Importing VFP data into SQL Server using SQL Server Integration Services (SSIS)

By Craig Bailey
www.craigbailey.net

In this article we will be investigating SQL Server 2005 Integration Services (SSIS), as a mechanism for importing VFP data.
We will be covering how to set up the Integration project, handle errors, debug the process, and then set it up as a SQL Agent job to run automatically.

For the purpose of the article we will use the Northwind database as the data, but at the end of the article, we'll cover off a real world implementation and the amazing results achieved.

Here's the details we'll go through:

Contents

SSIS versus VFP's SQL Upsizing Wizard

Before commencing let's look at why we would want to use SSIS instead of say the VFP Upsizing Wizard. There are two main considerations:

1. Performance
2. Database RI

As we shall see, the performance provided by SSIS is nothing short of incredible (eg in our real world example, we'll see how a 1GB VFP database was imported into a SQL Server database in under 4 minutes. And this while pulling the data over a network – it would likely be faster if the VFP data was in a local directory on the SQL Server.
However, the downside is that importing all the RI (ie triggers, constraints and validations) is difficult and requires hand coding. For this article we are only focussing on raw data import, and in general we don't use SSIS for these details.

In order to easily provide upsizing of all the RI associated with a database you should use the Upsizing Wizard shipped with VFP. I recommend downloading the latest Sedna update – the SQL Upsizing Wizard has been improved

greatly (available here: http://www.microsoft.com/downloads/details.aspx?familyid=05a0e7c9-43c1-417f-8810-ae7d7c66bac8&displaylang=en).

The downside of the Upsizing Wizard however is performance. In our tests the performance difference was drastic. Whereas SSIS was taking mere minutes, the Upsizing Wizard was taking hours...

Important: Hurdles that had to be overcome (Gotchas)

1. Permissions and mappings to drive letters (eg N: drive) - use UNC instead
2. Visual FoxPro 9.0 OLE DB driver needs to be installed on the actual SQL Server box

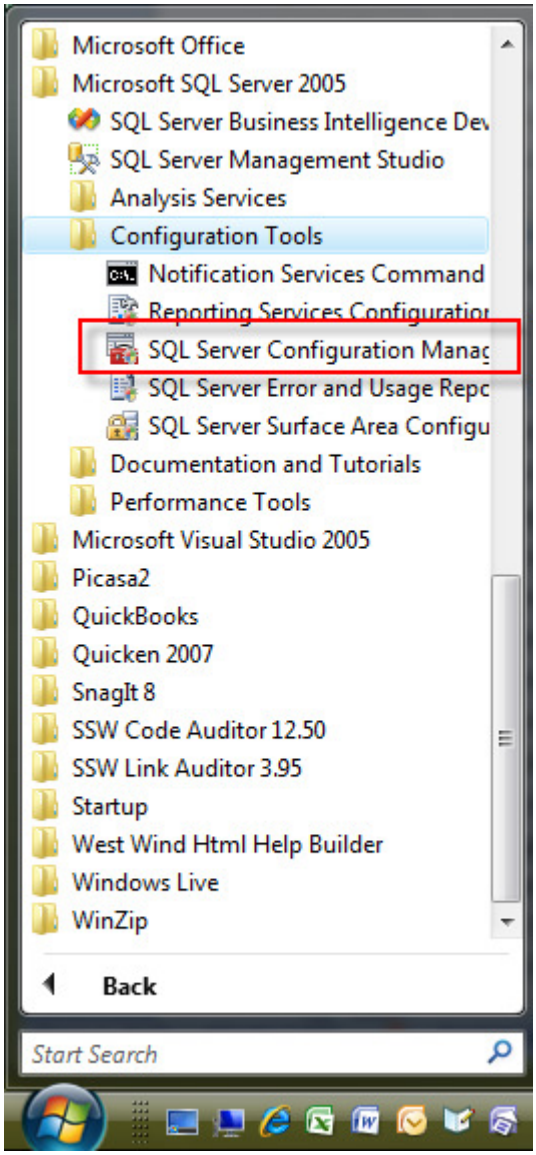
It is recommended to use UNC drive mappings where possible. Hard coding drive letters (eg N drive) can lead to problems if your SQL Server doesn't have the same drive mappings as the developer machine you build the package on.

You will also need to ensure you the VFP OLE-DB drivers installed on the SQL Server (see also Pre-flight check section later).

These points are very important (which is why I repeat them later in the article).

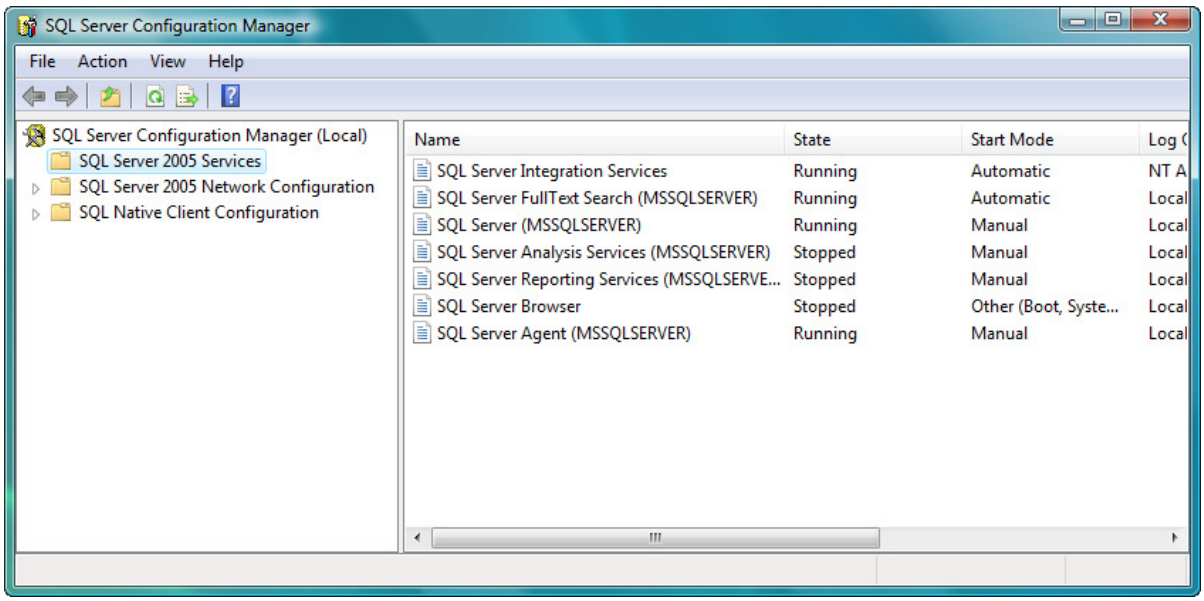
'Pre-flight check'

Before you start you need to check that three main SQL Server services are running. Open up the SQL Server Configuration Manager (from the Microsoft SQL Server 2005 > Configuration Tools menu)



Make sure the following services are running:

- SQL Server (MSSQLSERVER)
- SQL Server Integration Services
- SQL Server Agent



Overview of steps to set up the Integration Services package

Steps to follow

1. Open SQL Server Business Intelligence Development Studio
2. Create a New project
3. Select the Integration Services project type
4. Open the SSIS wizard
5. Select a data source – Visual FoxPro
6. Select a data destination – SQL Server
7. Select the tables to import
8. For each table open the Edit window and set the 'Drop and create table' checkbox and then scroll down through the fields and set any date fields to have a datetime type in SQL Server
9. After selecting and editing all the tables, click finish and the package is created
10. Run the package – it will fail with numerous errors, don't be concerned by this
11. Now go to Data flow view and double click on each of the destination targets
12. Go to the Error tab and set the error flow to be Ignore all errors
13. You need to do this for every table
14. Save
15. Run the package – it will take a long time on the first run
16. Run the package again – it will be much quicker this time
17. Save
18. Set the package to run as a scheduled task using SQL Agent

Overview of steps to set up the job to run as a Scheduled task via SQL Agent

1. Open SQL server Management Studio
2. Import the dtsx package into SQL Server Integration Services
3. Create SQL Agent job that uses the imported SSIS package

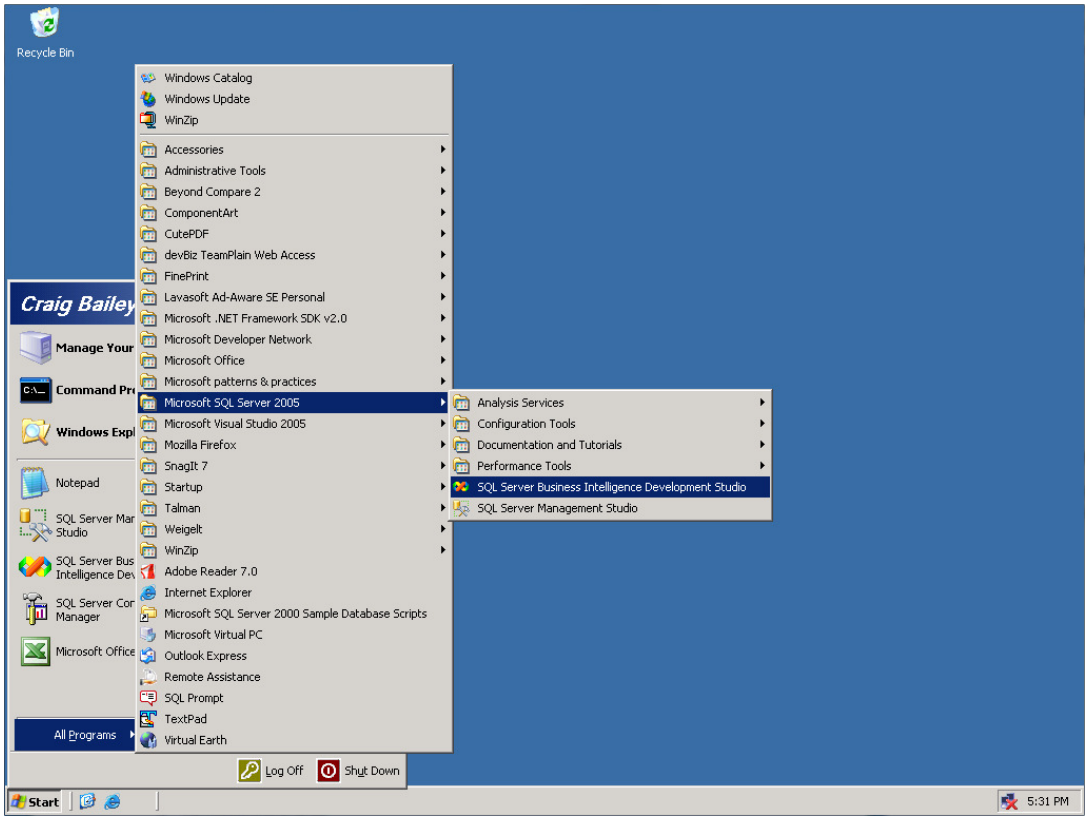
Prerequisites

1. Ensure the VFP OLE DB driver is installed on your SQL Server, and also the machine you are running the development tools from
2. Download it from here: http://www.microsoft.com/downloads/details.aspx?familyid=e1a87d8f-2d58-491f-a0fa-95a3289c5fd4&displaylang=en
3. Install it on the server and your development machine

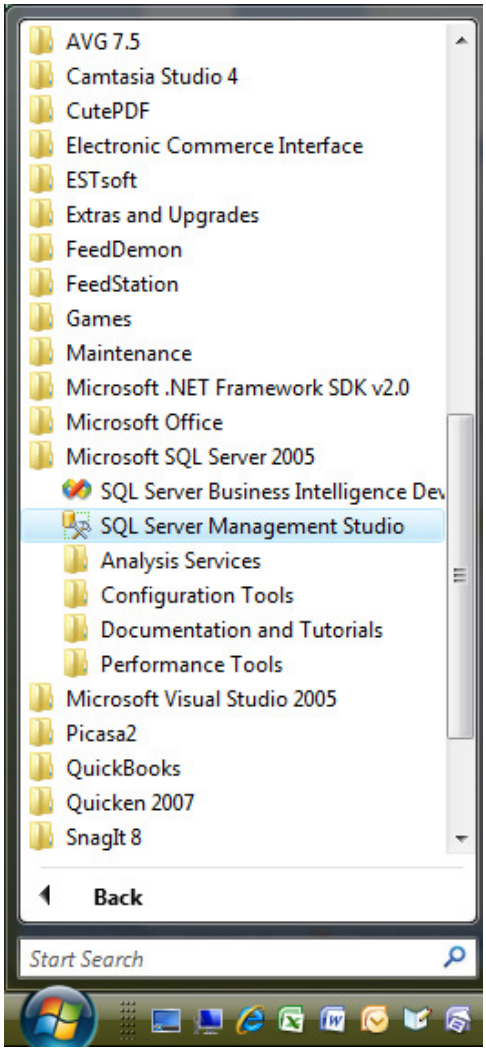
Building the Integration Services package - Step by Step instructions

OK, here we go – let's build our Import package.

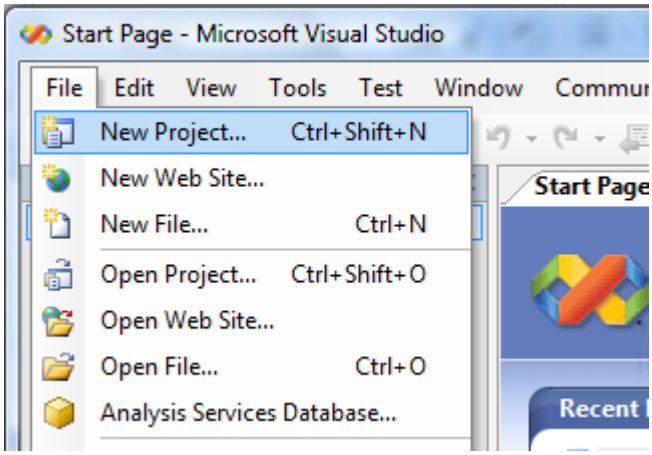
Start SQL Server Business Intelligence Development Studio
On a Windows 2003 box it'll be available through this path:



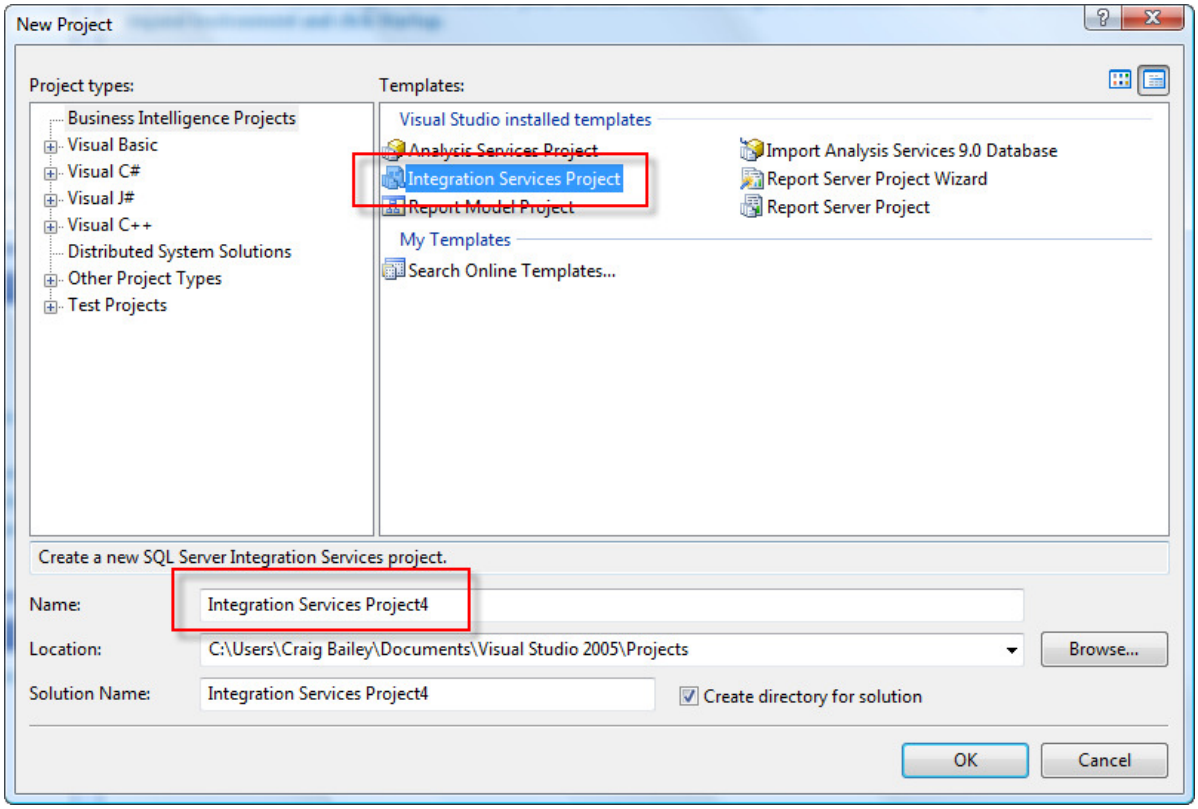
On Vista it'll be something like this



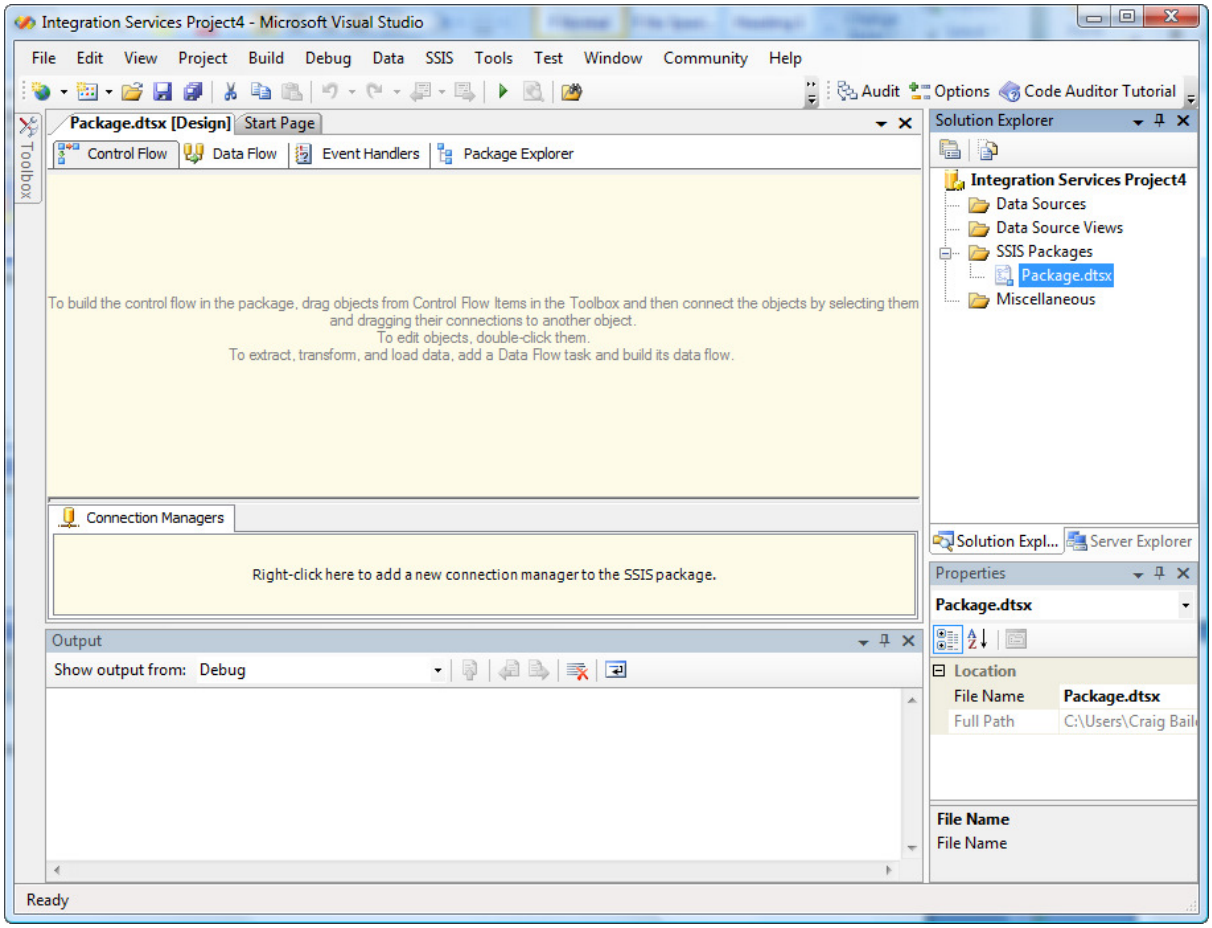
Create a New Project



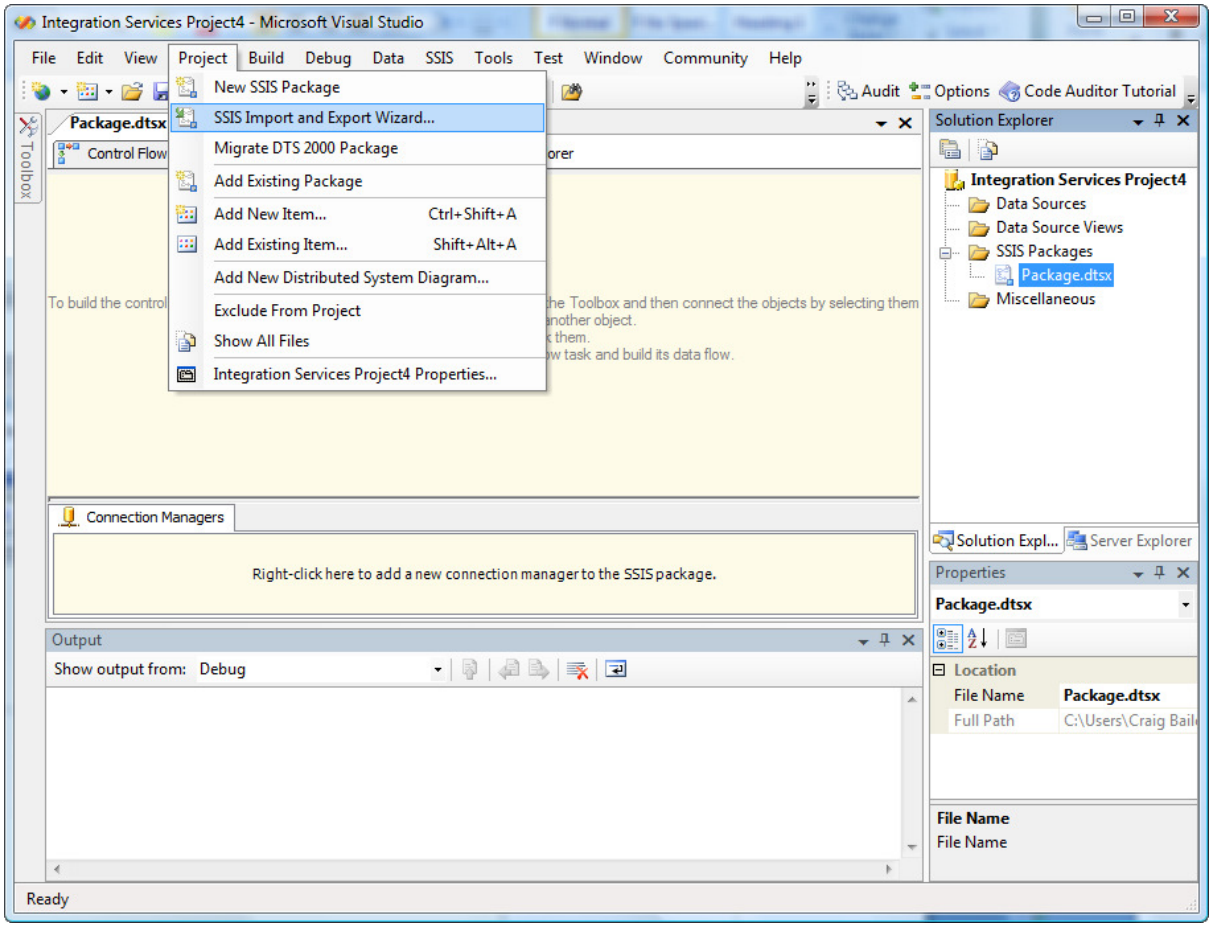
Select ‘Integration Services Project’, provide a name and click OK



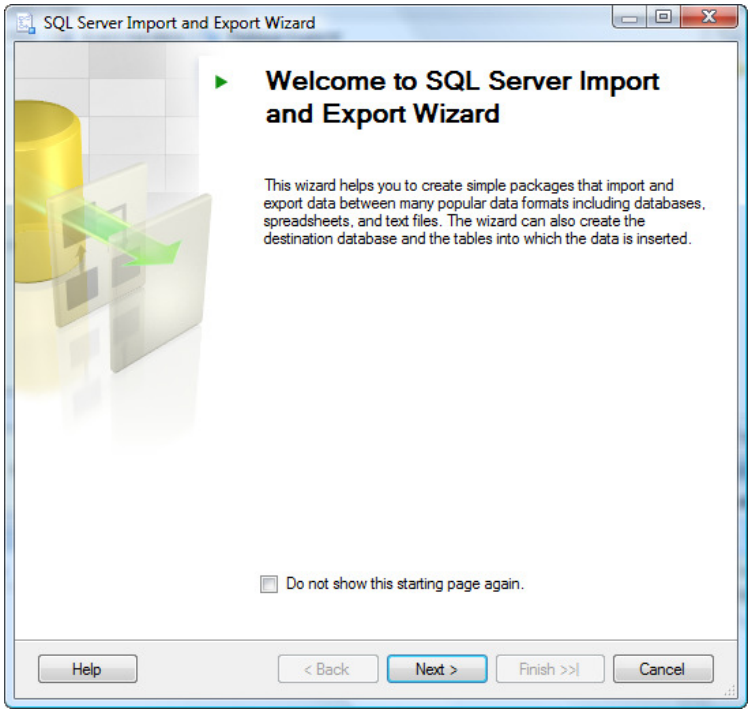
Your screen will look like this, all ready for you to start building the Integration Service package



