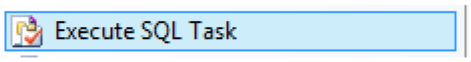
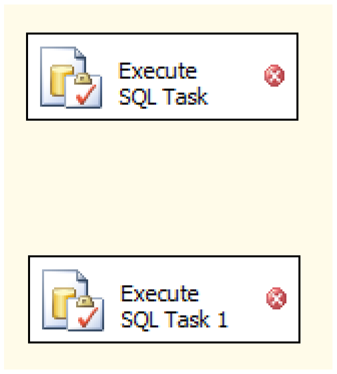
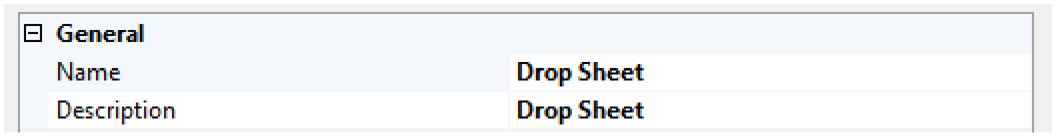
For this example we will need two Execute SQL tasks "Control Flow Items".



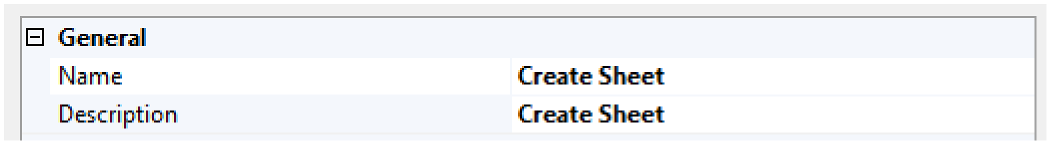
Drag Execute SQL Tasks to the Control Flow section two times.



At this point I would open each "Execute SQL Task" and name them appropriately. One is for dropping the Excel Sheet and the other is for Creating the Excel Sheet.

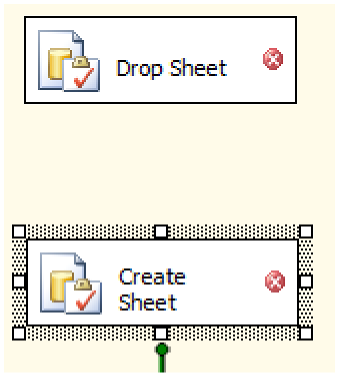


⊟ **General**	
Name	**Drop Sheet**
Description	**Drop Sheet**

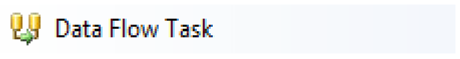


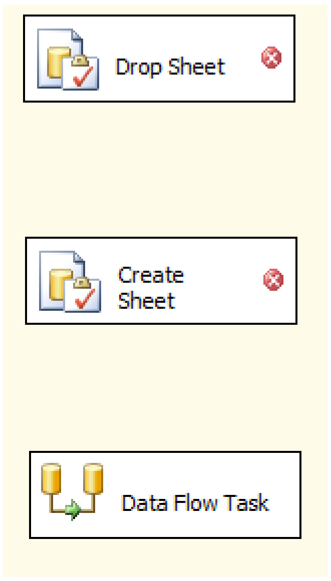
⊟ **General**	
Name	**Create Sheet**
Description	**Create Sheet**

You can double click on the "Execute SQL Task" and give the task a specific name.

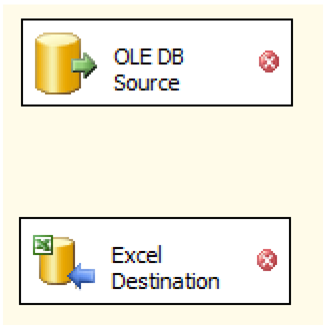


After giving each task a name it now will show in the Control Flow with its descriptive name. Now we can setup our Dataflow. Drag the "Data Flow Task" from the Control Flow Items to your Control Flow window.

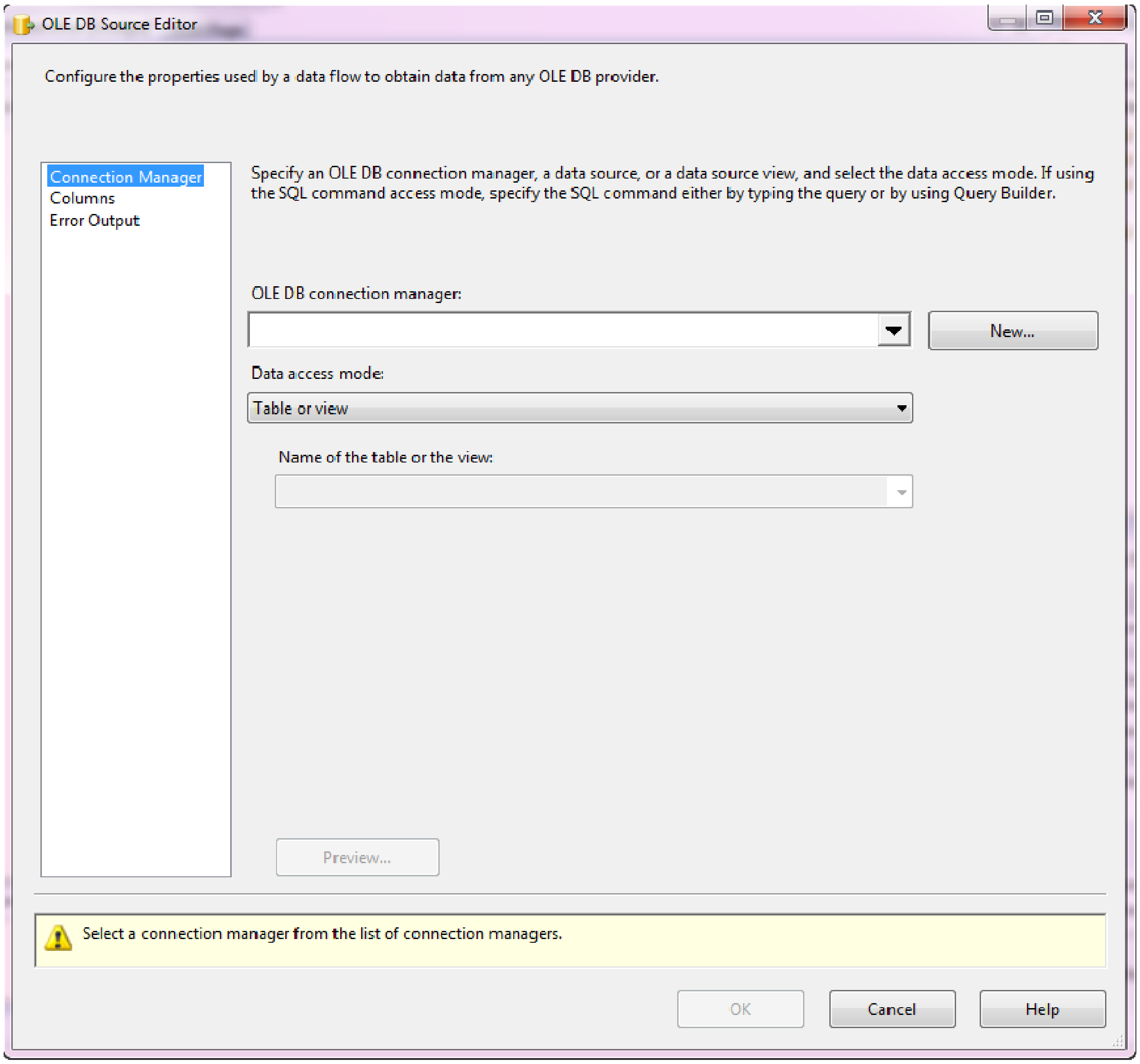




Now you can double click on the “Data Flow Task” and build your dataflow. We will need a source and a destination. For my example we will use a OLE DB Source and an Excel Destination.



Let’s double click on the OLE DB Source and configure it.



Configure the properties used by a data flow to obtain data from any OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder.

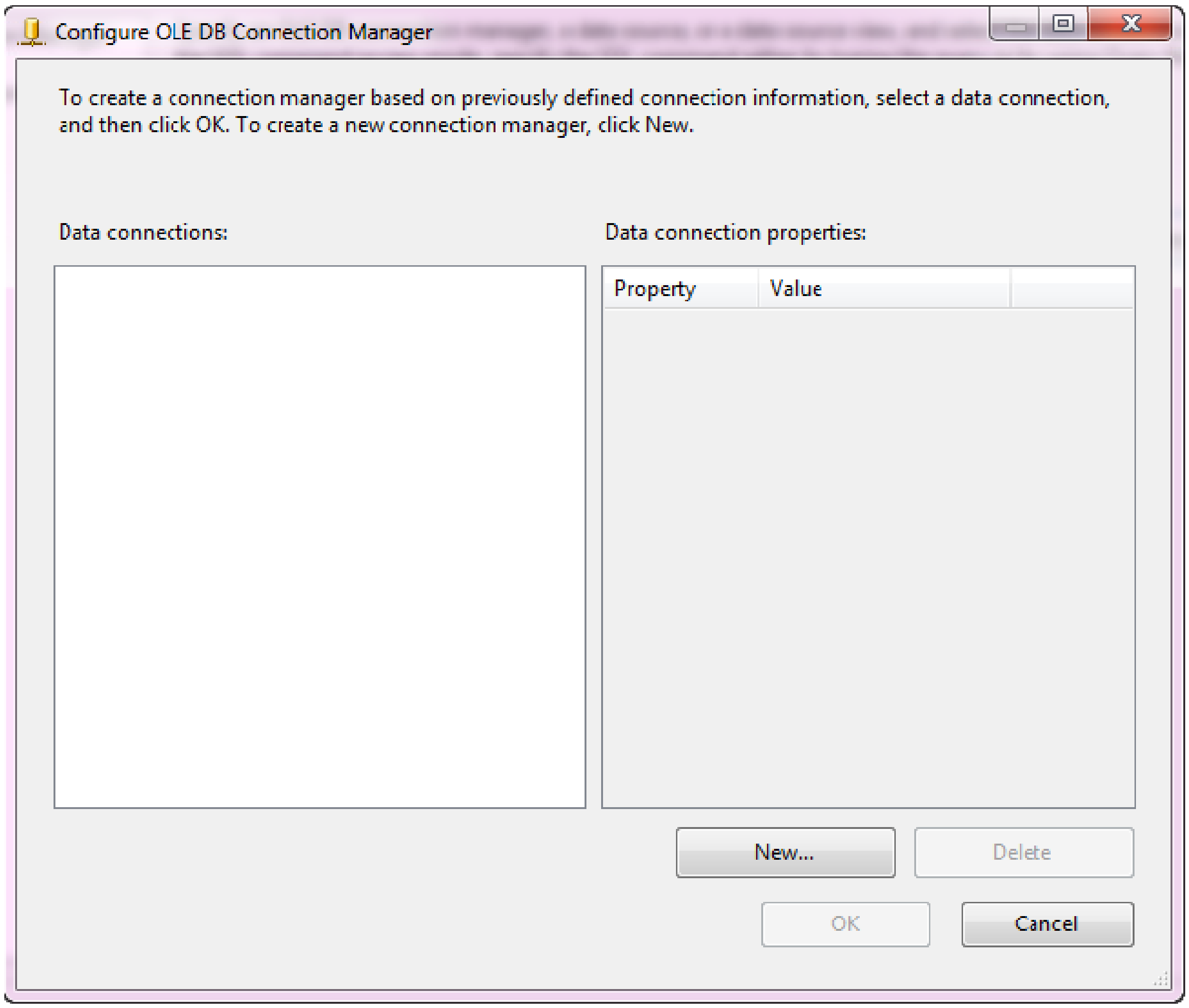
OLE DB connection manager:

