



Exercise Manual for Course 4555
Data Migration With SQL
Server[®] Integration Services

4555/MA/A.1/502/-

by Charles Kangai

Technical Editor:
Joseph Gagliardo

© LEARNING TREE INTERNATIONAL, INC.
All rights reserved.

All trademarked product and company names are the property of their respective trademark holders.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, or translated into any language, without the prior written permission of the publisher.

Copying software used in this course is prohibited without the express permission of Learning Tree International, Inc. Making unauthorized copies of such software violates federal copyright law, which includes both civil and criminal penalties.

Exercise Manual Contents

Legend for Course Icons.....	ii
Hands-On Exercise 1.1: Examining the Source Text Files and Target Database.....	1
Hands-On Exercise 1.2: Creating a New Package Project.....	5
Hands-On Exercise 2.1: Adding Connection Managers to a Package.....	9
Hands-On Exercise 2.2: Using the Execute SQL Task to Create a Database Table.....	19
Hands-On Exercise 2.3: Using the Data Flow Task to Import from a Text File.....	23
Hands-On Exercise 2.4: Transforming Data.....	29
Hands-On Exercise 3.1: Adding a Send Mail Task to the Package.....	35
Hands-On Exercise 3.2: Creating an SSIS Variable and Using the Script Task.....	39
Hands-On Exercise 4.1: Using the Foreach Loop Container to Import Multiple Files.....	45
Hands-On Exercise 4.2: Using the SSIS Wizard to Import From a Microsoft Access Database.....	53
Hands-On Exercise 5.1: Deploying Packages to a Server.....	57
Hands-On Exercise 5.2: Executing Packages from the Command Line.....	61
Hands-On Exercise 5.3: Scheduling Package Execution.....	63
Solutions to Hands-On Exercise 1.2: Creating a New Package Product.....	65
Solutions to Hands-On Exercise 2.3: Using the Data Flow Task to Import From a Text File.....	67
Solutions to Hands-On Exercise 4.2: Using the SSIS Wizard to Import From a Microsoft Access Database.....	69
Solutions to Hands-On Exercise 5.2: Executing Packages from the Command Line.....	71
Bonus Hands-On Exercise 1: Exporting from a Database Table.....	73
Bonus Hands-On Exercise 2: Using Precedent Constraints with Variables.....	75
After-Course Sandbox: Importing Files From an FTP Location.....	77



Legend for Course Icons

Standard icons are used in the hands-on exercises to illustrate various phases of each exercise.



Major step



Warning



Action



Hint



Checkpoint



Stop



Question



Congratulations



Information



Bonus



Solution/Answer



Hands-On Exercise 1.1: Examining the Source Text Files and Target Database

Objectives

In this exercise, you will familiarize yourself with the files and database you will use in class. You will use the **SQL Server Management Studio (SSMS)** tool to examine the database.

1. Log on to the virtual machine (VM) using the following information:

User name: **Administrator**

Password: **adminpw**

2. Use Notepad to view the files that you will import into a database table. The files are located in the folder `c:\4555\ImportFiles` and are named:
 - OrderDetails_1.csv
 - OrderDetails_2.csv
 - OrderDetails_3.csv
 - OrderDetails_4.csv
 - OrderDetails_5.csv.

More detailed steps...

- a. Click the Windows logo (hereafter referred to only as **Windows**), and on the **Start** program list click **Desktop**.
- b. On the Taskbar, click **File Explorer**. In File Explorer, navigate to the folder `c:\4555\ImportFiles`.
- c. Right-click the file `OrderDetails_1.csv` and select **Open with | Notepad**. Note the file contents.
- d. Repeat the previous step to view the contents of the files `OrderDetails_1.csv`, `OrderDetails_2.csv`, `OrderDetails_3.csv`, `OrderDetails_4.csv`, and `OrderDetails_5.csv`.



Hands-On Exercise 1.1: Examining the Source Text Files and Target Database (continued)



All five files contain data in text format, with fields *OrderID*, *ProductID*, *OrderDate*, *UnitPrice*, *Quantity*, *Discount*, and *EmployeeName*. The data fields in the file are comma-separated and each record starts on a new line.

```
OrderID,ProductID,OrderDate,UnitPrice,Quantity,Discount,EmployeeName
10248,11,1996-07-04 00:00:00,14,12,0,Steven Buchanan
10248,42,1996-07-04 00:00:00,9.8,10,0,Steven Buchanan
10248,72,1996-07-04 00:00:00,34.8,5,0,Steven Buchanan
10249,14,1996-07-05 00:00:00,18.6,9,0,Michael Suyama
10249,51,1996-07-05 00:00:00,42.4,40,0,Michael Suyama
10250,41,1996-07-08 00:00:00,7.7,10,0,Margaret Peacock
10250,51,1996-07-08 00:00:00,42.4,35,0.15000001,Margaret Peacock
10250,65,1996-07-08 00:00:00,16.8,15,0.15000001,Margaret Peacock
10251,22,1996-07-08 00:00:00,16.8,6,5.0000001E-2,Janet Leverling
10251,57,1996-07-08 00:00:00,15.6,15,5.0000001E-2,Janet Leverling
10251,65,1996-07-08 00:00:00,16.8,20,0,Janet Leverling
10252,20,1996-07-09 00:00:00,64.8,40,5.0000001E-2,Margaret Peacock
10252,33,1996-07-09 00:00:00,2,25,5.0000001E-2,Margaret Peacock
10252,60,1996-07-09 00:00:00,27.2,40,0,Margaret Peacock
10253,31,1996-07-10 00:00:00,10,20,0,Janet Leverling
10253,39,1996-07-10 00:00:00,14.4,42,0,Janet Leverling
10253,49,1996-07-10 00:00:00,16,40,0,Janet Leverling
10254,24,1996-07-11 00:00:00,3.6,15,0.15000001,Steven Buchanan
10254,55,1996-07-11 00:00:00,19.2,21,0.15000001,Steven Buchanan
10254,74,1996-07-11 00:00:00,8,21,0,Steven Buchanan
10255,2,1996-07-12 00:00:00,15.2,20,0,Anne Dodsworth
10255,16,1996-07-12 00:00:00,13.9,35,0,Anne Dodsworth
10255,36,1996-07-12 00:00:00,15.2,25,0,Anne Dodsworth
10255,59,1996-07-12 00:00:00,44,30,0,Anne Dodsworth
```

- Verify that there is currently no data in the target database `ClassDB`, which is located on your local SQL Server instance.


More detailed steps...

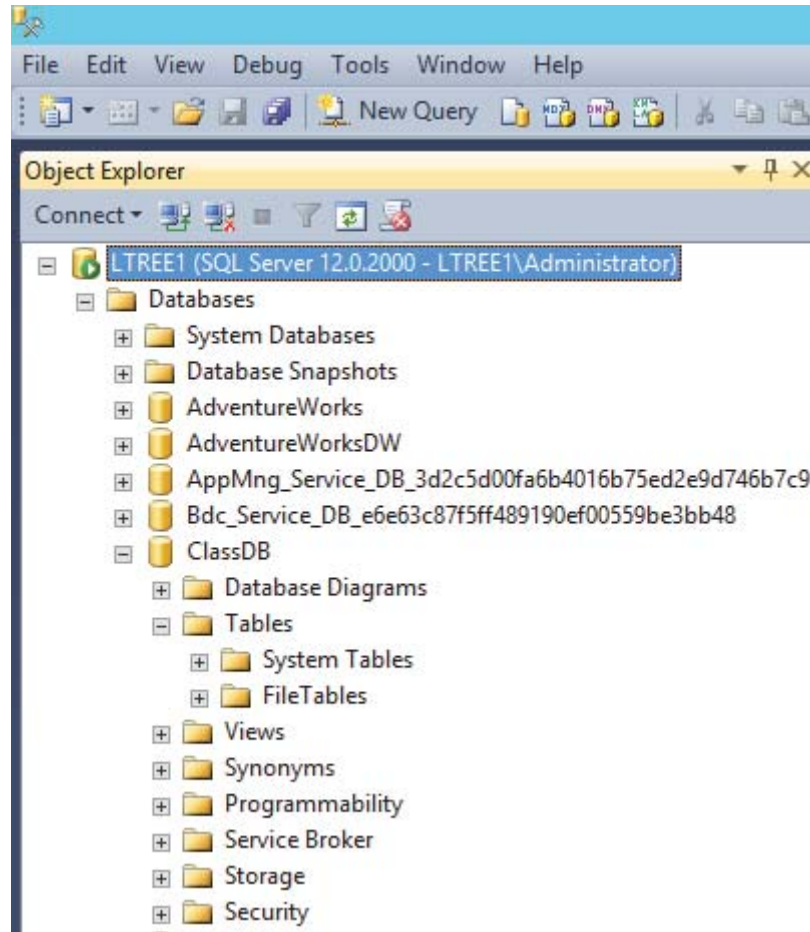
- Click **Windows**, and from the Start list, click **SQL Server 2014 Management Studio**.
- In the "Connect to Server" dialog box, make sure the "Server type" is **Database Engine** and the "Server name" is **LTREE1** or **localhost**. Click **Connect**.



**Hands-On Exercise 1.1:
Examining the Source Text Files and Target Database
(continued)**

- c. In the Object Explorer window on the left, expand the **Databases** node, expand the **ClassDB** database node, and expand **Tables**. There should be no tables listed.

 *There should be no user tables in the database. In particular, there is no table named `OrderDetails`.*



4. Leave SQL Server Management Studio (SSMS) open for a later exercise. Close any Notepad windows you might have open.



Hands-On Exercise 1.1: Examining the Source Text Files and Target Database (continued)



Congratulations! You have familiarized yourself with the files you are going to import and the database into which you will import the files, and you completed the exercise.



This is the end of the exercise.



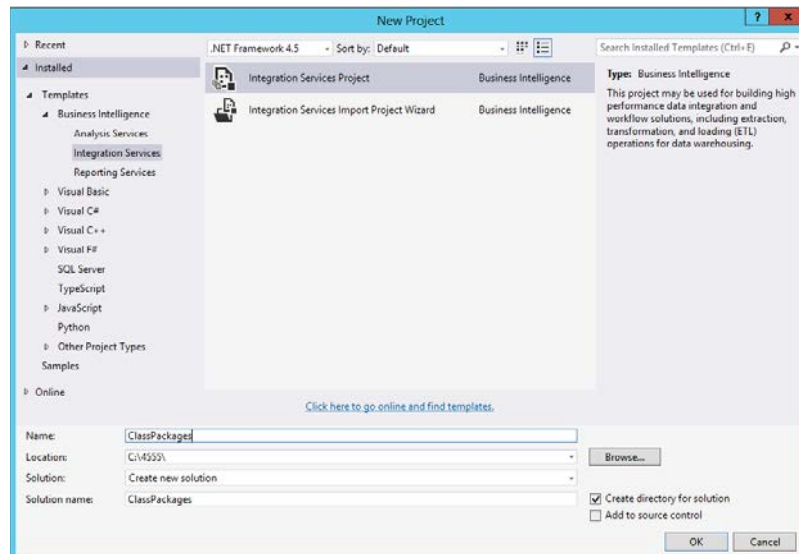
Objectives

In this exercise, you will create a new SQL Server Integration Services (SSIS) project. The project will contain the packages you will create.

1. Open **SQL Server Data Tools for Visual Studio 2013(SSDT)** and create a new project/solution named `ClassPackages` in the `c:\4555` folder.

More detailed steps...

- a. Click **Windows**, and in the Start program list, click **SQL Server Data Tools for Visual Studio 2013 (SSDT)**.
- b. From the **File** menu, select **New | Project**. In the New Project dialog box, expand **Templates**, expand **Business Intelligence**, and highlight **Integration Services**.
- c. In the Name box, enter the name `ClassPackages`. In the Location box, click **Browse**, navigate to the folder `c:\4555`, and click **Select Folder**. Click **OK**.



2. If a "Getting Started (SSIS)" window appears, close it.



Hands-On Exercise 1.2: Creating a New Package Project (continued)

3. □ View the contents of the following panes/windows that you will be using throughout the course: SSIS Toolbox, Solution Explorer, Properties, and Design. If you close any of these panes/windows, redisplay by clicking the corresponding command in the View menu.



a. What do you think the SSIS Toolbox is for?



b. What is the Solution Explorer pane for?



c. How do you think the Properties window will be used?

