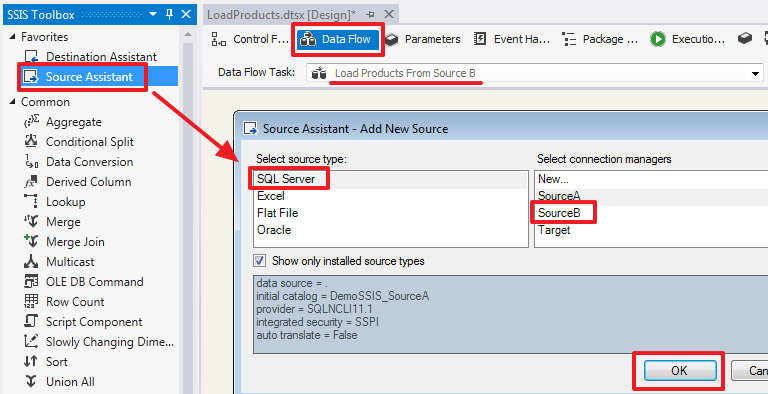
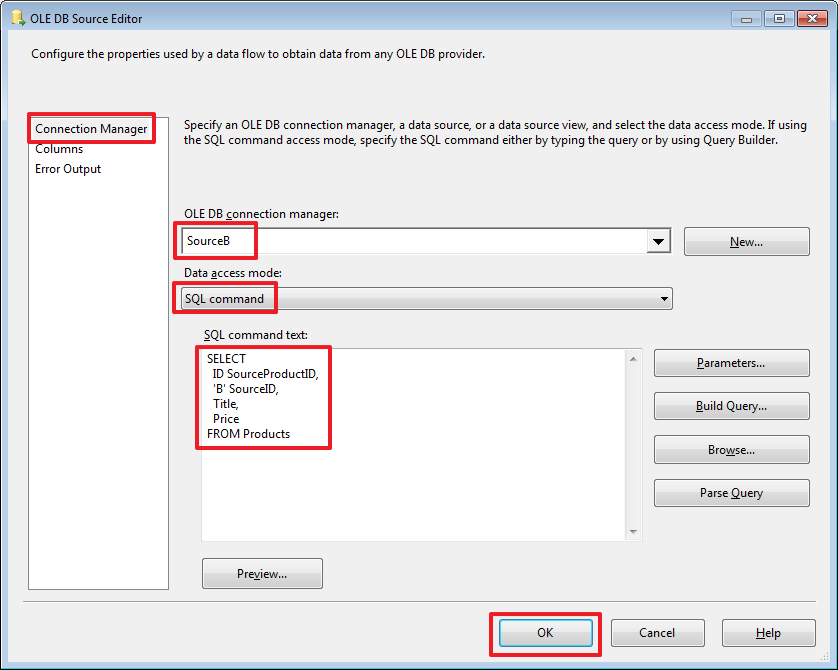
Теперь скинем в область «Control Flow» компонент «Data Flow Task» и переименуем его в «Load Products From Source A», а также протянем к этому компоненту зеленную стрелку от «Delete All Products From Target»:  
  
  
  
**Таким образом мы создали цепочку, которая будет выполняться последовательно.**  
  
Щелкнув дважды на «Load Products From Source A» мы попадаем в область «Data Flow» этого элемента.  
  
**Data Flow Task – это сложный компонент, который имеет свою область, в которой создаются вложенные элементы для работы с потоком данных.**  
  
Скинем в эту область компонент «Source Assistant»:  
  
  
  
  
  
Этот компонент отвечает за получение данных из источника. Дважды щелкнув по нему, мы сможем настроить его:  
  
  
  
Пока воспользуемся режимом «Data access mode» равным «Table or view». Это приведет к получению всех строк из таблицы Products. Посмотреть данные можно нажав на «Preview…».  
  
На закладке Columns мы можем выбрать только необходимые нам колонки и при необходимости переименовать их прописав новое имя в колонке «Output Columns»:  
  
  
  
Для получателя нужна еще одна дополнительная колонка SourceID, добавим ее к выходному набору при помощи компонента «Derived Column», который переименуем в «Add SourceID», так же протянем синюю стрелку к данному элементу от «OLE DB Source»:  
  
  
  
Дважды щелкнем по элементу «Add SourceID» и пропишем значение «A» в виде константы:  
  
  
  
Здесь я воспользовался функцией преобразования типа (DT\_STR,1,1251) для того чтобы превратить Unicode строку в ANSI.  
  
Теперь создадим компонент «Destination Assistant»:  
  
  
  
Направим в него поток от «Add SourceID»:  
  
  
  
Дважды щелкнем по «OLE DB Destination» и произведем настройки:  
  
  
  
Здесь мы показываем в какую таблицу будет записываться полученный набор.  
  