Data access mode: Table or view

Name of the table or the view:

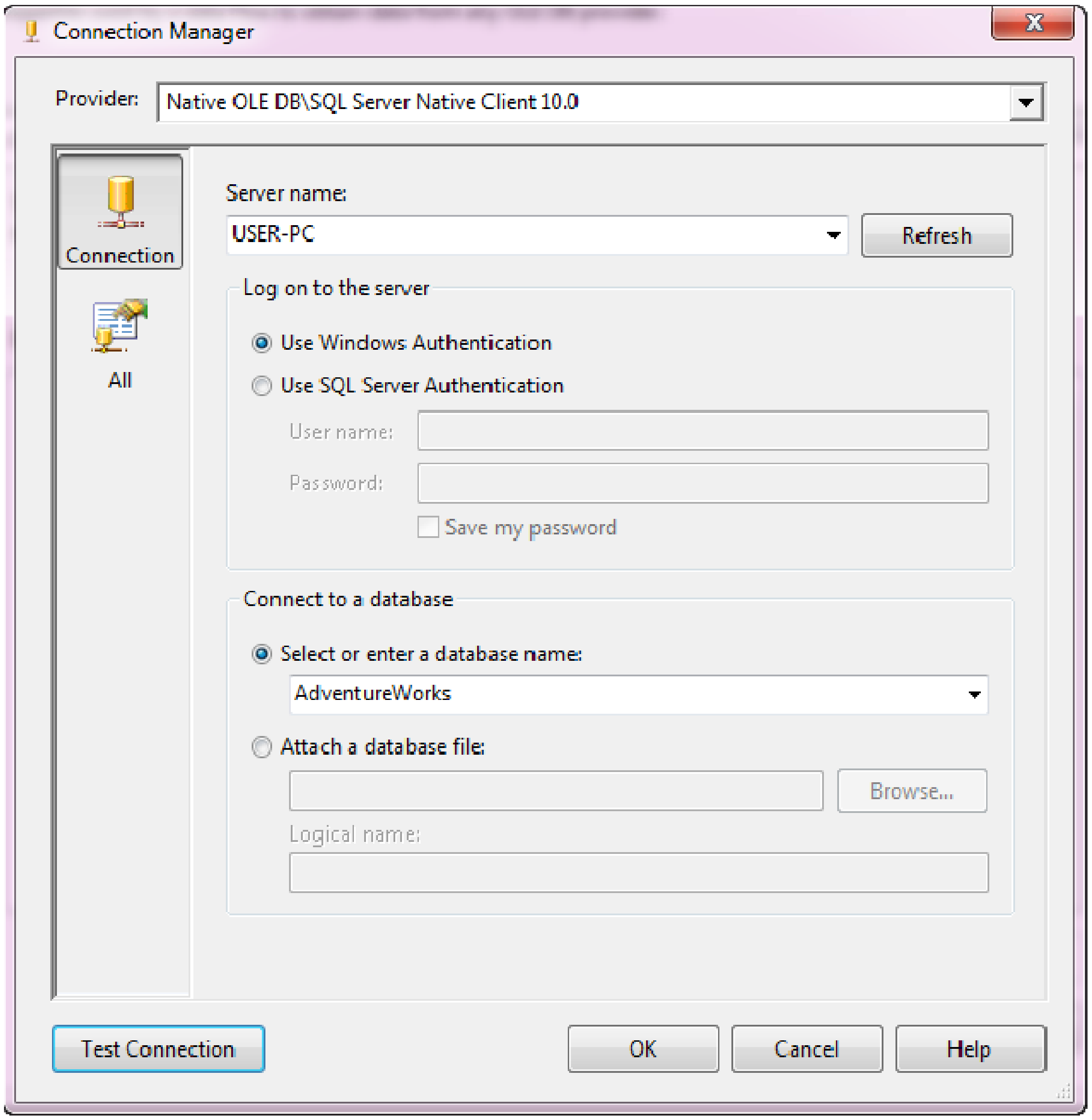
Select a connection manager from the list of connection managers.

For this example we will create a new connection so click NEW. A new dialog box will open.


Configure OLE DB Connection Manager

To create a connection manager based on previously defined connection information, select a data connection, and then click OK. To create a new connection manager, click New.

If you had any existing connections they would be listed here. Click New to create a new connection to your data source. For this example I will use AdventureWorks.



Provider: Native OLE DB\SQL Server Native Client 10.0

Server name: USER-PC

Log on to the server: [✓ Use Windows Authentication] Use SQL Server Authentication

User name:

Password:

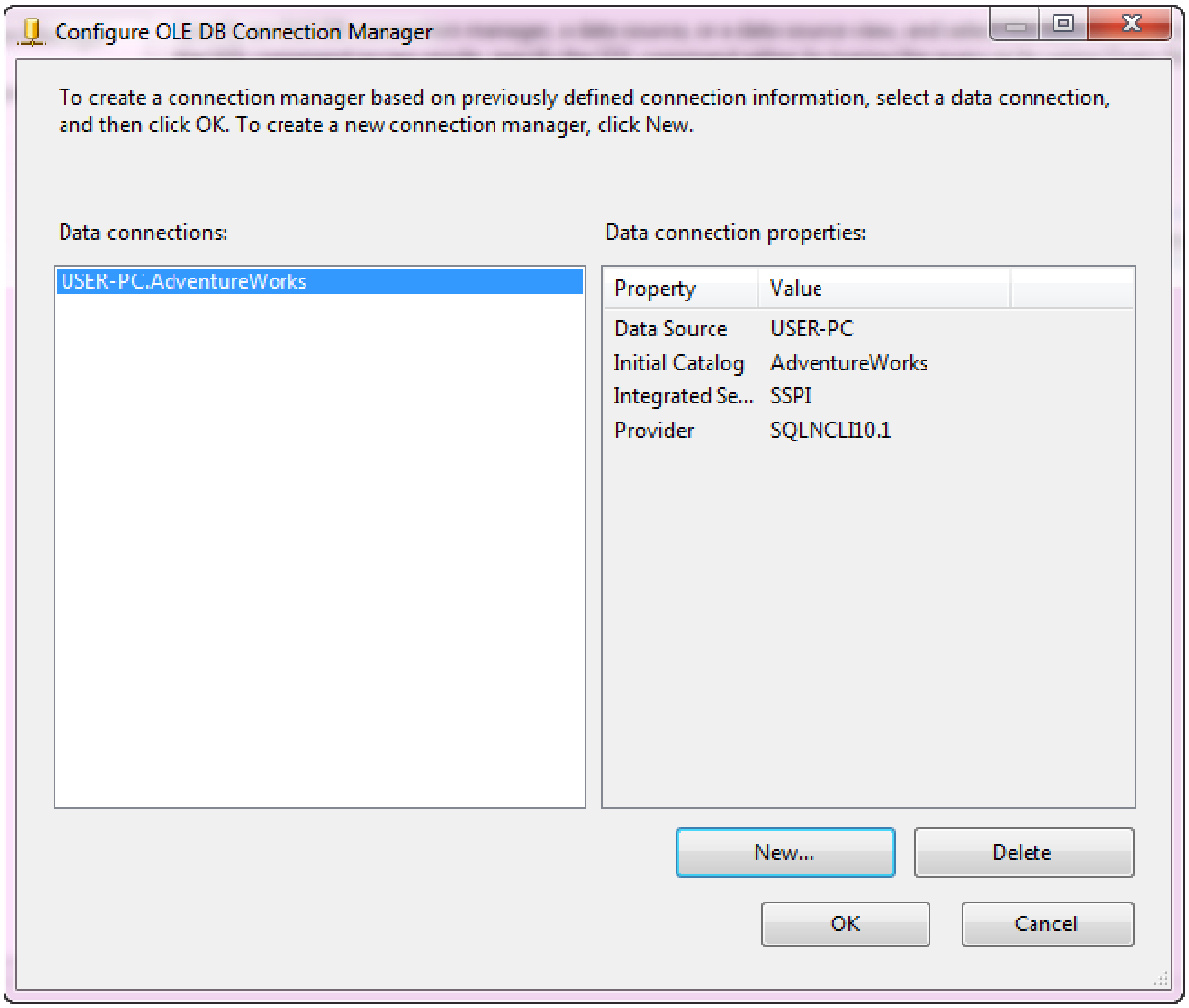
Save my password

Connect to a database: [✓ Select or enter a database name:] AdventureWorks

Attach a database file:

Logical name:

Type the name of your database server and your database. You can click “Test Connection” but it would be sort of redundant if you type the name of your server and use the drop down to find your database. Click OK to accept your settings.