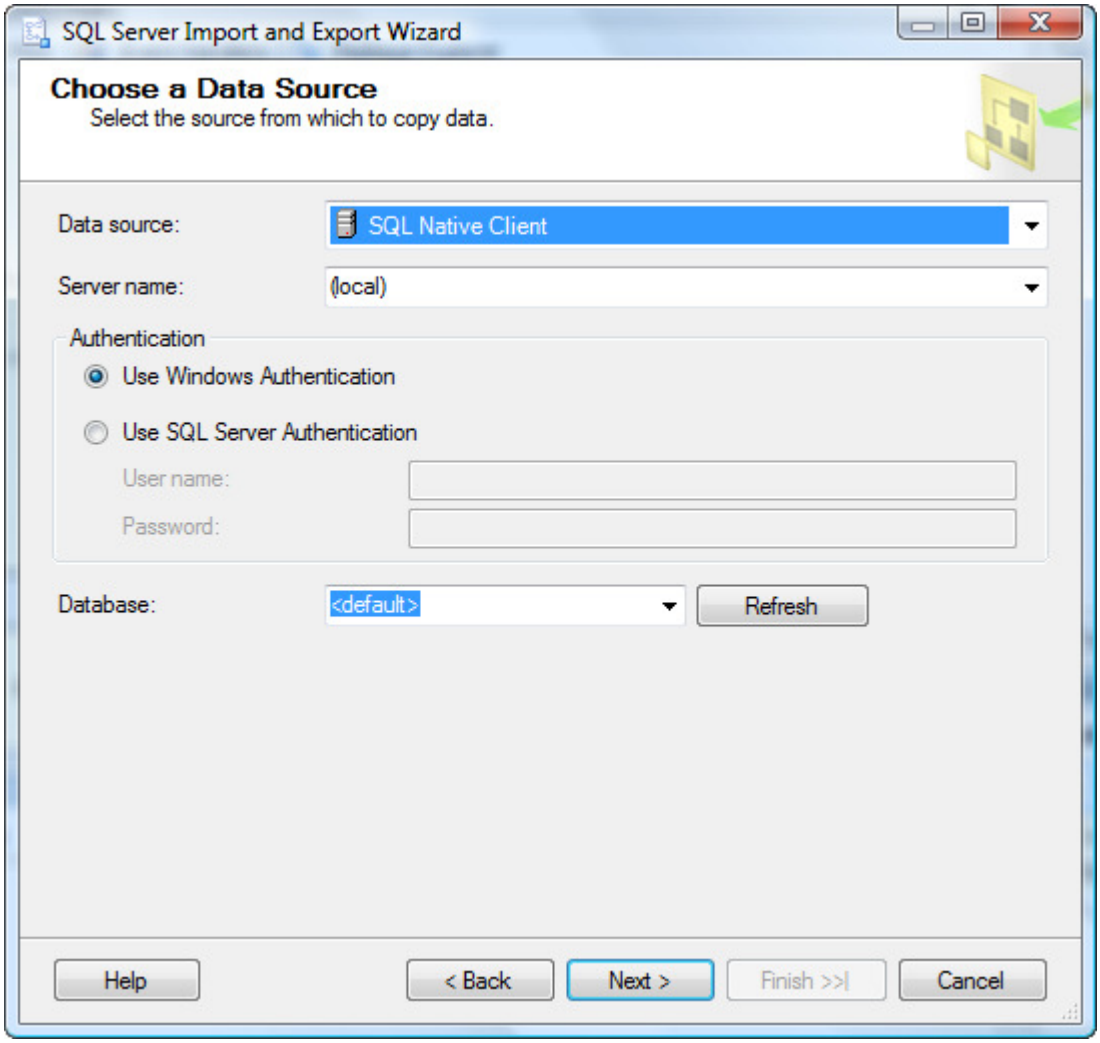
We will be using the SSIS Wizard to create our package, available under the Project menu.



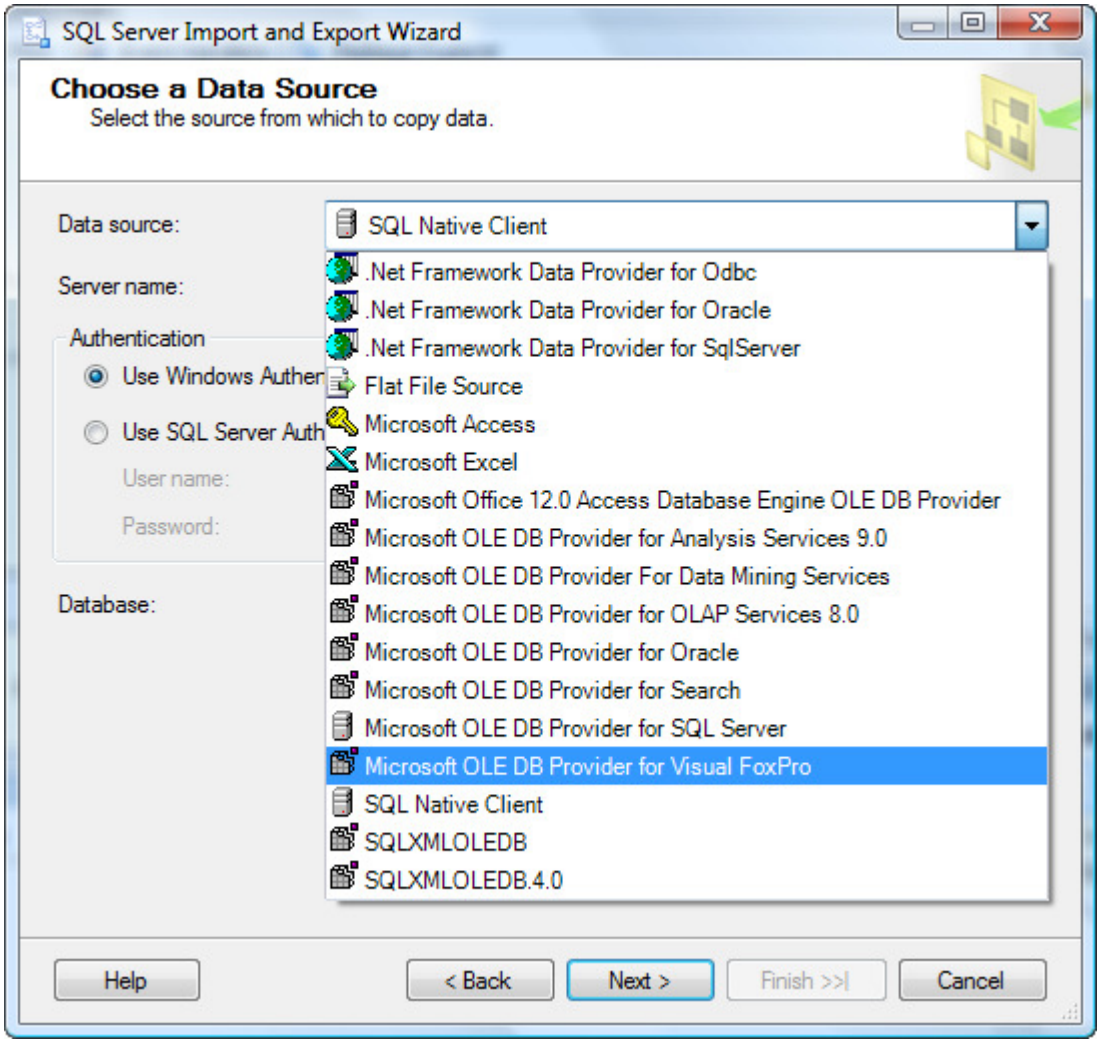
The default screen will look like this

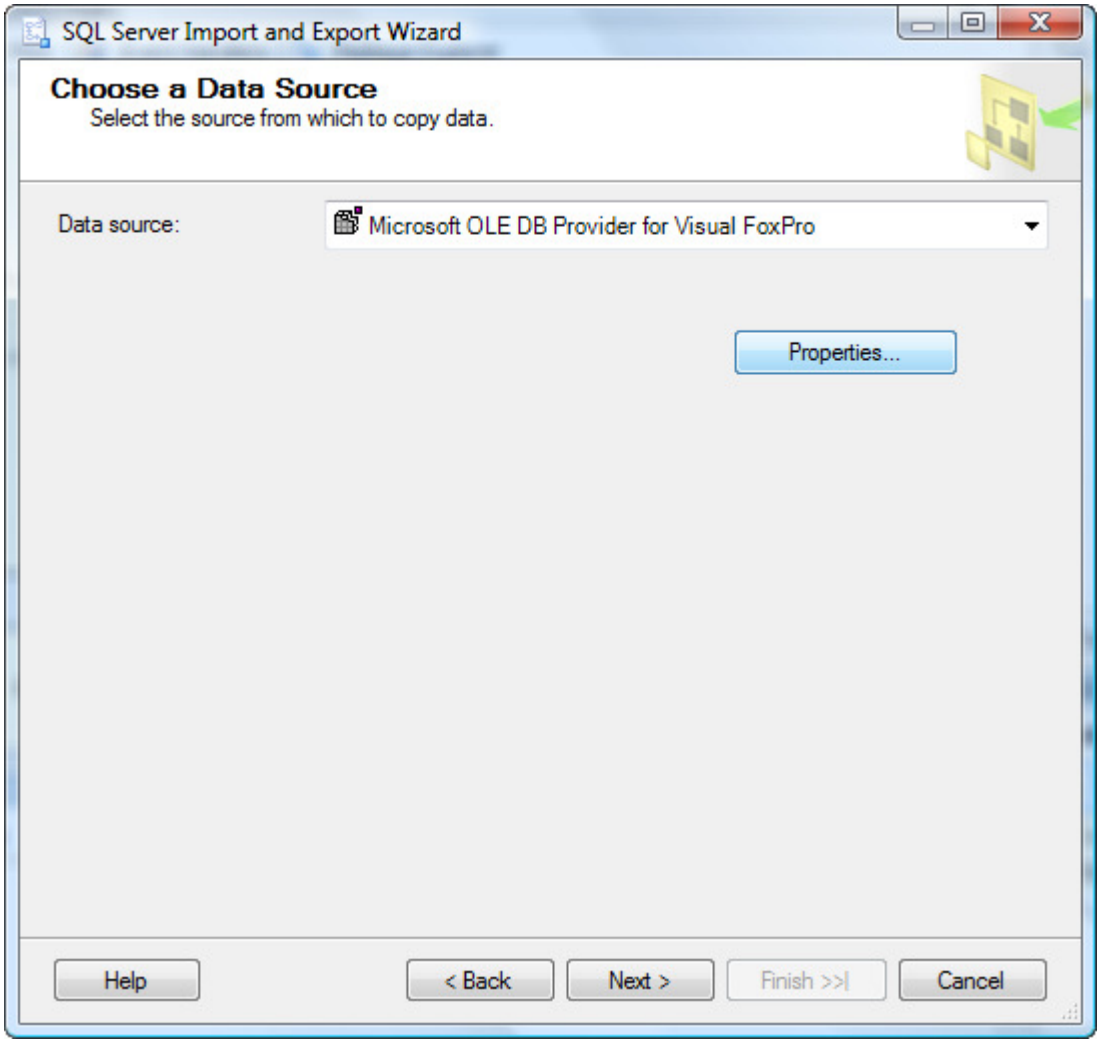


Then you will be prompted for your Data Source

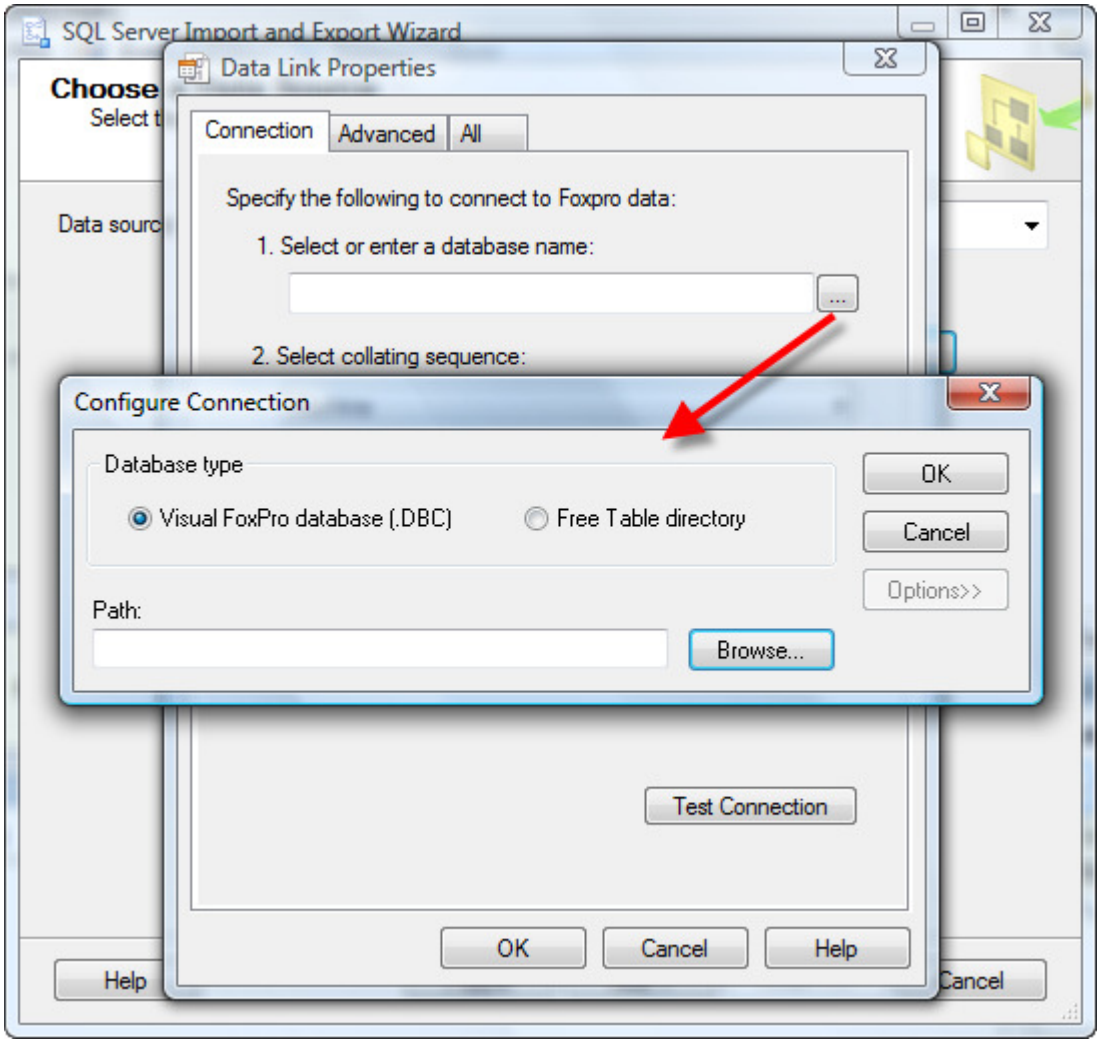


Change the Data source to be the Microsoft OLE DB Provider for Visual FoxPro


SQL Server Import and Export Wizard
Choose a Data Source
Select the source from which to copy data.
Data source:
SQL Native Client
Server name:
Authentication
Use Windows Auther
Use SQL Server Auth
User name:
Password:
Database:
.Net Framework Data Provider for Odbc
.Net Framework Data Provider for Oracle
.Net Framework Data Provider for SqlServer
Flat File Source
Microsoft Access
Microsoft Excel
Microsoft Office 12.0 Access Database Engine OLE DB Provider
Microsoft OLE DB Provider for Analysis Services 9.0
Microsoft OLE DB Provider For Data Mining Services
Microsoft OLE DB Provider for OLAP Services 8.0
Microsoft OLE DB Provider for Oracle
Microsoft OLE DB Provider for Search
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Provider for Visual FoxPro
SQL Native Client
SQLXMLOLEDB
SQLXMLOLEDB.4.0
Help
< Back
Next >
Finish >>|
Cancel

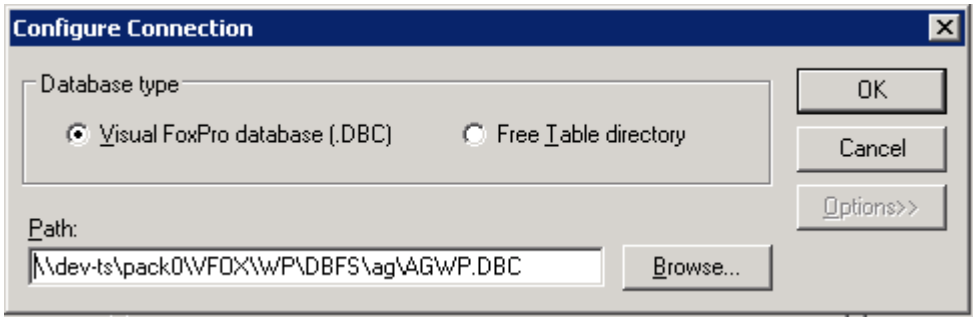


Click on Properties and set the path to your Visual FoxPro database (or tables)

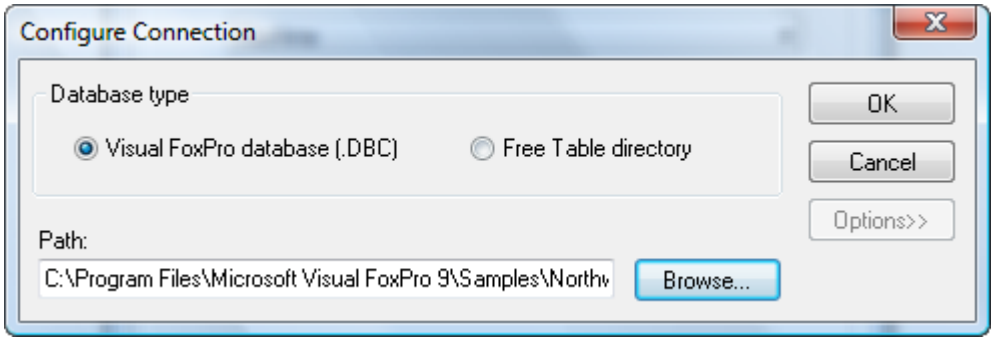


Note: It is a good idea to set the UNC path to the files.

Below is an example the form filled out with UNC mapping (this is the data that is used in our real world example later)

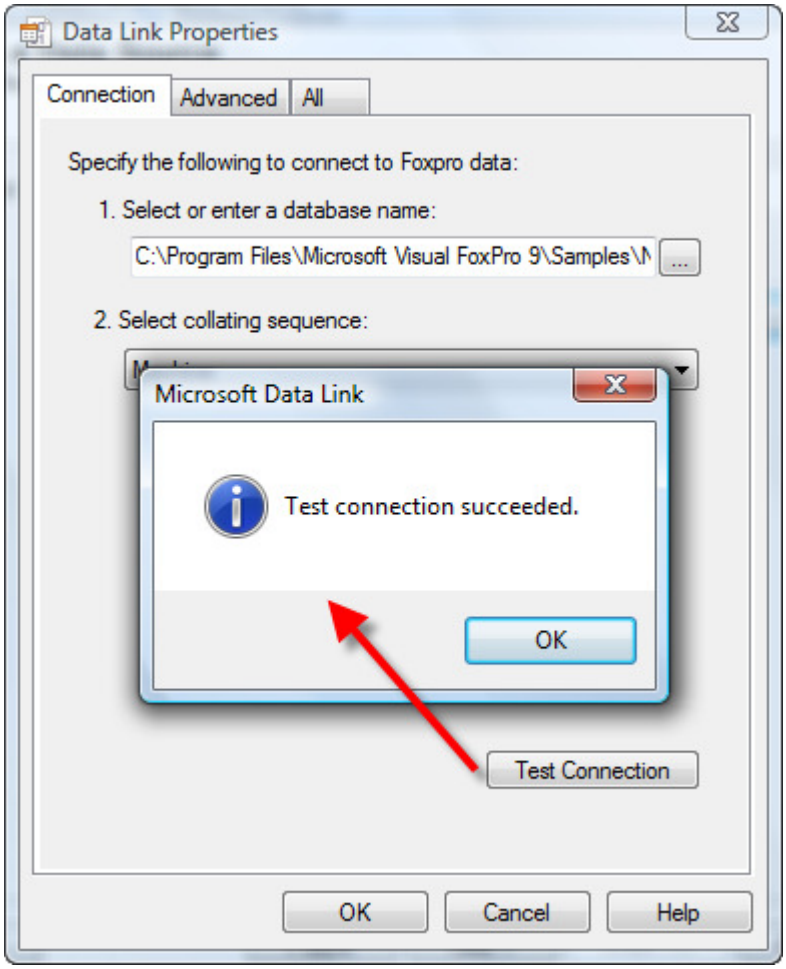


However, for our purposes we will just use a local mapping to the VFP sample database in Northwind



Click OK

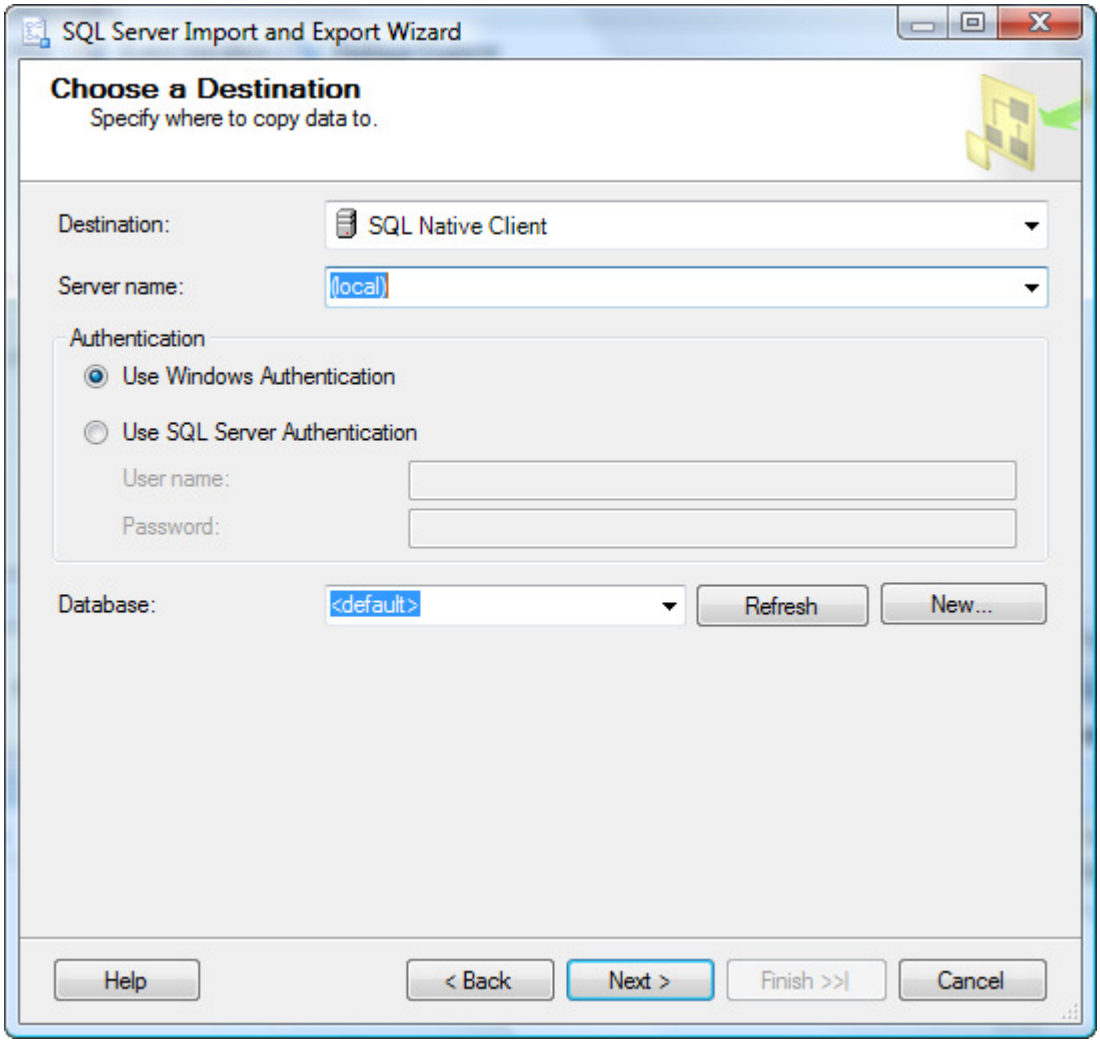
Click Test Connection to ensure the connection works fine