«**Keep identity**» используется в случае если в принимающей таблице есть поле с флагом IDENTITY и мы хотим, чтобы значения в него тоже записывались из источника (это аналогично включению опции **SET IDENTITY\_INSERT Products ON**).  
  
Перейдя на закладку Mappings осуществим привязку полей источника с полями получателя:  
  
  
  
Так как у нас поля источника и приемника именуются одинаково, то привязка осуществилась автоматически.  
  
**Можем протестировать работу пакета и убедиться, что данные залились в таблицу Products базы DemoSSIS\_Target.**  
  
Запускаем пакет на выполнение из Visual Studio нажав **Start** или клавишу **F5**:  
  
  
  
Так же пакет можно выполнить, воспользовавшись командой из контекстного меню:  
  
  
  
При помощи «**Set as StartUp Object**» можно задать пакет, который будет запускаться по нажатию на **Start (F5)**.  
  
**Какой пакет будет запускаться при нажатии на Start (F5) можно переопределить в свойствах проекта:**  
  
  
  
**По умолчанию будет запускаться пакет открытый в текущий момент, об этом говорит значение StartObjectID равное <Active Package>.**  
  
Запустив проект мы должны увидеть следующую картину:  
  
  
  
Пакет выполнился без ошибок, о чем говорит зеленый значок и текст в нижней части.  
  
В случае наличия ошибок их можно будет увидеть вкладке Progress.  
  
**Нажмем на ссылку «Package execution completed…» или на кнопку «Stop Debugging» расположенную на панели инструментов для остановки выполнения пакета.**  
  
https://habrastorage.org/web/39d/de0/fef/39dde0fefd6e494abb62424cd31d92ff.png  
  
Выполним запрос:

USE DemoSSIS\_Target

GO

SELECT \*

FROM Products

И убедимся, что данные были записаны в принимающую таблицу.  
  
Перейдем в область «Control Flow» и создадим еще один компонент «Data Task Flow», который назовем «Load Products From Source B», протянем на него зеленную стрелку от «Load Products From Source A»:  
  
  
  
Двойным щелчком зайдем в область «Data Flow» этого элемента и создадим «Source Assistant»:  
  
  
  
**Дважды щелкнув на этом элементе, настроим его по-другому:**  
  
  
  
Выберем режим «SQL command» и пропишем следующий запрос:

SELECT

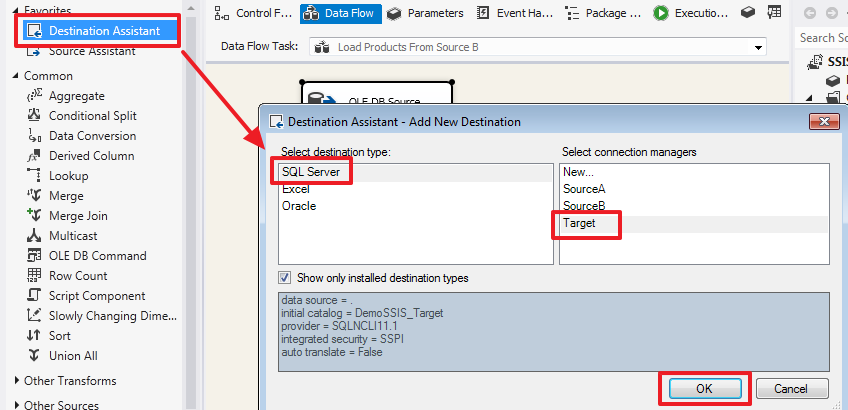
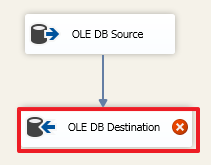
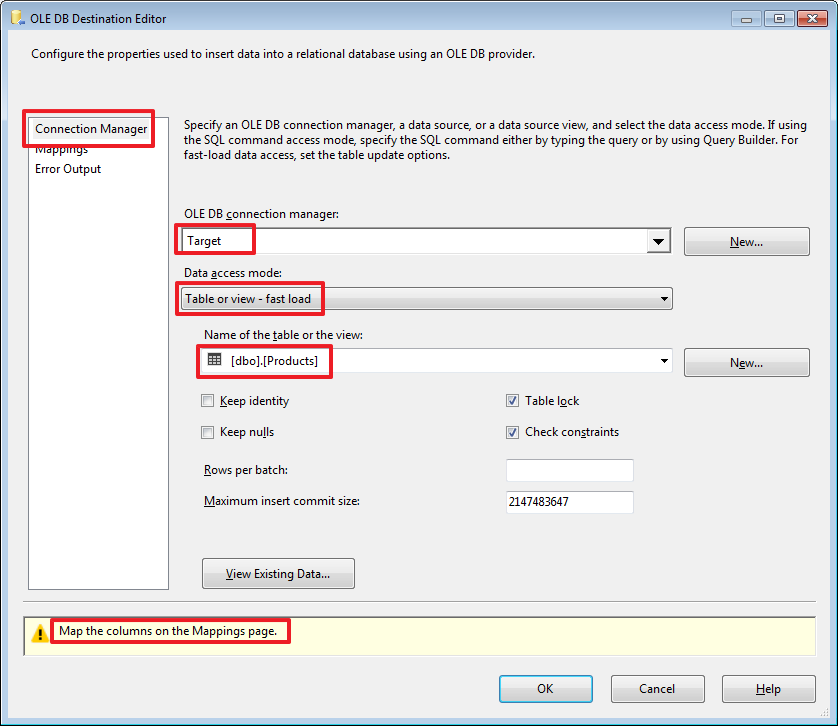
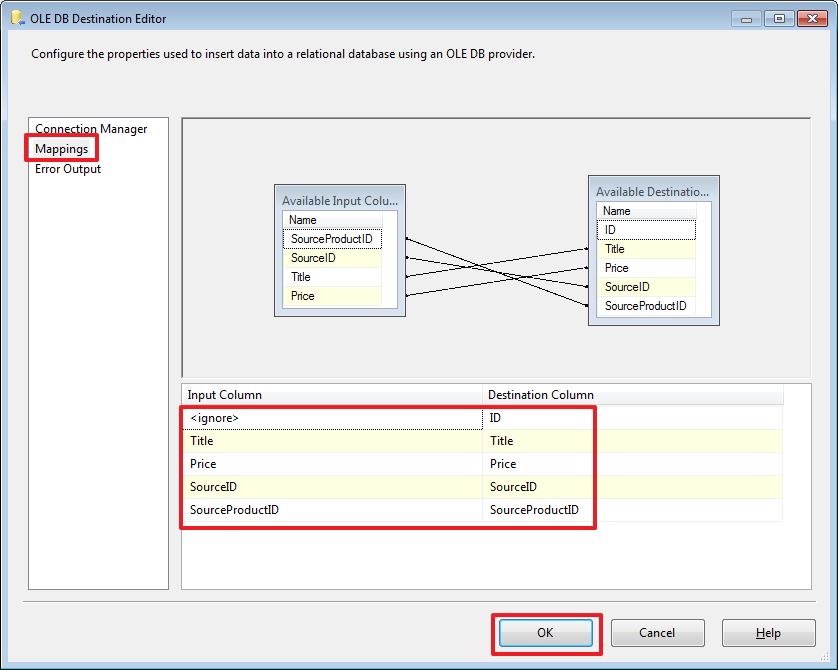
ID SourceProductID,

'B' SourceID,

Title,

Price

FROM Products

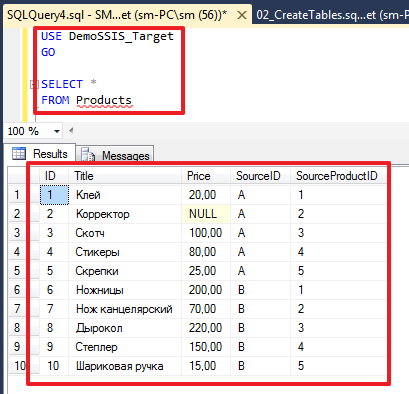
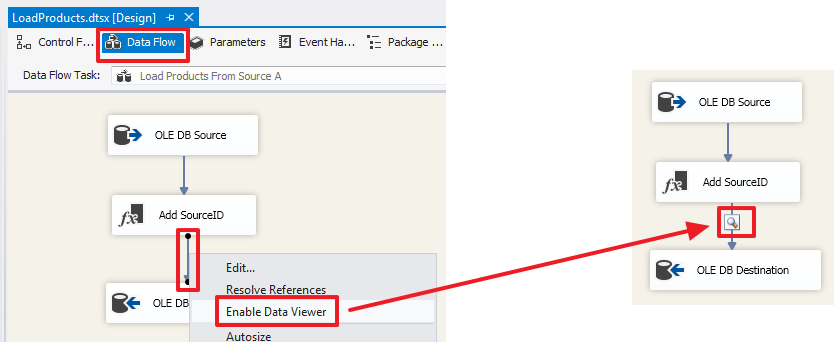
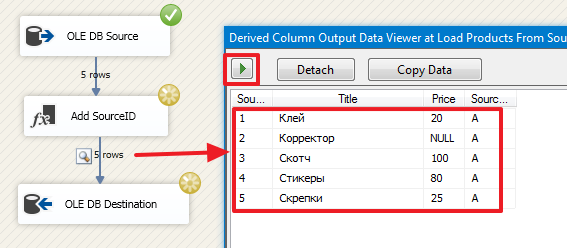
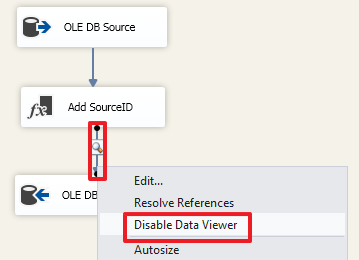
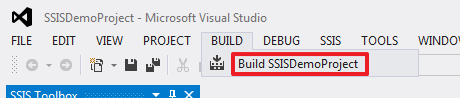
Дальше сразу создадим компонент «Destination Assistant» и протянем на него синюю стрелку от «OLE DB Source»:  
  