Configure OLE DB Connection Manager

To create a connection manager based on previously defined connection information, select a data connection, and then click OK. To create a new connection manager, click New.

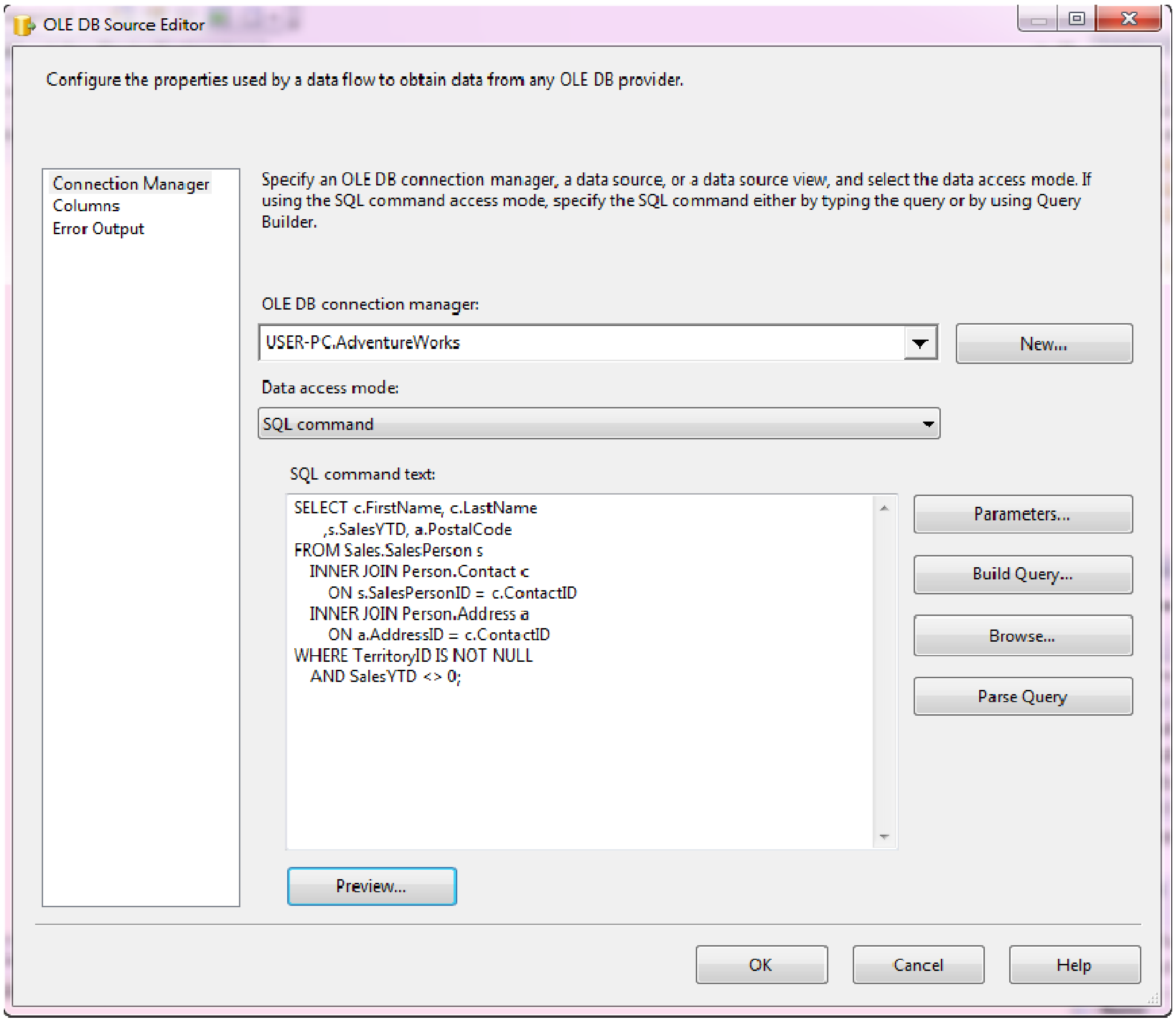
Data connections:

USER-PC.AdventureWorks

Data connection properties:

Property	Value
Data Source	USER-PC
Initial Catalog	AdventureWorks
Integrated Se...	SSPI
Provider	SQLNCLI10.1

Click OK



Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder.

OLE DB connection manager: USER-PC.AdventureWorks

Data access mode: SQL command

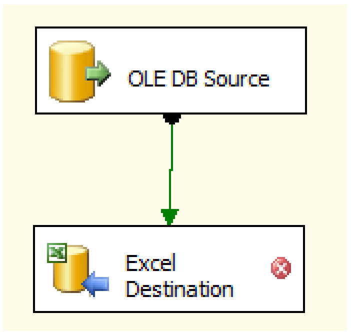
SQL command text:

```
SELECT c.FirstName, c.LastName
    ,s.SalesYTD, a.PostalCode
FROM Sales.SalesPerson s
  INNER JOIN Person.Contact c
     ON s.SalesPersonID = c.ContactID
  INNER JOIN Person.Address a
     ON a.AddressID = c.ContactID
WHERE TerritoryID IS NOT NULL
  AND SalesYTD <> 0;
```

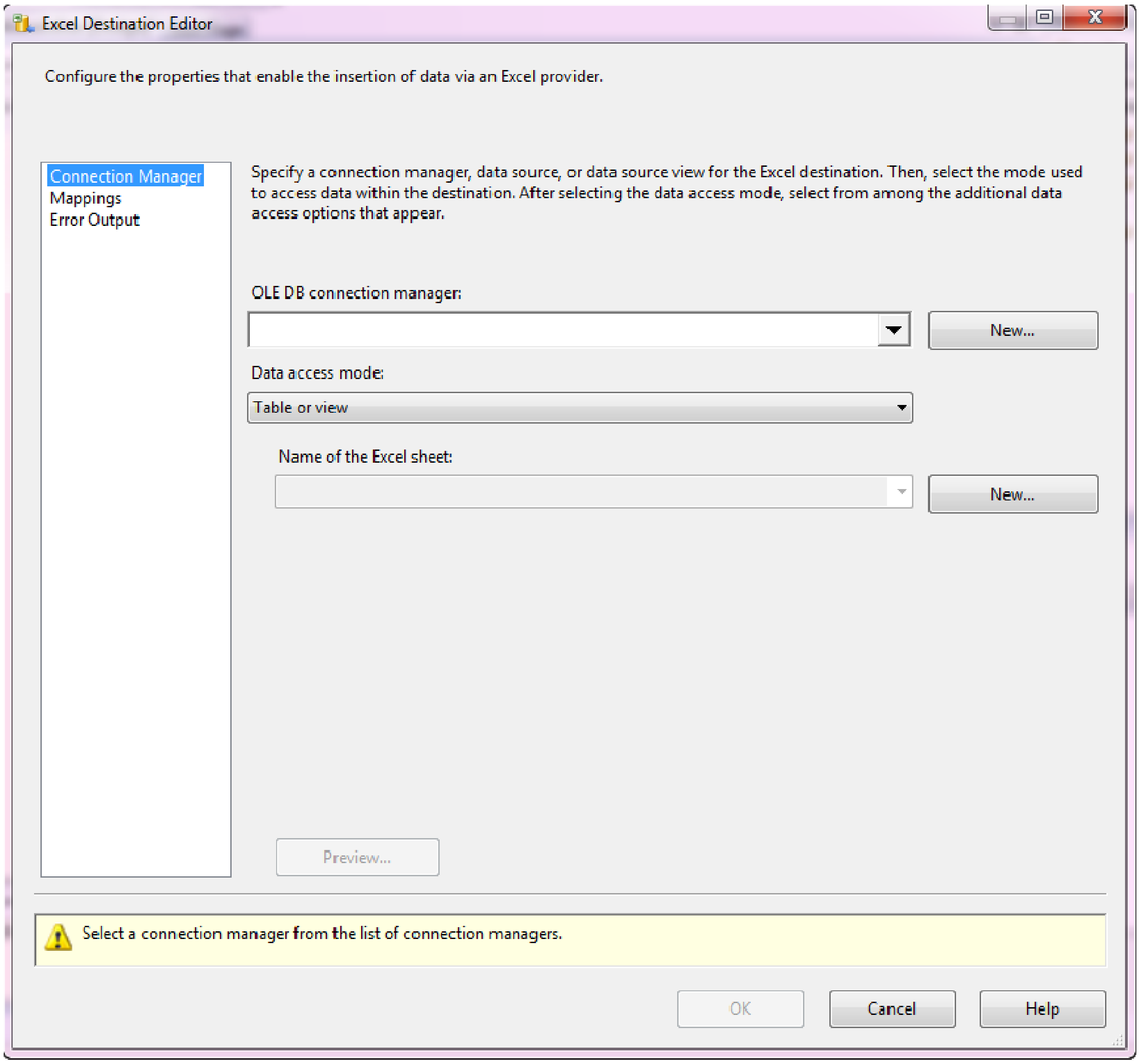
For this example we will change the “Data Access mode” to “SQL command”. You can now paste your query into the “SQL command text” window. For this example my query is

```
SELECT c.FirstName, c.LastName
        ,s.SalesYTD, a.PostalCode
FROM Sales.SalesPerson s
    INNER JOIN Person.Contact c
        ON s.SalesPersonID = c.ContactID
    INNER JOIN Person.Address a
        ON a.AddressID = c.ContactID
WHERE TerritoryID IS NOT NULL
    AND SalesYTD <> 0;
```

I typically click Preview to ensure everything is good with my query and the results come back as expected.



Drag the green arrow to your “Excel Destination”.



Configure the properties that enable the insertion of data via an Excel provider.