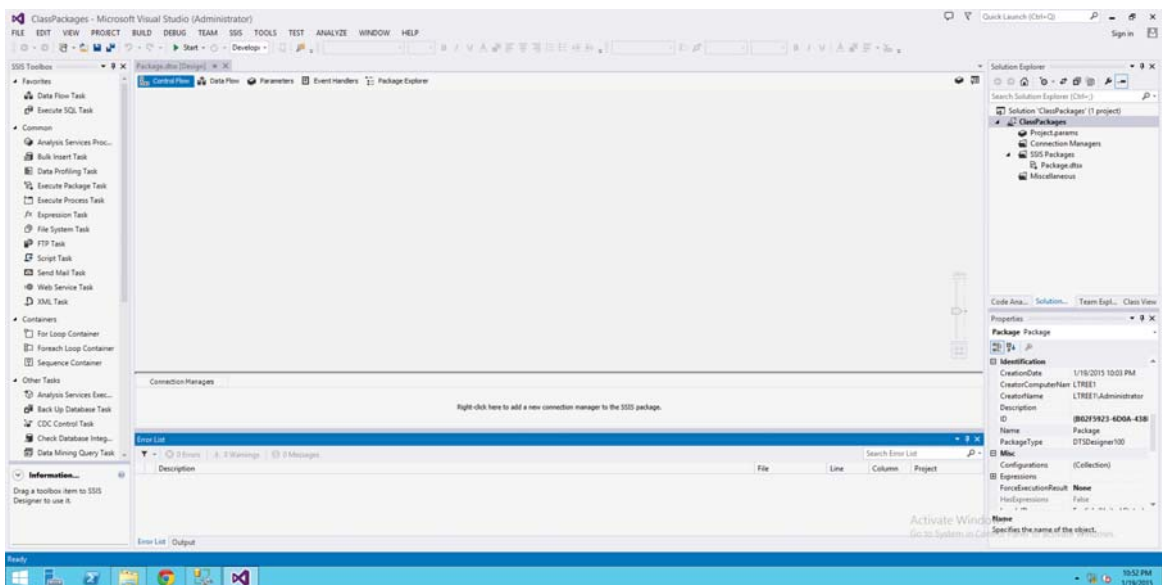


d. What kind of activity is performed within the Design window?



e. What do you think the lower part of the Design screen labeled "Connection Managers" is for?

4. □ Click the **Save All** icon in the toolbar and leave SSDT open for the next exercise.





Congratulations! You have created a new SSIS project and completed the exercise.



This is the end of the exercise.





Objectives

In this exercise, you will add connection managers for the data sources and servers you will use. Connection managers define how your package should connect to the data sources or servers. There are two types: package level and project level. You will create a project-level connection manager to point to your database on SQL Server, a package-level connection manager to point to the text file you will import, and an SMTP connection manager to point to the mail server that your package will connect to in order to send e-mail.

1. Create a *package-level*, not project-level, Flat File connection manager to point to the text file `OrderDetails_1.csv`, which is located in the folder `c:\4555\ImportFiles`. Be sure to name the connection manager `OrderDetails`. Use the data types in the table below to configure the source columns of the connection manager.

Column	Data Type
OrderID	four-byte signed integer [DT_I4]
ProductID	four-byte signed integer [DT_I4]
OrderDate	database timestamp [DT_DBTIMESTAMP]
UnitPrice	currency [CT_CY]
Quantity	two-byte signed integer [DT_I2]
Discount	float [DT_R4]
EmployeeName	Unicode string [DT_WSTR]

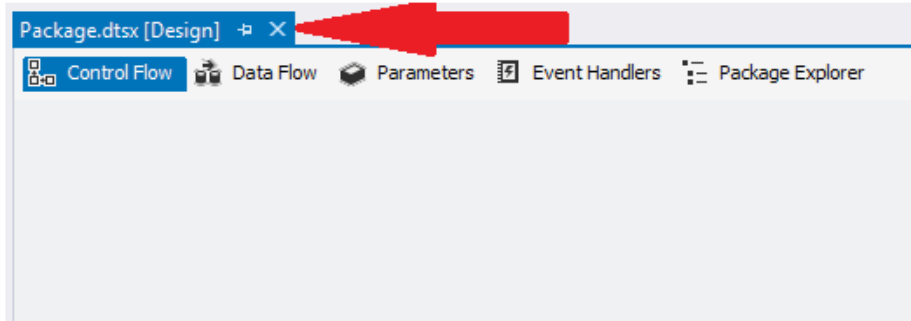
More detailed steps...

- a. Switch to SQL Server Data Tools. (It should still be open from the last exercise).



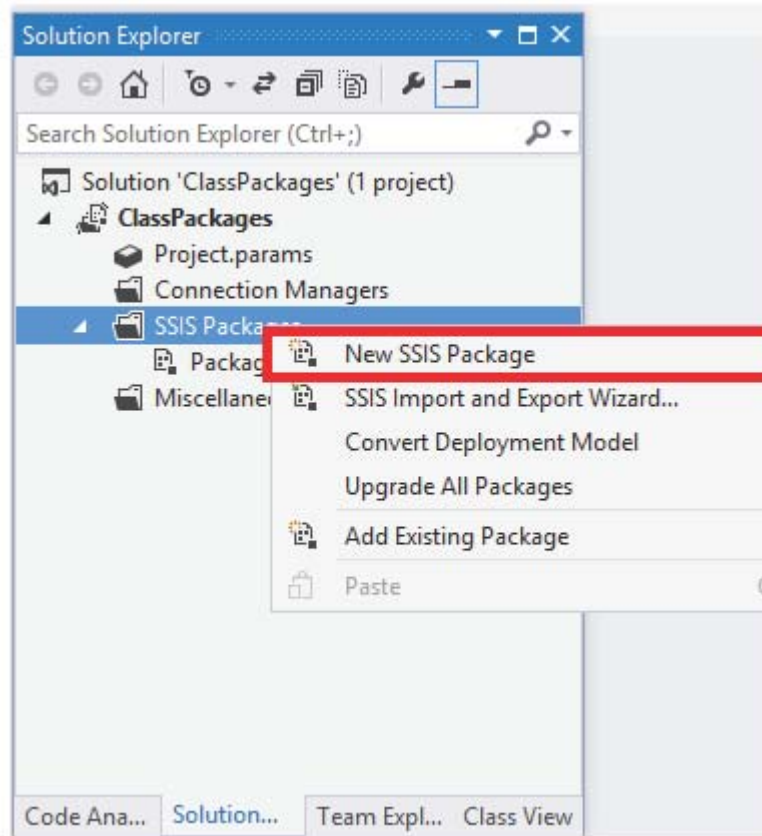
Hands-On Exercise 2.1: Adding Connection Managers to a Package (continued)

- b. In the Design window, close the default package that is created when you create a new project. It should be named `Package.dtsx`. Close it by clicking its **X** button. (We could use it to build our package, but we want you to know how to create one yourself).



Hands-On Exercise 2.1: Adding Connection Managers to a Package (continued)

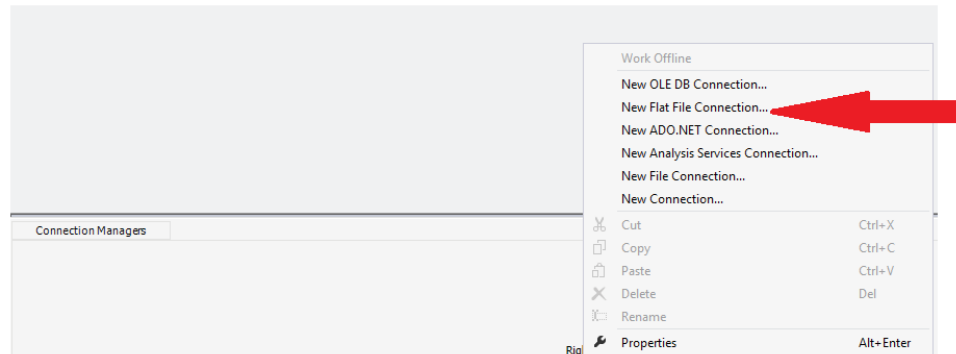
- c. □ In the Solution Explorer window, right-click the the **SSIS Packages** node and select **New SSIS Package**.



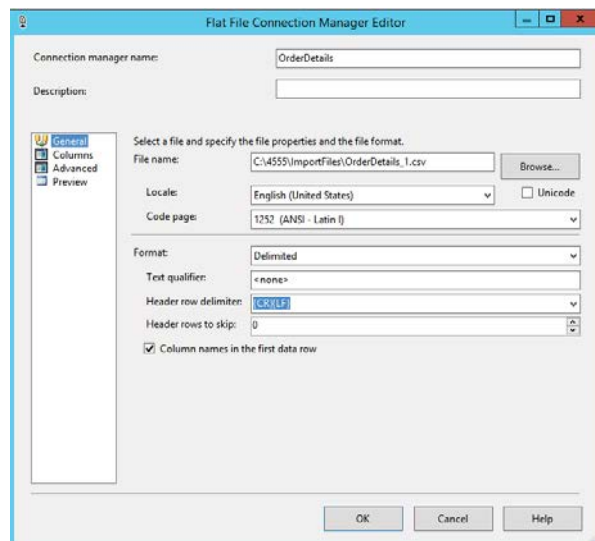
- d. □ In the Solution Explorer window, right-click the newly created package (it should be `Package1.dtsx`), and select **Rename**. Change the name to **ImportOrders.dtsx**

Hands-On Exercise 2.1: Adding Connection Managers to a Package (continued)

- e. □ With your `ImportOrders` package open in the designer, right-click the **Connection Managers** tray at the bottom of the screen and select **New Flat File Connection...**

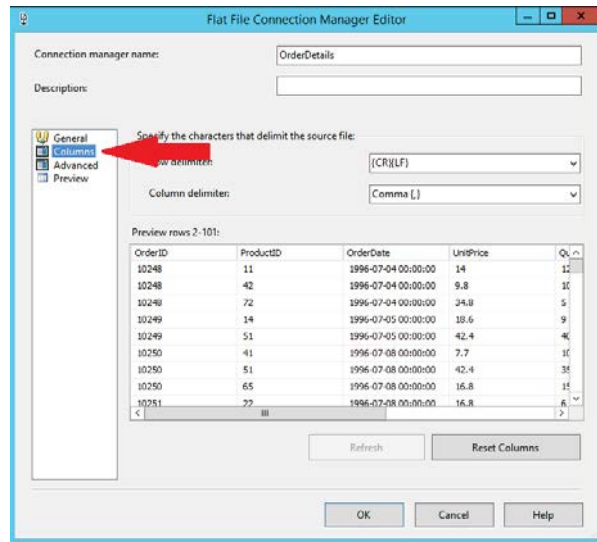


- f. □ In the Flat File Connection Manager Editor dialog box, in the Connection Manager Name box, enter `OrderDetails`. Click the **Browse** button and navigate to the folder `c:\4555\ImportFiles`. Change the file type to **CSV files (*.csv)**. Select the file `OrderDetails_1.csv` and click **Open**.
- g. □ Make sure the checkbox for **Column names in the first data row** is selected.



Hands-On Exercise 2.1: Adding Connection Managers to a Package (continued)

- h. Click the **Columns** page. Make sure the Column delimiter is set to **Comma{,}** and that the data preview makes sense.

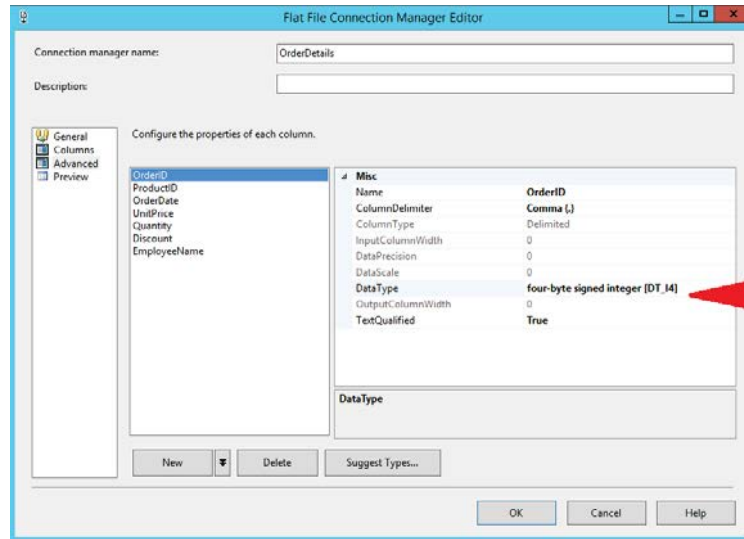


- i. On the Advanced page, click each column one by one and change its Data Type property as indicated in the table:

Column	Data Type
OrderID	four-byte signed integer [DT_I4]
ProductID	four-byte signed integer [DT_I4]
OrderDate	database timestamp [DT_DBTIMESTAMP]
UnitPrice	currency [CT_CY]
Quantity	two-byte signed integer [DT_I2]
Discount	float [DT_R4]
EmployeeName	Unicode string [DT_WSTR]



Hands-On Exercise 2.1: Adding Connection Managers to a Package (continued)



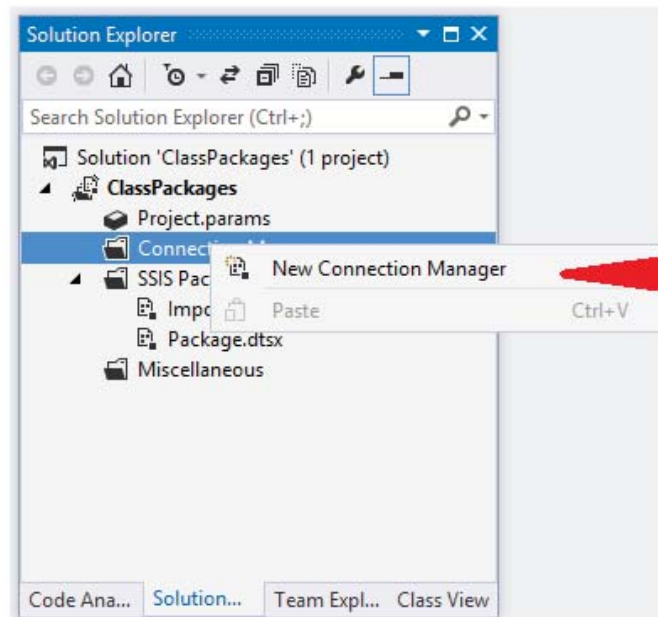
- j. Click **OK** when finished.
2. Create a *project-level*, not package-level, OLE DB connection manager to point to the `ClassDB` database on your local SQL Server instance. (Creating the connection manager at project level will save us from having to create one for each package we create).