Click OK and OK again

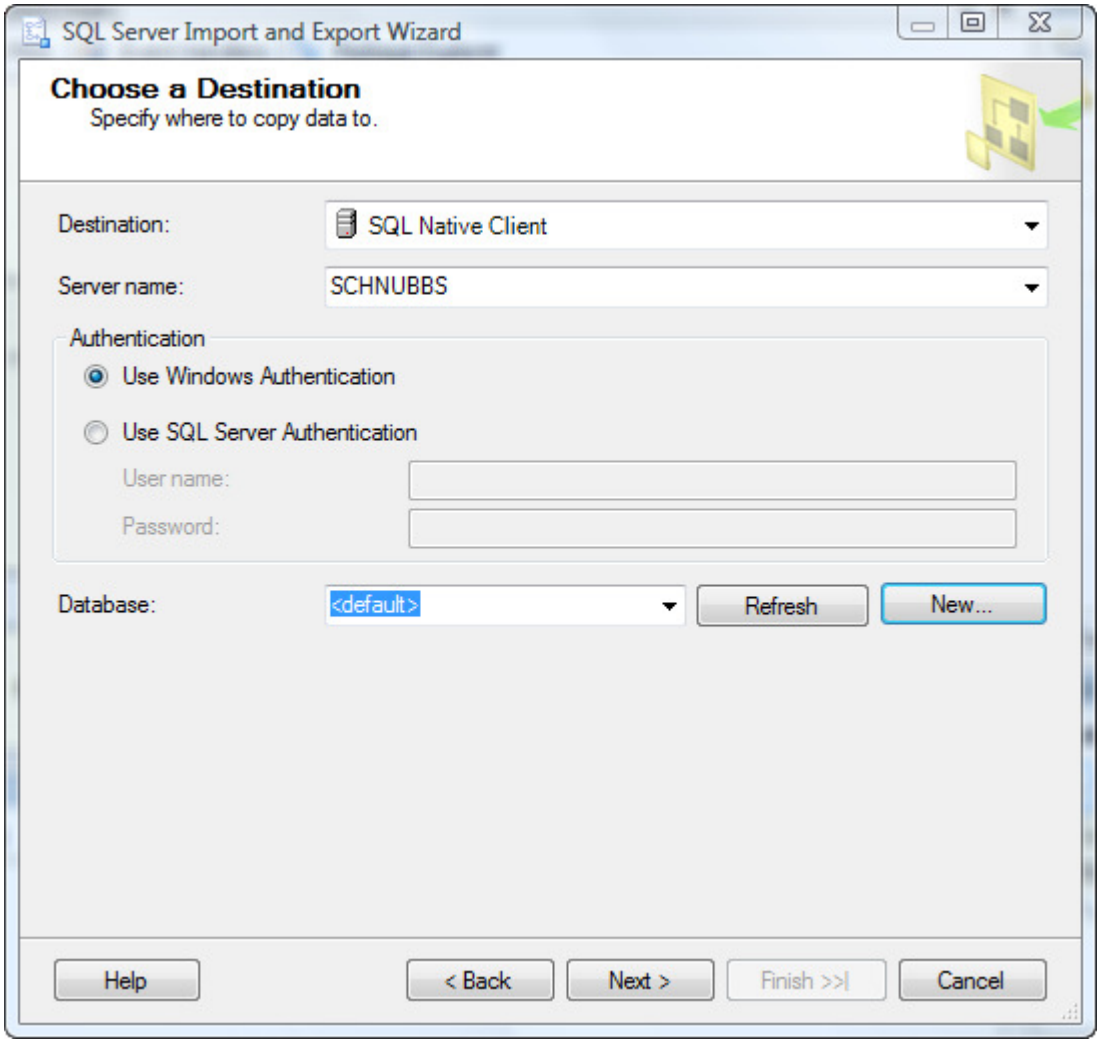
Once back at the Wizard click Next

Here's how it will look by default



Leave the destination name as SQL Native Client

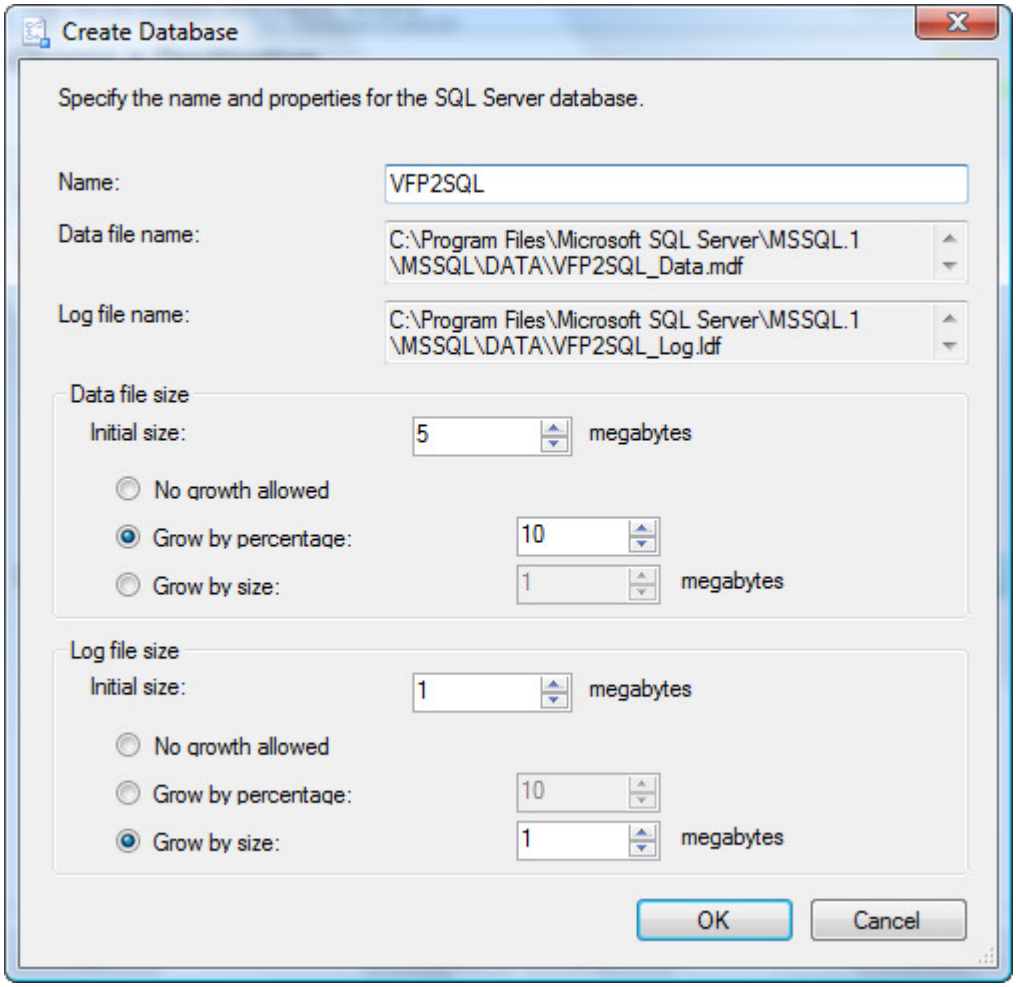
Choose the Server name you want to import into (my local Server is called SCHNUBBS)



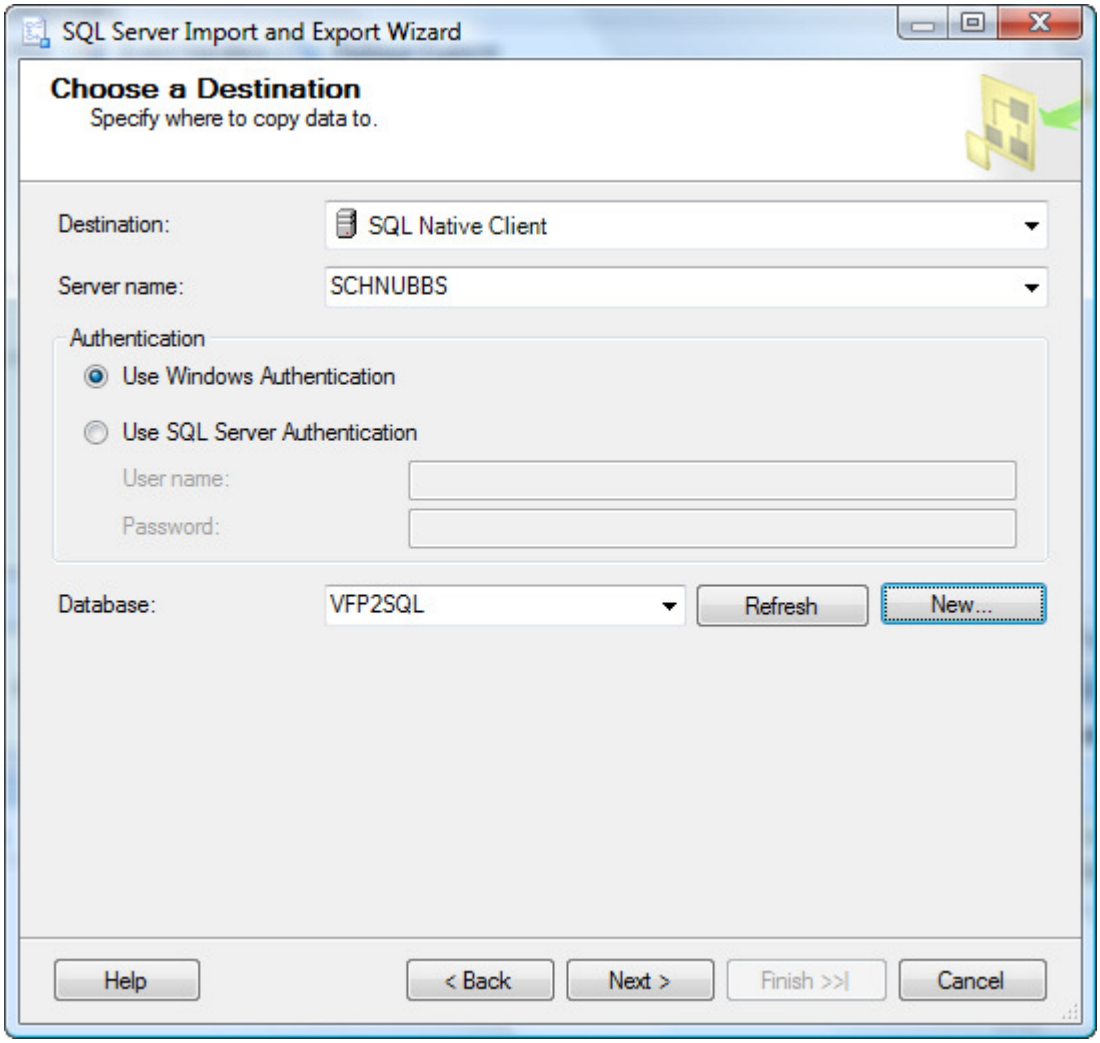
Click New to create a new database

Set the name and Initial data file size. It is a good idea at this point to think carefully about how big your database is likely to be. If it is likely to be in the hundreds of megabytes, then you may as well allocate much of that now.

If you don't, then your first Import run is going to be very slow, as the process will need to grow the database numerous times (and you will have likely left the growth at 10%). Thus a 500MB database will have to grow from 5MB to 5.5 to 6.05 to 6.655 etc all the way up to the 500MB (49 growth steps later).



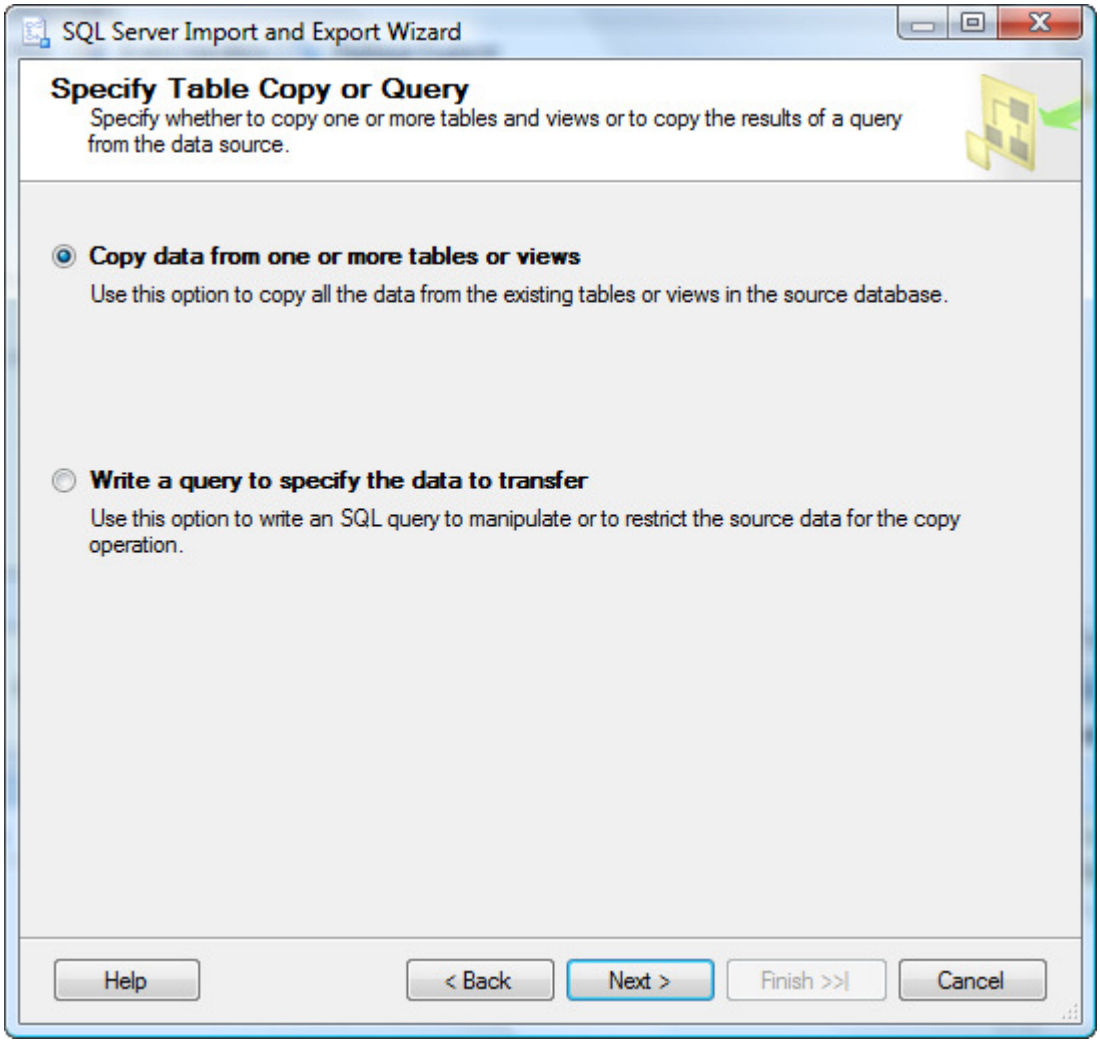
You should now have a screen like the following



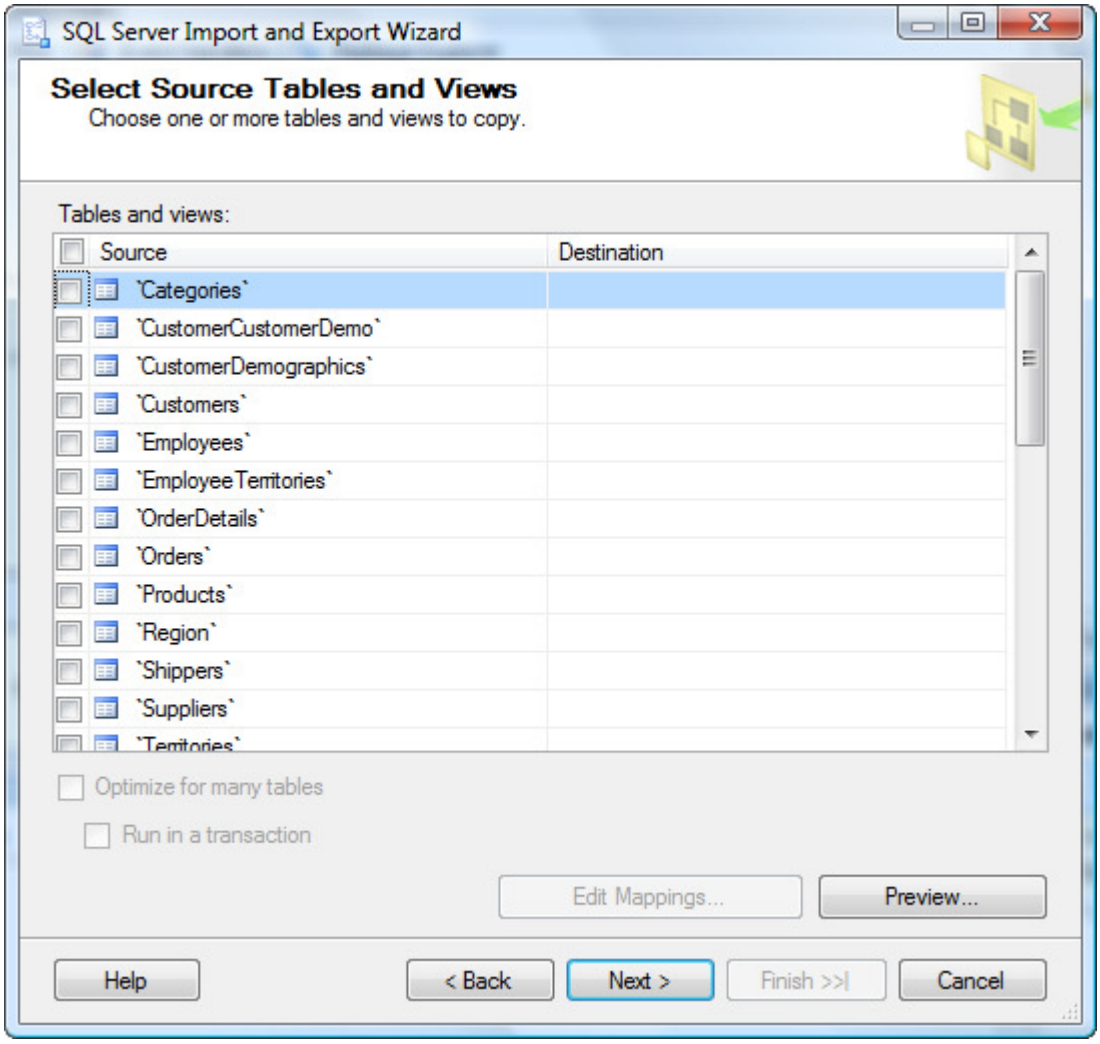
Click Next

Stay with the first option (Copy data)

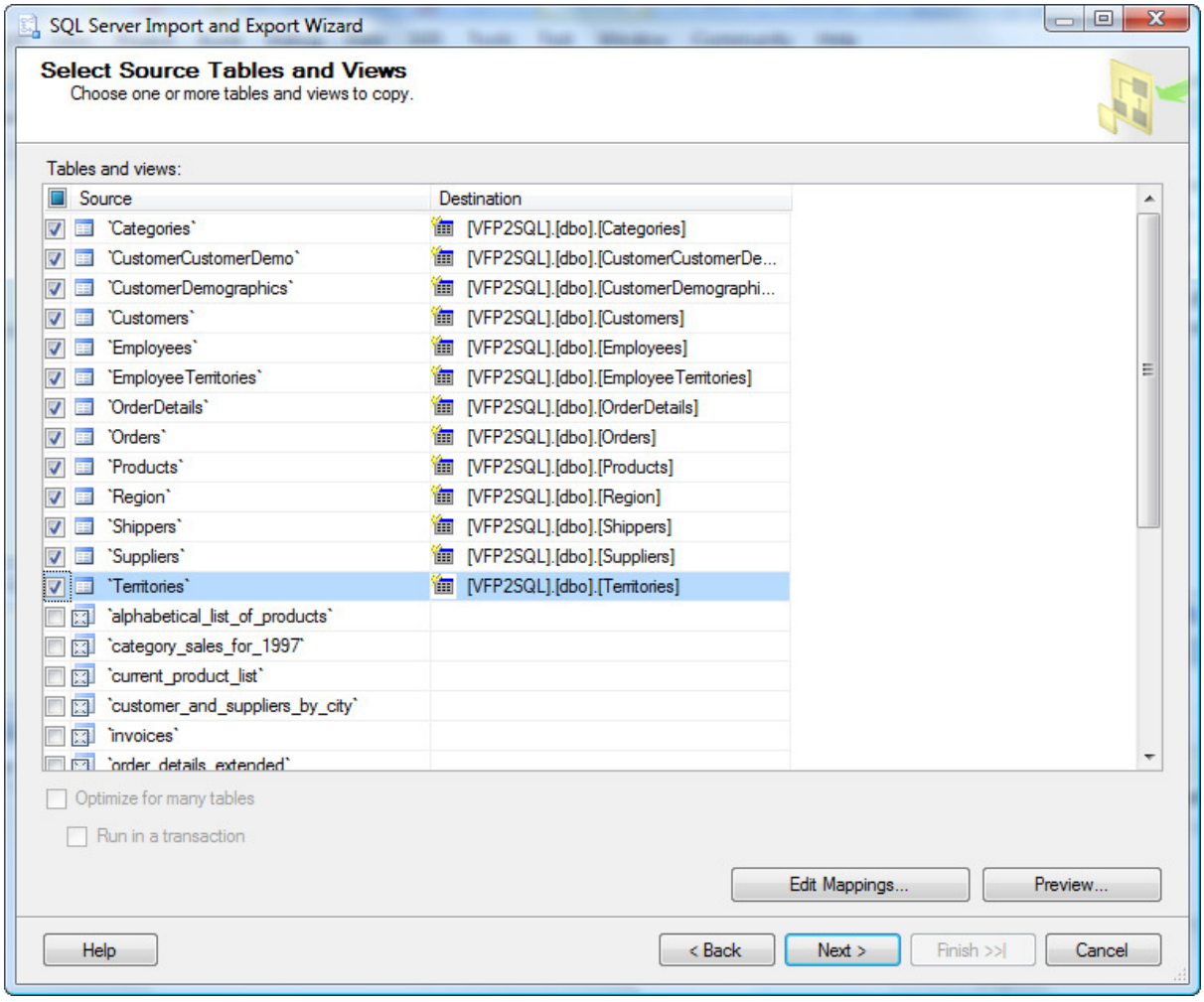
Click Next



Time now to choose which tables to import. We will be limiting our import to just the 13 main tables

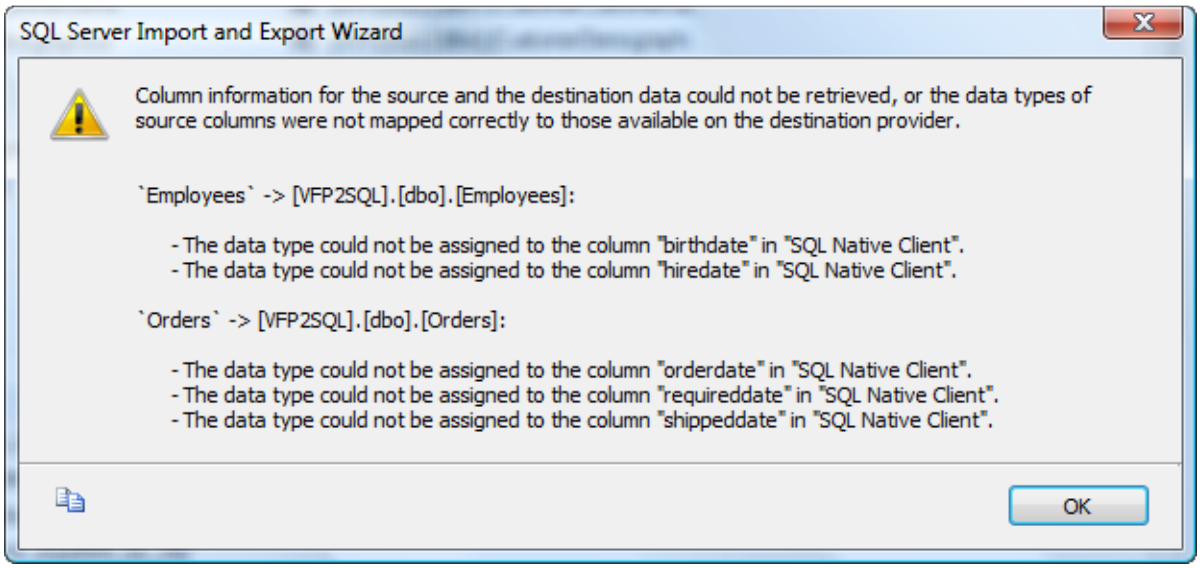


Tick next to each table that you wish to import. We will select only the tables (not the views).



Now comes the tedious part. We need to set all tables to Drop and Create, and we will also need to manually edit some of the field mappings.

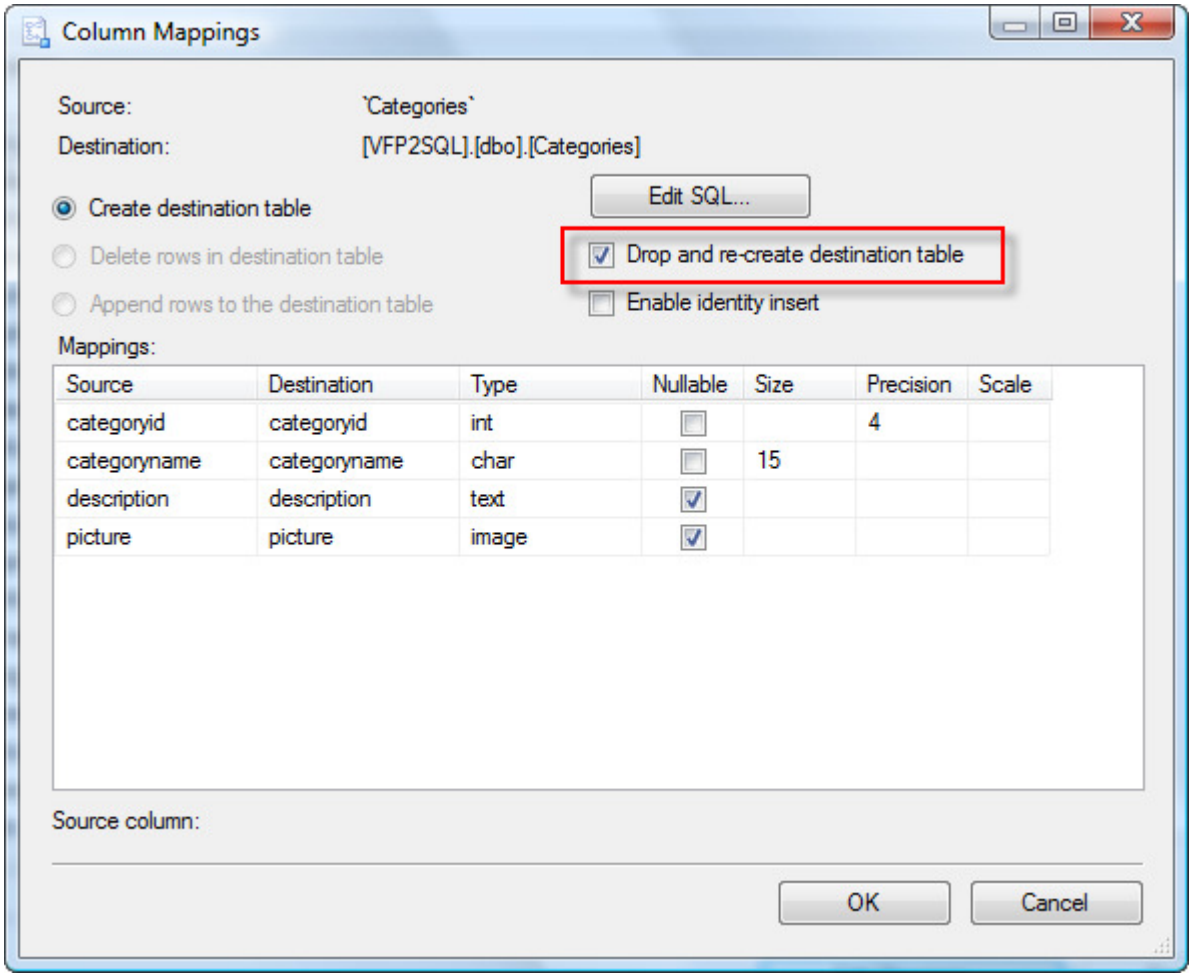
Since SQL Server doesn't have a date only field type we will need to manually set those field mappings. There are a few tables that will require this. You can easily find out the extent of the missed mappings by clicking the Next button (it won't progress if there are issues). A form like this will display the issues that need to be resolved.