  
  
  
  
Двойным щелчком зайдем в редуктор этого элемента и настроим его:  
  
  
  
  
  
**Запустим проект на выполнение и убедимся, что данные с двух источников попали в таблицу в базе Target:**

USE DemoSSIS\_Target

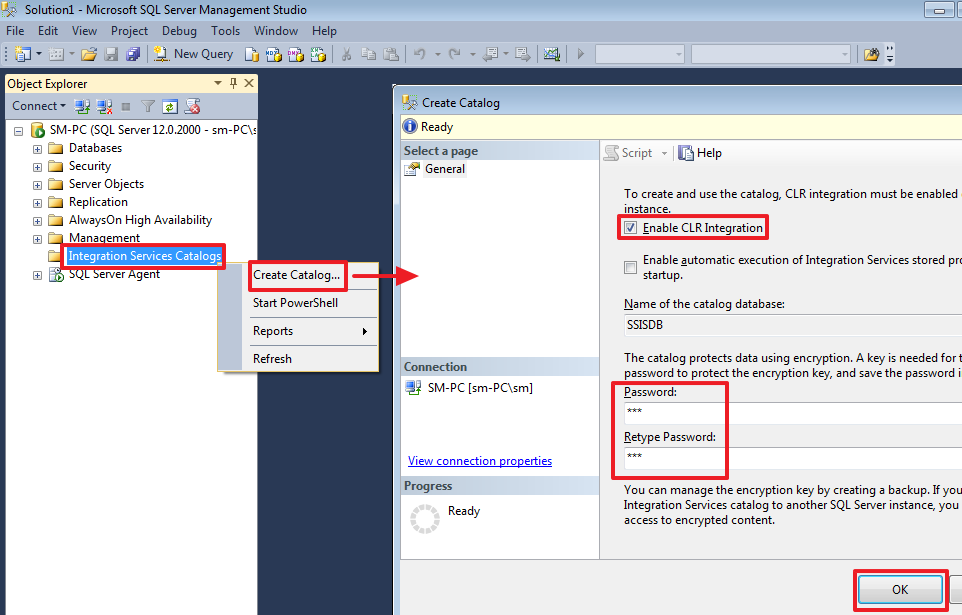