Hands-On Exercise 2.1: Adding Connection Managers to a Package (continued)

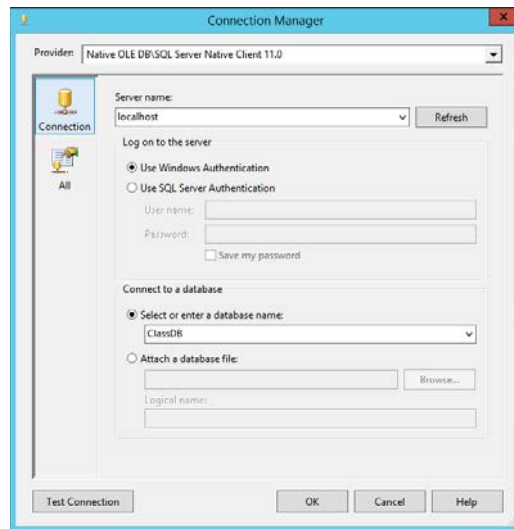
More detailed steps...

- a. In the Solution Explorer window on the right of the screen, right-click the **Connection Managers** node and select **New OLE DB Connection**.



Hands-On Exercise 2.1: Adding Connection Managers to a Package (continued)

- b. Click **New**. The **Native OLE DB/SQL Server Native Client 11.0** provider is already selected by default. In the "Server name" box, enter `localhost`. In the Select or enter database name drop-down, select **ClassDB**. Click **OK** twice to finish.



3. Create a package-level connection manager to connect to your SMTP server, which is located on your local machine. This will enable your package to connect to the server and send e-mail. Name the connection manager `My SMTP Server`

More detailed steps...

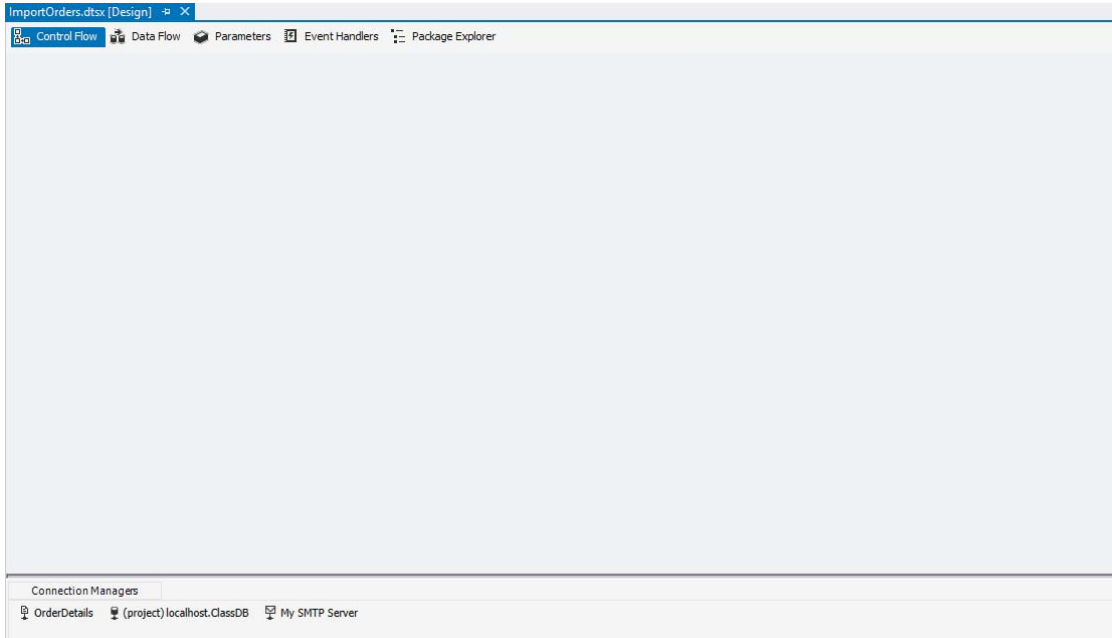
- a. Right-click the **Connection Managers** tray at the bottom of your screen and click **New Connection**. In the Add SSIS Connection Manager window, select **SMTP** and click the **Add** button. Click **New**. In the SMTP Server box, enter `localhost`. Select the **Use Windows Authentication** checkbox and click **OK** to finish.
- b. In the Connection Managers tray, your connection manager should have been created with the default name of `SMTP Connection Manager`. Right-click **SMTP Connection Manager** and select **Rename**. Change the name to `My SMTP Server` and click **OK**.



Hands-On Exercise 2.1: Adding Connection Managers to a Package (continued)



On completion, your package should look like the screenshot below. (The Solution Explorer window is not included as part of the screenshot).



Congratulations! You have created package- and project-level connection managers and completed the exercise.



This is the end of the exercise.





Objectives

In this exercise, you will use SSIS's "Execute SQL" task to create the `OrderDetails` table, into which you will import data from the `OrderDetails_1.csv` text file. Your SQL script will first check whether the table exists. If it does, delete the table.

1. Add an **Execute SQL** task to your `ImportOrders` package and configure it to use the OLE DB connection manager you created in the last exercise to run the SQL script in the file `CreateOrderDetails.sql`. Name the task `Create OrderDetails Table`

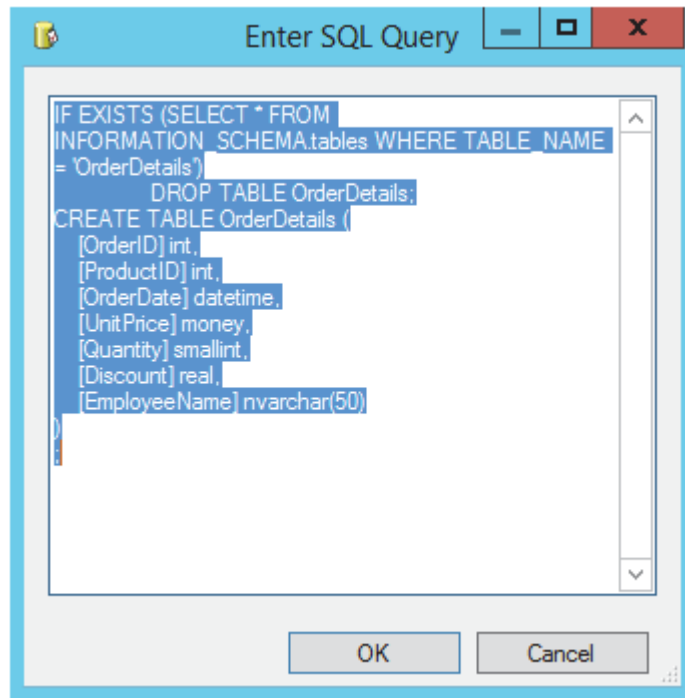
More detailed steps...

- a. With the **Control Flow** tab of the package active, drag the **Execute SQL** task from the toolbox onto the screen. Change the task's name to `Create OrderDetails Table`
- b. Double-click the task to open its task editor.
- c. On the General page of the task editor, make sure the `ConnectionType` property is **OLE DB**. Then, in the `Connection` property below it, select the connection `localhost.ClassDB` that you created.



Hands-On Exercise 2.2: Using the Execute SQL Task to Create a Database Table (continued)

- d. Click the **Browse** button and navigate to the folder `c:\4555\Exercises`. Select the file `CreateOrderDetails.sql` and click **Open**. The script in the file should be copied to the SQLStatementbox. Click the **SQLStatement** property and click the ellipsis button (...) to the right. Read the script and click **OK**.



Click **OK** to close the Execute SQL Task Editor window.

- e. Click **Save All** in the toolbar to save your project and package.
2. Run the package and use SQL Server Management Studio (SSMS) to check that the table has been created. Also check that the table has no data and leave SSMS open for the next exercise.

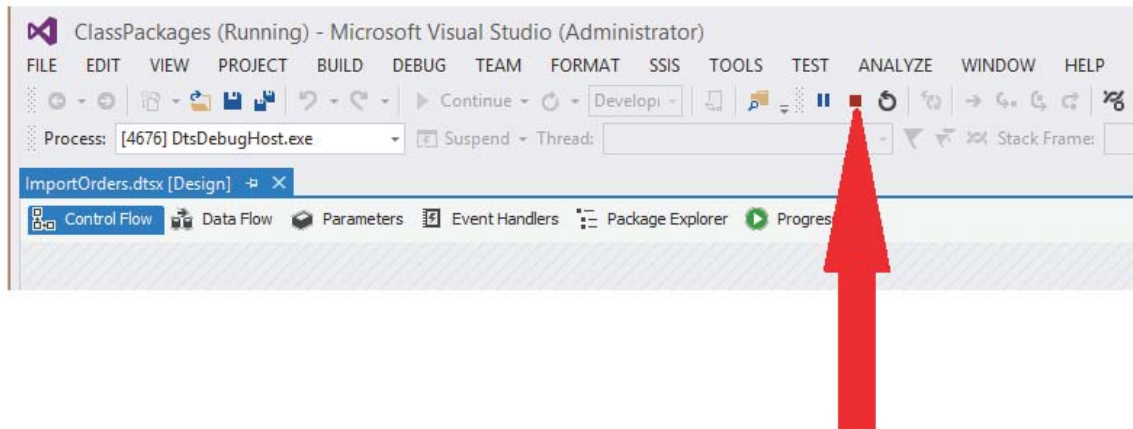
More detailed steps...

- a. With your `ImportOrders` package open in the designer, click the **Start** button in the toolbar. If it executes successfully, your task should display a green checkmark. If not, troubleshoot.



Hands-On Exercise 2.2: Using the Execute SQL Task to Create a Database Table (continued)

- b. SQL Server Management Studio should be open from an earlier exercise. If not, then open it using the instructions from Exercise 1.1. Switch to SQL Server Management Studio. With the `ClassDB` database node expanded, right-click the **Tables** node and select **Refresh**. You should see that the `OrderDetails` table has been created.
 - c. Right-click the `OrderDetails` table in the Object Explorer pane and click **Select Top 1000 Rows**. You should see that the table has no data.
 - d. Leave SQL Server Management Studio running for the next exercise.
3. Switch to SSDT and stop the package that is running by clicking the **Stop Debugging** button, as shown in the screenshot.



Congratulations! You have used SSIS's "Execute SQL" task to create a table on a SQL instance and completed the exercise.



This is the end of the exercise.



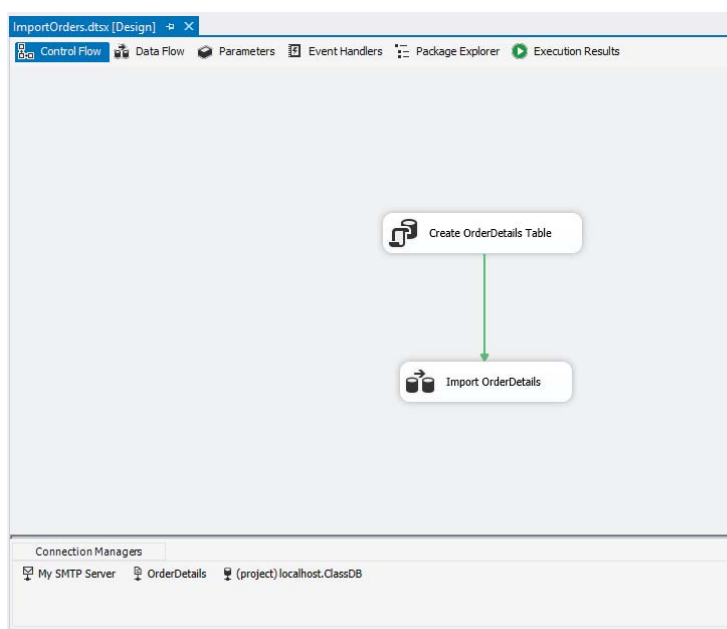


Hands-On Exercise 2.3: Using the Data Flow Task to Import from a Text File

Objectives

In this exercise, you will use the Data Flow task to import data from the `OrderDetails_1.csv` file to the `OrderDetails` table on your SQL Server instance. In previous exercises, you created the connection managers to the text file and to the SQL Server database. You also configured a task to create the table in your database.