Specify a connection manager, data source, or data source view for the Excel destination. Then, select the mode used to access data within the destination. After selecting the data access mode, select from among the additional data access options that appear.

OLE DB connection manager:

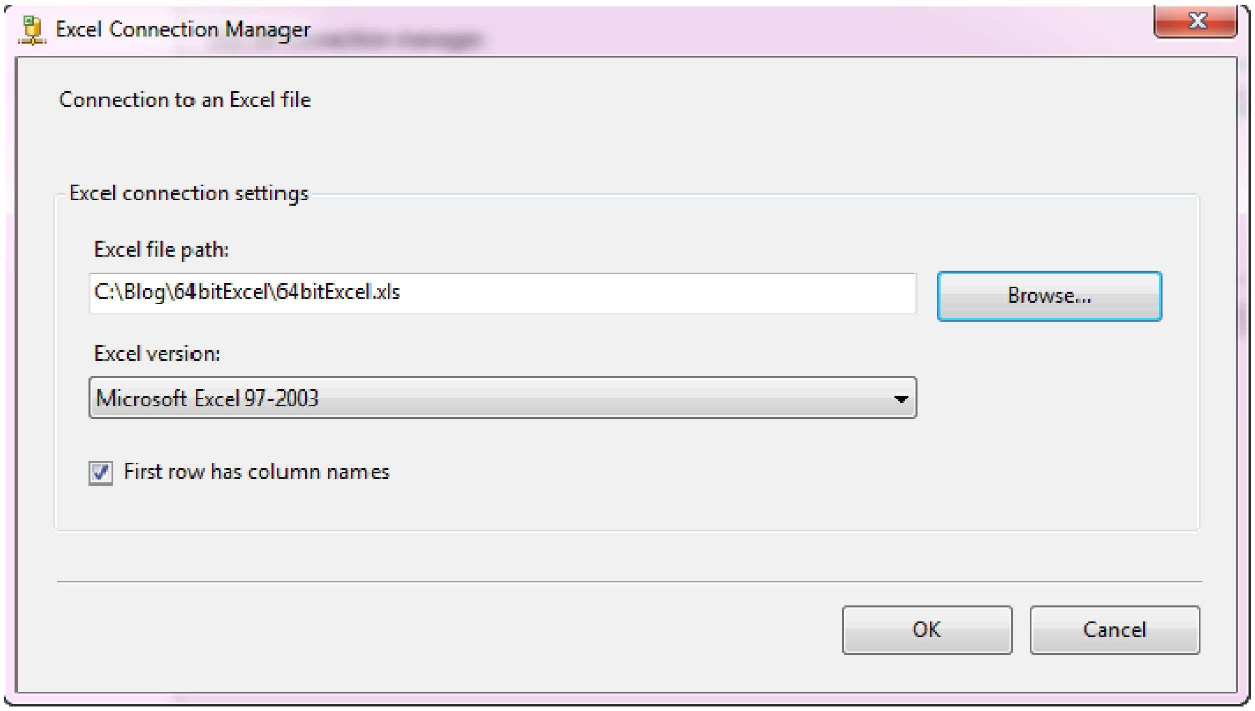
Data access mode:

Table or view

Name of the Excel sheet:

Select a connection manager from the list of connection managers.

Double click your “Excel Destination”. We will be creating a new Excel file. Click New.


Excel Connection Manager

Connection to an Excel file

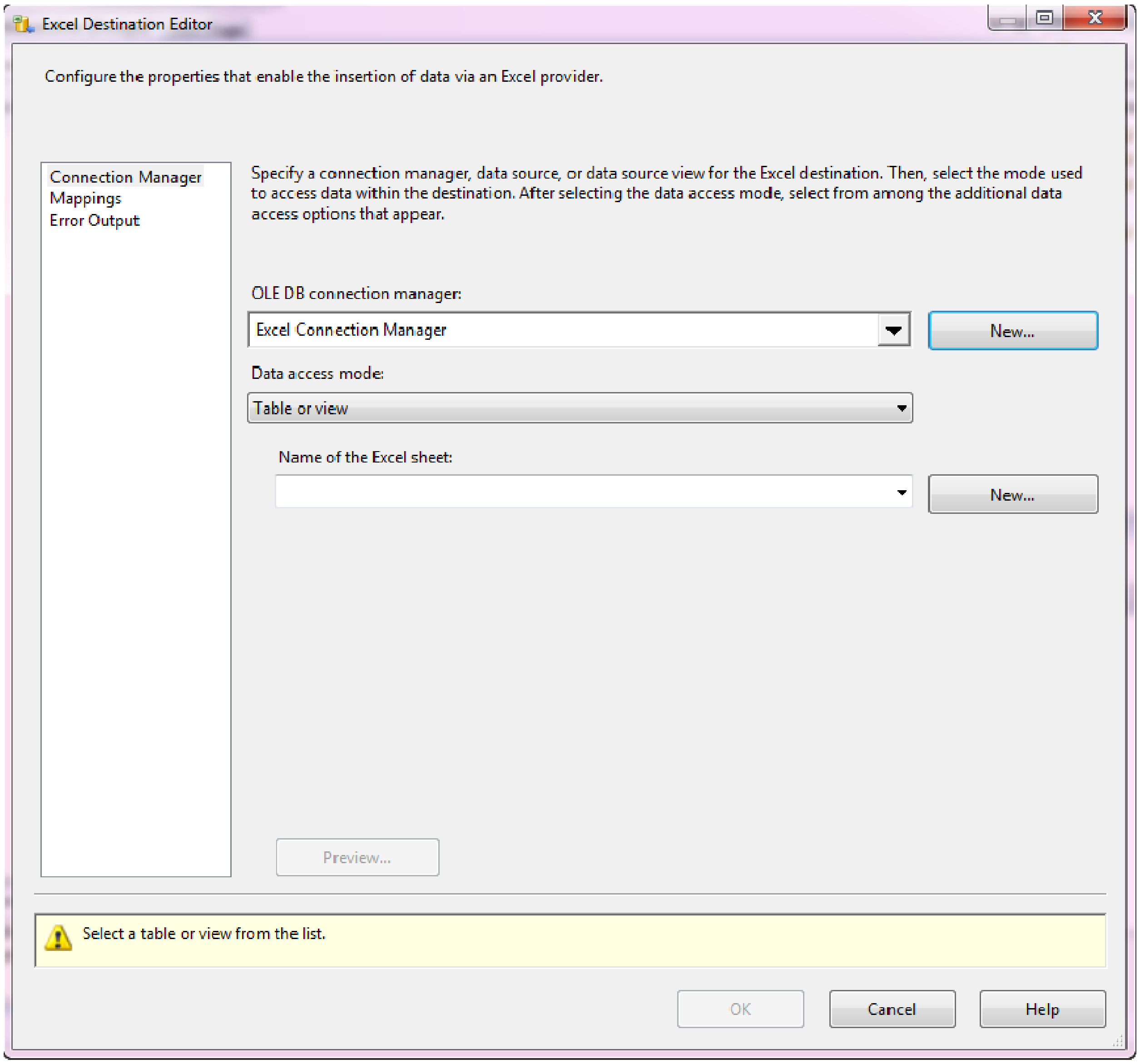
Excel connection settings

Excel file path: C:\Blog\64bitExcel\64bitExcel.xls

Excel version: Microsoft Excel 97-2003

[✓] First row has column names

Type the path and name of your Excel file and click OK.



Configure the properties that enable the insertion of data via an Excel provider.

Specify a connection manager, data source, or data source view for the Excel destination. Then, select the mode used to access data within the destination. After selecting the data access mode, select from among the additional data access options that appear.

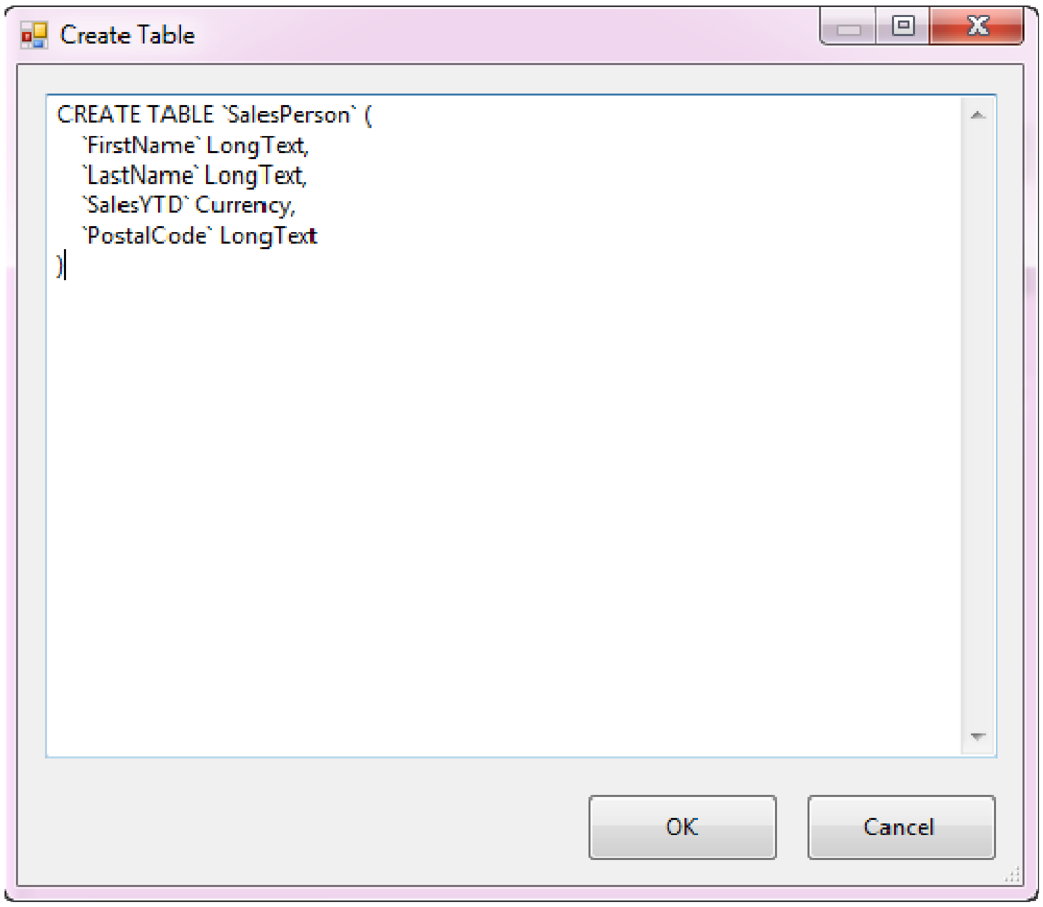
OLE DB connection manager: Excel Connection Manager

Data access mode: Table or view

Name of the Excel sheet:

Select a table or view from the list.