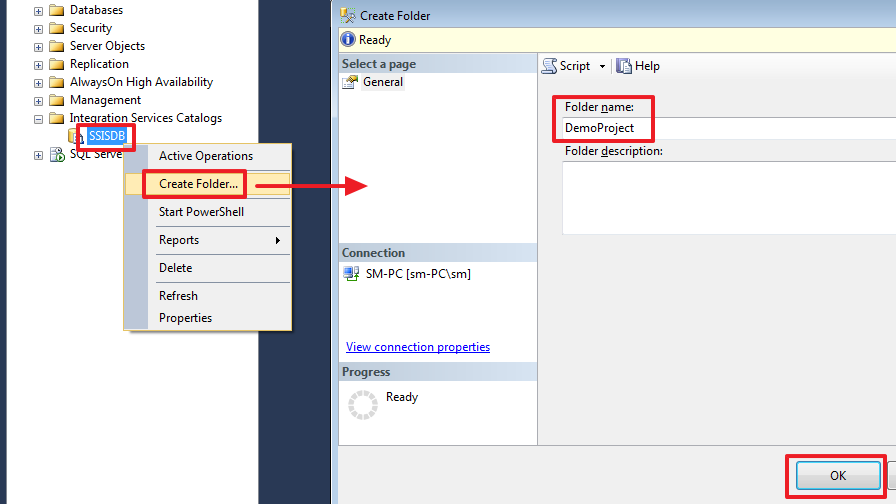
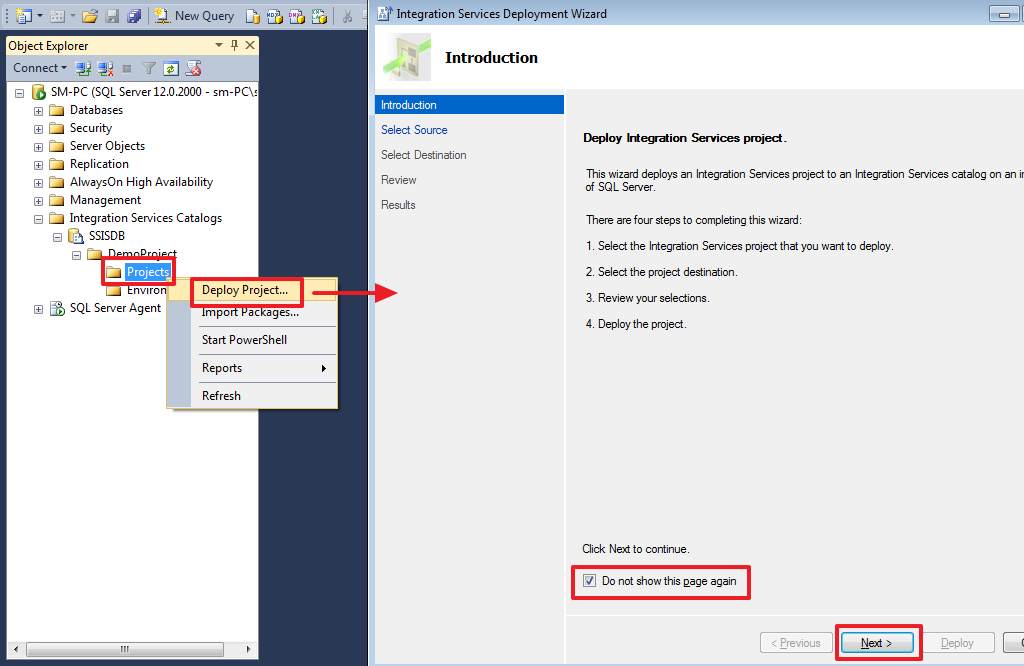
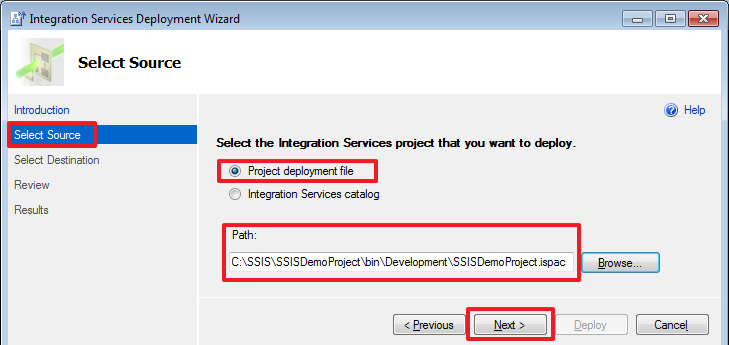
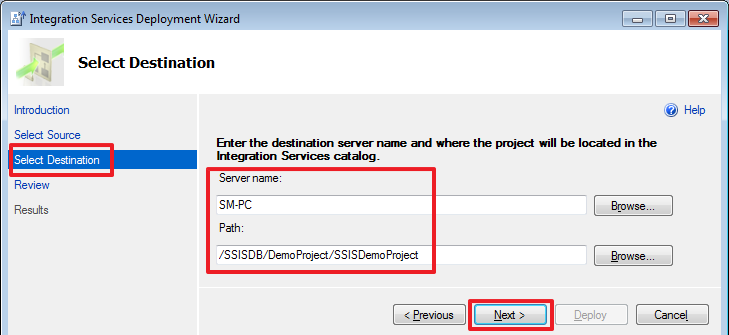
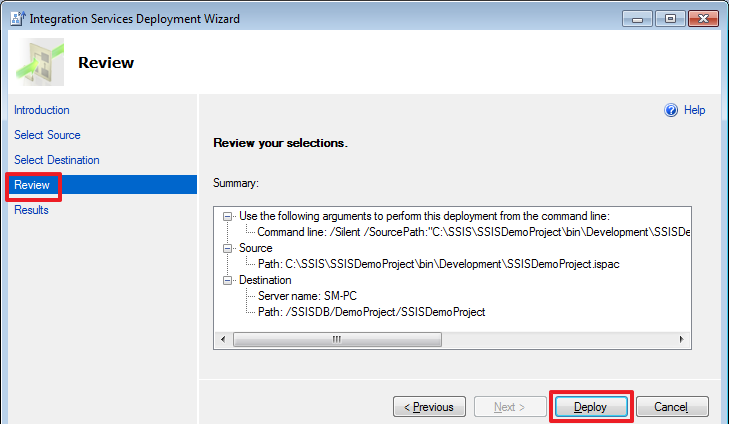
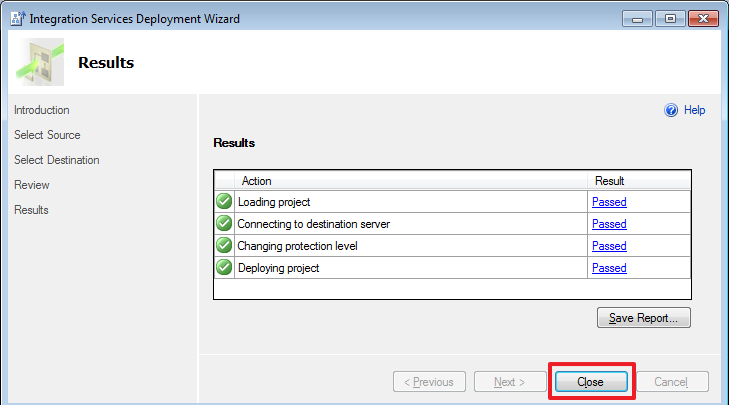
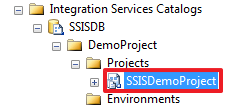
GO