The completed package will look like the screenshot below:



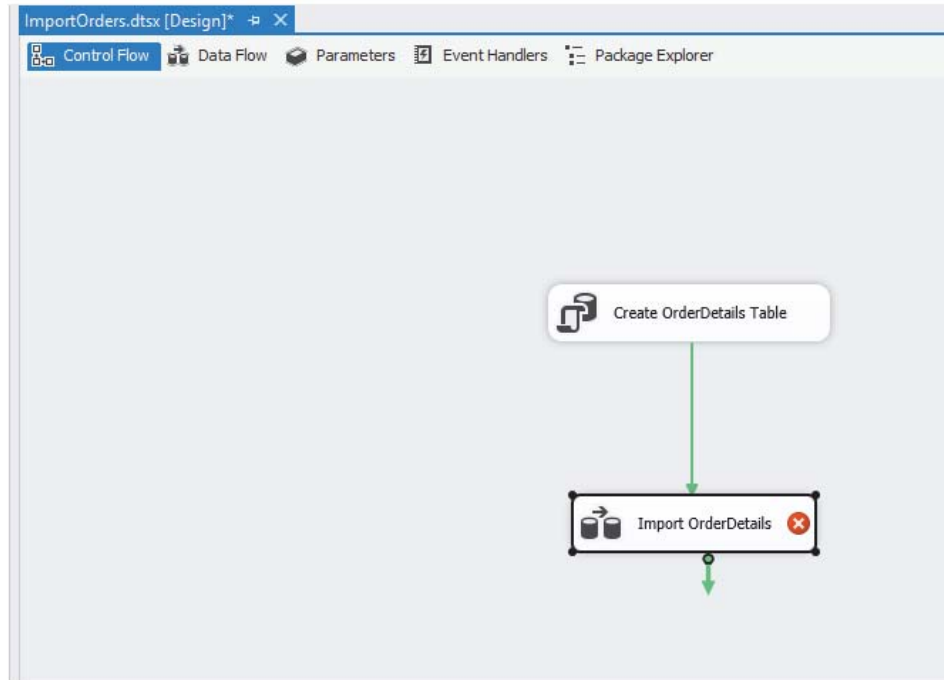
1. □ Add a Data Flow task instance to your package and name it **Import OrderDetails**. Configure it to execute after the Create OrderDetails Table task. On the Data Flow tab of the Data Flow task, configure a Flat File Source component to take data from the OrderDetails connection manager, and use a SQL Server Destination component to write the data to the SQL Server OrderDetails table that is in the ClassDB database. Be sure to change the name of the SQL Server Destination component to **OrderDetails Table**



Hands-On Exercise 2.3: Using the Data Flow Task to Import from a Text File (continued)

More detailed steps...

- a. Drag a **Data Flow** task from the Toolbox to the SSIS Designer, to the right of or below the Create OrderDetails Table task. Change the name of the task to **Import OrderDetails**

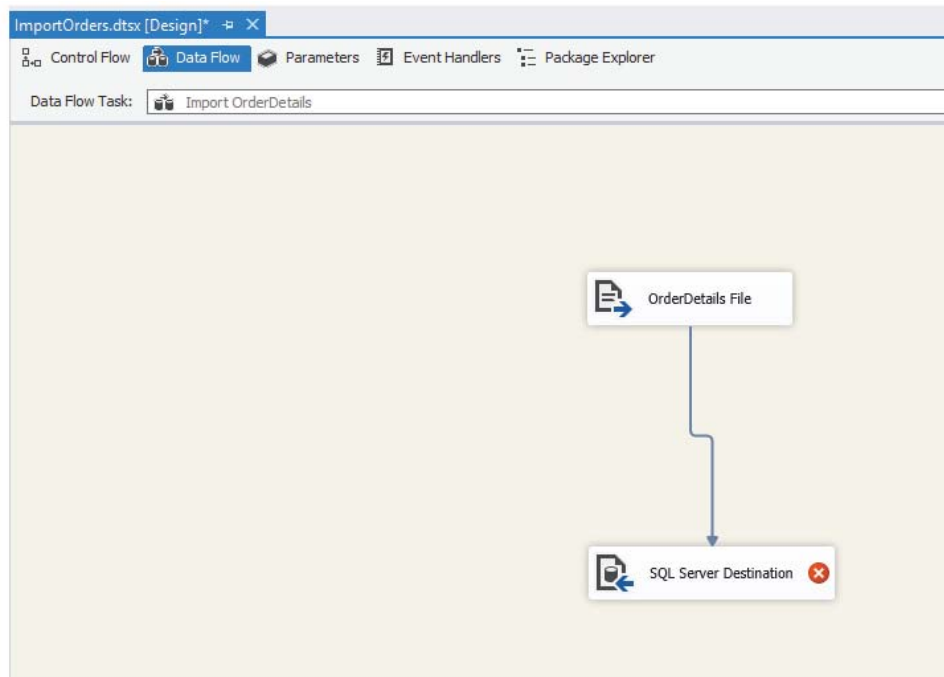


- b. Click the **Create OrderDetails Table** task. Drag the green arrow below it onto the Import OrderDetails task to create a success precedence constraint.
- c. Double-click the **Import OrderDetails** task to edit it on the Data Flow tab.
- d. Drag the **Flat File Source** component from the Other Sources section of the Toolbox onto the Designer. Rename it **OrderDetails File**
- e. Double-click the **OrderDetails File** component you have just renamed to open its Flat File Source Editor dialog box.
- f. Click **Columns** to open the Columns page, view the list of output columns of the component, and click **OK**.



**Hands-On Exercise 2.3:
Using the Data Flow Task to Import from a Text File
(continued)**

- g. Drag the **SQL Server Destination** component from the Toolbox to the SSIS Designer. (The SQL Server Destination component is near the bottom of the Toolbox, in the Other Destinations category.) Change its name to **OrderDetails Table**
- h. Click the **OrderDetails File** component. Drag the blue connector from the OrderDetails File component and drop it onto the OrderDetails Table component.



- i. Right-click the **OrderDetails Table** component and click **Edit** to open its editor. Make sure the connection manager is set to **localhost.ClassDB**.
- j. From the "Use a table or view" drop-down, select **dbo.OrderDetails**. Click **Mappings** to open the Mappings page of the dialog box.
- k. Make sure the input columns from the previous component map to columns of the same name in the destination.
- l. Click **OK**.



Hands-On Exercise 2.3: Using the Data Flow Task to Import from a Text File (continued)

- m. Click the **Save All** icon in the toolbar and press <F5> to execute the package.

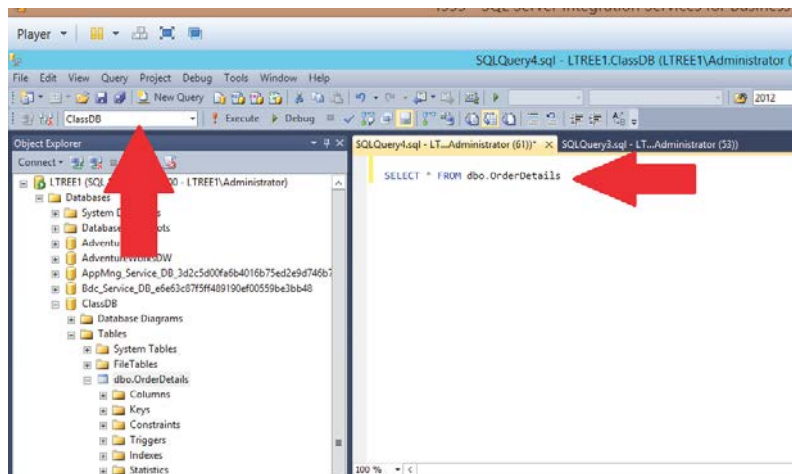


How many rows were migrated?

- n. Select the **Progress** tab and note the time taken by the Import OrderDetails task. Record it here (time to execute this task only, not the whole package):
-

- o. Switch to SSMS. Click the **New Query** button and verify that the import was successful by running either of the queries below. Be sure to change the database context in the toolbar to **ClassDB**

```
SELECT count(*) FROM dbo.OrderDetails  
OR  
SELECT * FROM dbo.OrderDetails
```



Does the result agree with what you found at step m?



Congratulations! You have migrated data from a text file into a database table using the Data Flow task!



**Hands-On Exercise 2.3:
Using the Data Flow Task to Import from a Text File
(continued)**

2. Switch back to SSDT and stop debugging the package.



Congratulations! You have completed the exercise.



This is the end of the exercise.





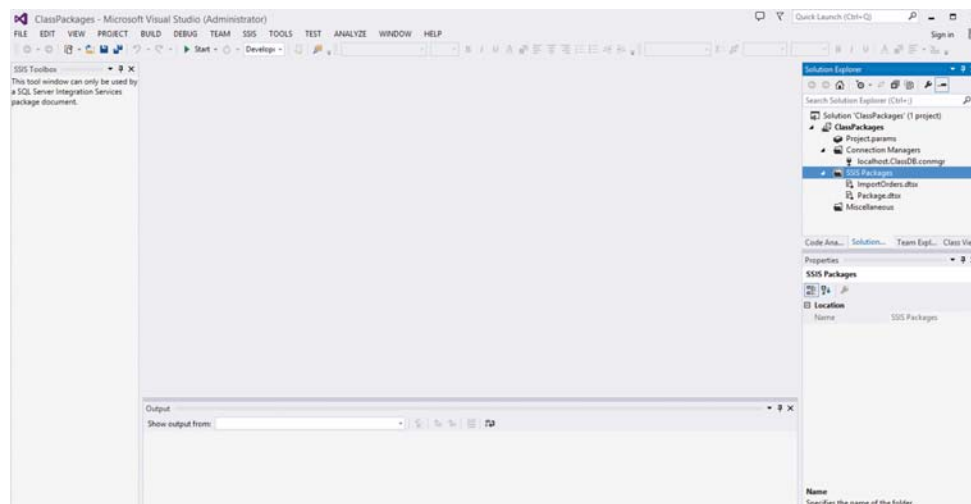
Objectives

In this exercise, you will transform data using the Character Map transformation. With this transformation, you will capitalize the data extracted from the text file before writing it to the `OrderDetails` table. First, you will create a copy of your existing `ImportOrders` package.

1. Make sure your package is closed from the designer so that its name is visible only in the Solution Explorer window. Copy and paste the package, then rename the copy `TransformAndImportOrders.dtsx`

More detailed steps...

- a. Make sure your package is closed in the designer. If not, close it by clicking the **X** at the top right of the package window. After closing it, you should only see its name in the Solution Explorer window, as in the screenshot below.

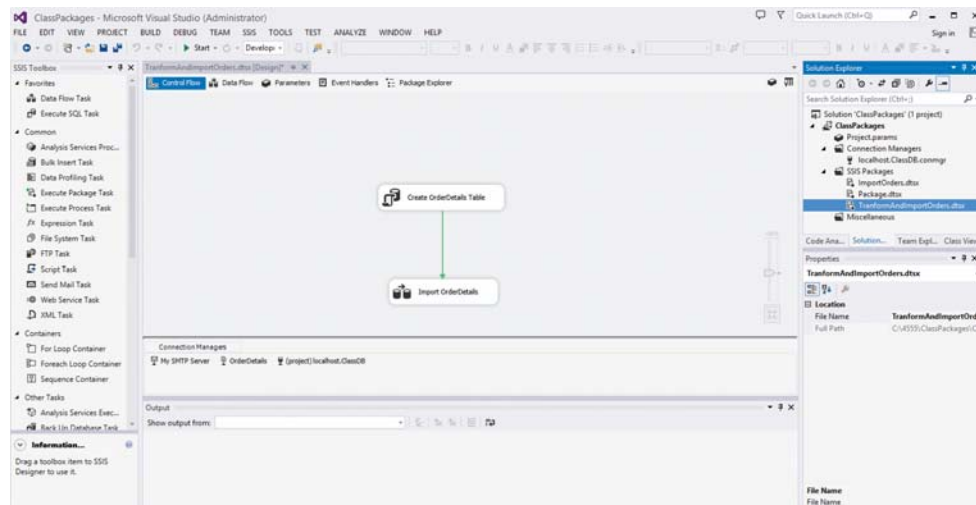


- b. In the Solution Explorer window, right-click the package name `ImportOrders.dtsx` and click **Copy**.
- c. In the Solution Explorer window, right-click the **SSIS Packages** node and click **Paste** to create a copy of the `ImportOrders` package.



Hands-On Exercise 2.4: Transforming Data (continued)

- d. In the Solution Explorer window, right-click the copied package. It should be named `ImportOrders_1.dtsx`. Click **Rename**. Change its name to `TransformAndImportOrders.dtsx` and press `<Enter>`.



2. On the Data Flow tab of the new `TransformAndImportOrders` package, delete the existing data mapping between the `OrderDetails` Source and `OrderDetails` table components, then place a **Character Map** transformation component between the two. Be sure to name the component `Capitalize EmployeeName` and configure it to capitalize the `EmployeeName` field before writing it to the `OrderDetails` table in SQL Server.

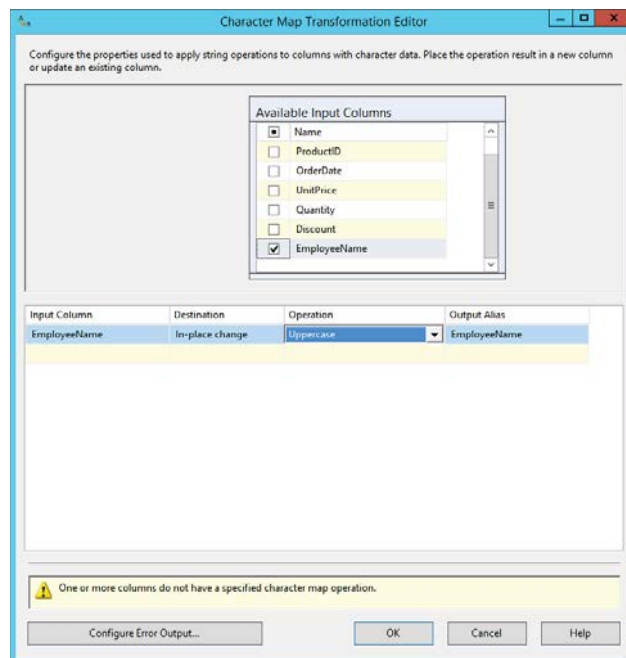
More detailed steps...