Next we will click on NEW to create our Excel Sheet.

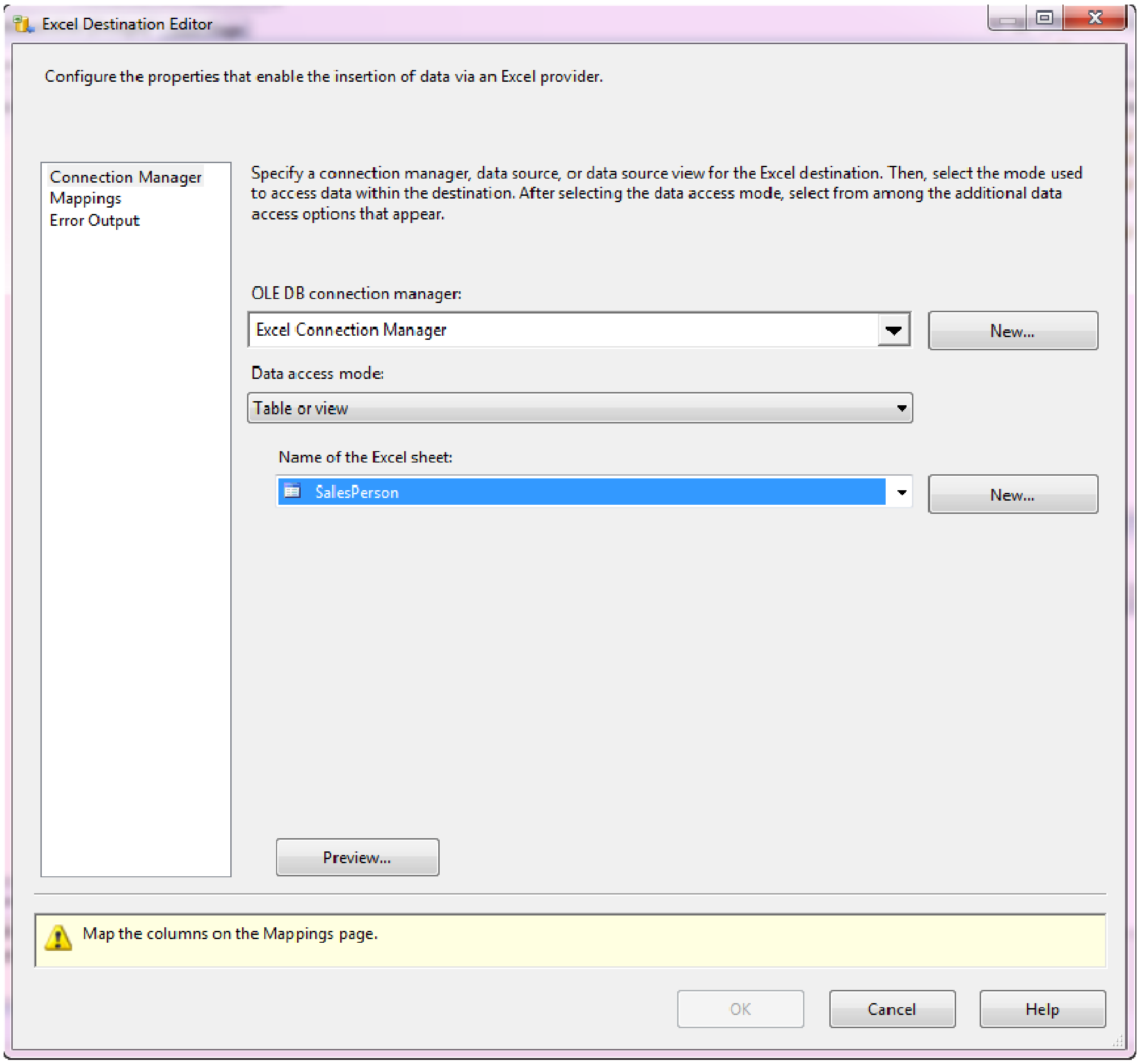


```
CREATE TABLE `SalesPerson` (
    `FirstName` LongText,
    `LastName` LongText,
    `SalesYTD` Currency,
    `PostalCode` LongText
)
```

I always rename the Worksheet to something specific. This really helps out if you have multiple worksheets that you are populating. That will be another blog later. Make sure to select and copy the code to create the worksheet. You will need this for your Create Sheet SQL Task on the Control Flow.

```
CREATE TABLE `SalesPerson` (

   `FirstName` LongText,

   `LastName` LongText,

   `SalesYTD` Currency,

   `PostalCode` LongText

)
```

Click OK.



Configure the properties that enable the insertion of data via an Excel provider.

Specify a connection manager, data source, or data source view for the Excel destination. Then, select the mode used to access data within the destination. After selecting the data access mode, select from among the additional data access options that appear.

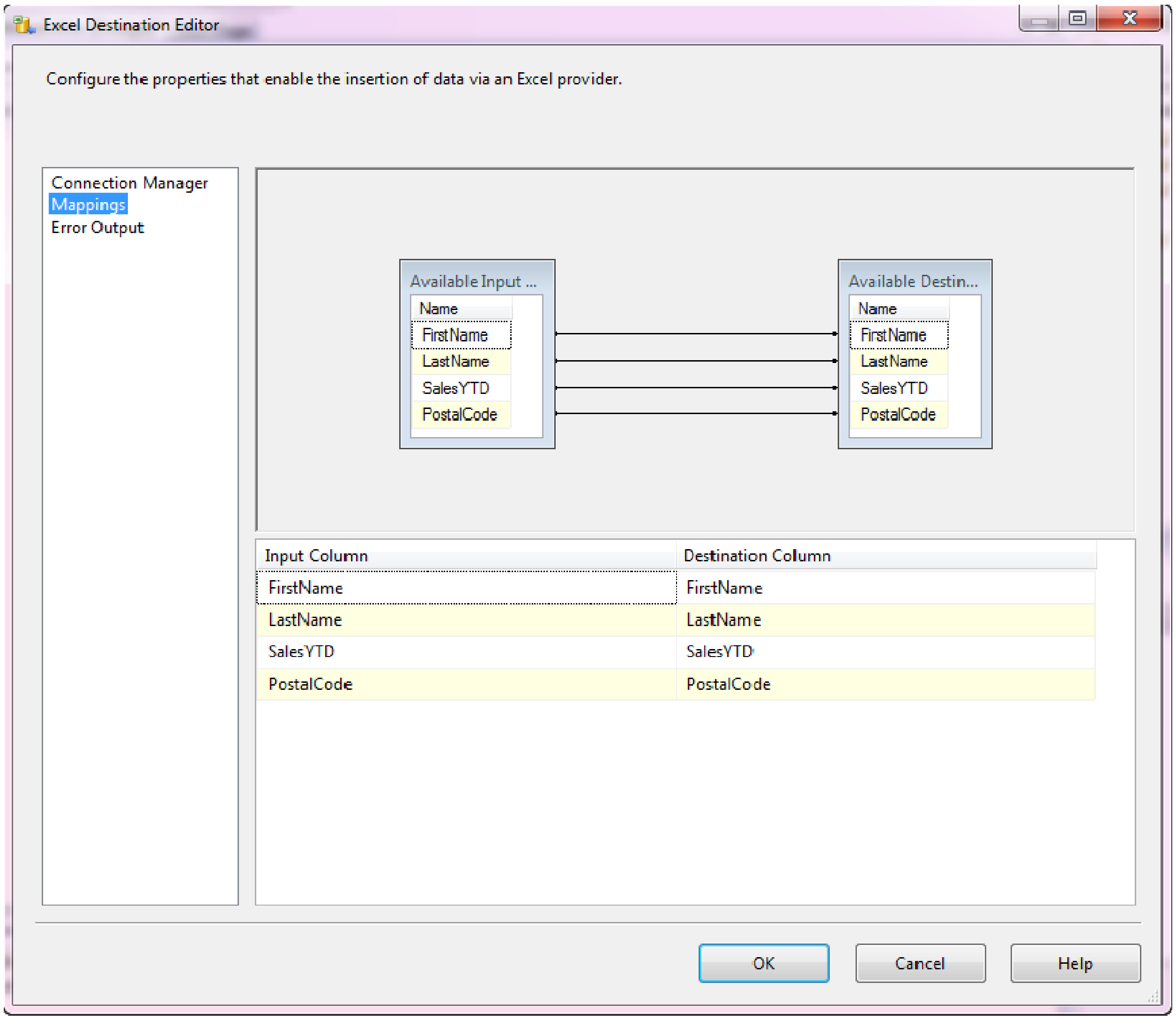
OLE DB connection manager: Excel Connection Manager

Data access mode: Table or view

Name of the Excel sheet: SalesPerson

Map the columns on the Mappings page.

Now click on Mappings.