Firstly lets go through and set the Drop and re-create option. Select a table (eg Categories) and click the Edit Mappings... button

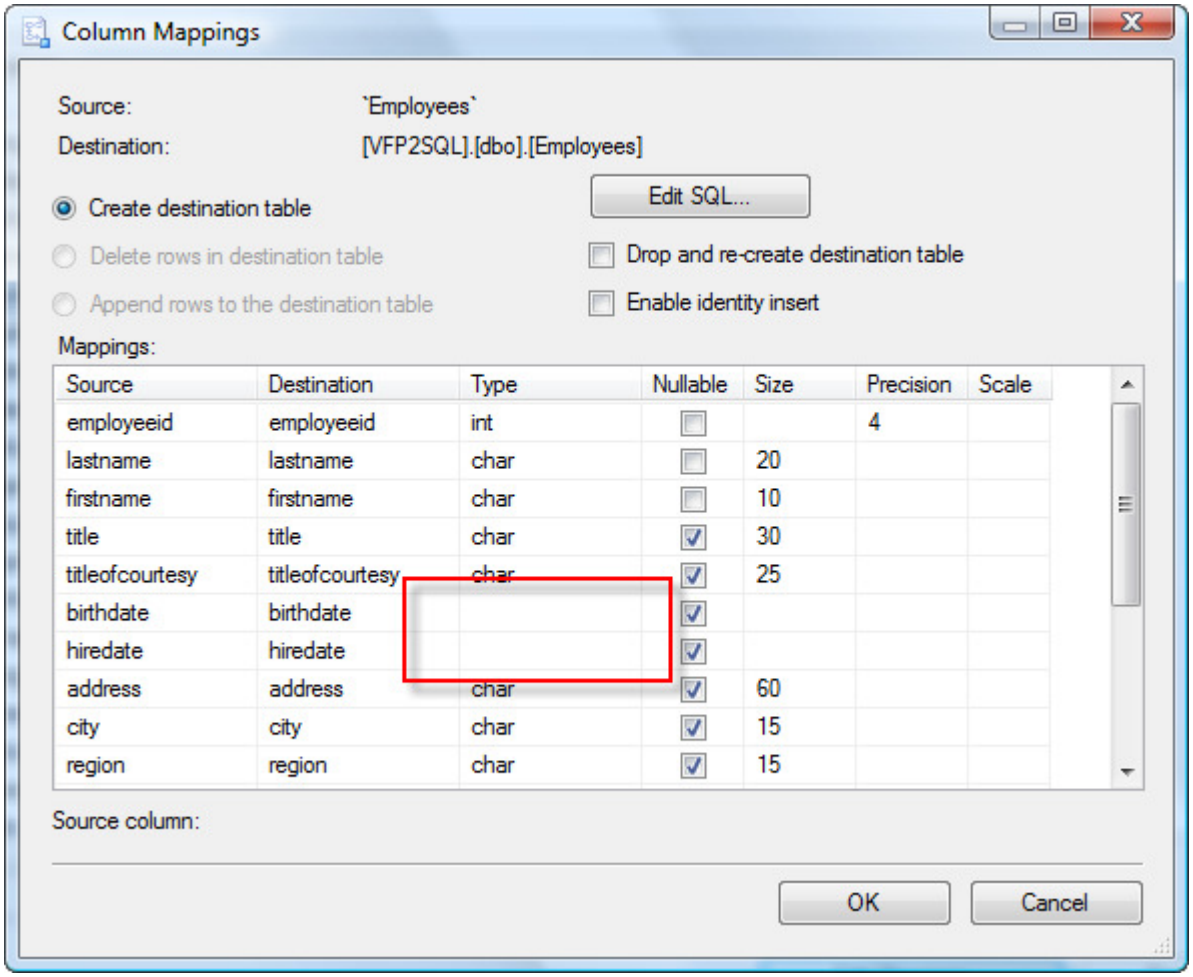
In the form tick the 'Drop and re-create destination table' option.

We are doing this because the likely use of an Import package is that it will be re-run multiple times (perhaps a daily import) so we want the import process to drop the tables and re-create them. Otherwise we would have to have multiple packages – one to create the tables initially, and then others for doing the record deletions and re-import. If your table has special primary key record numbers that you wish to insert, you can tick the 'Enable identity insert' checkbox as well (assuming you have an identity column set).

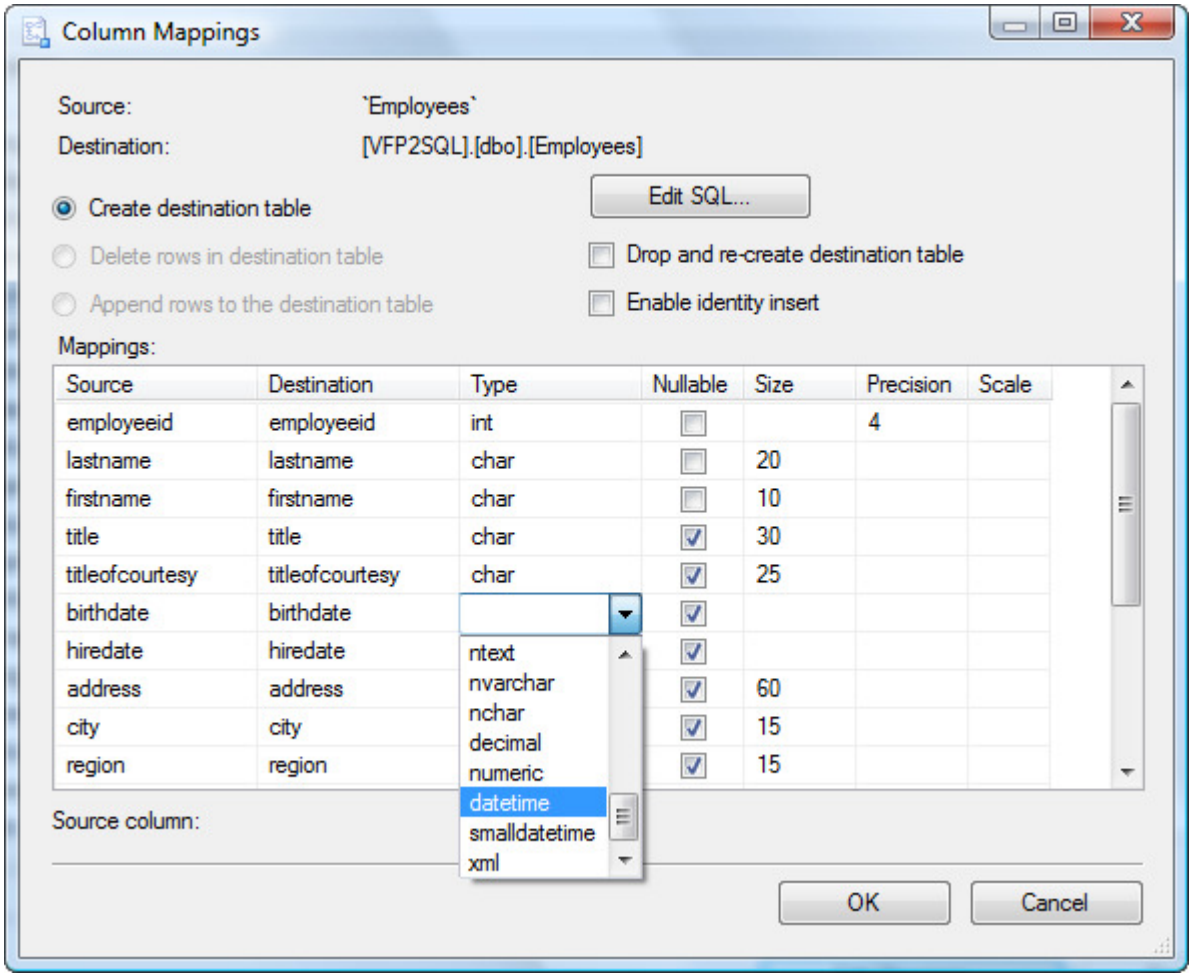


Do this for all tables.

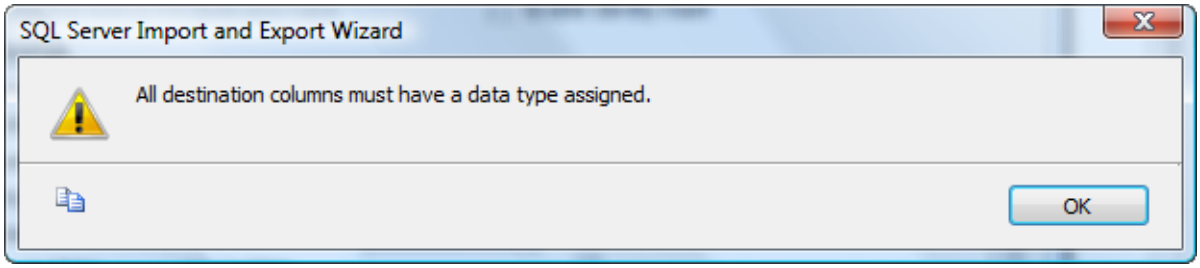
As you go through tables, check that all of the Type fields are set. The date to datetime field issue flagged earlier will appear like this



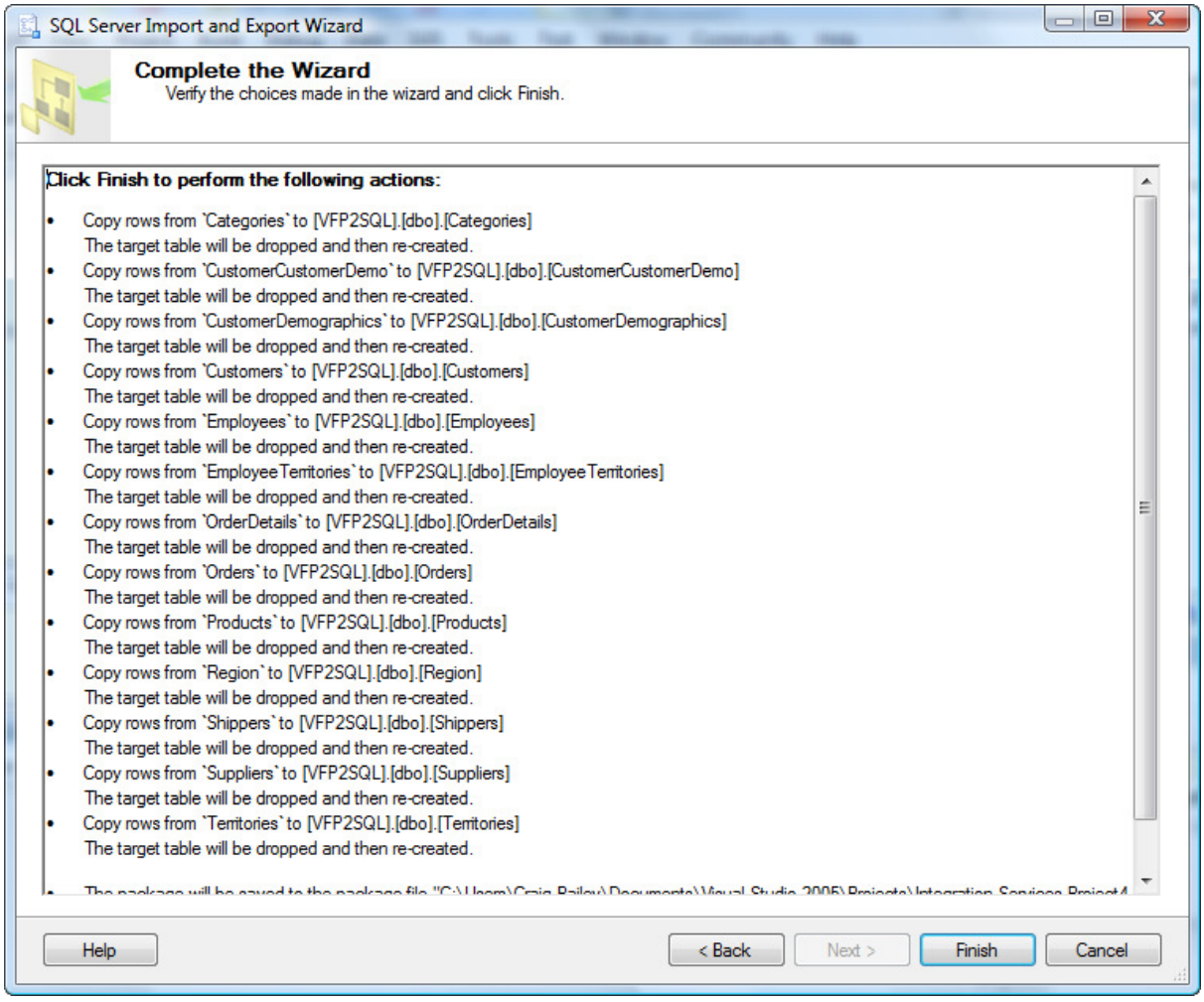
In cases like this, click on the type and then choose the appropriate field type from the drop down. To save time you can also just click on the grid and press the ‘d’ key twice to set datetime.



Note, if you don't fill out all the types you will be prompted with a message

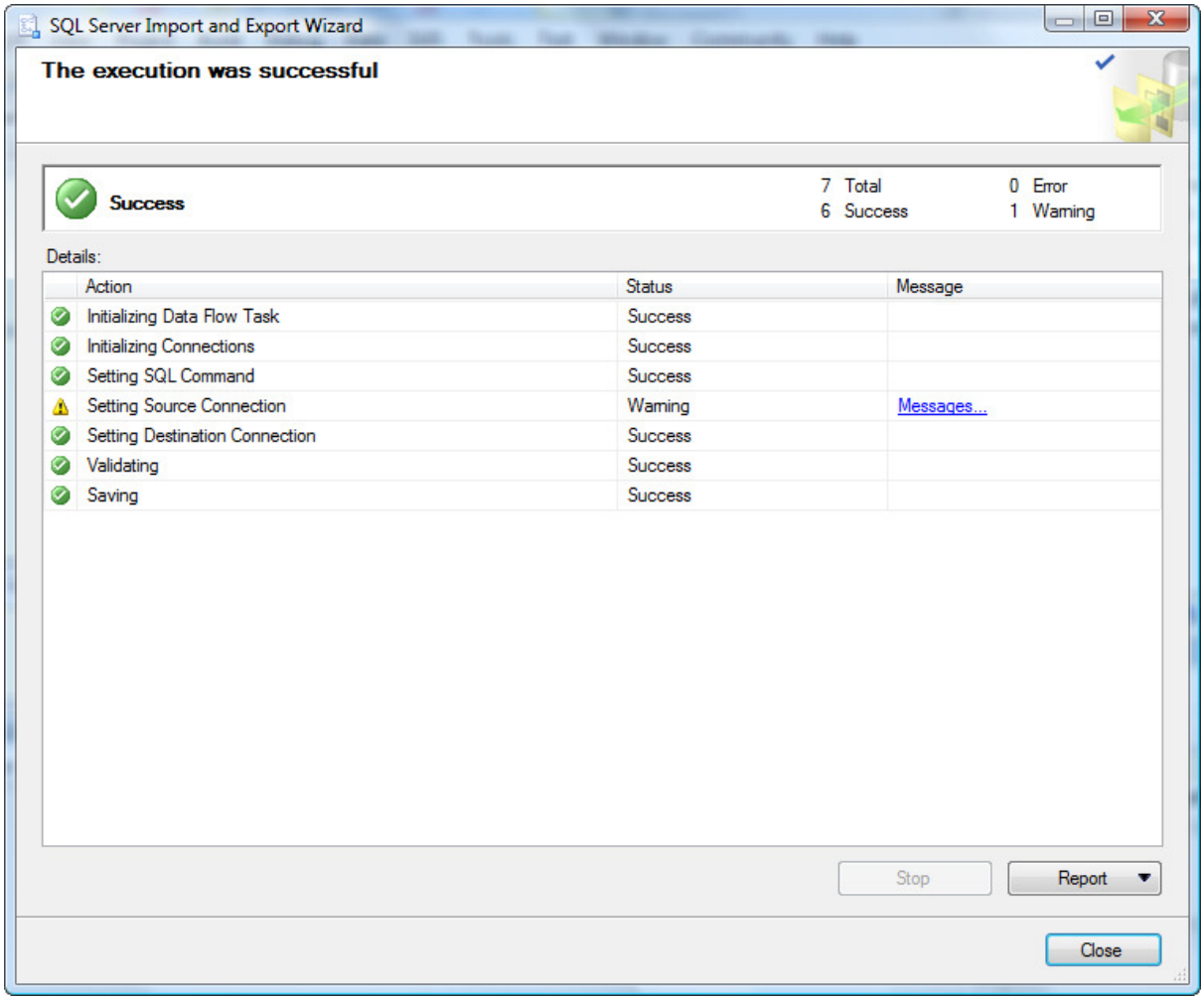


Once all the tables have been set, click Next to be presented with your summary of actions to perform

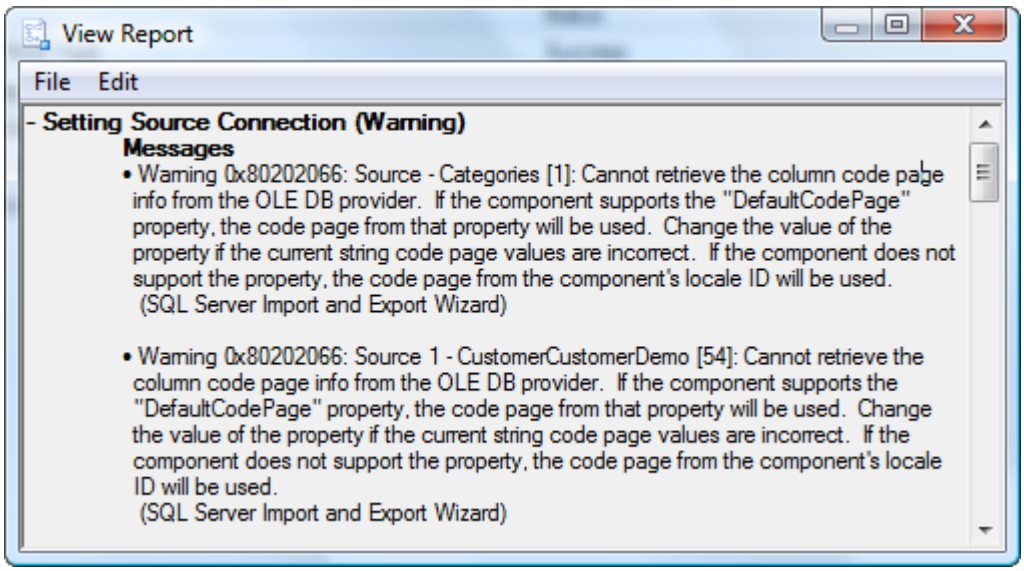


Click Finish

The wizard will go through and pre-populate the package based on all those settings. Check for any errors or warnings.



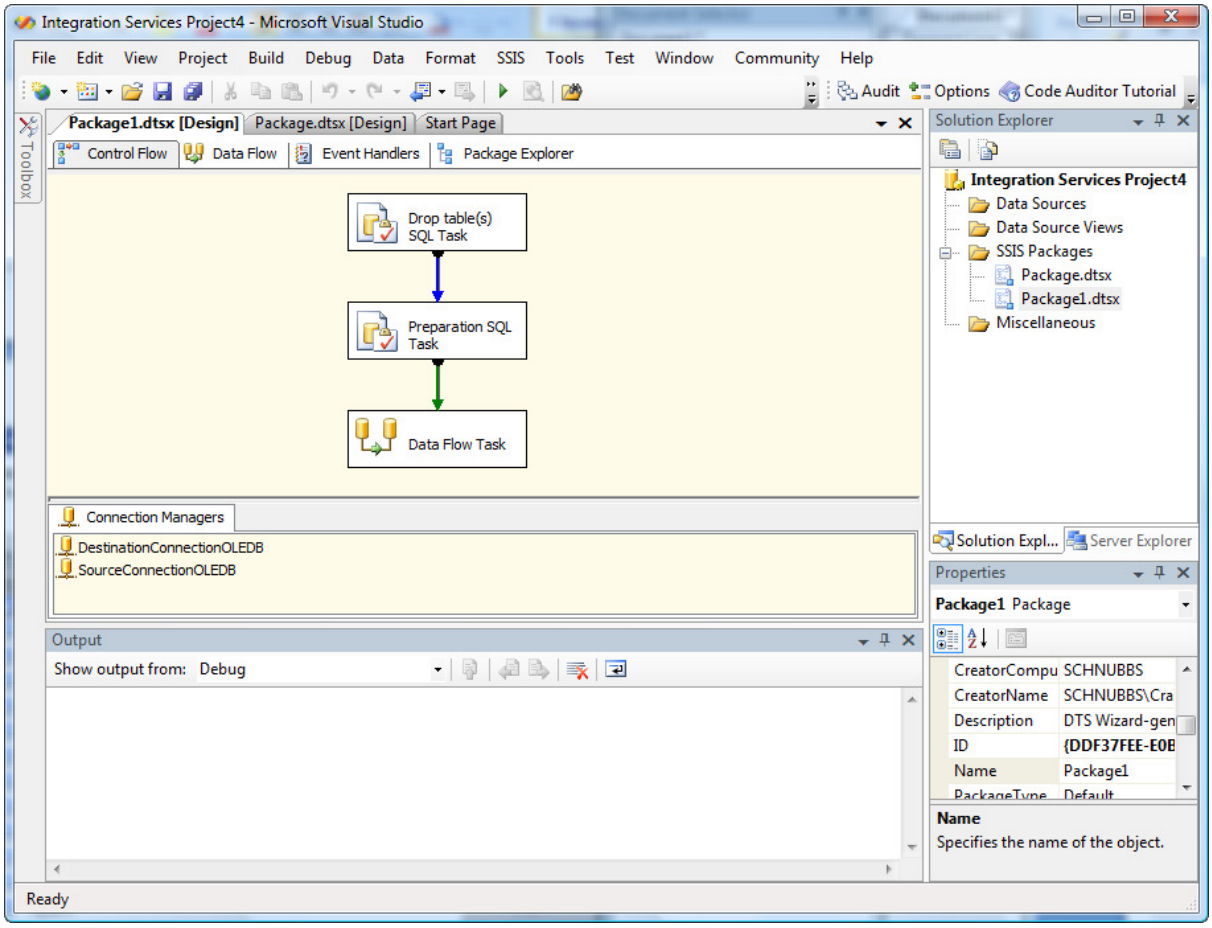
A common warning concerns failure to determine the code page used. However SQL will work fine with the default settings, so don't be alarmed by this.