- a. Your `TransformAndImportOrders` package should have opened in the designer when you renamed it. Click the **Data Flow** tab to open the Data Flow task for internal configuration. Click and delete the line between the `OrderDetails` Source and `OrderDetails` Table components.
- b. From the toolbox, drag the **Character Map** transformation and place it between the `OrderDetails` Source and `OrderDetails` Table components. Change the name of the newly added component to `Capitalize EmployeeName`



Hands-On Exercise 2.4: Transforming Data (continued)

- c. Click the **OrderDetails Source** component, then drag the blue line from the OrderDetails Source component and drop it onto the Capitalize EmployeeName component, so the two components are connected.
- d. Double-click the **Capitalize EmployeeName** component to open it for configuration. In the Character Map Transformation Editor dialog box, in the Available Input Columns list pane, select the checkbox in front of **EmployeeName** to make the **EmployeeName** field appear in the grid below.
- e. In the grid, in the Destination column, note that SSIS is proposing to create a new column. You want to replace the existing column instead, so click **New Column**, and in the drop-down, click **In-place change**. Click inside the Operation column. In the drop-down, select the checkbox in front of **Uppercase**.



- f. Click **OK**.

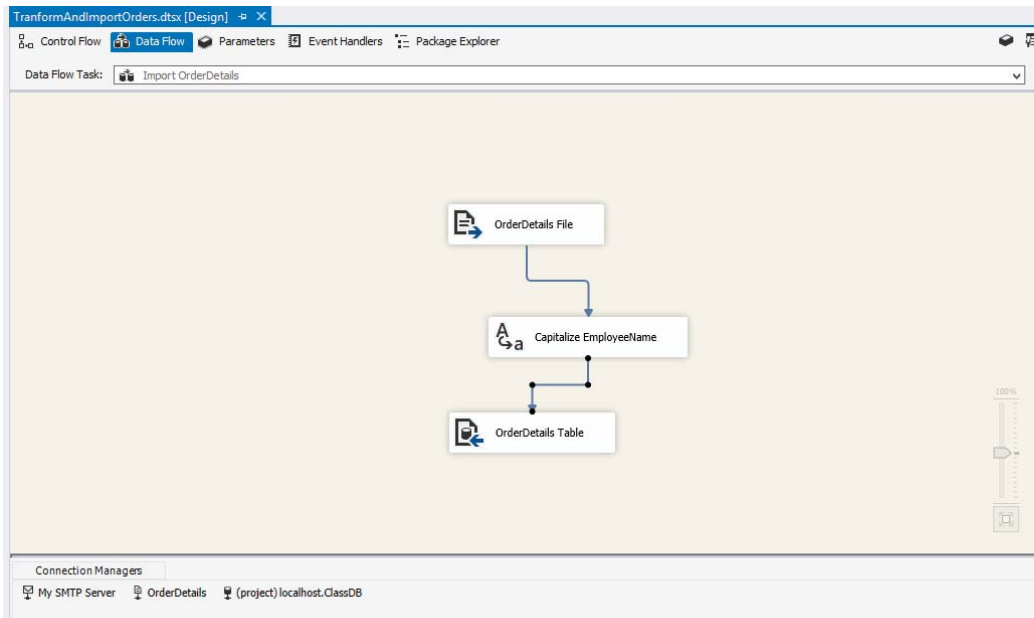


Hands-On Exercise 2.4: Transforming Data (continued)

- g. Click the **Capitalize EmployeeName** component if it does not have the focus. Drag the blue line from the Capitalize EmployeeName to the OrderDetails Table component. The latter component automatically maps the fields between the two components because it remembers the field mapping that you originally configured.



On completing the major step, the Data Flow tab of your package should look like the screenshot below.



3. Execute the package and check the results using SQL Server Management Studio.

More detailed steps...

- a. Click the **Save All** icon in the toolbar. Execute the package by clicking the **Start** button in the toolbar.
- b. If your package successfully executes, switch to SQL Server Management Studio, which should still be open with your query from the last exercise. Execute the query `SELECT * FROM OrderDetails`. The imported data should have the EmployeeName column data capitalized.



- c. Leave SQL Server Management Studio open, switch back to SQL Server Data Tools, and stop debugging the package. Close the package from the designer by clicking its **X** button.



Congratulations! You have transformed data before importing it and completed the exercise.



This is the end of the exercise.





Objectives

In this exercise, you will use the **Send Mail** task to send an e-mail on successfully importing from the `OrderDetails.csv` file.

1. Modify the `ImportOrders` package by adding and configuring a **Send Mail** task that is to execute after the data import into the `OrderDetails` table successfully completes. Name the task `Notify Administrator`. Send the e-mail from yourself to yourself; i.e., `Administrator@ssiscourse.com`. As there is no mail server in the class, you will open the e-mail directly by double-clicking it in the folder `c:\InetPub\Mailroot\Drop`. Add a simple message to indicate that that the package successfully executed.

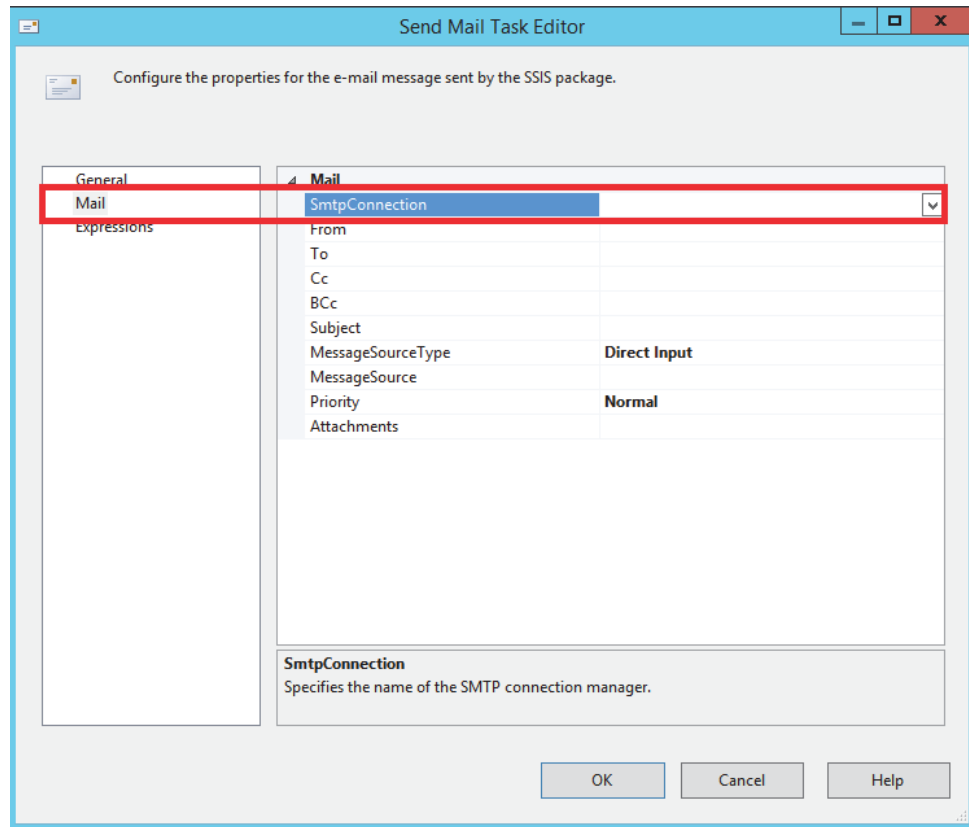
More detailed steps...

- a. Double-click the `ImportOrders` package to open it in the SSIS designer.
- b. From the Toolbox, drag the **Send Mail** task and drop it on the designer. Rename the task `Notify Administrator`
- c. On the Control Flow tab, click the **Import OrderDetails** task. You should see a protruding green arrow.
- d. Drag the green arrow from the `Import OrderDetails` task to the `Notify Administrator` task you just created to indicate that you want the `Notify Administrator` task to execute after the `Import OrderDetails` task successfully executes.



Hands-On Exercise 3.1: Adding a Send Mail Task to the Package (continued)

- e. Double-click the **Send Mail** task to open its editor. Click the **Mail** tab.



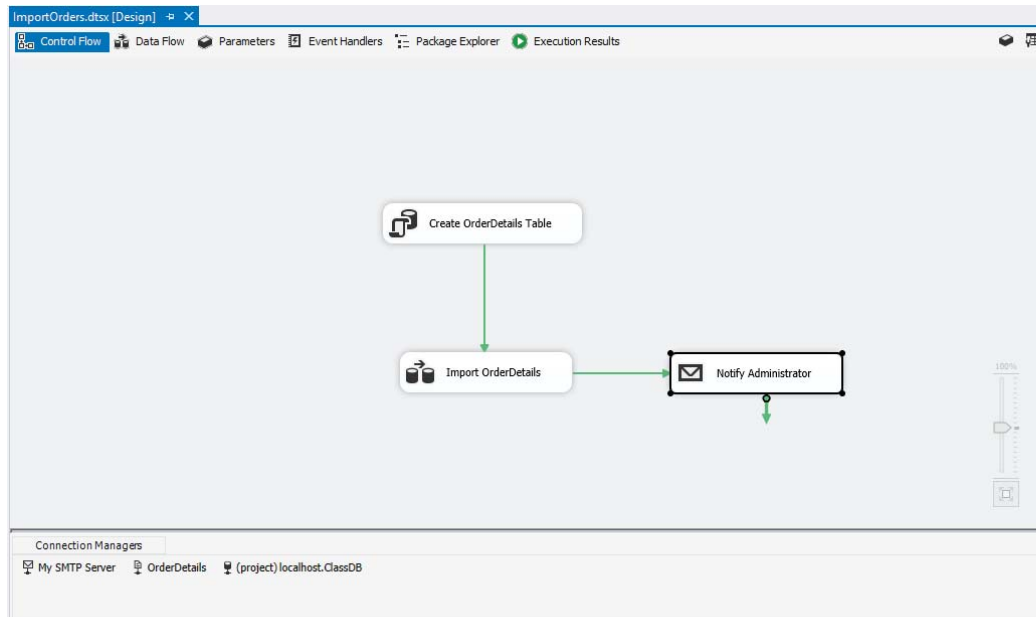
- f. Click the **SmtpConnection** drop-down and select **My SMTP Server**. You configured this connection manager in an earlier exercise.
- g. In the From box, enter your e-mail address as below:
Administrator@ssiscourse.com
- h. Enter the same e-mail address in the To box.
- i. In the Subject box, enter **Mail from SSIS**
- j. In the MessageSource box, enter some text. Click **OK** to close.
- k. Save and execute the package. Stop debugging the package.



Hands-On Exercise 3.1: Adding a Send Mail Task to the Package (continued)



When you finish, your package's Control Flow tab should look like the screenshot below.



2. Switch to the folder `c:\inetpub\mailroot\Drop` and double-click the mail to open it in Outlook. Close Outlook when you're finished viewing the e-mail.

More detailed steps...

- a. Switch to Windows Explorer and switch to the folder `C:\inetpub\mailroot\Drop`.



Your e-mail should be there.

- b. Double-click the e-mail file to view it in Outlook.
- c. Close Outlook.



Hands-On Exercise 3.1: Adding a Send Mail Task to the Package (continued)



Congratulations! You have completed the exercise.



This is the end of the exercise.



Objectives

In this exercise, you will create an SSIS variable. You will use an Expression task to assign a value to the variable and the Script task to display the content of this and a system variable.

1. Create a new package and name it `scriptingSSIS.dtsx`

More detailed steps...

- a. In the Solution Explorer window, right-click the **SSIS Packages** node and select **New SSIS Package**.
 - b. SSIS creates a new package with a default name of `Package 1.dtsx`. Right-click the newly created package and select **Rename**. Change its name to `scriptingSSIS.dtsx`
2. Create a package-level variable named `MyLocation`. The data type should be `string`; it do not assign it a value yet.

More detailed steps...

- a. Display the Variables window by right-clicking a blank part of the Control Flow tab and selecting **Variables** from the context menu.
 - b. With the focus on the Control Flow tab, create a variable in the Variables window by clicking the first icon in the Variables window toolbar. Call the variable `MyLocation`. It should be `String` type, but you should not give it a value.
3. Add an Expression task to your package and name it `Specify Location`. Configure it so that it assigns the name of your location to the `MyLocation` variable you just created.

More detailed steps...