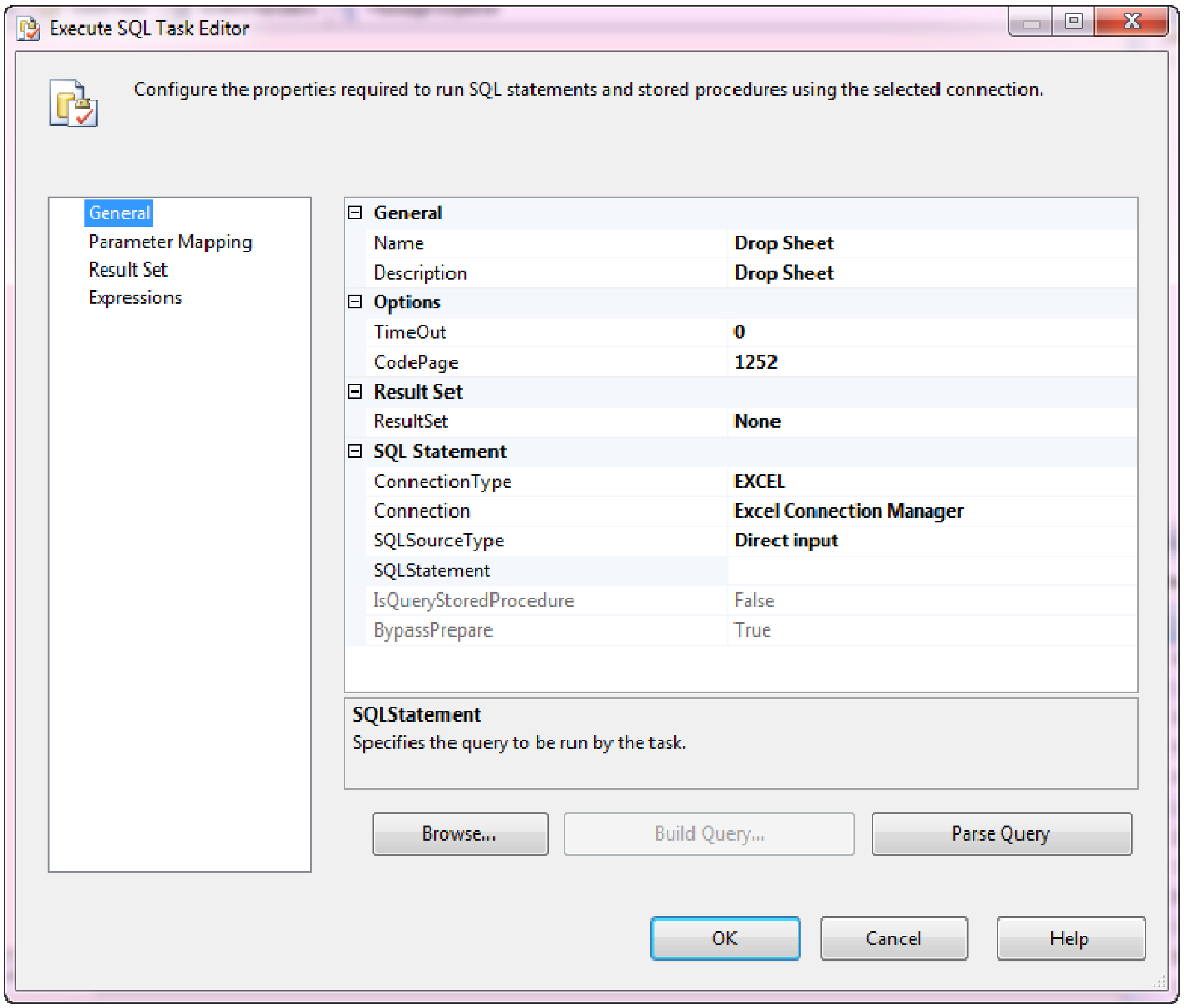

Excel Destination Editor

Configure the properties that enable the insertion of data via an Excel provider.

Input Column	Destination Column
FirstName	FirstName
LastName	LastName
SalesYTD	SalesYTD
PostalCode	PostalCode

Click OK.

Click on the Control Flow tab. Control Flow and double click the “Drop Sheet” SQL Task.

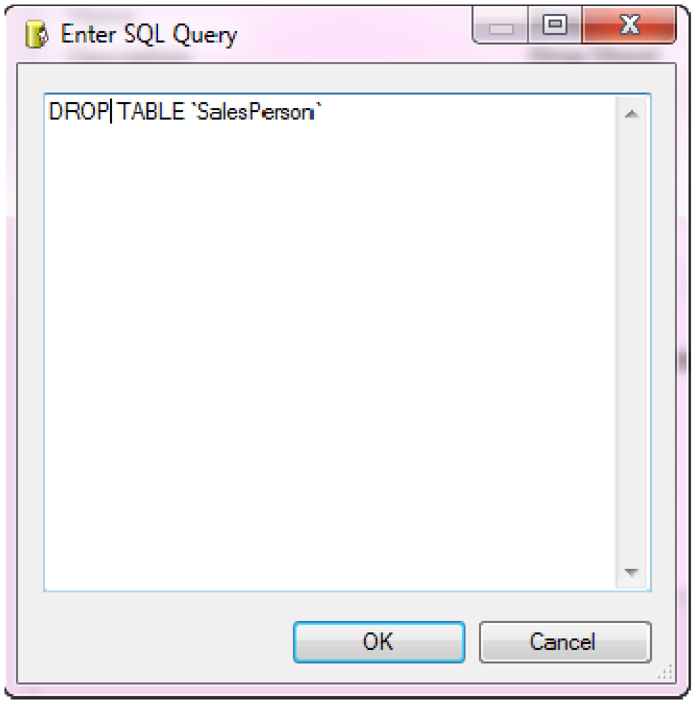

Execute SQL Task Editor

Configure the properties required to run SQL statements and stored procedures using the selected connection.

General	
Name	**Drop Sheet**
Description	**Drop Sheet**
Options	
TimeOut	**0**
CodePage	**1252**
Result Set	
ResultSet	**None**
SQL Statement	
ConnectionType	**EXCEL**
Connection	**Excel Connection Manager**
SQLSourceType	**Direct input**
SQLStatement	
IsQueryStoredProcedure	False
BypassPrepare	True

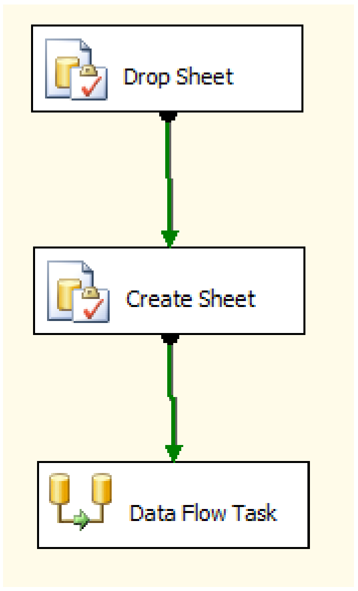
SQLStatement
Specifies the query to be run by the task.

Change the Connection Type to EXCEL, click the down arrow on Connection and chose your Excel Connection Manager. The SQLSourceType should be Direct Input. Now click the elipses (…) next to SQL Statement so you can paste the Create statement from the Excel Destination.



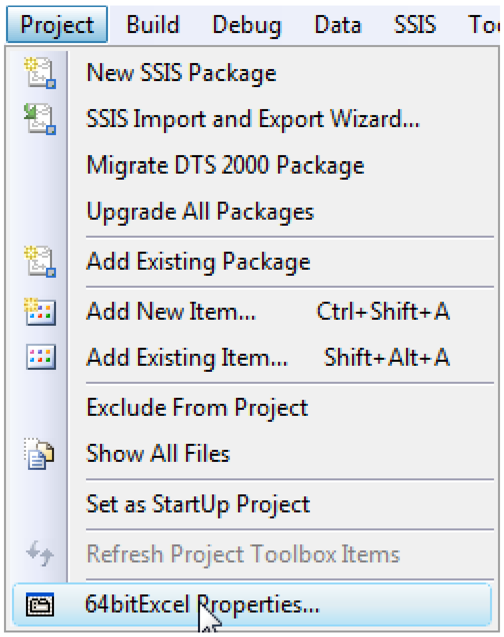
```
DROP TABLE `SalesPerson`
```

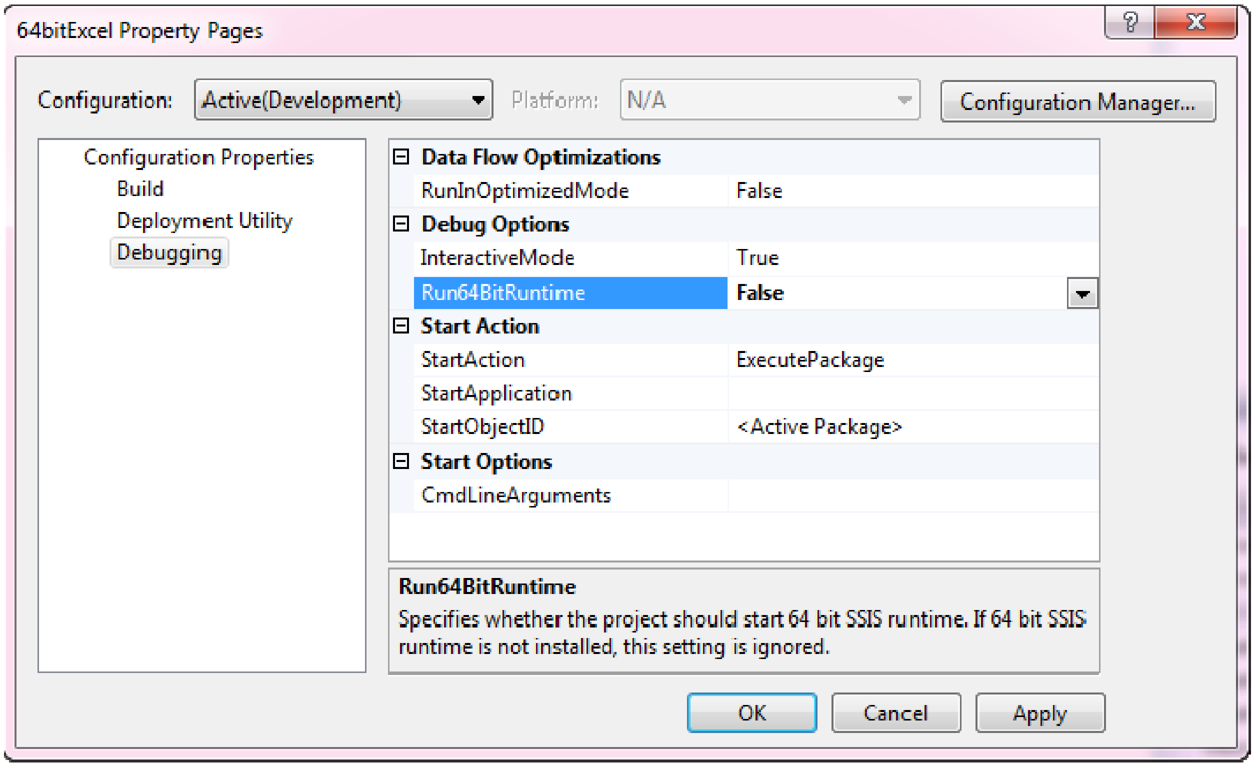
Change “create” to “drop” and remove the reference to the columns.   Now click OK and then OK again. Now double click on the Drop Sheet SQL Task. Repeat the steps above but this time in the SQL Statement paste the entire create table query.