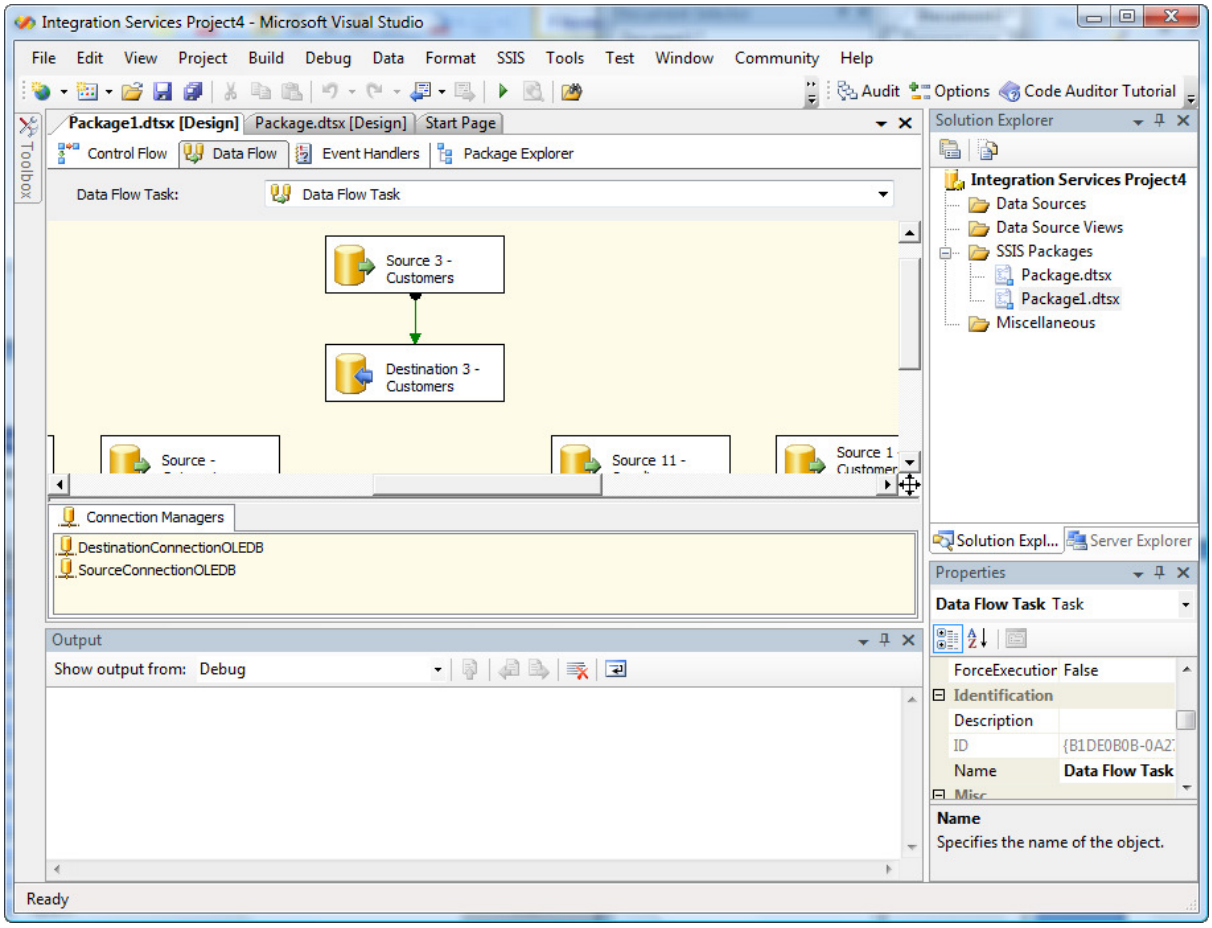
Close the warnings box.

Click Close on the Wizard.

Your generated SSIS package will be displayed



The Control Flow tab shows the overall high level process (we'll come back to that in a minute). The Data Flow shows the specifics of each table being imported

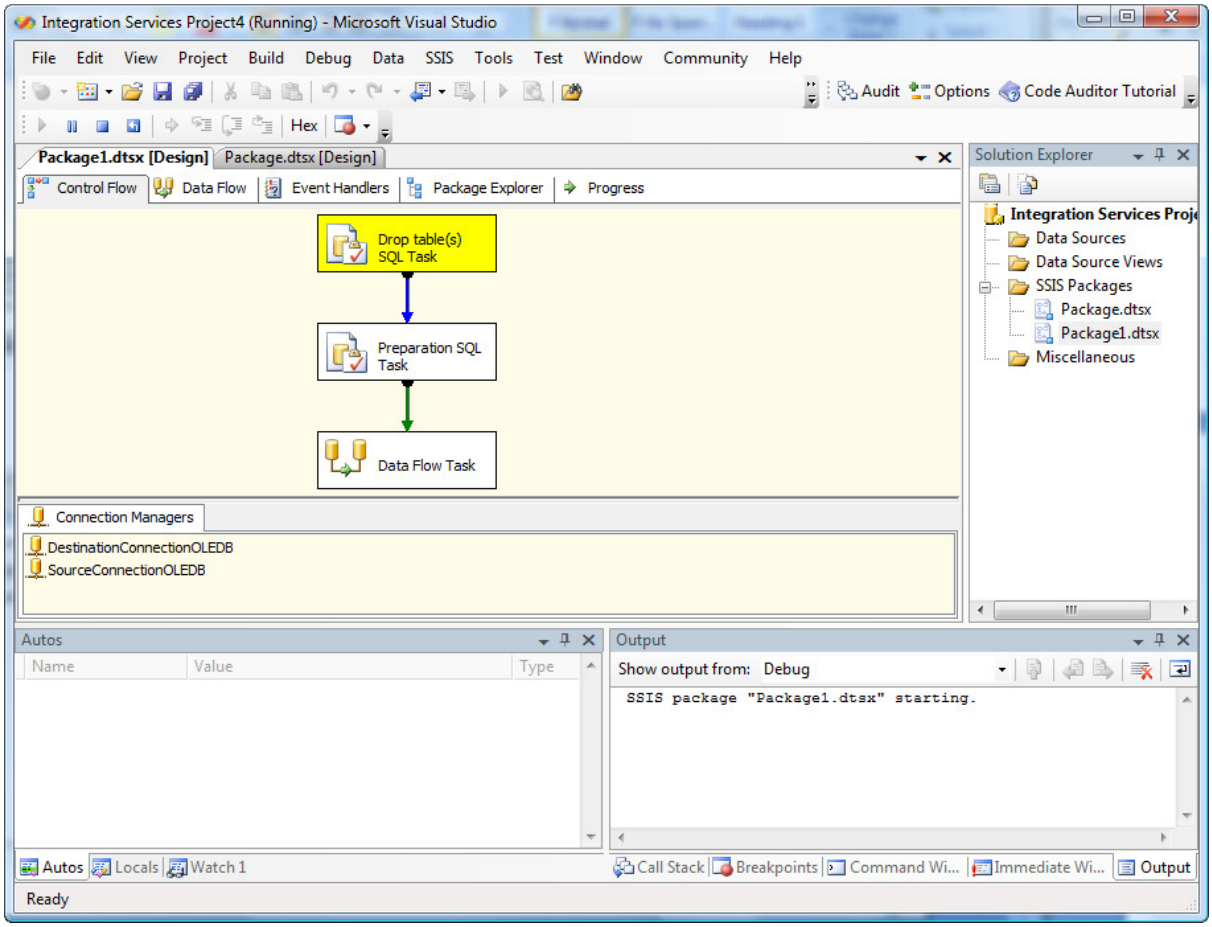


Event Handlers allows us to build error handlers (we won't be doing this here) and the Package Explorer is a nice way to navigate the various components.

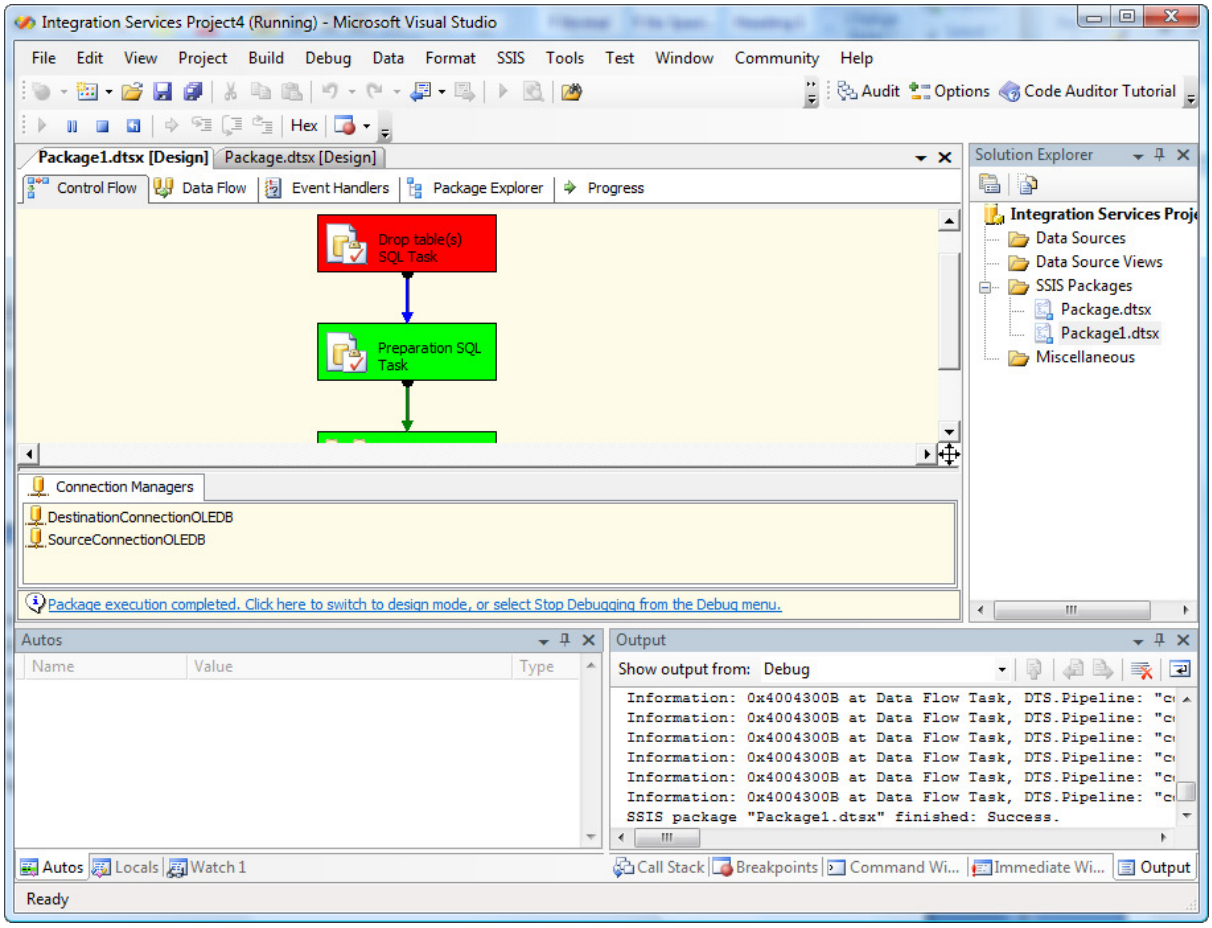
You are now going to run the process a few times.
The first time the package will fail. This is because of two things:

1. The package has been set to Drop and then re-create tables. Since the tables don't exist the first time it runs, there is an error generated.
2. The import will likely stumble on some data issues in the source files (eg dates that are wrong – this can happen at times if we don't validate entry. I've seen dates with years of 0227 instead of 2007 manage to find their way into date fields, causing problems for SQL Server) – we will fix and overcome this shortly.

Click the green arrow (or press F5) to start the import. Here's how it will look whilst in progress

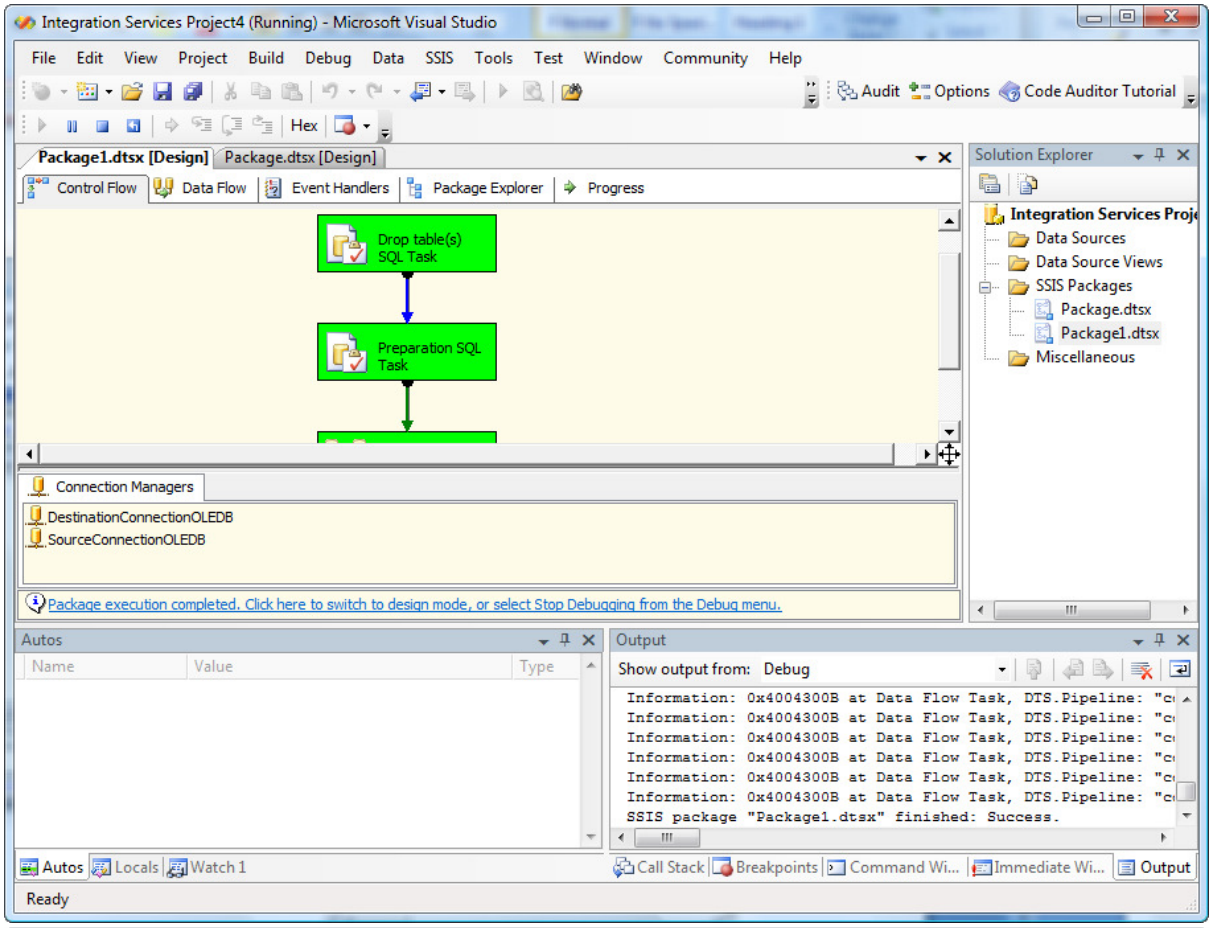


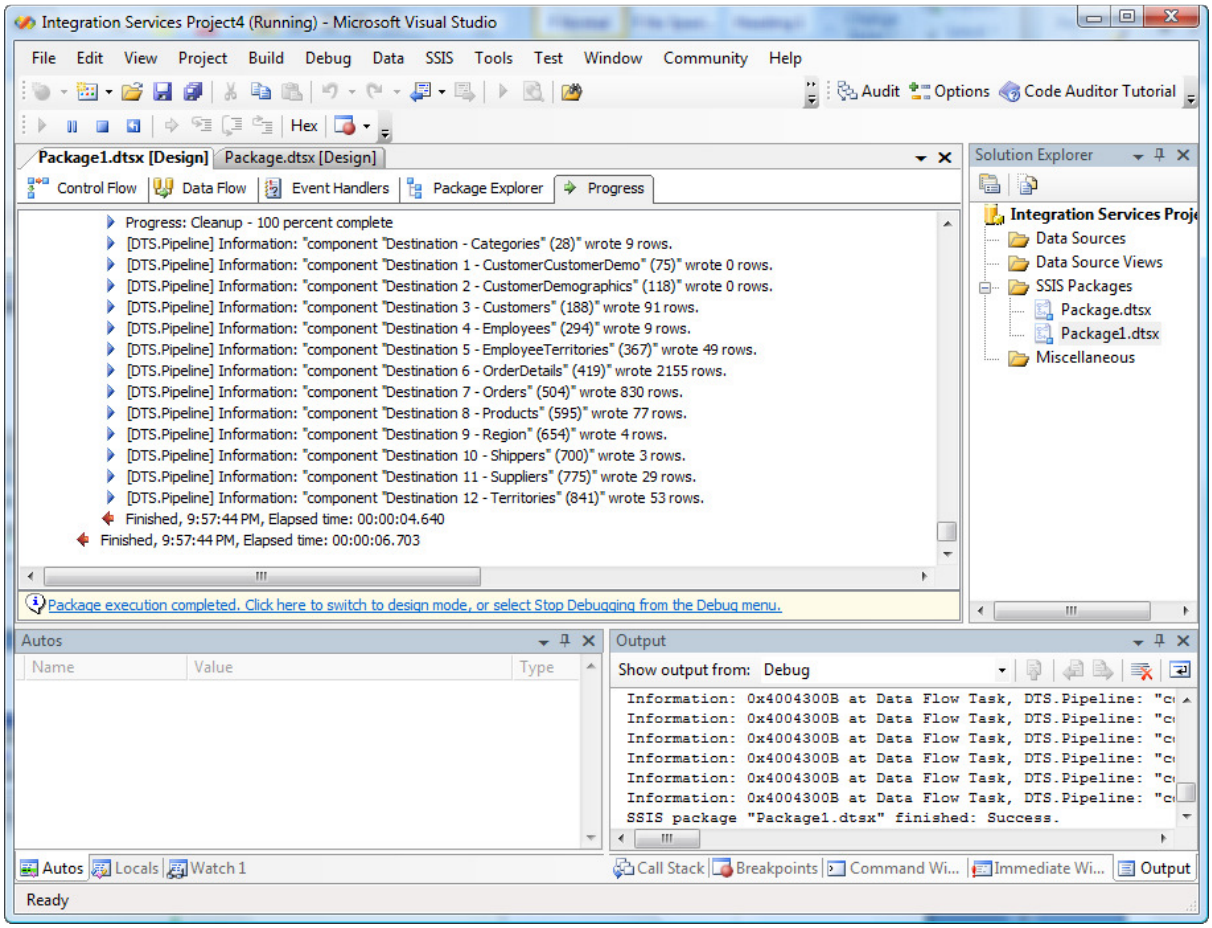
The red means error, green success and yellow means in progress.
The red block is because of the error mentioned above.
Once complete click the link to stop debugging (or Press Shift F5)



Run the process a second time (eg press F5)

This time the first action will succeed, as will the second, but you may have an error in the third (the actual import). Also you can always pop over to the Progress tab to check what has happened. In our case we have completed the import without issue.





Click Shift F5 to stop debugging.

Click on the Data Flow tab.

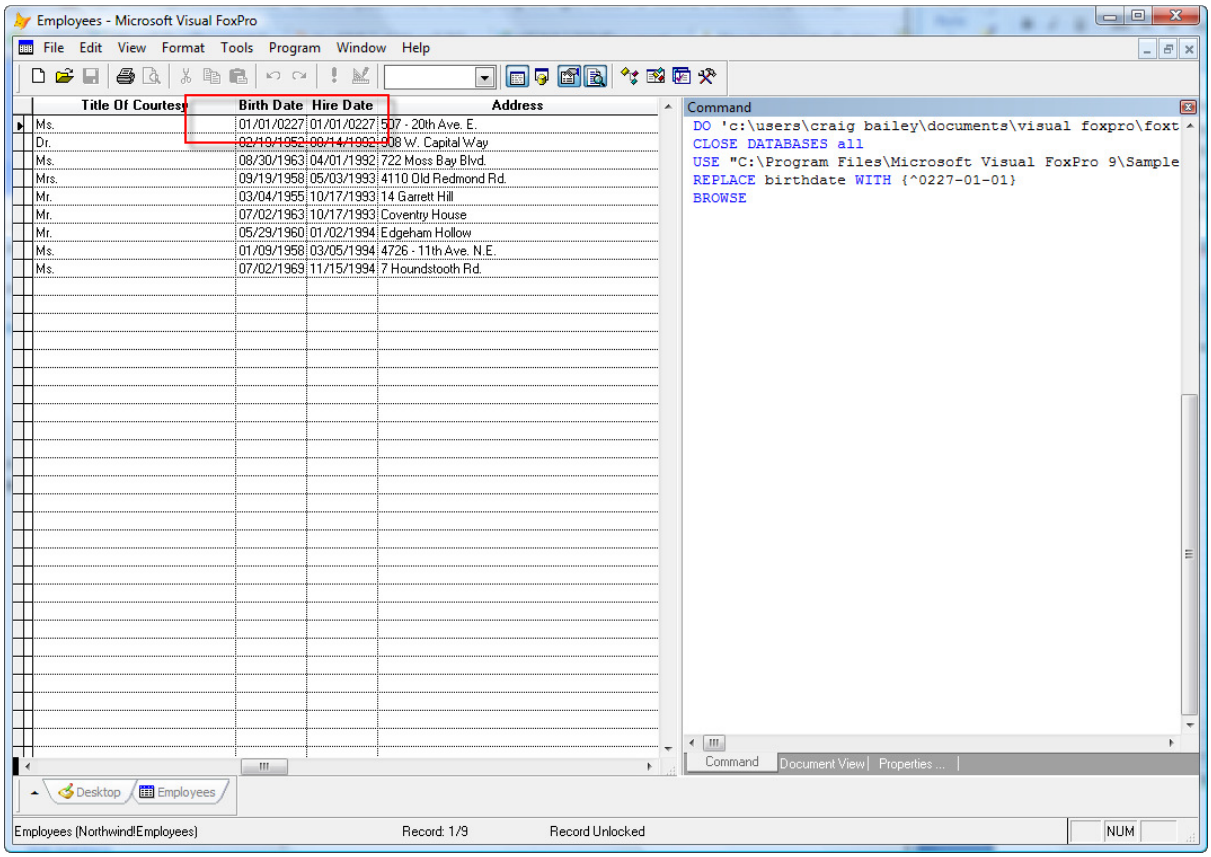
Error handling

If you have failures in the import, then you can set how errors are handled on each of the import destination blocks.

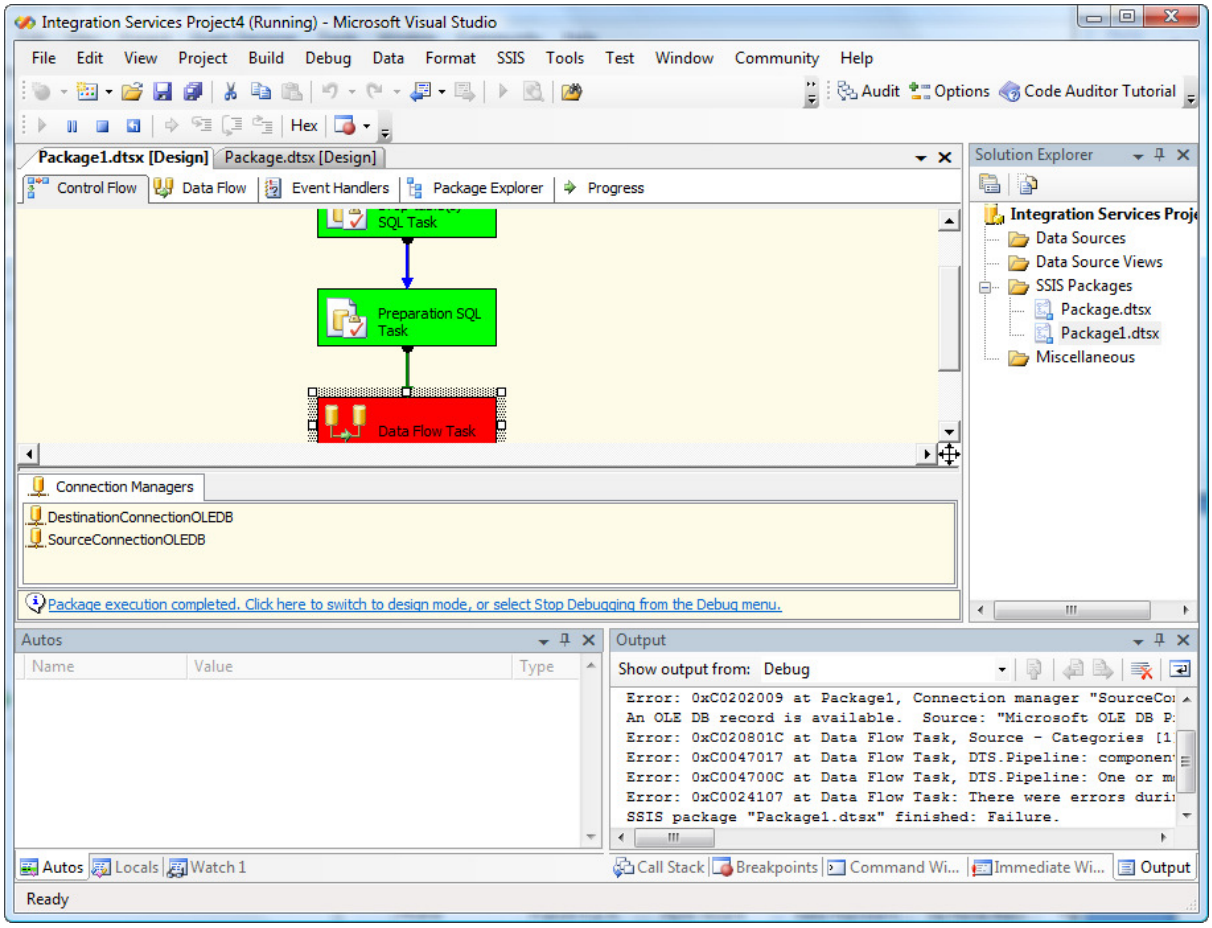
Firstly, let's edit our VFP source data to force an error.