- a. Drag an **Expression** task from the Toolbox to the Control Flow tab of the SSIS designer. Name the task `Specify Location`



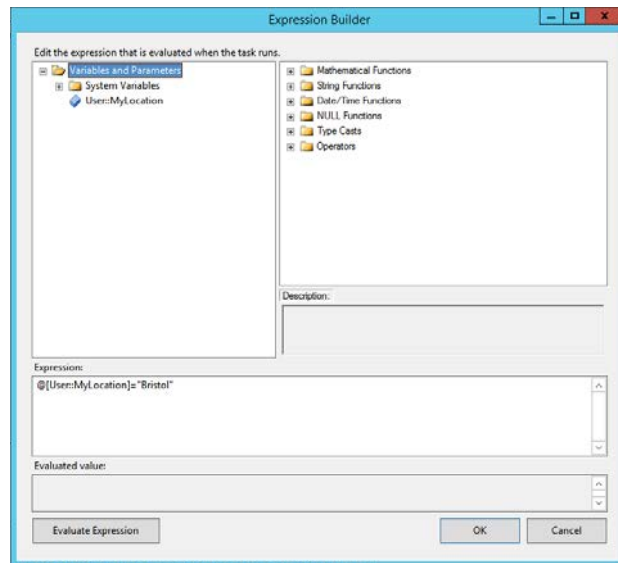
Hands-On Exercise 3.2: Creating an SSIS Variable and Using the Script Task (continued)

- b. Double-click the task to edit it. In the Expression Builder window, expand the **Variables and Parameters** node. You should see the variable you created, `User::MyLocation`, listed there. Drag `User::MyLocation` to the Expression area near the bottom of the window. Complete the expression by entering the following to the right of `@[User::MyLocation]`:

```
= "<Your place name>"
```

replacing `<Your place name>` with the name of the place where you are located; for example:

```
@[User::MyLocation] = "Bristol"
```



- c. Click the **Evaluate Expression** button and make sure it returns the name of the location you specified. Click **OK**.
4. Add to your package a Script task named `Display Greeting`, and configure it to execute after the `Specify Location` task has executed. Edit the task and enable script access to the variables `User::MyLocation` and `System::UserName`. Finally, import the simple code to display a greeting from the file `DisplayGreeting.cs`, which is located in the folder `c:\4555\Exercises`.



Hands-On Exercise 3.2: Creating an SSIS Variable and Using the Script Task (continued)

More detailed steps...

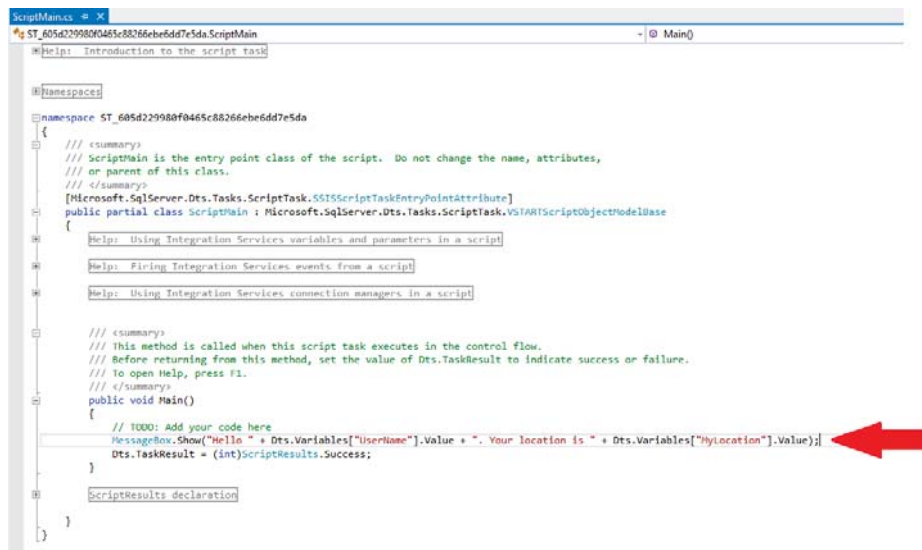
- a. Add a Script task from the Toolbox to the Control Flow tab of the SSIS designer. Name the task **Display Greeting**. Drag the green arrow from the Specify Location task to the Display Greeting task so that the Display Greeting task executes after the Specify Location task.
- b. Double-click the task to edit it. In the `ReadOnlyVariables` property, click the ellipsis (...) to the right of the property and select the checkboxes corresponding to the following two variables:
`User::MyLocation` and `System::UserName`.
- c. Click the **Edit Script** button. Note the template script that is provided. You will need to enter your code between the curly braces below `public void Main()` (see next step).



Hands-On Exercise 3.2: Creating an SSIS Variable and Using the Script Task (continued)

- d. Click between the braces below the line `public void Main()`, and between the comment that reads: `//TODO: Add your code here` and the line that reads `Dts.TaskResult = (int)ScriptResults.Success;`. From the **Edit** menu, select **Insert File As Text**. Navigate to the file `C:\4555\Exercises\DisplayGreeting.cs`. Highlight it and click **Open**. If you prefer to type it in, the code reads:

```
MessageBox.Show("Hello " +  
Dts.Variables["UserName"].Value + ". Your location  
is " + Dts.Variables["MyLocation"].Value);
```



```
ScriptMain.cs - x  
ST_605d229980f0465c88266e8ebdd7e5da.ScriptMain - | Main() |  
# Help: Introduction to the script task  
  
[Namespace]  
[Namespace ST_605d229980f0465c88266e8ebdd7e5da  
{  
    /// <summary>  
    /// ScriptMain is the entry point class of the script. Do not change the name, attributes,  
    /// or parent of this class.  
    /// </summary>  
    [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]  
    public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase  
    {  
        # Help: Using Integration Services variables and parameters in a script  
        # Help: Firing Integration Services events from a script  
        # Help: Using Integration Services connection managers in a script  
  
        /// <summary>  
        /// This method is called when this script task executes in the control flow.  
        /// Before returning from this method, set the value of Dts.TaskResult to indicate success or failure.  
        /// To open Help, press F1.  
        /// </summary>  
        public void Main()  
        {  
            // TODO: Add your code here  
            MessageBox.Show("Hello " + Dts.Variables["UserName"].Value + ". Your location is " + Dts.Variables["MyLocation"].Value);  
            Dts.TaskResult = (int)ScriptResults.Success;  
        }  
        # ScriptResults declaration  
    }  
}
```

- e. From the **File** menu, select **Exit**, then click **OK** to close the task editor window.
- f. Run the package and make sure the information gets displayed correctly. If not, go back and edit the script until it works correctly.
5. Stop debugging the package. Save the package and close it.



Hands-On Exercise 3.2: Creating an SSIS Variable and Using the Script Task (continued)



Congratulations! You created an SSIS variable, assigned the variable a value using the Expression task, and displayed the variable value from a script inside the Script task, and you completed the exercise.



This is the end of the exercise.

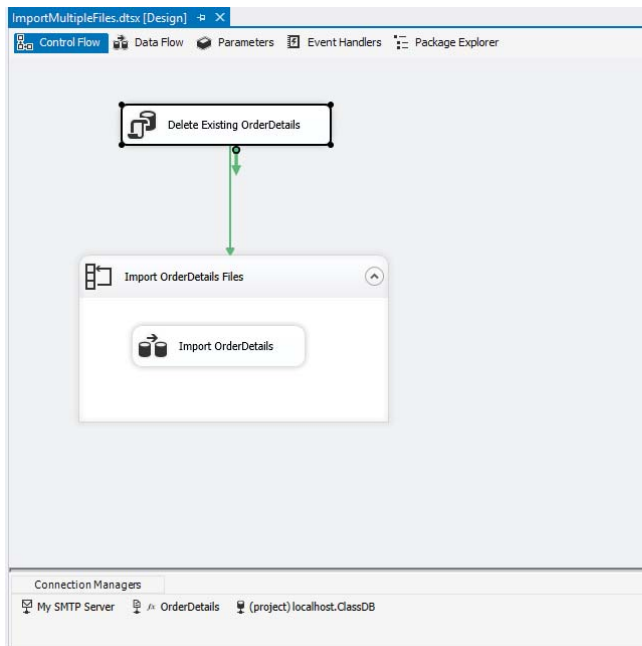




Hands-On Exercise 4.1: Using the Foreach Loop Container to Import Multiple Files

Objectives

In this exercise, you will develop a package that imports multiple files using a single Data Flow task. You will use a Foreach Loop container to get the Data Flow import task to be executed multiple times, once for each file.



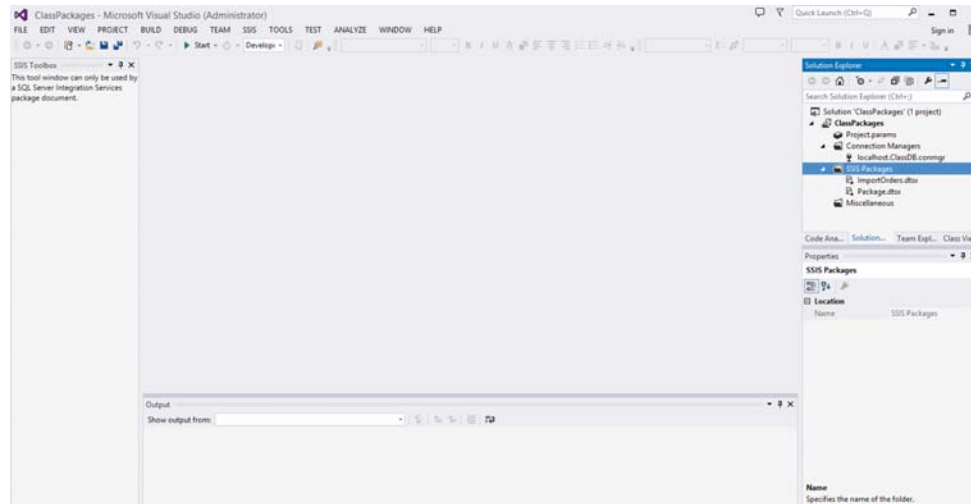
1. Make sure the package `TransformAndImportOrders.dtsx` is closed from the designer so that its name is visible only in the Solution Explorer window. Copy and paste the package, then rename the copy `ImportMultipleFiles.dtsx`.



Hands-On Exercise 4.1: Using the Foreach Loop Container to Import Multiple Files (continued)

More detailed steps...

- a. Make sure your package is closed in the designer. If not, close it by clicking the X at the top right of the package window. After closing it, you should only see its name in the Solution Explorer window, as in the following screenshot.



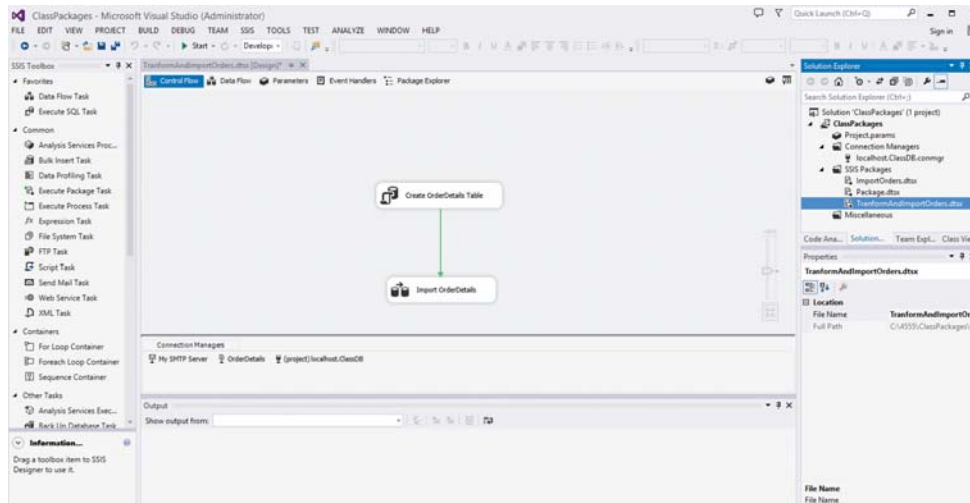
In the Solution Explorer window, right-click the package name `TranformAndImportOrders.dtsx` and click **Copy**.

- b. In the Solution Explorer window, right-click the **SSIS Packages** node and click **Paste** to create a copy of the `TranformAndImportOrders` package.



Hands-On Exercise 4.1: Using the Foreach Loop Container to Import Multiple Files (continued)

- c. In the Solution Explorer window, right-click the copied package. It should be named `TransformAndImportOrders_1.dtsx`. Click **Rename**. Change its name to `ImportMultipleFiles.dtsx` and press `<Enter>`.



2. On the Control Flow tab of your `ImportMultipleFiles` package, delete the precedent constraint between the `Create OrderDetails Table` task and the `Import OrderDetails` task. Rename the `Create OrderDetails Table` task to **Delete Existing OrderDetails**, and change the SQL query within the task to:
`DELETE FROM OrderDetails`

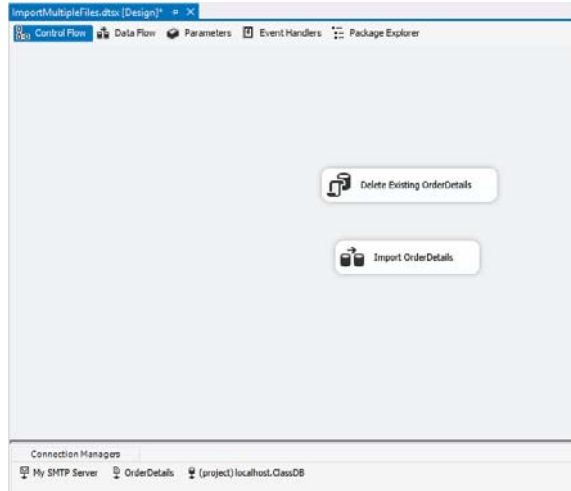
More detailed steps...