Connect your Drop Sheet task to your Create Sheet and then your Create Sheet to your Data Flow Task.

Now since we are running on an x64 platform we need to set debugging to allow us to debug the package. Click Project and chose the name of the package.





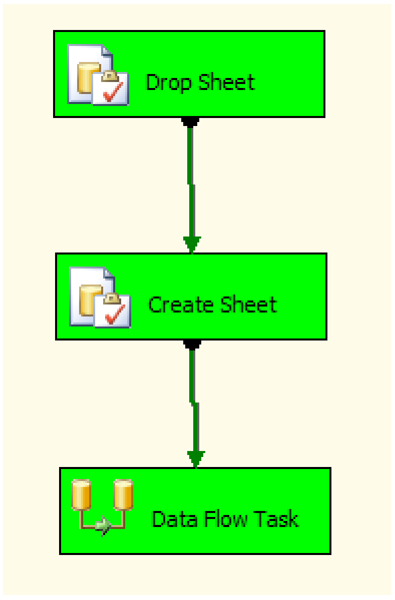
Configuration: Active(Development)
Platform: N/A

Data Flow Optimizations	
RunInOptimizedMode	False
Debug Options	
InteractiveMode	True
Run64BitRuntime	**False**
Start Action	
StartAction	ExecutePackage
StartApplication	
StartObjectID	<Active Package>
Start Options	
CmdLineArguments	

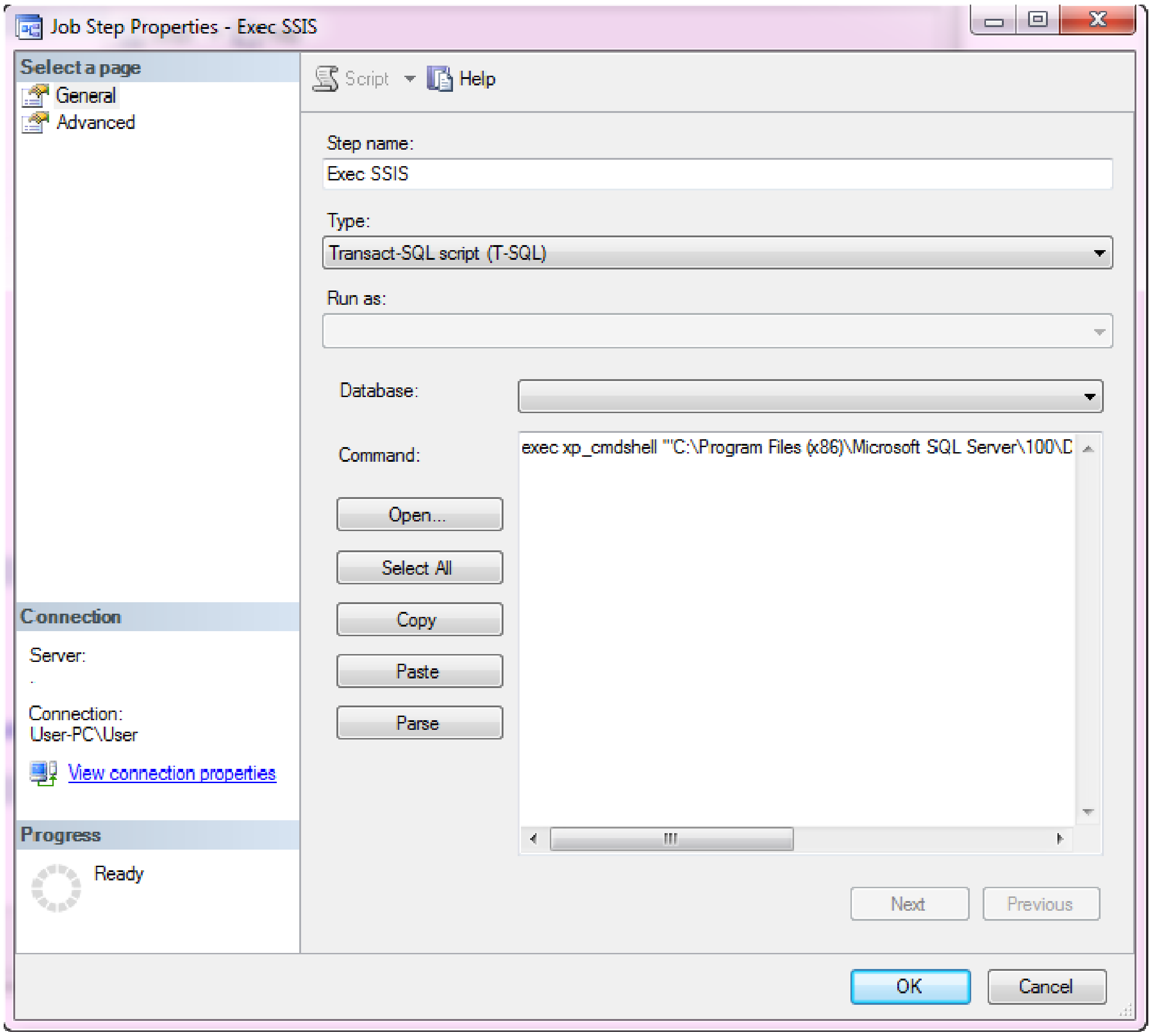
Run64BitRuntime
Specifies whether the project should start 64 bit SSIS runtime. If 64 bit SSIS runtime is not installed, this setting is ignored.

Change the “Run64BitRuntime” from True to False and click OK.

Now execute your Package.



You should now save your work if you haven't been saving as you went along. Next you will most likely want to schedule this package to run as a SQL Job. To schedule this to run, you will have to invoke SSIS in 32 bit mode. To do so you will have to call it from command line.



Step name: Exec SSIS

Type: Transact-SQL script (T-SQL)

Run as:

Database:

Command:

```
exec xp_cmdshell '"C:\Program Files (x86)\Microsoft SQL Server\100\D
```

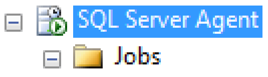
Connection

Server: .

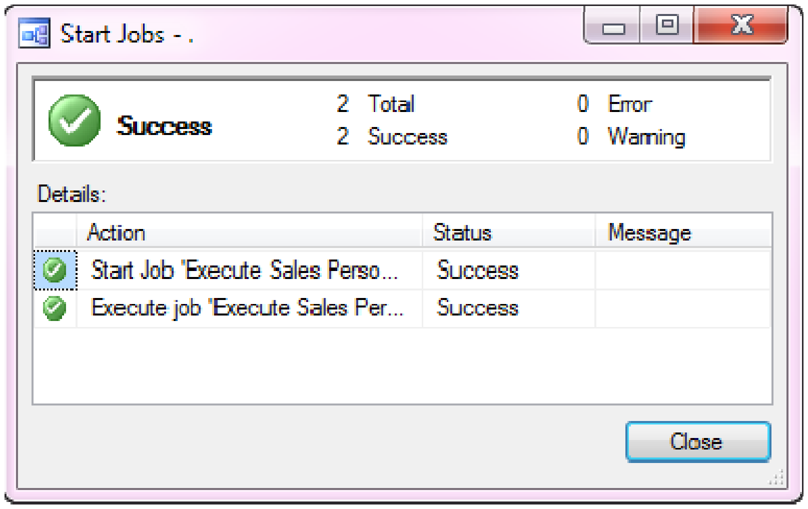
Connection: User-PC\User

Progress

Ready

Lets get started. I am not going to show each step for creating a SQL job, you should know this. On my system I right clicked on my jobs folder  and chose “New Job”. I gave the job a name and then clicked steps. I named the step and change the Type to “Transact-SQL script (T-SQL) since we will be using xp_cmdshell to call the 32bit version of SSIS. The syntax of the command is below. Since I am running SQL 2008 R2 Developer I chose \100. If you are running SQL 2005 it will be \90\.

```
exec xp_cmdshell '"C:\Program Files (x86)\Microsoft SQL
Server\100\DTS\Binn\dtexec.exe" /f C:\Blog\64bitExcel\64bitExcel.dtsx'
```



Start Jobs - .