SELECT \*

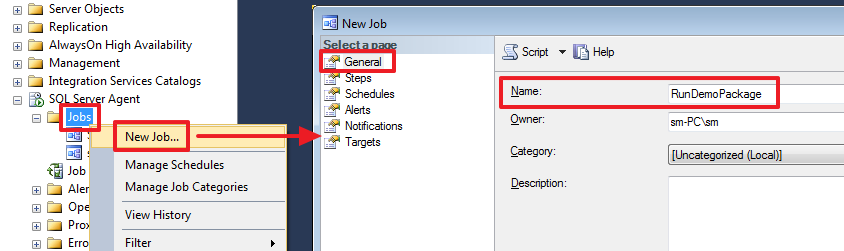
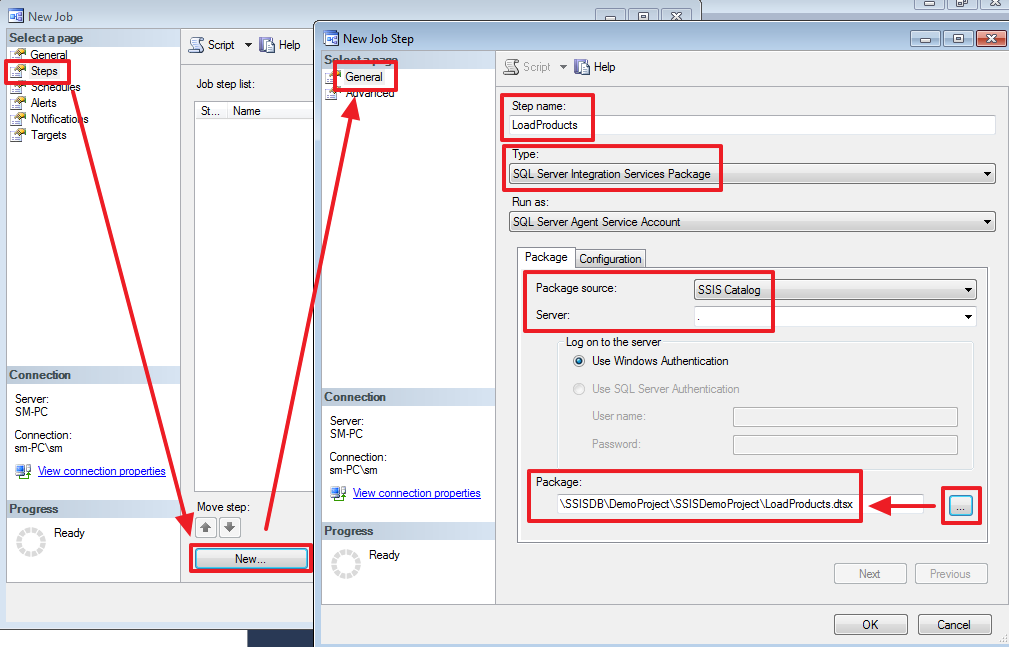
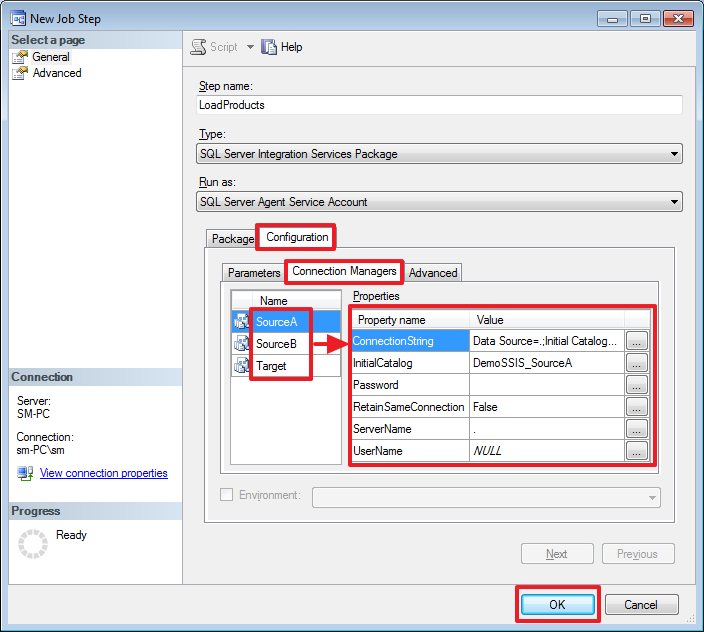
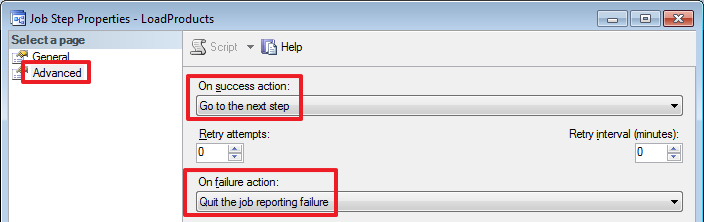
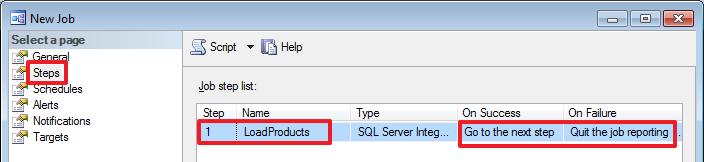