- a. With the `ImportMultipleFiles` package open in the designer, click the **Control Flow** tab. Delete the green line (precedent constraint) between the `Create OrderDetails Table` and the `Import OrderDetails` tasks.
- b. Right-click the `Create OrderDetails Table` task and choose **Rename**. Change the name to `Delete Existing OrderDetails`.



Hands-On Exercise 4.1: Using the Foreach Loop Container to Import Multiple Files (continued)

- c. Double-click the task Delete Existing OrderDetails to edit it.



- d. In the Execute SQL Task Editor dialog box, click the `SQLStatement` property. Click the ellipsis to the right of the property and delete the existing SQL statement there. Enter the following code in its place:
- ```
DELETE FROM OrderDetails
```
- e.  Click **OK** twice.
3.  Create a package-level variable named `strFileName`. The data type should be `String`. Do not assign a value. You will need it to store the name of each file being imported.

More detailed steps...

- a.  Display the Variables window by right-clicking a blank part of the Control Flow tab and selecting **Variables** from the context menu.
- b.  With the focus on the Control Flow tab, create a variable in the Variables window by clicking the first icon in the Variables window toolbar. Call the variable `strFileName`. It should be `String` type, but you should not give it a value.

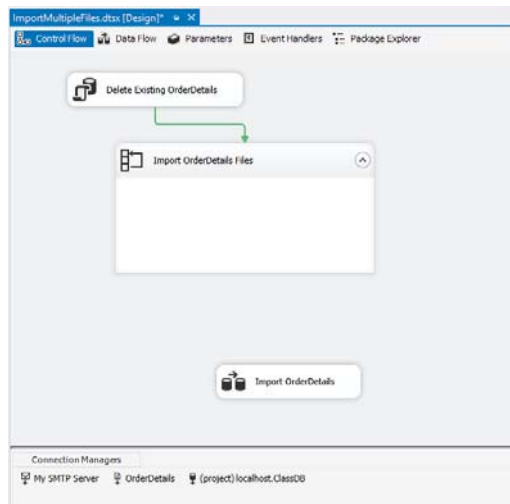


## Hands-On Exercise 4.1: Using the Foreach Loop Container to Import Multiple Files (continued)

4.  Drag a Foreach Loop container from the toolbox to the package designer and configure it to execute after the Delete Existing OrderDetails task. Configure the container to loop around all files with the wildcard pattern `OrderDetails_*.csv` that are in the folder `c:\4555\ImportFiles` and, for each file iteration, you will save the filepath to the variable `strFileName` that you already created.

More detailed steps...

- a.  Drag a **Foreach Loop** container from the Toolbox to the Control Flow tab. Rename it **Import OrderDetails Files**. Create a precedence constraint so that the Import OrderDetails Files container executes after the Delete Existing OrderDetails task succeeds.

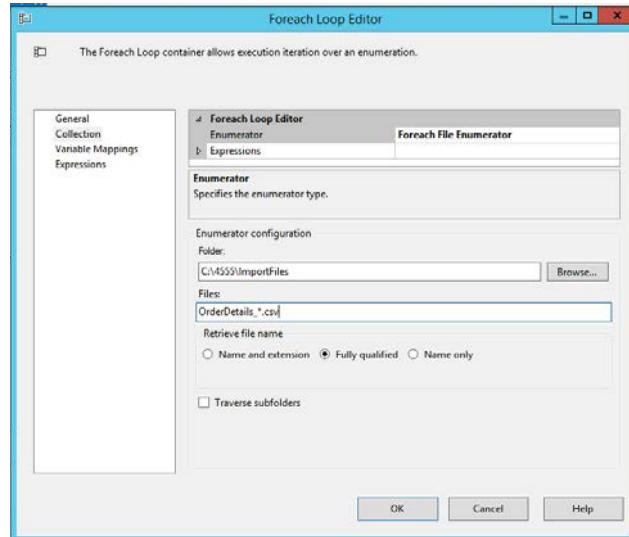


- b.  Double-click the **Foreach Loop** container to open its editor dialog. Click the **Collection** page. In the Enumerator drop-down, select **Foreach File Enumerator**.

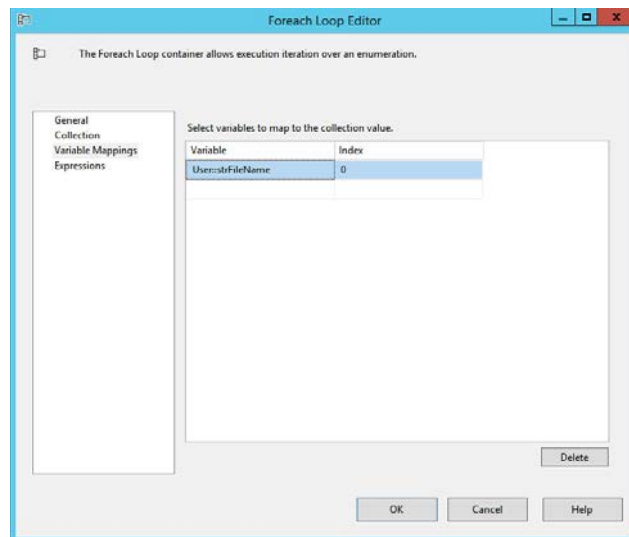


## Hands-On Exercise 4.1: Using the Foreach Loop Container to Import Multiple Files (continued)

Click **Browse** and select the folder `C:\4555\ImportFiles`. Set the Files text box to read `OrderDetails_*.csv`



- c.  Click the **Variable Mappings** page. In the Variable column, select `strFileName`, the variable you created near the beginning of this exercise. Click **OK**.



**Hands-On Exercise 4.1:**  
**Using the Foreach Loop Container to Import Multiple Files**  
**(continued)**

---

5.  Drag the Import OrderDetails task inside the Foreach Loop container so that it executes for each file identified by the container. Change the `ConnectionString` property of the OrderDetails connection manager so that it is a variable containing the name of the current file in the looping. (Locate the `ConnectionString` property through the `Expressions` property of the connection manager). The name of the current file is stored in the variable `strFileName` that you created for that purpose.

More detailed steps...

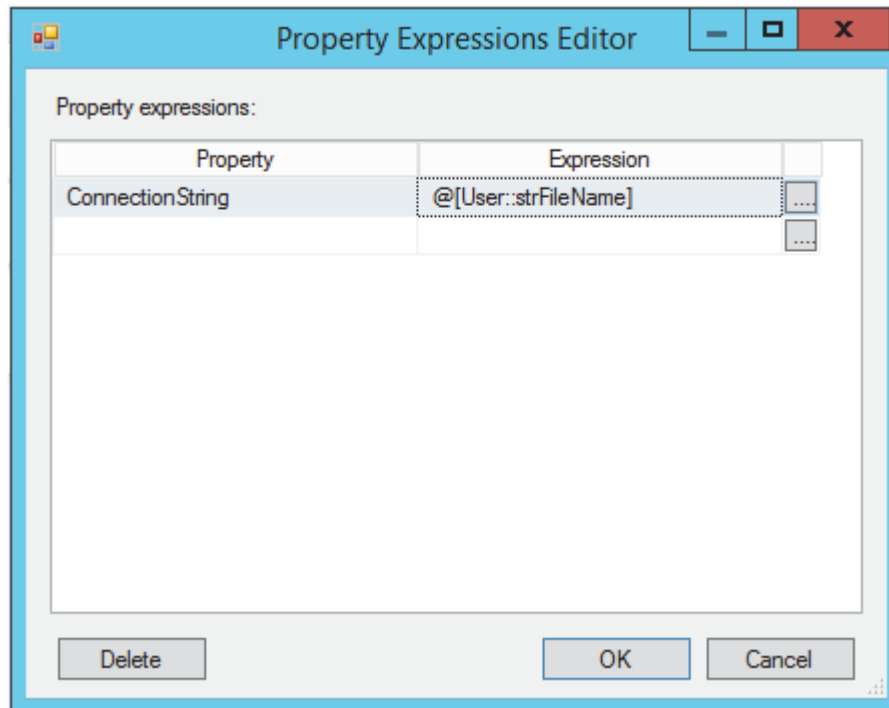
- a.  Drag the **Import OrderDetails** task and drop it inside the Import OrderDetails Files container. (This causes it to be executed for each file the container returns).
- b.  Right-click the **OrderDetails** connection manager and select **Properties**.
- c.  In the Properties sheet, locate the `Expressions` property. Click the ellipsis to the right of the property.
- d.  From the Property drop-down, select **ConnectionString**.
- e.  Click the ellipsis in the Expression column.
- f.  In the Expression Builder window, expand the Variables and Parameters folder and drag the variable `User::strFileName` to the Expression box.



## Hands-On Exercise 4.1: Using the Foreach Loop Container to Import Multiple Files (continued)

---

- g.  Click **OK** to close the Expression Builder, and click **OK** again to close the Expressions dialog box.



- h.  Save the package and execute it.
- i.  Use SQL Server Management Studio (SSMS) to verify that a total of 2,155 rows have been imported from the text files.



**Congratulations! You have created a package that imports from any number of files of the same structure, and you have completed the most complex exercise in the course.**



***This is the end of the exercise.***





## Objectives

In this exercise, you will use a wizard to import from an Access database. The wizard is good for simple imports where you don't need to perform transformations, there are no additional tasks required, and no looping is required.

1.  Run the SSIS Import and Export Wizard, accessible from the Packages folder of the SSDT tool, to import the table `Ledger2000` from the Microsoft Access database file `c:\4555\Exercises\CompanyInformation` to a SQL Server table in the course database `ClassDB`. Rename the package created by the wizard from the default `Package 1.dtsx` to `ImportLedgerInfo.dtsx`. Run the package and check that the data has been imported successfully into SQL Server.

More detailed steps...

- a.  Right-click the `SSIS Packages` folder in the Solution Explorer window and select **SSIS Import and Export Wizard**. At the Welcome screen, click **Next**.



*Hint...*

If the Solution Explorer window is not visible, select the Solution Explorer command from the View menu to make it visible.

- b.  On the "Choose a Data Source" screen, click the **Data source** drop-down and select **Microsoft Access (Microsoft Access Jet Database Engine)**. To the right of the File Name text box, browse to and select the file `C:\4555\Exercises\CompanyInformation.mdb`. Click **OK**, then click **Next**.
- c.  In the "Choose a Destination" screen, select **SQL Server Native Client 11.0** and type `localhost` in the "Server name" text box. Set the Authentication to **Use Windows Authentication**. In the Database drop-down, select **ClassDB**. Click **Next**.
- d.  In the "Specify Table Copy or Query" screen, leave the default selected; i.e., "Copy data from one or more tables or views." Click **Next**.
- e.  Select the checkbox next to `Ledger2000`.



## Hands-On Exercise 4.2: Using the SSIS Wizard to Import From a Microsoft Access Database (continued)

---



*Which table name does the wizard propose to create at the destination?*

---

- f.  Click **Next** and **Finish**.
- g.  The last screen should show that the wizard completed successfully. Click **Close**.
- h.  In the Solution Explorer, locate `Package1.dtsx`, which was created on completion of the wizard. Right-click it and select **Rename**. Change its name to **ImportLedgerInfo.dtsx**
- i.  Double-click the `Source - Ledger2000` and `Destination - Ledger2000` tasks created by the wizard and see whether you understand what the wizard has done.
- j.  Select **File | Save All** or click the corresponding icon in the toolbar.
- k.  Right-click the `ImportLedgerInfo.dtsx` package in the Solution Explorer window and select **Execute Package**. The package should execute successfully, as indicated by all tasks on the Control Flow tab showing green.



*How many rows does it say were transferred from the source to the destination? (Look at the Data Flow tab.)*

---

- l.  Switch to SQL Server Management Studio. Right-click **ClassDB** and select **New Query**.
- m.  Enter the following query:

```
SELECT * FROM dbo.Ledger2000;
```

Click **Execute** (the red exclamation icon) and verify that the expected number of rows has been imported.