In VFP from the command window, USE the table and enter some dodgy date data into the birthdate or hiredate fields:

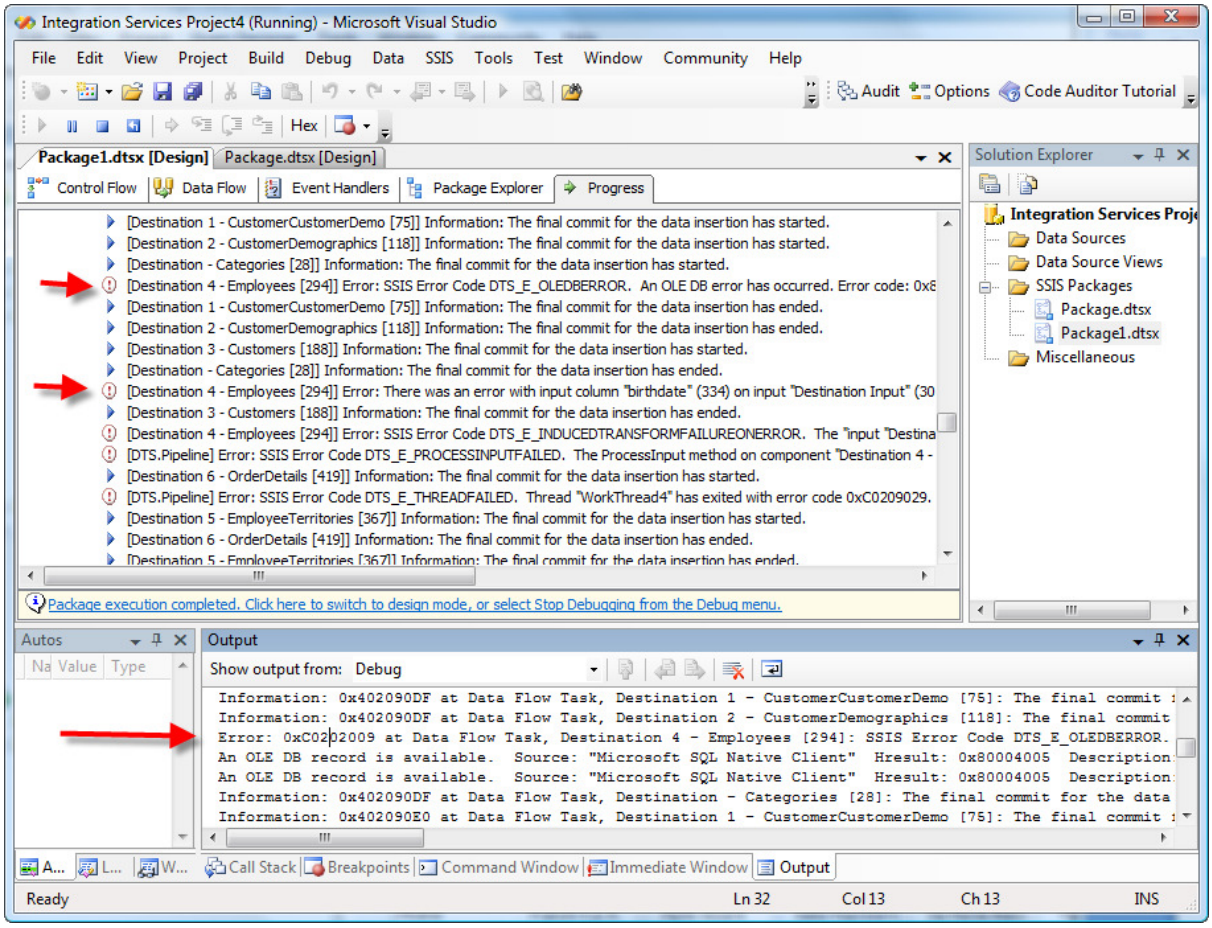
```
USE 'C:\Program Files\Microsoft Visual FoxPro 9\Samples\Northwind\Employees' IN 0
SHARED
GO top
REPLACE hiredate WITH {^0227-01-01}
BROWSE
```



Now, back in the SQL Server package, try running the Import again by hitting F5. This time it should fail.



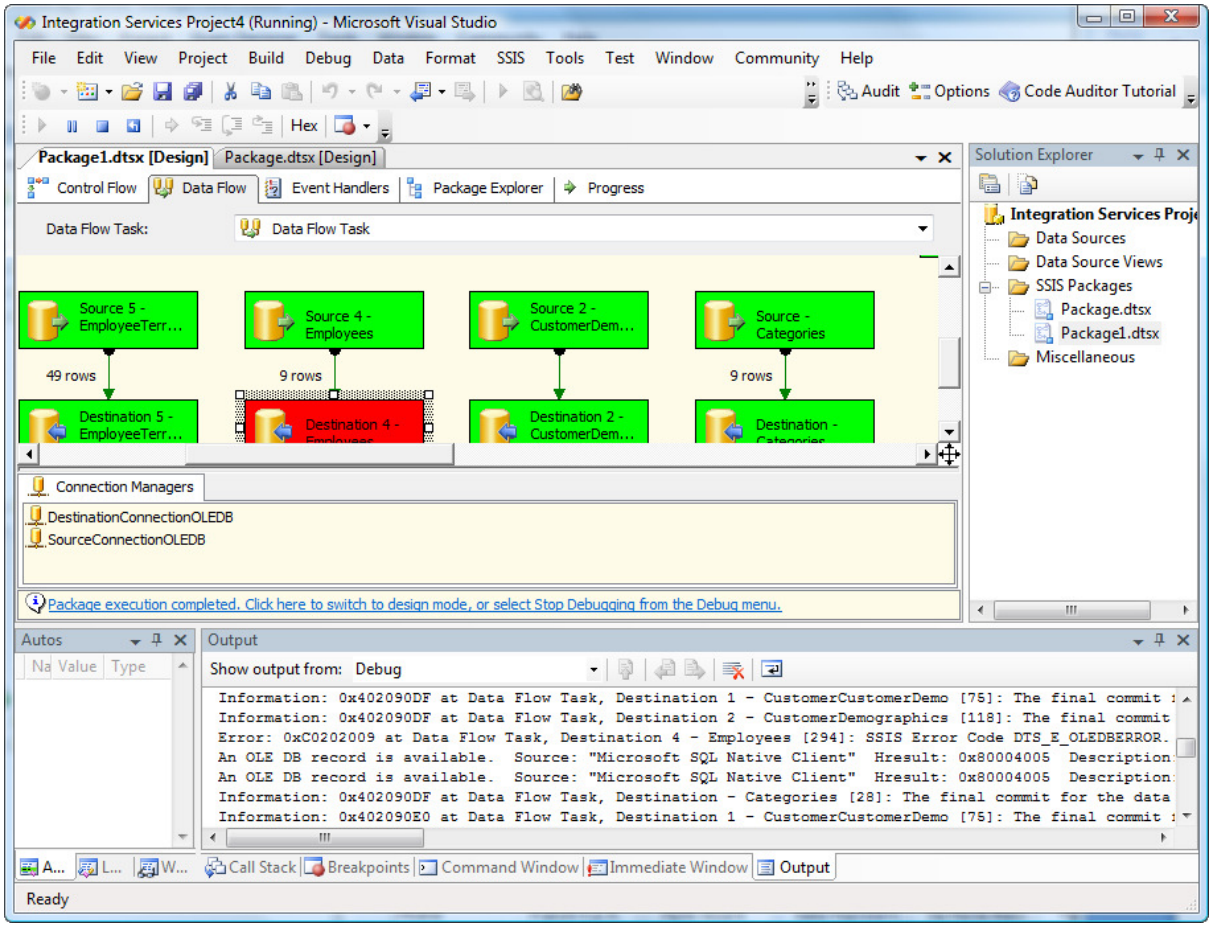
Click on the Progress tab to isolate where it occurred. You can also check the Output window for details.



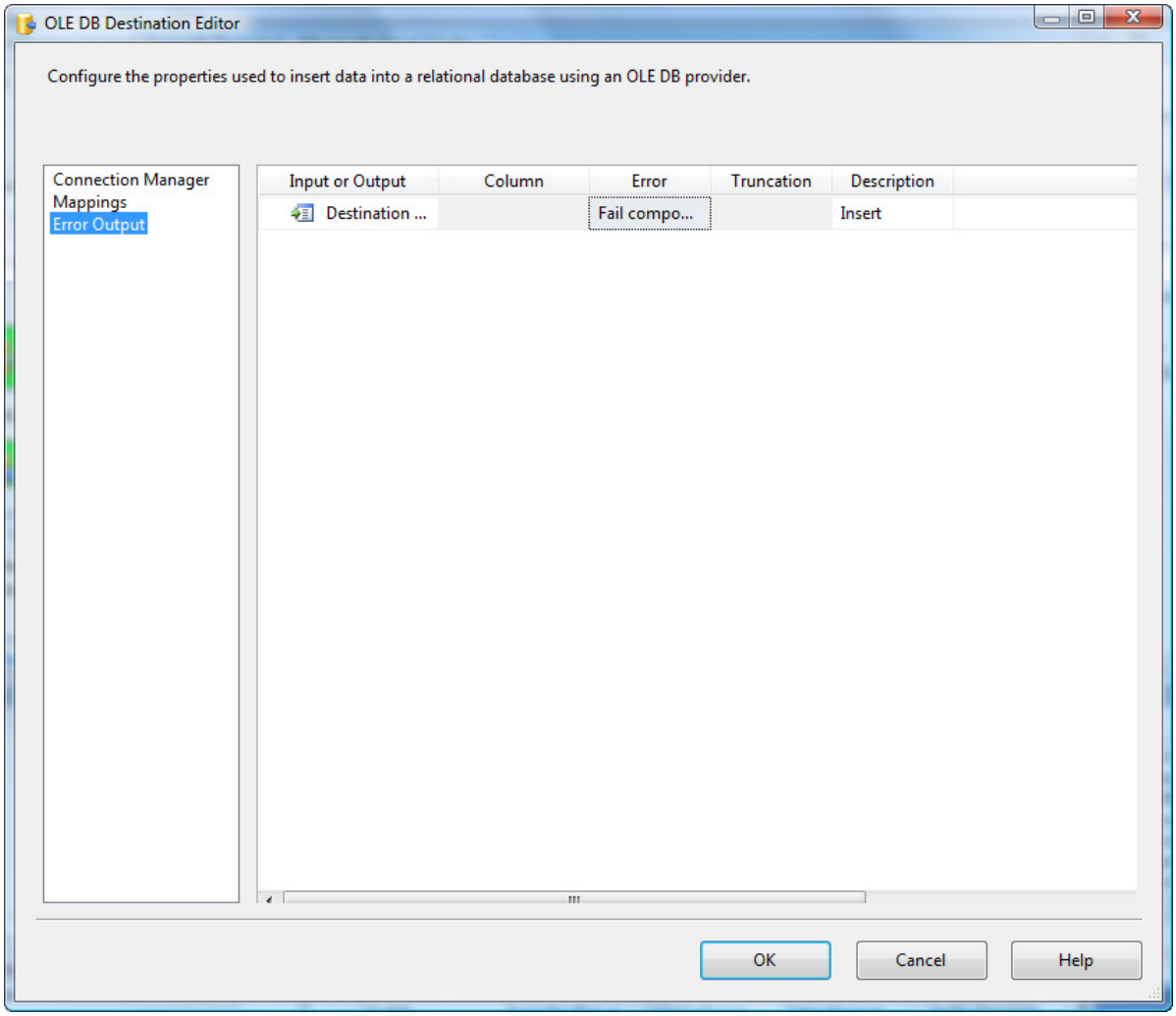
Clicking in the Data Flow tab will also show you what was finished (green), what had an error (red) and what didn't get started (white).

From here you would probably try to fix the source data. However in large data sets this isn't always possible. And thus we need to handle the errors to ensure they don't terminate the entire process.

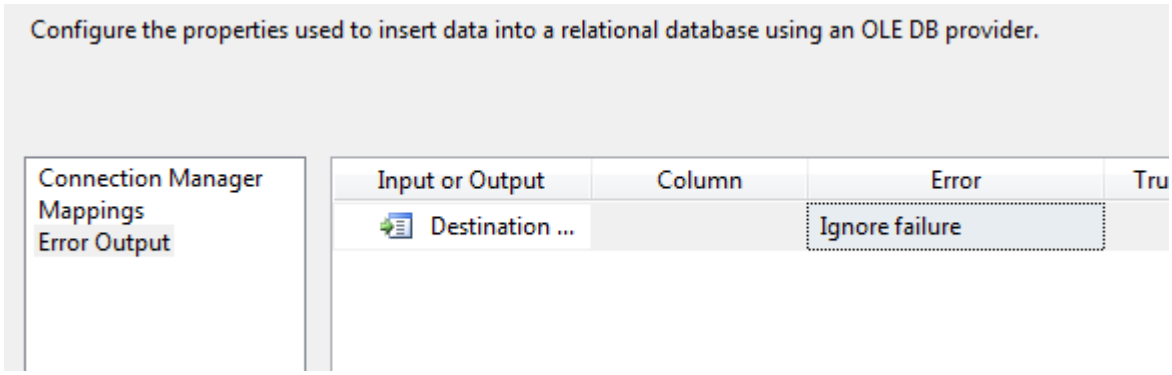
From the Data flow tab double-click on a destination block



Click on the Error Output section

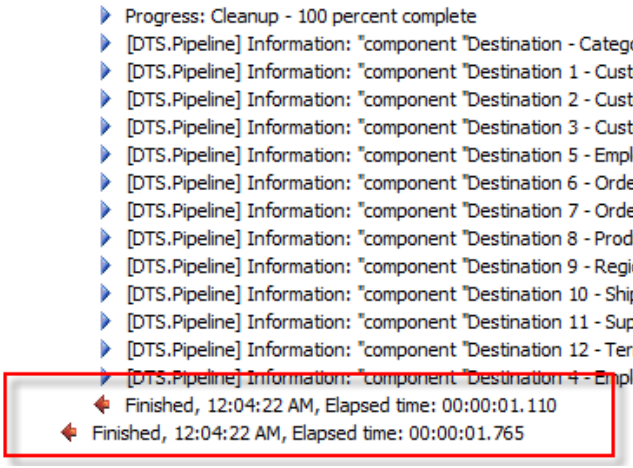


In the grid to the right we will change the way errors are handled from 'Fail component' to 'Ignore failure'. This is directing the Import process to ignore the error and continue on.

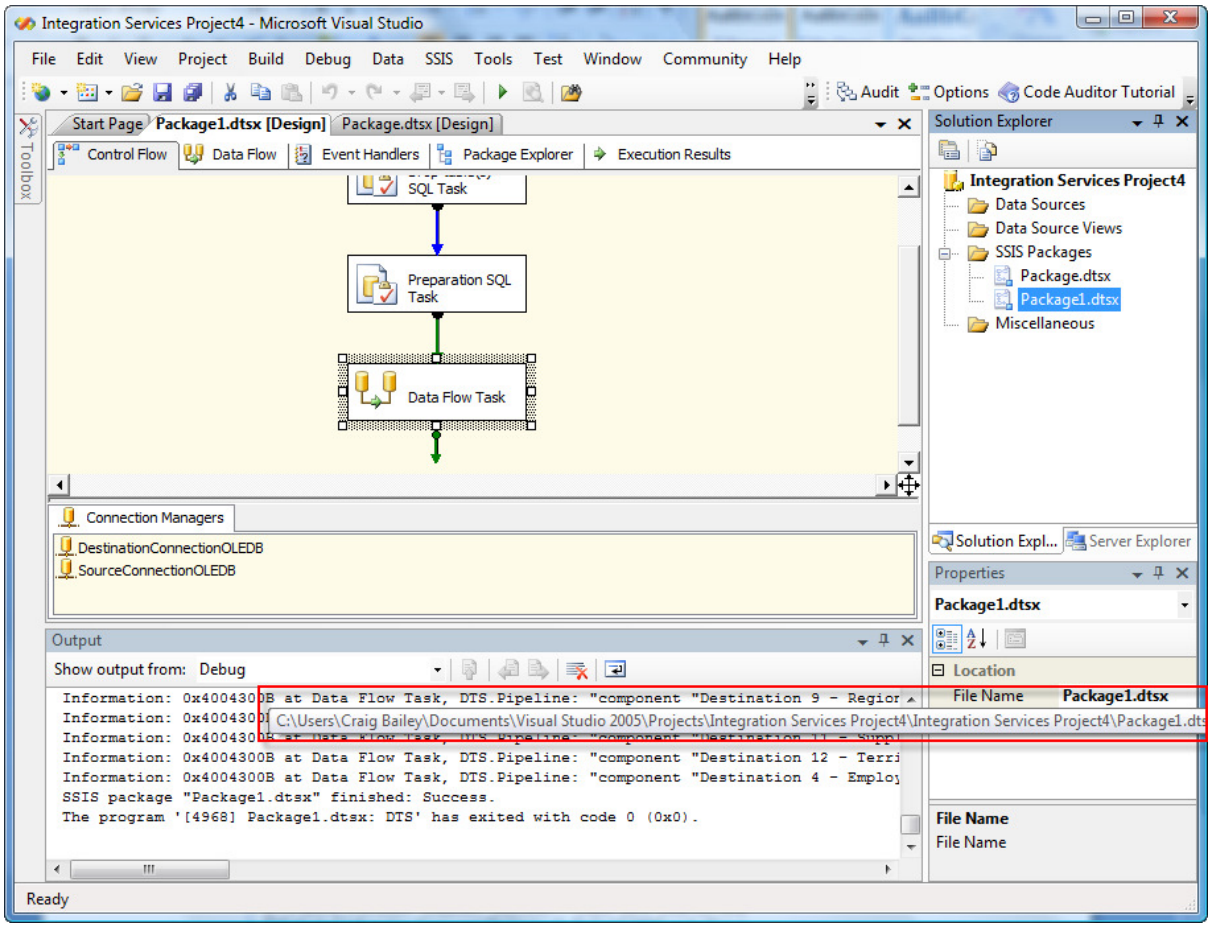


Click OK and run the package again (you may need to cancel the package with Shift F5 if it is still running).
This time the package should complete.
(Note, if you go into SQL Server and look at the data you will find that the record with problematic data has been ignored completely.)

On the progress tab you will also get a summary of how long the import process took (just under 2 seconds in our case)



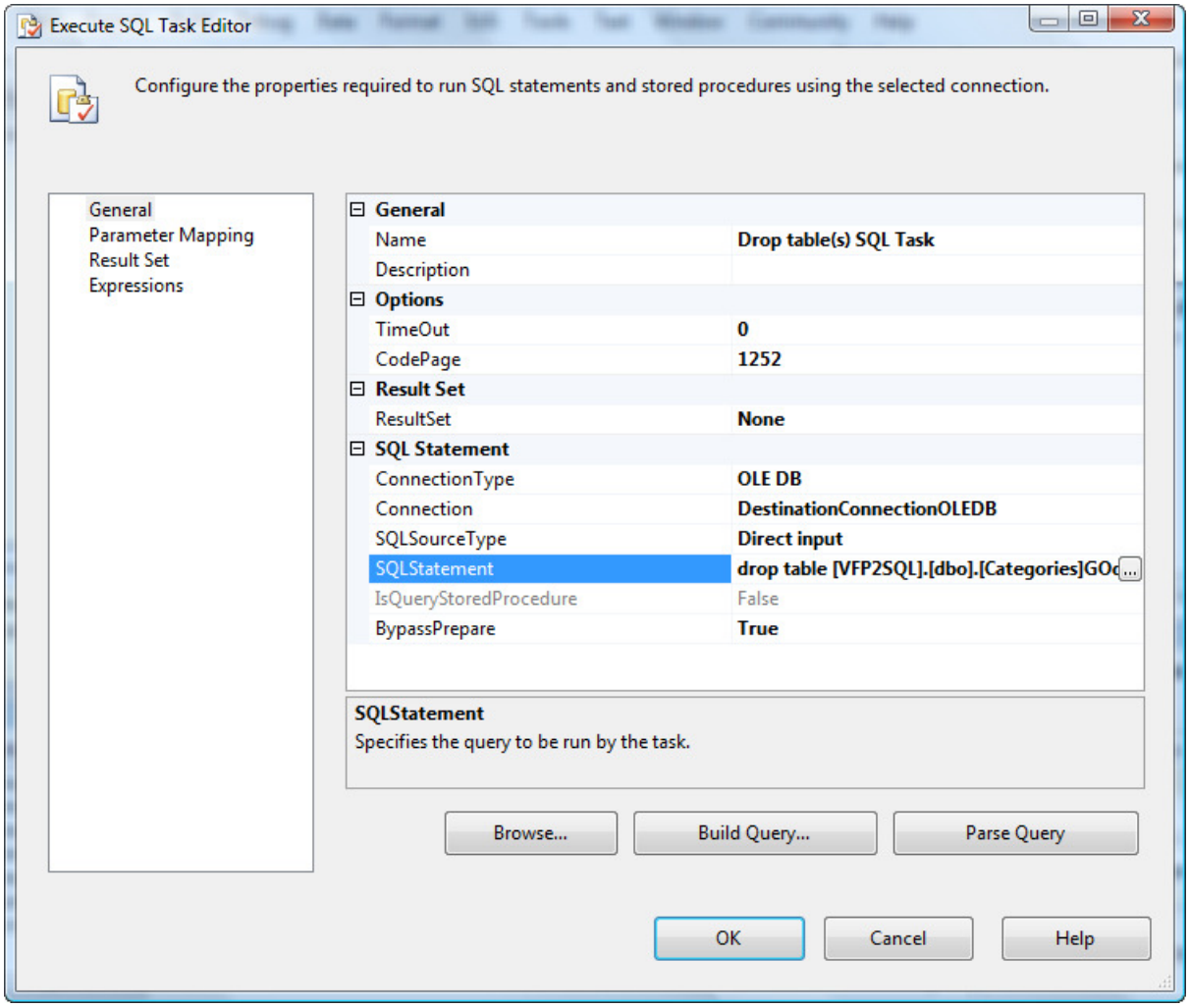
Save the package and take a note of where it is stored (by mouse hovering over the Full path property). You will need to know this location for the next steps, so that the SQL Agent job can be set up to run this import automatically on a regular schedule.



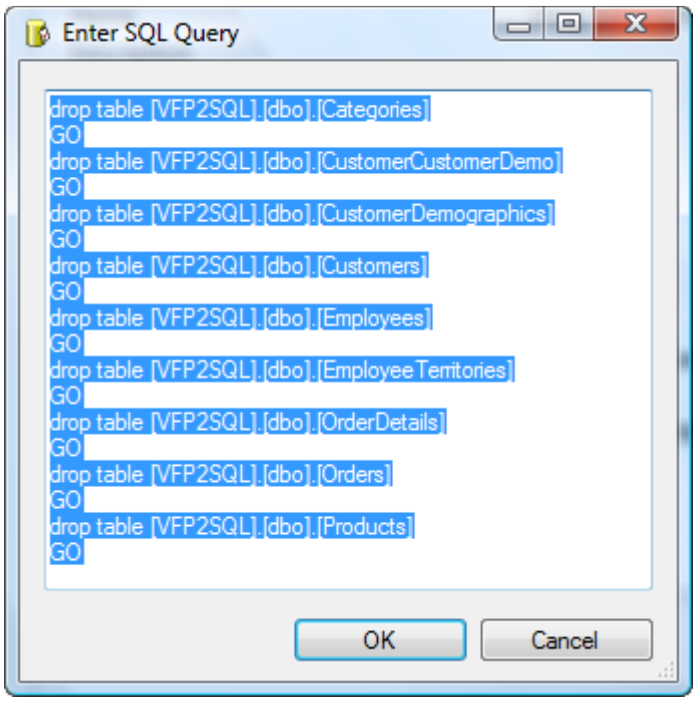
Digging into the Import package

This is an optional part that shows where you can get access to all the details of the package.

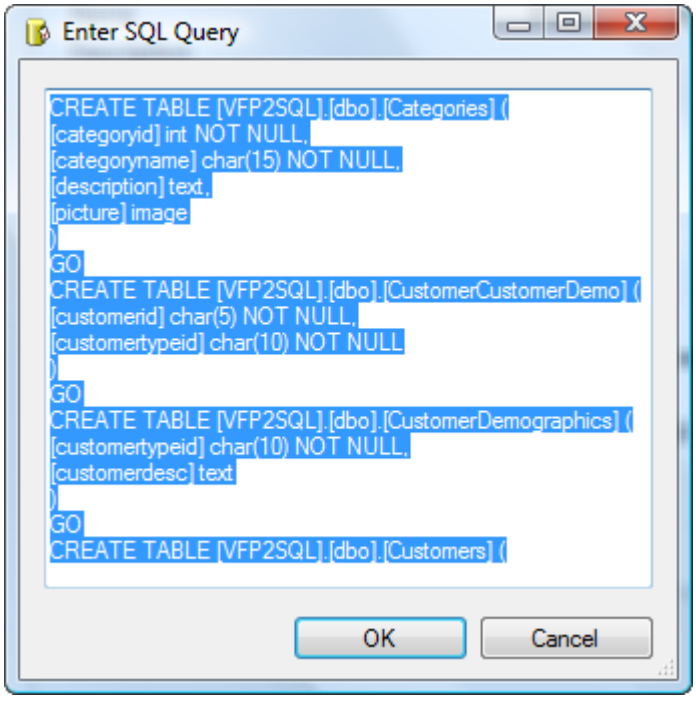
Double click on one of the tasks in the Control Flow tab (eg the Drop table block). You will be presented with a form like the following.



Click on the SQL Statement property. Then click on the button with the elipses [...] to display the full SQL statement. Here you can see the statements that are sent to SQL Server. This is the code that was automatically generated by the SSIS Wizard, based on us ticking the 'Drop and re-create table' option.



Close that block and double click on the Preparations SQL Task block back in the Control Flow tab
Look at the code for that SQL Statement and it will be similar to the following



This is all the code for creating the tables.