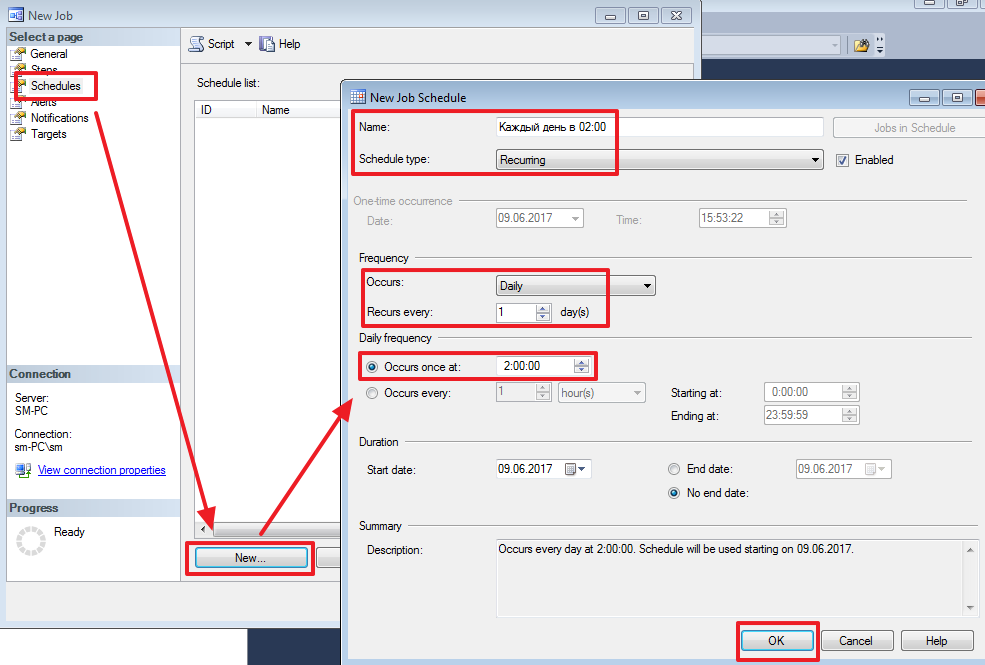
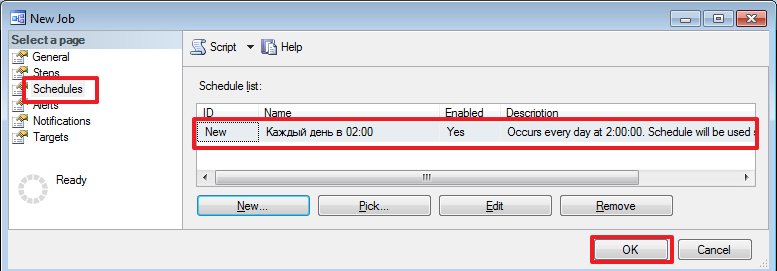
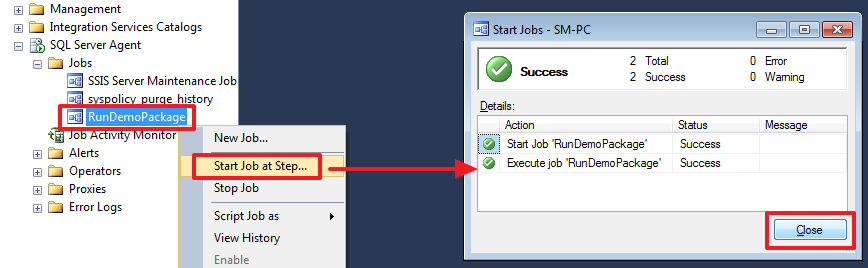
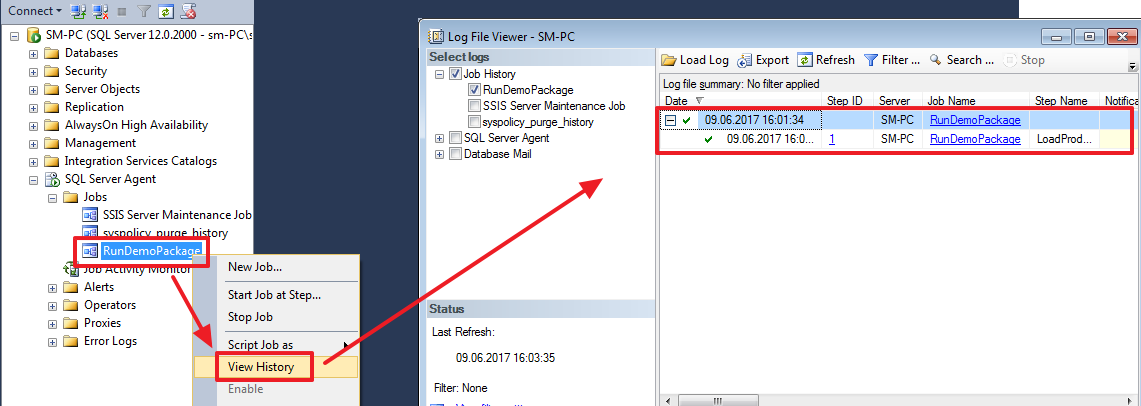
FROM Products