**Hands-On Exercise 4.2:  
Using the SSIS Wizard to Import From a Microsoft Access Database  
(continued)**

---



**Congratulations! You have used the SSIS wizard to create a package that imports data from an Access database into a SQL Server table, and you have completed the exercise.**



***This is the end of the exercise.***

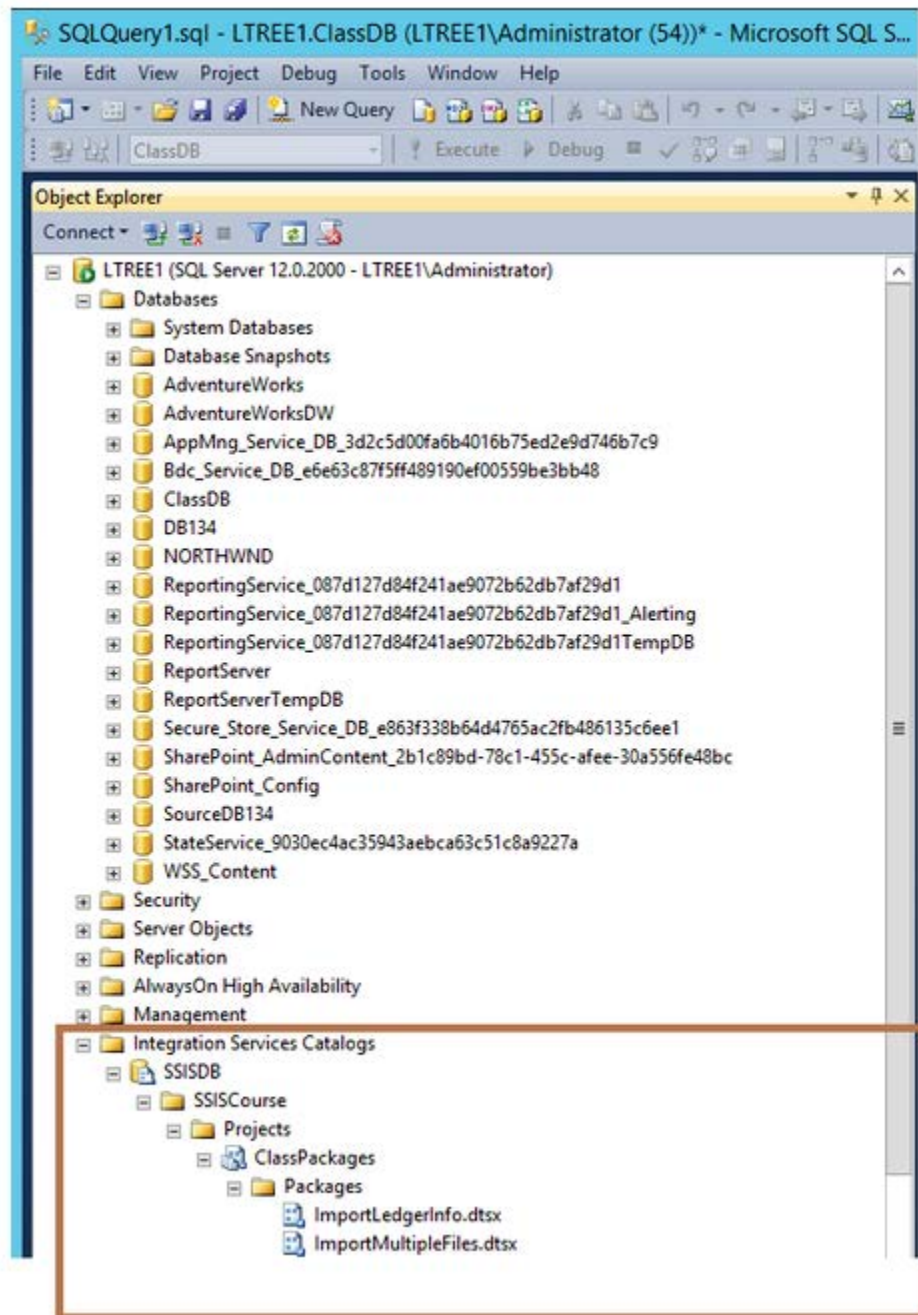




## Hands-On Exercise 5.1: Deploying Packages to a Server

### Objectives

In this exercise, you will deploy all the packages you have created in class to the server so they are stored in a SQL Server catalog database.



## Hands-On Exercise 5.1: Deploying Packages to a Server (continued)

---

1.  Create the SSISDB catalog on the server and create an `SSISCourse` folder to contain the packages that you will deploy later.

More detailed steps...

- a.  If SQL Server Management Studio is not already running, start it and connect to the Database Engine. Locate the `Integration Services Catalogs` folder. Right-click and select **Create Catalog**. Select the checkboxes `Enable CLR Integration` and `Enable automatic execution of Integration Services stored procedures at SQL Server startup`. Enter the password `4555` twice and click **OK**.
  - b.  Right-click the **SSISDB** node that appears and select **Create Folder**. Enter a folder name of `SSISCourse` and click **OK**.
  - c.  Refresh the Databases node in the Object Explorer window. Expand it and notice that a new database SSISDB has been created.
2.  Deploy the project to the path `SSISDB/SSISCourse/ClassPackages`.

More detailed steps...

- a.  Leave SSMS running and switch back to SSDT.
  - b.  Right-click the project node `ClassPackages` within the Solution Explorer window and select **Deploy**. Click **Next** on the first screen. For Server name, enter `localhost` if it is not already selected. For Path, click **Browse**. Be sure the path is set to `/SSISDB/SSISCourse/ClassPackages`. Click **Next**.
  - c.  Click **Deploy**, and click **Close** when it completes.
- Close SSDT and switch to SSMS.
3.  Execute the deployed package `ImportMultipleFiles`.



More detailed steps...

- a.  Expand the `SSISCourse` folder under `SSISDB` and expand **Projects**, **ClassPackages**, and **Packages**. Right-click the package `ImportMultipleFiles.dtsx` and select **Execute**. Click **OK**. Click **No** when asked to see the overview report.



*The overview report is one of many interesting reports you can see as part of monitoring report execution on the server. You can right-click the `SSIDB` node, `ClassPackages` node, `Packages` node, and any package and see a menu of reports that you can view.*

- b.  Click **OK** to dismiss the pop-up dialog box that appears as part of the report execution.



**It may be hiding behind the SSMS window. Look in the task bar.**



**Congratulations! You have deployed SSIS packages to a server and completed the exercise.**



***This is the end of the exercise.***







## Objectives

In this exercise, you will use the `dtexec` command line utility to run the `ImportOrders` package from the command line.

1.  Run the `ImportOrders` package that has been deployed to your server from the command line. If you followed the instructions of the last exercise, it should be stored at the following server path: `\SSISDB\SSISCourse\ClassPackages`.

More detailed steps...

- a.  Switch to the command line, then type and execute the following command:

```
dtexec /ISServer "\SSISDB\SSISCourse\ClassPackages
\ImportOrders.dtsx" /Server LTREE1
```



**Congratulations! You have executed a package from the command line and completed the exercise.**



***This is the end of the exercise.***

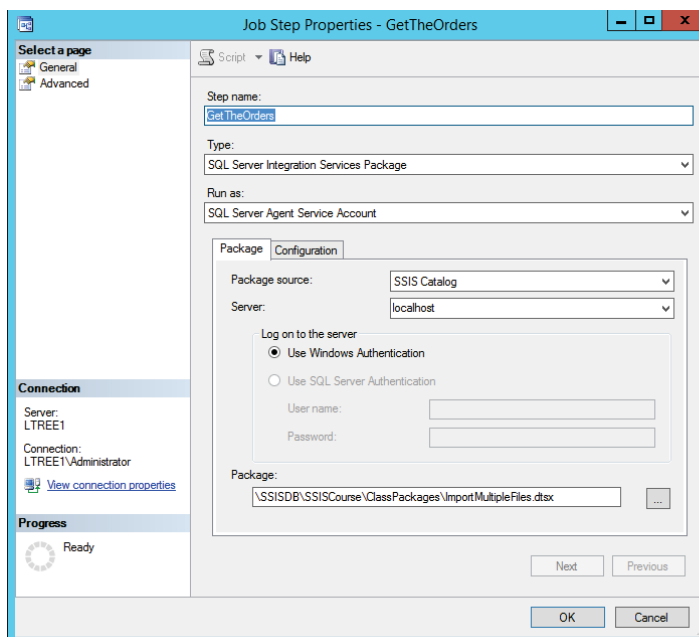




### Objective

In this exercise, you will schedule execution of your SSIS packages using SQL Server Agent's job feature. Scheduling package execution helps automate routine data imports.

1.  Use SQL Server Agent to schedule the `ImportMultipleFiles` package to execute once only in the next two minutes, then check the job history to make sure it executed successfully.



More detailed steps...

- a.  In the Object Explorer window, click the **Connect** drop-down in the toolbar. Select **Database Engine**. In the "Connect to Server" dialog box, make sure the "Server name" is set to `localhost` or your `L'TREE1` server name. Click **Connect**.
- b.  In the Object Explorer window, expand the **SQL Server Agent** node. Right-click the **Jobs** node and click **New Job**. For the job Name, enter `OrdersImportJob`



## Hands-On Exercise 5.3: Scheduling Package Execution (continued)

---

- c.  Click the **Steps** page. Click **New** to create a new job step. For Step name, enter **GetTheOrders**. In the Type drop-down, select **SQL Server Integration Services Package**. In the Package source drop-down, make sure **SSIS Catalog** is selected. In the Server box, enter your **localhost**.

In the Package box, click the ellipsis to browse, select the **ImportMultipleFiles** package, and click **OK**. Click **OK** again to complete configuration of the step.

- d.  Click the **Schedules** page. Click **New** to create a new schedule. For the schedule name, enter **OrdersSchedule**. In the Schedule type drop-down, select **One time**. Under "One-time occurrence," make sure today's date is displayed; in the Time box, select a time that is two minutes later than the current time. Click **OK** and **OK** again to complete configuration of the job.
- e.  After the time specified in the previous step, right-click **OrdersImportJob** and select **View History**. Verify that the job executed successfully. If it did not, troubleshoot.



**Congratulations! You have scheduled a deployed package for execution and completed the course.**



*This is the end of the exercise.*



## Solutions to Hands-On Exercise 1.2: Creating a New Package Product

---

3a. It contains the tasks that make up the steps that do the work when a package executes.

3b. It has the list of files and parameters that are part of the project.

3c. We will use this to configure package components by setting property values.

3d. This is where the package tasks are placed and the execution order and flow are configured.

3e. The Connection Managers tray is where definitions of connections to data sources are stored.





## Solutions to Hands-On Exercise 2.3: Using the Data Flow Task to Import From a Text File

---

1m. 431

1n. Various, e.g. 0.731 seconds

1o. Yes







## Solutions to Hands-On Exercise 4.2: Using the SSIS Wizard to Import From a Microsoft Access Database

---

1e. dbo.Ledger2000

1k. 856





## Solutions to Hands-On Exercise 5.2: Executing Packages from the Command Line

---

```
dtexec /ISServer "\SSISDB\SSISCourse\ClassPackages
\ImportOrders.dtsx" /Server LTREE1
```





## Objectives

**This is a bonus exercise. You should do this on your own without asking the instructor for help. Use your notes or Exercise Manual if necessary. You are tasked with exporting the contents of the database table `OrderDetails` to a text file with the same filename. Use two methods to do the export.**

1.  Export the contents of the `OrderDetails` table to a text file `OrderDetails.txt` by using the SSIS Import/Export wizard. Figure out by yourself how to do this. If necessary, refer to your notes or a previous exercise.
2.  Export the contents of the table `OrderDetails` to a text file. This time, do not use the wizard.



***This is the end of the exercise.***





## Bonus Hands-On Exercise 2: Using Precedent Constraints with Variables

### Objectives

In this bonus exercise, you are tasked with using a variable value to determine the execution path of a package. Task 1 will assign a value to the variable. Then, you will use the variable value to determine whether Task 2 or Task 3 should be executed. You will change the value being assigned in Task 1 and make sure it works.



*Again, you are on your own. You will get more out of this bonus exercise by investigating it yourself and getting it to work. A slide in Chapter 3 should give you all the hints you need.*

*Just one more thing: If you are testing whether a variable is equal to a certain value, you use `==`, not just `=`. For example:*

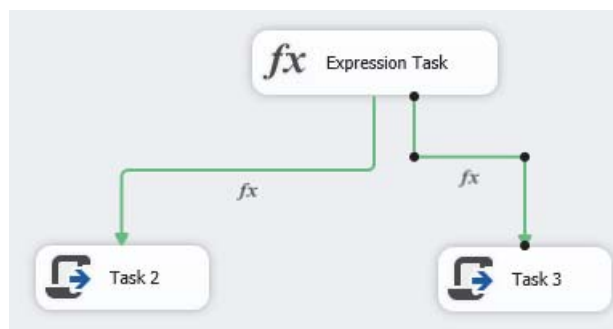
```
[@User::WhatToDo] == 1
```

*If you testing non-equality, use `!=`. For example:*

```
[@User::WhatToDo] != 1
```

*However, if you are assigning a value, you use `=`. Task 2 should display a suitable message; e.g., "I am Task 2!" Similarly for Task 3.*

*Your package structure should look like the following.*



***This is the end of the exercise.***









## Objectives

This is an after-course project. In class, we covered only a subset of tasks and transformations that SSIS has. Hopefully, this will have given you a base from which to extend your knowledge of how to configure other tasks without needing further instruction.

 You will find two more tasks quite useful: FTP task and File System task. We suggest you briefly research what these two tasks do, either on the Microsoft Developer Network or TechNet. You can also simply do a web search for the task names and see the links that come up.

1.  Copy the files named `authors0.txt` through `authors4.txt` from the `ImportFiles` folder to the ftp location: `c:\inetpub\ftproot`. Create a package that downloads the files from the ftp location, then imports them in a loop. After all the files are imported successfully, delete them from the ftp server. If you are feeling up to it, you can archive the downloaded files after a successful import.

 We have already given you a hint that the FTP and File System tasks might be required.



***This is the end of the exercise.***