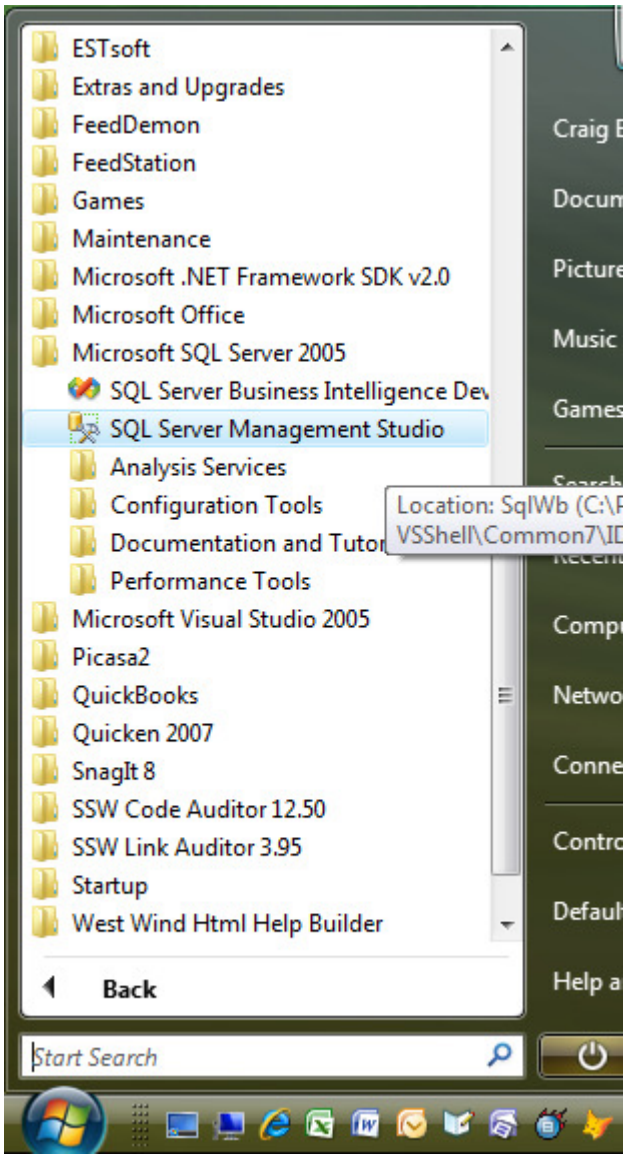
Double-clicking on the Data Flow block will take you to the Data Flow tab where each of the processes for each table to be imported is outlined.

This completes the section on setting up the Import package.

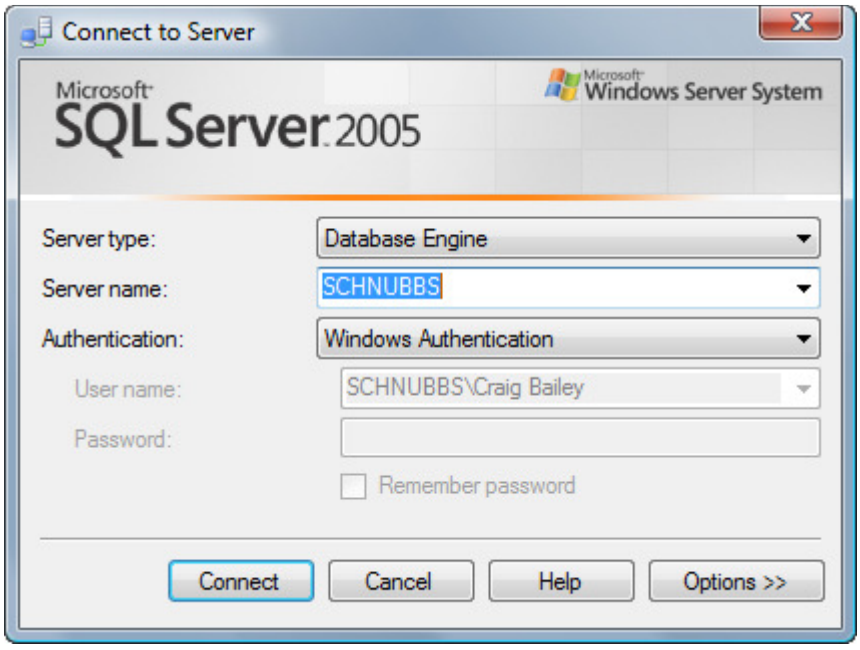
Next we will look at how to import the package to run automatically via SQL Agent.

Setting up SQL Agent to run the package

Start SQL Server Management Studio

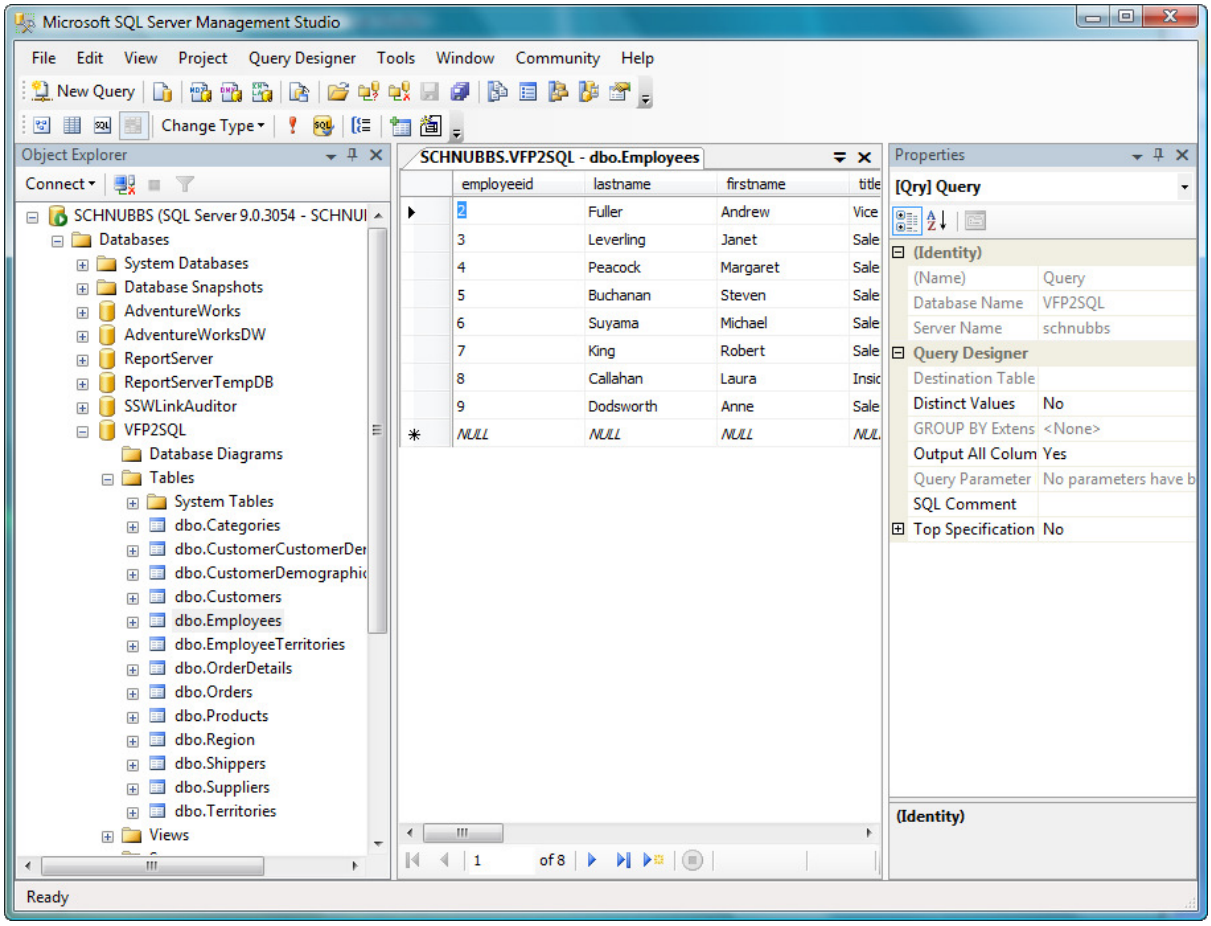


Connect to the database engine first

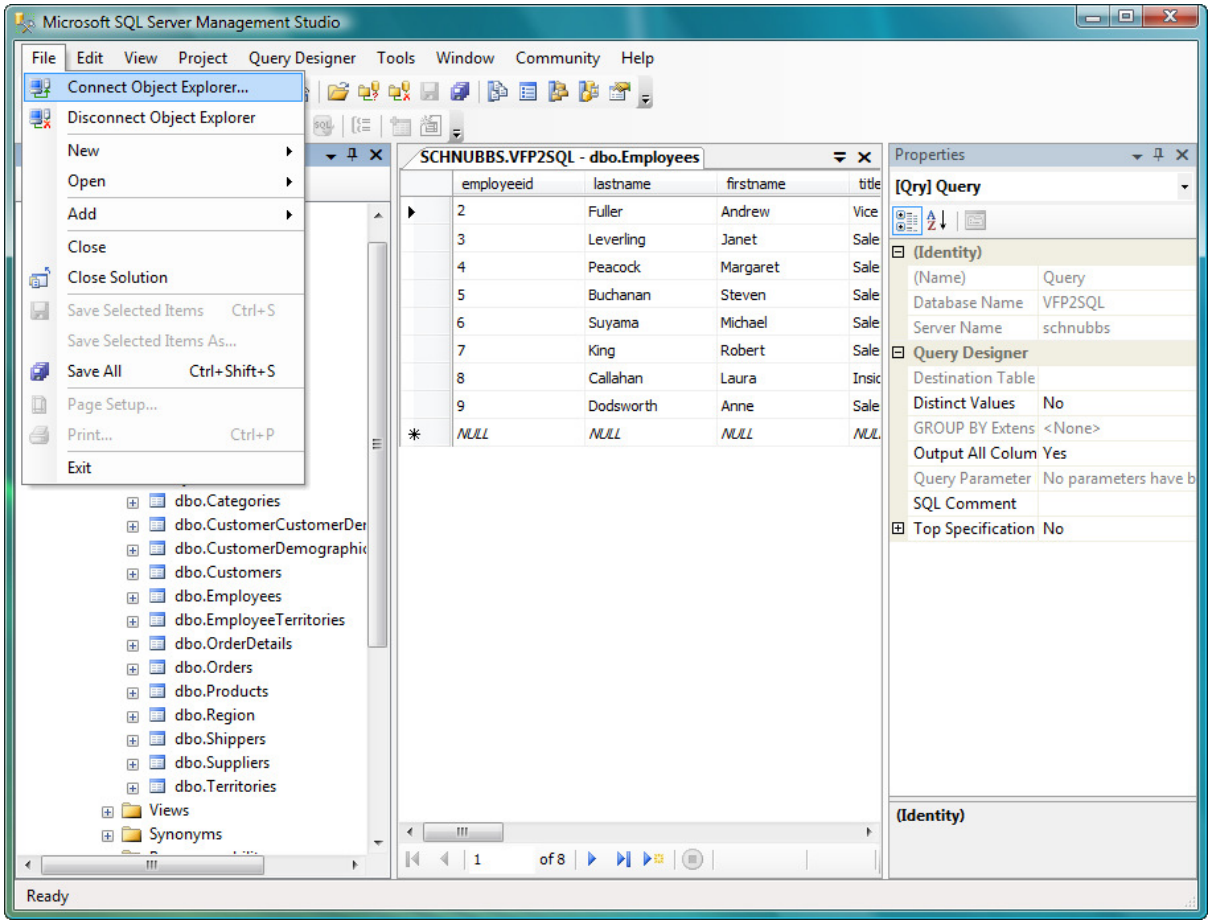


Once connected, expand out the Databases tree and find your newly created database (eg VFP2SQL). Expand the tree and check the tables.

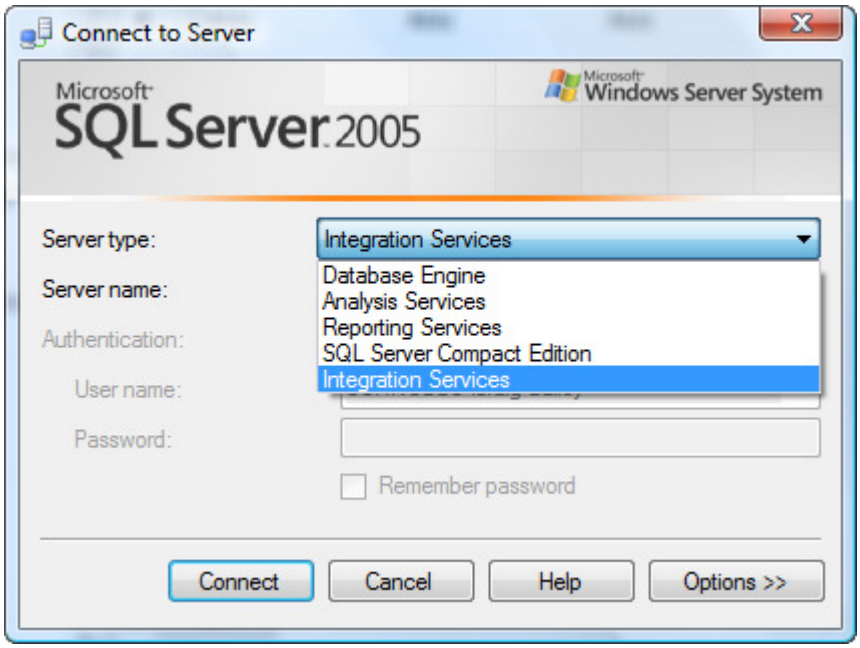
Check the Employees table as an example (notice the first record is missing – it had our dodgy date data).



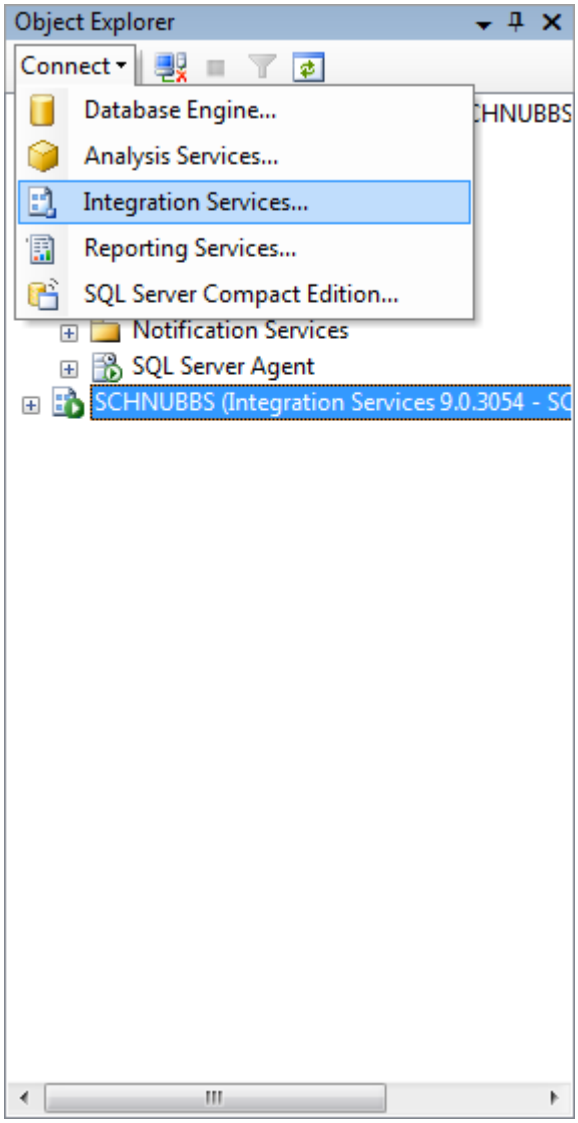
Next, you will need to connect to the Integration Services (previously you connected to the Database Server). From the File menu, choose Connect Object Explorer.



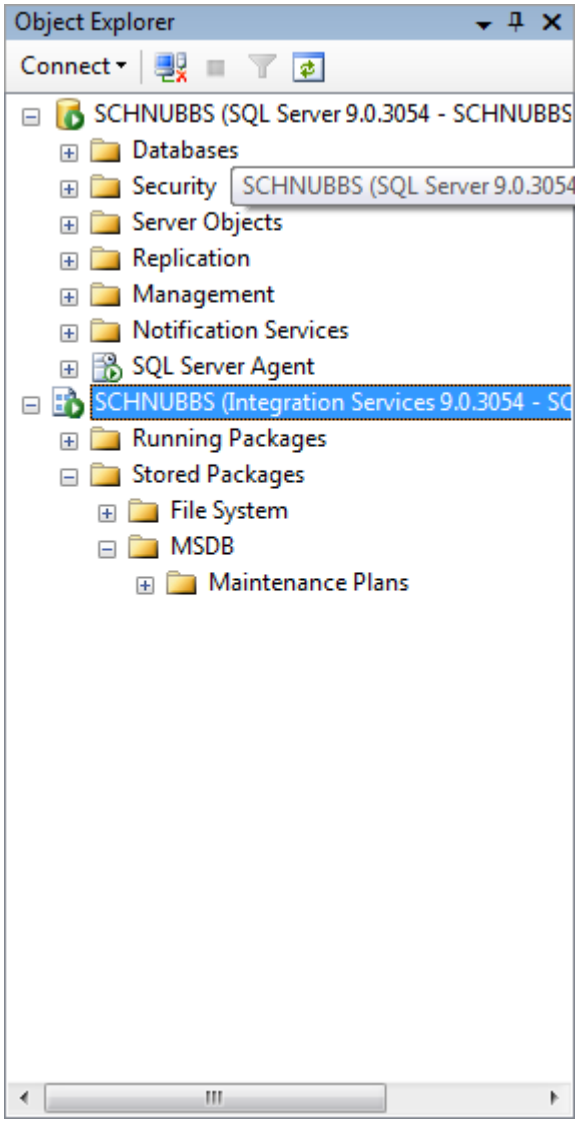
Select Integration Services in the Server type drop down



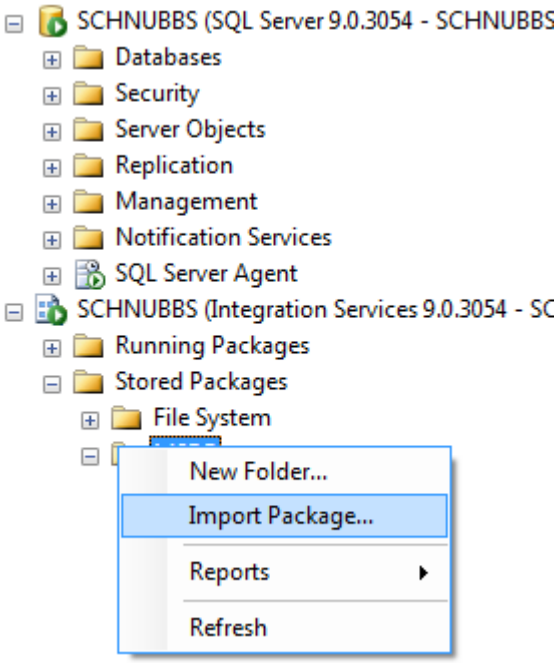
(Note, you could also achieve this by selecting Connect from the Object Explorer)



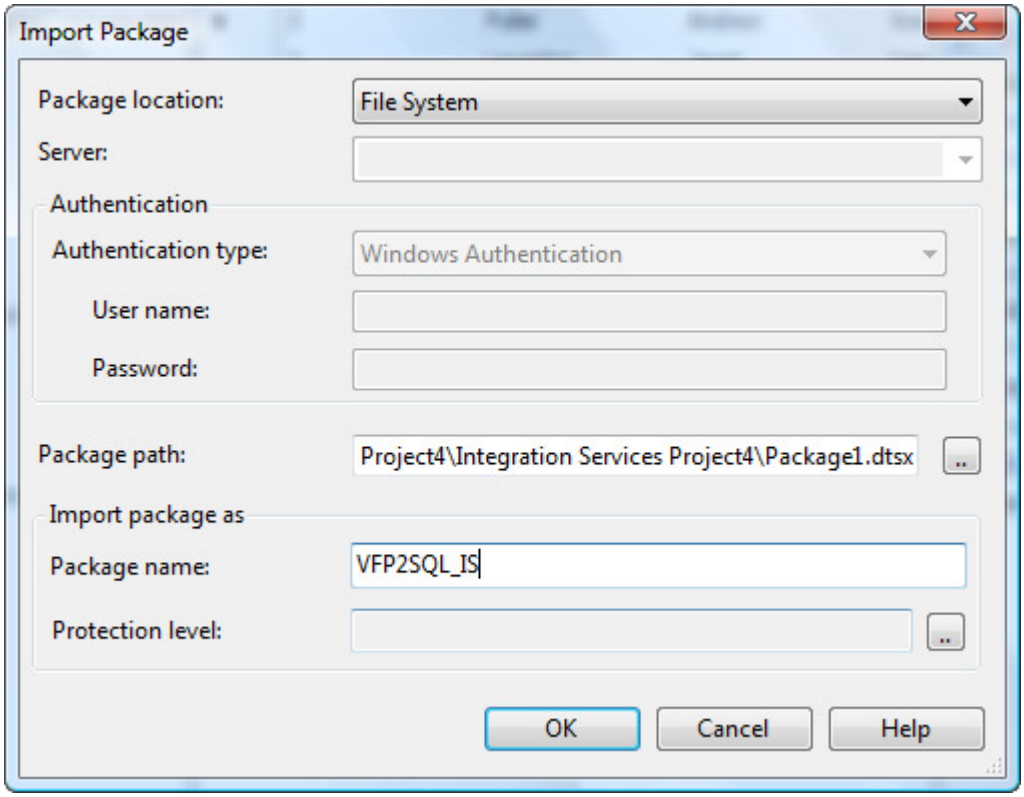
Once connected, expand out the Integration Services Stored Packages



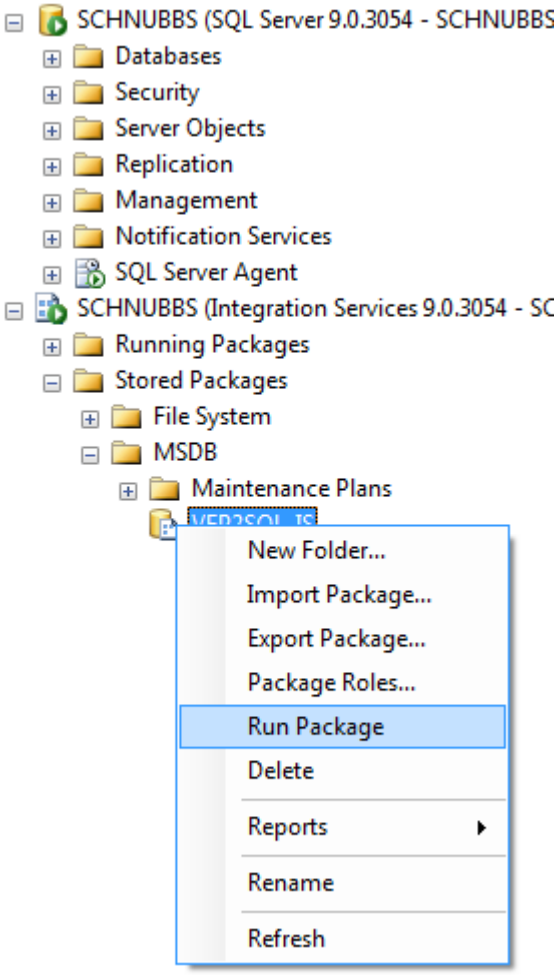
Right click on the MSDB folder and select Import package



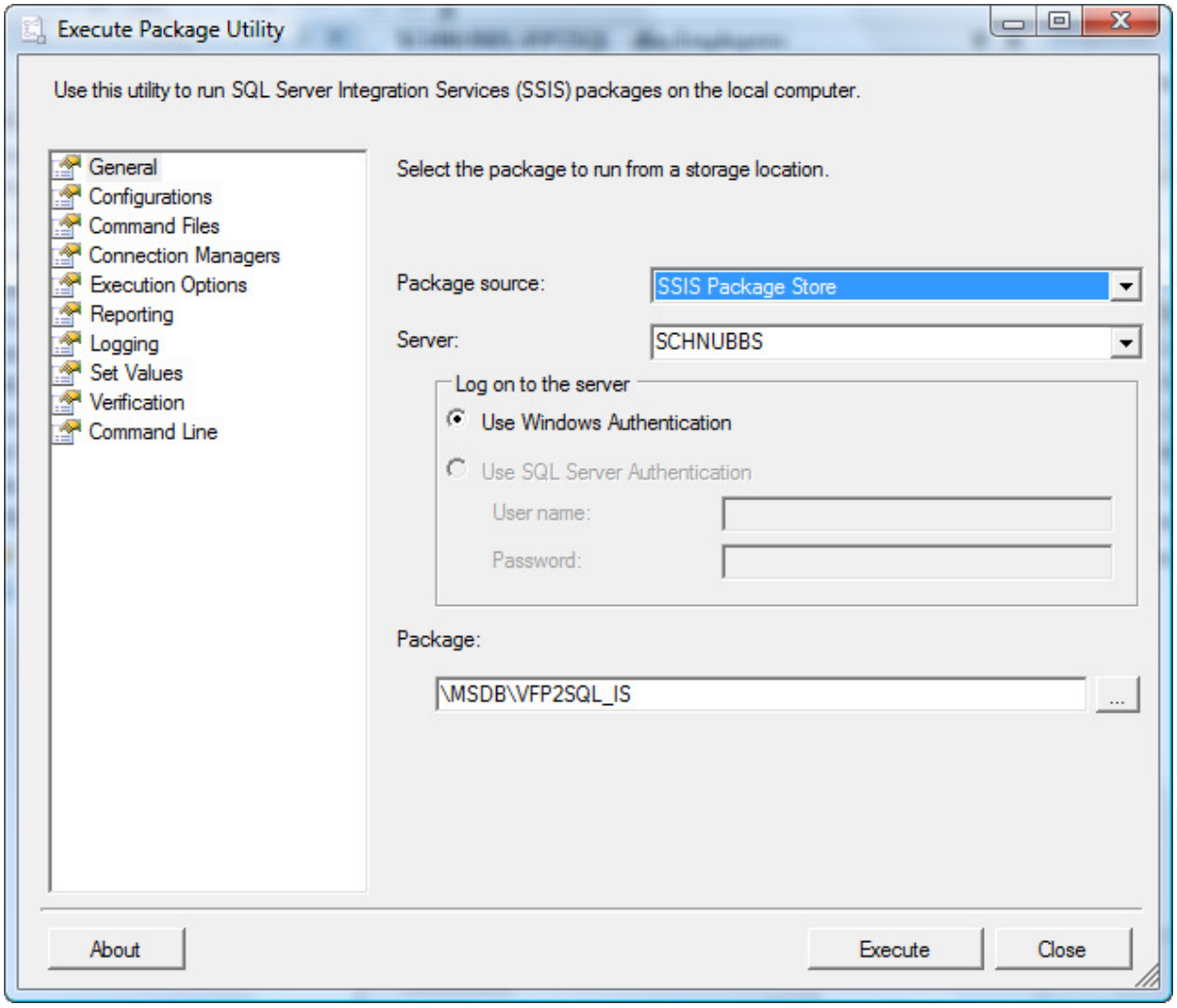
Se the Package location to File System and then set the Package Path to the location of your package (you may need to switch back to the Integration Services project and get the Full Path)
Finally give your package a name