  
  
**Дополнительно в контекстном меню стрелки можно активизировать «Data Viewer»:**  
  
  
  
Теперь при запуске пакета на выполнение в этой точке будет сделана остановка и нам будут показаны данные этого потока:  
  
  
  
**Для продолжения выполнения пакета нужно нажать на кнопку со стрелкой или просто закрыть окно просмотра данных.**  
  
Для отключения этой функции в контекстном меню стрелки выбираем «Disable Date Viewer»:  
  
  
  
**Для первой части думаю этого будет достаточно.**  
  
Создадим сборку:  
  
  
  
В результате мы получим файл «C:\SSIS\SSISDemoProject\bin\Development\**SSISDemoProject.ispac**».  
  
**Рассмотрим каким образом делается развертывание этого проекта на SQL Server.**

Развертывание SSIS

Все последующие действия будем делать в SSMS.  
  
Создание каталога SSISDB:  
  
  
  
Здесь вводим любой пароль.  
  
Теперь создаем папку, в которой будет располагаться наш проект:  
  
  
  
Разворачиваем сам проект:  
  
  
  
  
  
  
  
  
  
В завершении мы должны увидеть следующую картину:  
  
  
  
**После обновления (F5) мы увидим наш проект:**  
  


Создание задачи в SQL Server Agent

Создадим задачу в SQL Agent, для выполнения пакета по расписанию:  
  
  
  
Создаем новый шаг:  
  
  
  
На вкладке «**Configuration → Parameters**» можно задать параметры пакета   
  
На вкладке «**Configuration → Connection Manager**» мы можем изменить параметры подключения для каждого соединения, которое мы создали в проекте:  
  
  
  
На закладке Advanced можно изменить логику, которая будет использоваться при успешном или неуспешном завершении шага:  
  
  
  
Шаг создан:  
  
  
  
Осталось создать расписание для данной задачи:  
  
  
  
Расписание можно задать разнообразным образом. Думаю, здесь все должно быть интуитивно понятно:  
  
  
  
Все, задача создана.  
  
**Делаем тестовый запуск:**  
  
  
*Так как шаг у нас всего один, то задача запустится сразу, иначе нужно было бы указать с какого шага нужно начать выполнение.*  
  
Результат выполнения задачи можно увидеть в следующем журнале:  
  
  
*В данном журнале можно увидеть успешность завершения каждого шага, а также время выполнения и прочие параметры.*  
  
**Более подробный отчет о выполнении пакета можно посмотреть при помощи следующего отчета:**  
  