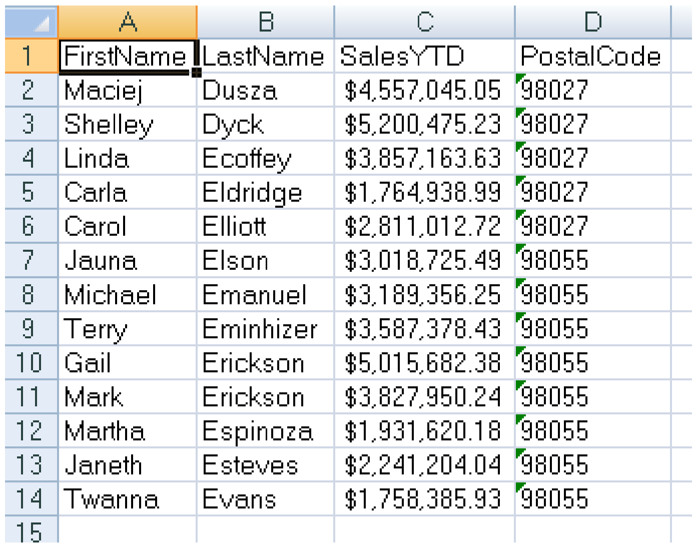
Success

2 Total
2 Success
0 Error
0 Warning

Details:

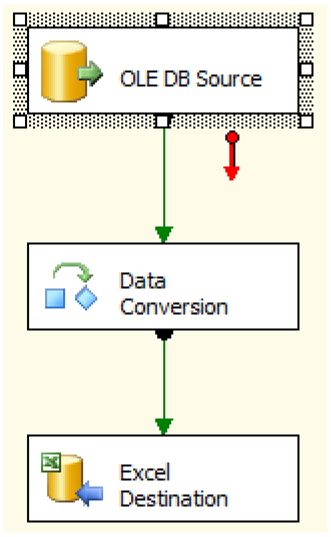
Action	Status	Message
Start Job 'Execute Sales Perso...	Success	
Execute job 'Execute Sales Per...	Success	

I can then browse to my folder where I placed the Excel file and view my data.

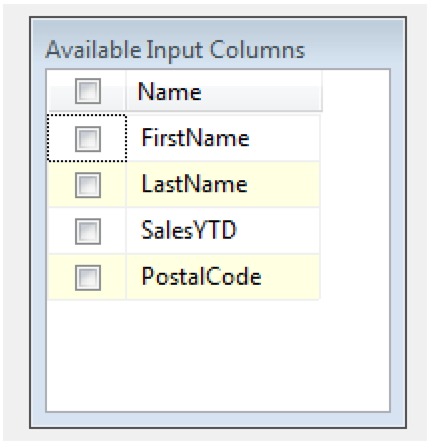


	A	B	C	D
1	FirstName	LastName	SalesYTD	PostalCode
2	Maciej	Dusza	$4,557,045.05	98027
3	Shelley	Dyck	$5,200,475.23	98027
4	Linda	Ecoffey	$3,857,163.63	98027
5	Carla	Eldridge	$1,764,938.99	98027
6	Carol	Elliott	$2,811,012.72	98027
7	Jauna	Elson	$3,018,725.49	98055
8	Michael	Emanuel	$3,189,356.25	98055
9	Terry	Eminhizer	$3,587,378.43	98055
10	Gail	Erickson	$5,015,682.38	98055
11	Mark	Erickson	$3,827,950.24	98055
12	Martha	Espinoza	$1,931,620.18	98055
13	Janeth	Esteves	$2,241,204.04	98055
14	Twanna	Evans	$1,758,385.93	98055
15				

Something else worth mentioning is when you export data to Excel, you might have to use a "Data Conversion" task Data Conversion located in the Data Flow Transformations toolbox **Data Flow Transformations**. You will need to drag the Data Conversion task onto your Data Flow Task screen and place it between your OLE Source and your Excel Destination.



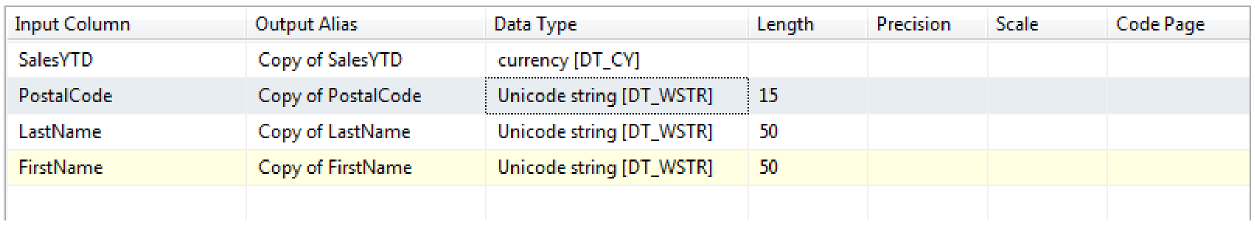
To do this highlight your green arrow and press delete. Then line up your task and connect the Green arrow from OLE DB Source to the Data Conversion task, then the Data Conversion task to your Excel Destination.



☐	Name
☐	FirstName
☐	LastName
☐	SalesYTD
☐	PostalCode

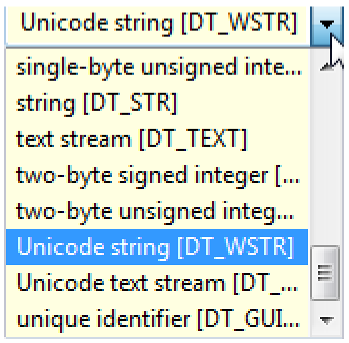
Double click the Data Conversion task and it will bring up a box for you to check which columns you want to convert.

In my example I am going to check each one of them.

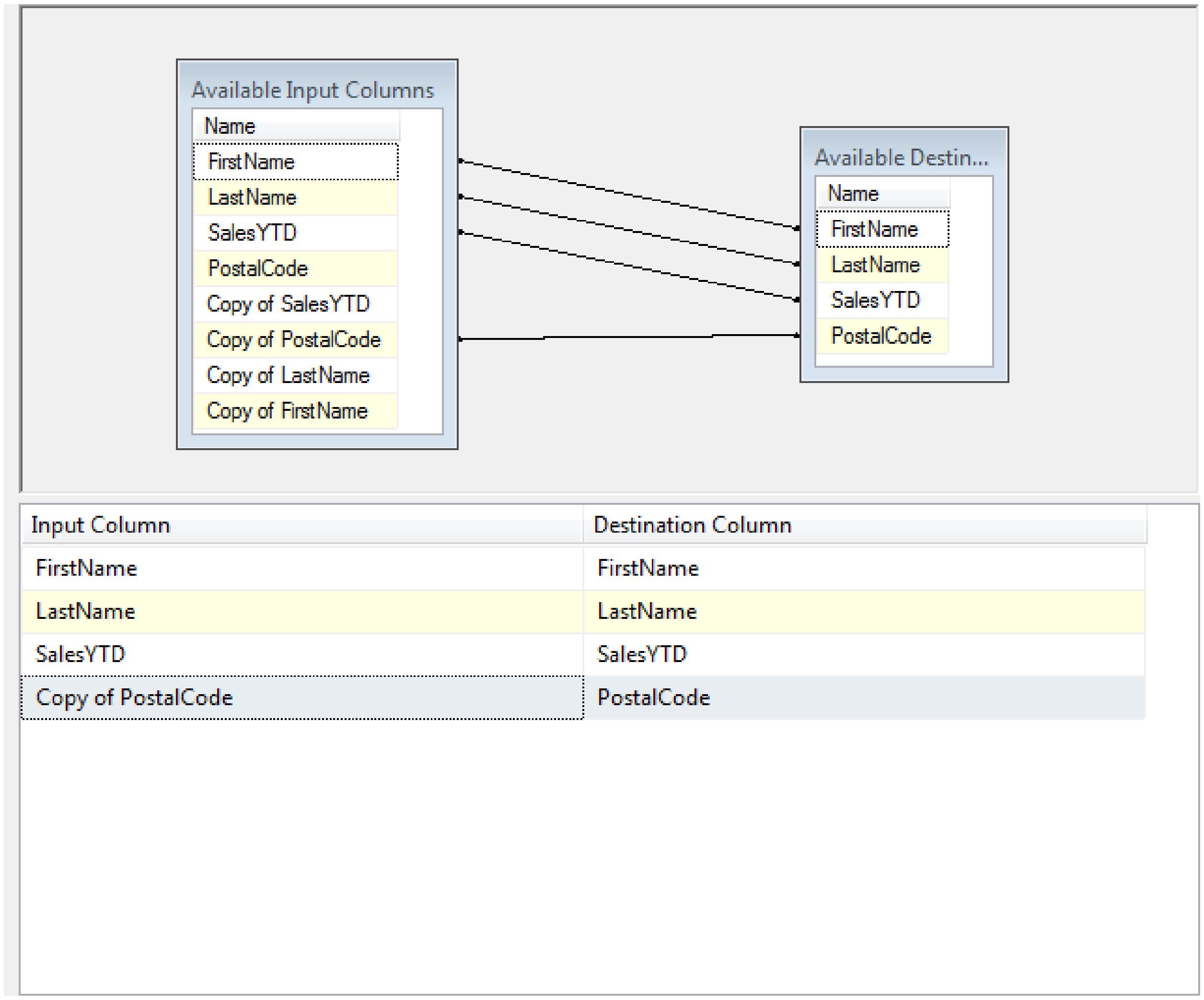


Input Column	Output Alias	Data Type	Length	Precision	Scale	Code Page
SalesYTD	Copy of SalesYTD	currency [DT_CY]				
PostalCode	Copy of PostalCode	Unicode string [DT_WSTR]	15			
LastName	Copy of LastName	Unicode string [DT_WSTR]	50			
FirstName	Copy of FirstName	Unicode string [DT_WSTR]	50			

They are then added to the bottom of your window. At this point you can click the drop down and change the Data Type. This will be very helpful for many datasets. You will find dealing with strings to be quite frustrating. If you click the down arrow you will see something like this.


Unicode string [DT_WSTR]
single-byte unsigned inte...
string [DT_STR]
text stream [DT_TEXT]
two-byte signed integer [...
two-byte unsigned integ...
Unicode string [DT_WSTR]
Unicode text stream [DT_...
unique identifier [DT_GUI...

There are lots of choices here.



Input Column	Destination Column
FirstName	FirstName
LastName	LastName
SalesYTD	SalesYTD
Copy of PostalCode	PostalCode

If you have to do a data conversion the default name of the “newly converted column” is Copy of column_name. For the purpose of demo I have changed my mapping for the PostalCode to be the converted column.

That is all there is to it. Once you suffer through creating your first Excel destination on an x64 system it really doesn’t seem that difficult. I have heard a lot of grumbling about how tedious SSIS can be to work with but with great control and precision come a few extra steps of configuration. Once you create a few of these it will seem like child’s play.