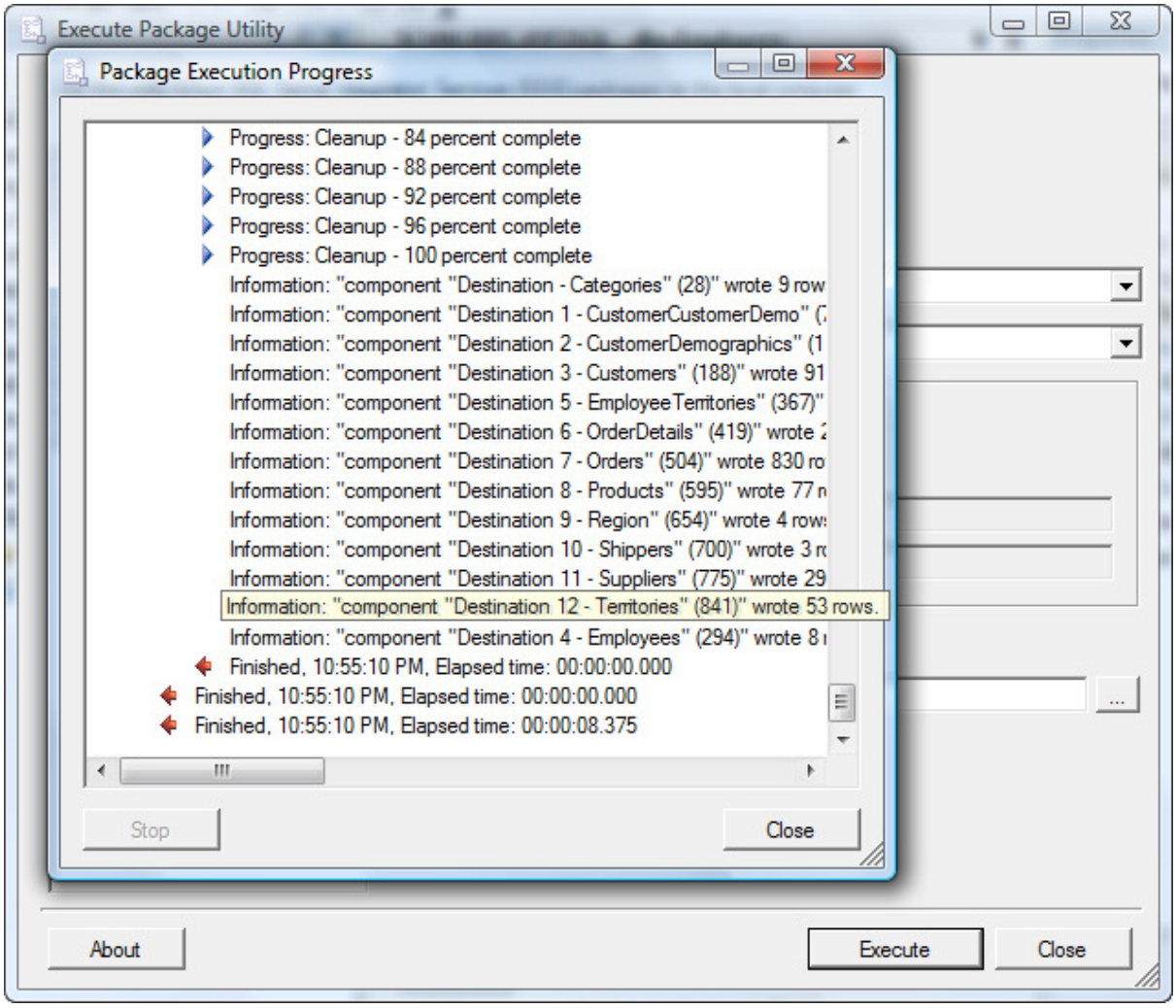
The package will appear under the folder
Right click and choose Run package to run it



Click Execute to run it

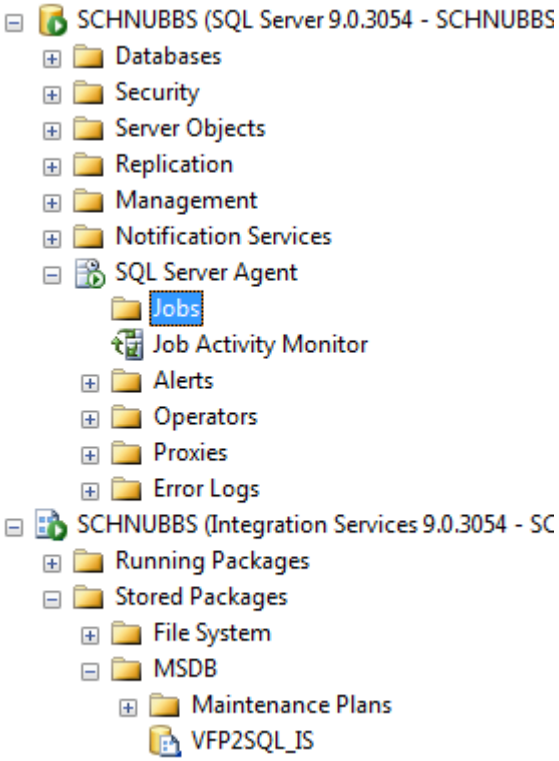


The package will run through



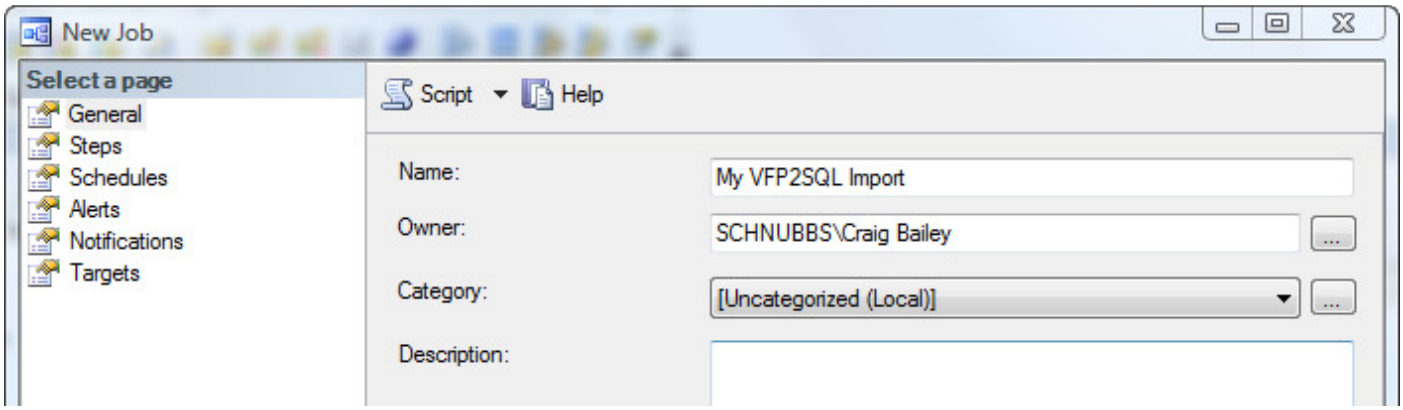
Click Close when it completes

Now expand the SQL Agent jobs



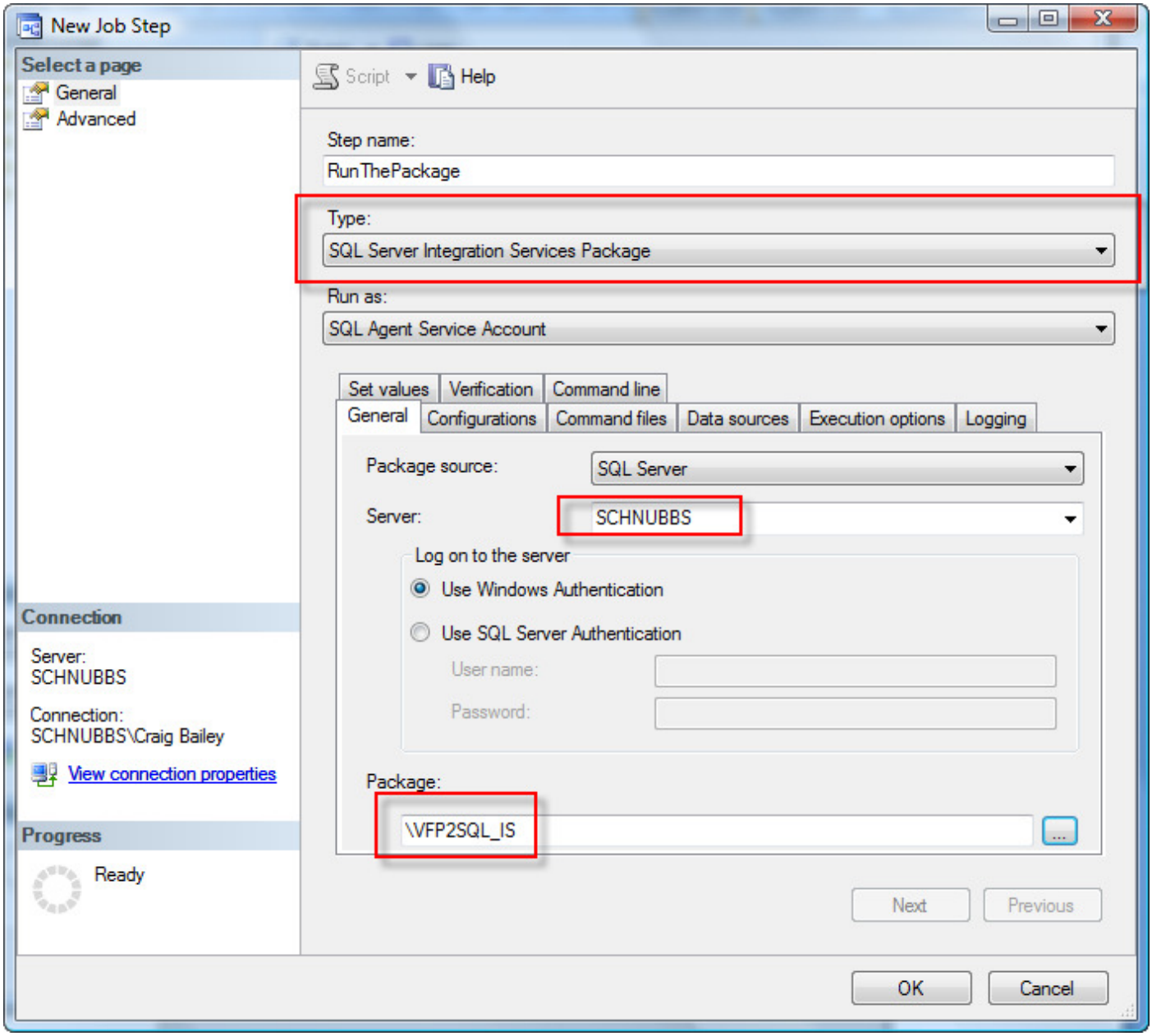
Right click and Select New Job...

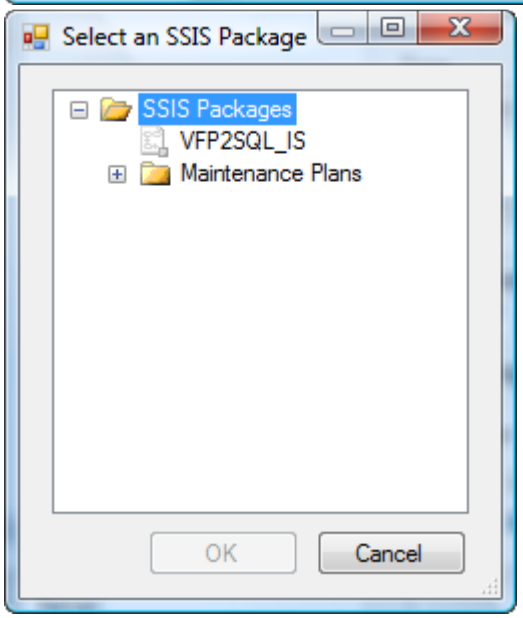
Name the job and click on the Steps page at the left



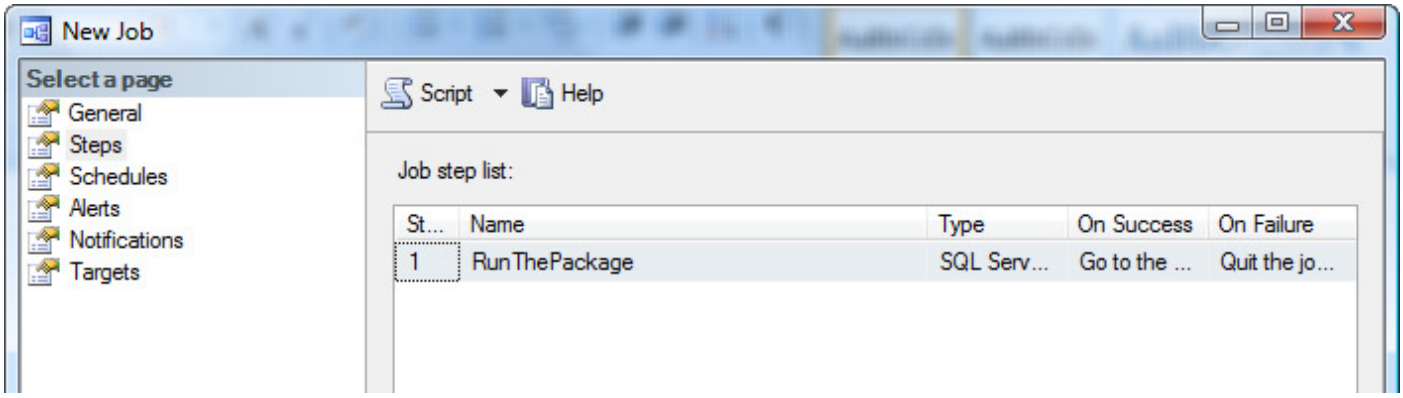
In the Steps page click New...

Add a Step name
Set the Type to be SQL Server Integration Services Package
In bottom part of the screen, set the Package source to SQL Server
Set the server to your server (the same one you have been using all along). You may have to manually type in the Server name here, sometimes it doesn't seem to know about the server)
In the Package area, select the package that we imported earlier (in our case this is VFP2SQL_IS)



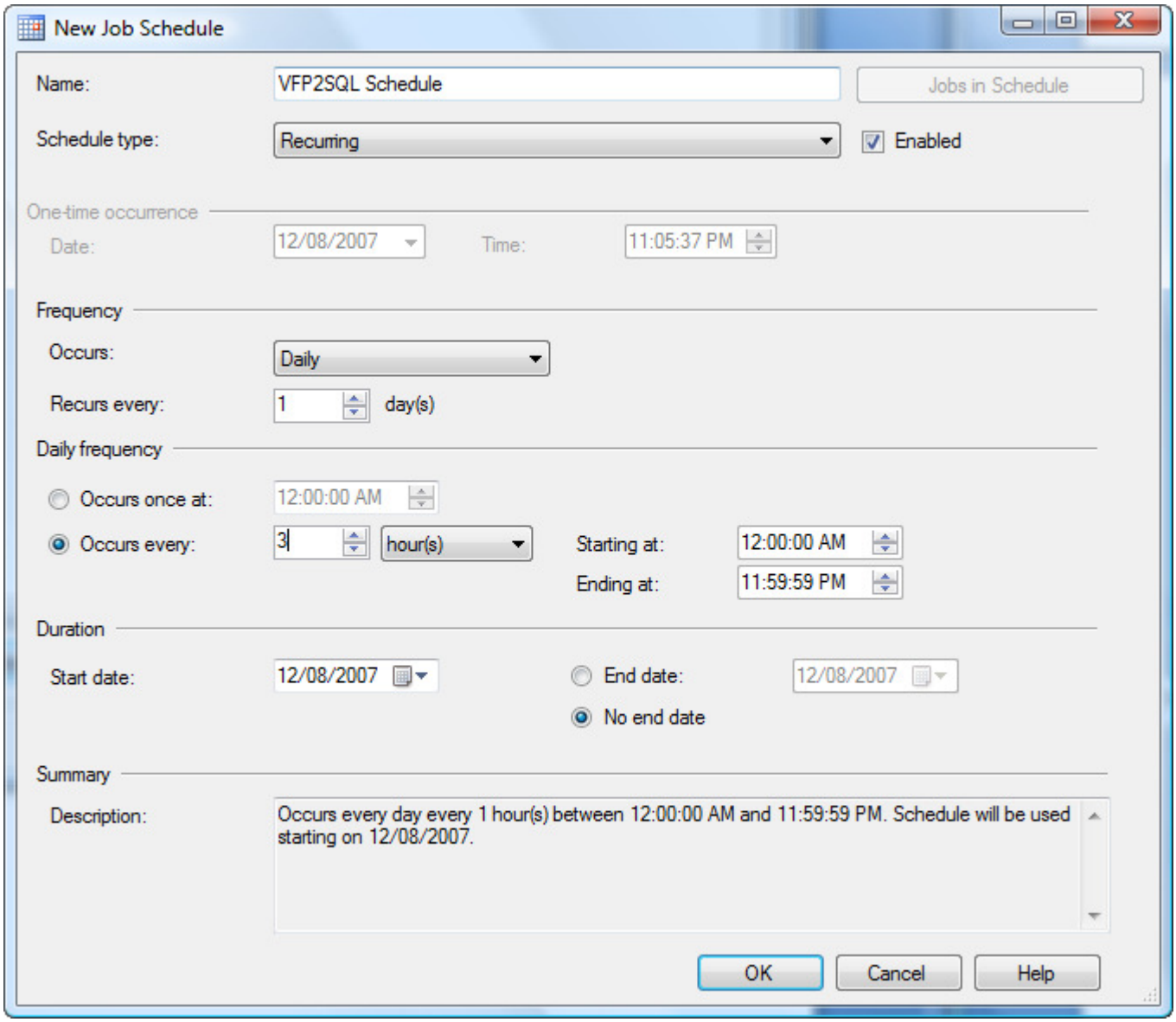


The steps should be completed along the lines of



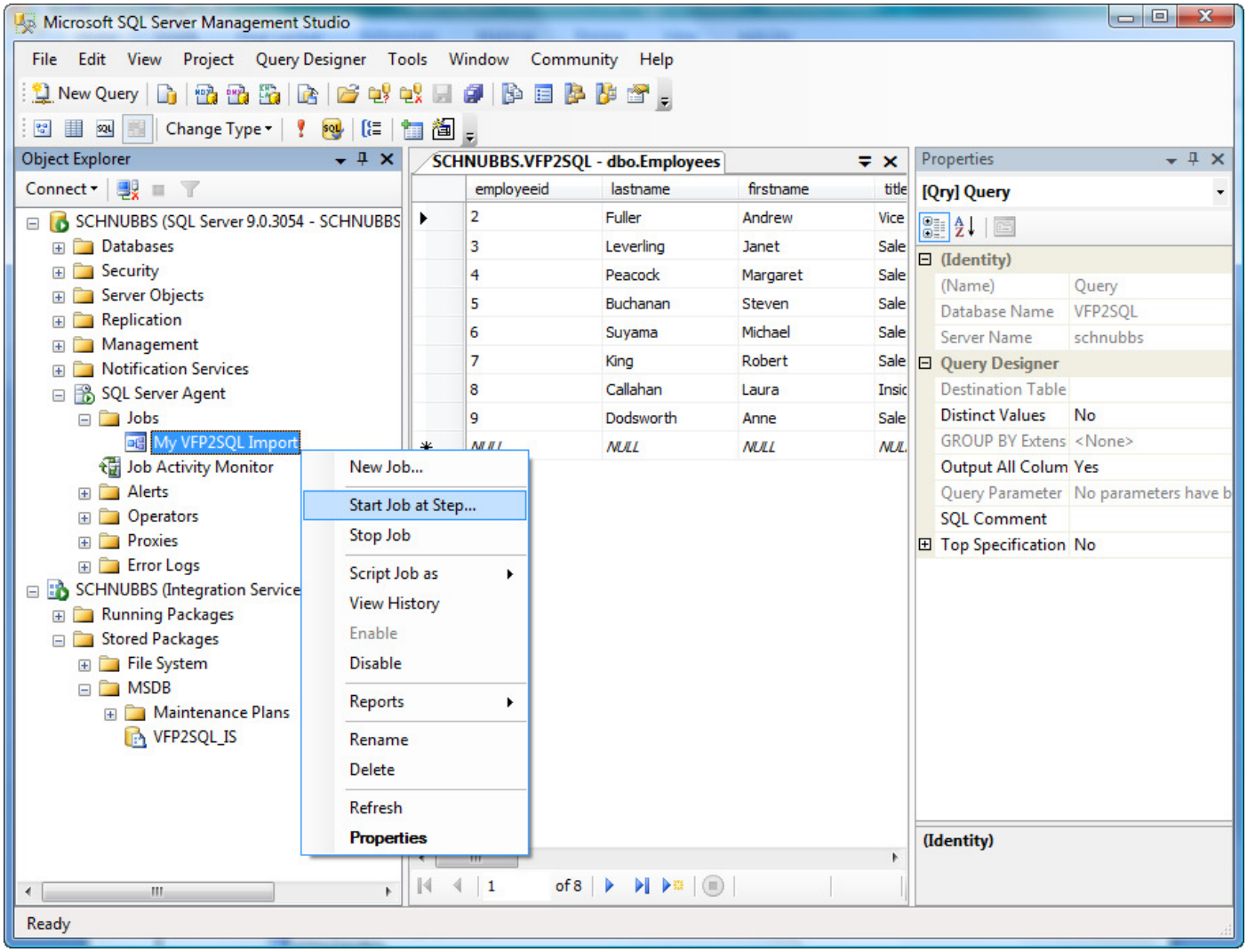
Now click on the Schedules page

Put in a name and set the frequency items.
In this case the schedule has been set to run every3 hours.
Once complete it should look something like the following

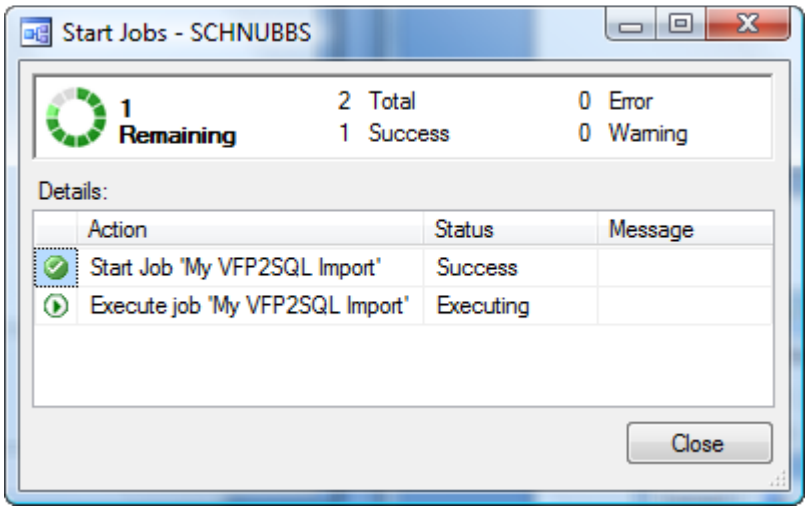


Click OK, and then OK again to finish setup of the job

You can test the job by right clicking on it and choosing Start Job at Step...



Whilst running the following form will display



Once complete click Close.

Your SQL Agent job is now all set up, and will run according to the schedule you have specified.
If the Job fails, then check that you have set the proper UNC mapping to the source Visual FoxPro files. It may be that you had a drive hard coded (eg N drive) which the SQL agent doesn't know about.
Keep in mind that you are running the Job from the SQL Server's perspective. In our steps above the SQL Server has been my local machine. This will not normally be the case. Usually you will prepare the SSIS package on a dev machine, then import it into a Server machine. That Server machine will need to know the full UNC path to the

source files. Plus it will need to have the VFP OLE DB driver installed on it. It is easy to create a package on your dev machine (that has the VFP OLE DB driver installed) and then deploy it to a server that doesn't have them installed.

Real world experience

At Talman (where I work) we're using SSIS to import a few large databases (eg >1 GB) into SQL.
Here's the results from one client import: This import occurs over a network (ie it is not a local import) and includes tables that have many memo fields.
The total import time is between 3 and 4 minutes depending on network traffic, server usage, etc.
Although not huge (we are only talking hundreds of thousands of records, not millions) it is still very quick.
Running this same process using the VFP Upsizing Wizard took hours (I don't have the exact times – it was more than half the day).

This was developed on a developer machine and deployed to a main company server. The server had UNC mapped access privileges to a N drive that stored the source VFP database files.

Summary

So there's the process for importing VFP into SQL using SQL Server Integration Services (SSIS). It has only just brushed the surface of SSIS, but demonstrates the incredible performance that can be achieved.

About the author

Craig Bailey is a Visual FoxPro MVP living in Sydney, Australia.
For further details please visit his site at www.craigbailey.net