MCTS Self-Paced Training Kit (Exam 70-448: Microsoft SQL Server 2008—Business Intelligence Development and Maintenance

by Erik Veerman, Teo Lachev and Dejan Sarka
Microsoft Press. (c) 2009. Copying Prohibited.

Chapter 1: Developing SSIS Packages

Overview

A *package* is the core object within SQL Server Integration Services (SSIS) that contains the business logic to handle workflow and data processing. You use SSIS packages to move data from sources to destinations and to handle the timing precedence of when data is processed. You can create packages by using the SQL Server Import And Export Wizard in SQL Server Management Studio (SSMS) or by using the SSIS Designer in the Business Intelligence Development Studio (BIDS). This chapter looks at creating and defining packages in SSIS and using the main components of the control flow and data flow objects with sources and destinations.

SSIS is designed for many data integration and processing applications. One of those applications is the processing of data into a data mart or data warehouse, where data is used exclusively for business intelligence (BI) analytics and reporting. Although many businesses use SSIS for BI, there are many other applications of SSIS. For example, many organizations use SSIS to move data from legacy systems into new systems during application migrations, to integrate data from multiple systems by passing data back and forth, to extract data for sending to vendors or partners, to cleanse data, to import data from vendors or partners—the list goes on. Because this Training Kit focuses on BI, part of the SSIS content and lessons cover using SSIS for data warehouse extraction, transformation, and loading (ETL), but the SSIS chapters and lessons also explain how to take advantage of SSIS for other purposes.

This initial chapter explains how to create SSIS packages and defines the basic objects contained in the control flow and data flow. Later chapters describe the advanced features, deployment, and implementation details of SSIS.

Exam objectives in this chapter:

- Implement control flow.
- Implement data flow.
- Implement package logic by using variables.
- Extend SSIS packages by using .NET code.
- Identify and resolve issues related to SSIS solution deployment.
- Install and maintain SSIS components.

Before You Begin

To complete this chapter, you must have:

- Knowledge of Microsoft SQL Server 2008, including SSIS features and components.
- Experience working with SQL Server Business Intelligence Development Studio (BIDS) projects and solutions.
- Experience working in SQL Server Management Studio (SSMS).
- The AdventureWorks2008 and AdventureWorksDW2008 sample databases installed. You can download these databases from the CodePlex community Web site at *http://www.codeplex.com/MSFTDBProdSamples*.

Lesson 1: Creating SSIS Packages and Data Sources

Estimated Lesson Time: 50 Minutes

The core object within SSIS is a *package*. A package contains the business logic to handle the data extraction, manipulation, and transformation tasks needed to move data to destinations. Packages also contain workflow elements to help process data. These workflow elements might involve running a stored procedure, moving a file from an FTP server to a destination folder on your server, or sending an e-mail message when an error occurs. When you execute a package, the logic within performs the designed steps.

Packages also contain connections to data sources and data destinations. You set up these connections to connect to different external systems such as databases, files, File Transfer Protocol (FTP) servers, Simple Mail Transfer Protocol

(SMTP) servers, and so on. Connections are used for the SSIS data processing engine (called the *data flow*) as well as the workflow engine (called the *control flow*).

Creating SSIS Packages

The first step in getting started with SSIS is to create a package. You can accomplish this in one of two ways:

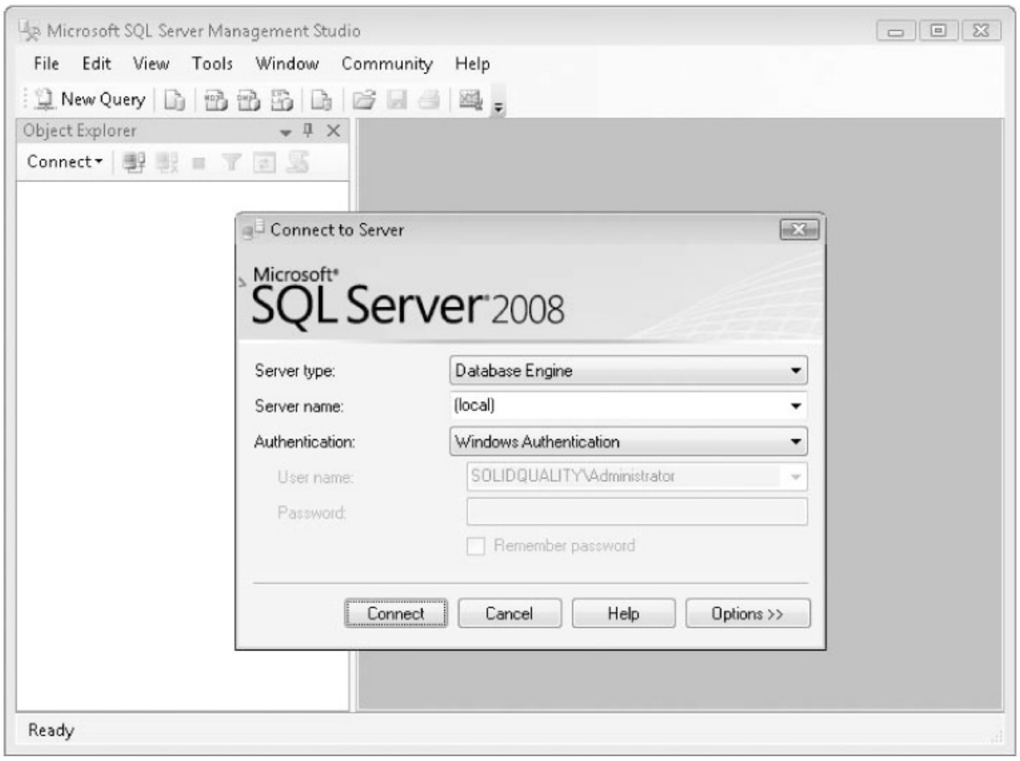
- By using the built-in Import And Export Wizard in SQL Server 2008, which asks you about moving data from a source to a destination and then automatically generates an SSIS package. After you create a package in the wizard, you can execute it immediately, schedule it, or associate it with an SSIS project.
- By explicitly creating a package inside an SSIS project in BIDS. BIDS in SQL Server 2008 uses the Microsoft Visual Studio 2008 interface with specific templates installed to create BI objects such as SSIS packages. Within the BIDS development environment, you first create an SSIS project and then create and develop new packages.

The remainder of this lesson explains using both methods to develop SSIS packages.

Using the Import And Export Wizard

With SQL Server 2008, you use the Import And Export Wizard to copy data without going through the process of creating an SSIS project. When you use the wizard, it generates an SSIS package that you can execute immediately or save and then manually modify or schedule.

You typically start the Import And Export Wizard through SSMS when you are connected to the SQL Server relational engine. SSMS is the SQL Server management tool designed primarily for managing databases, and you will be using SSMS many times in the lessons throughout this Training Kit. To launch SSMS, from the Start menu, select Microsoft SQL Server 2008 and then SQL Server Management Studio. Figure 1-1 shows the Connect To Server dialog box, where you first connect to the Database Engine.


Microsoft SQL Server Management Studio
File Edit View Tools Window Community Help
New Query
Object Explorer
Connect
Connect to Server
Microsoft SQL Server 2008
Connect
Cancel
Help
Options >>
Ready

Server type: Database Engine
Server name: (local)
Authentication: Windows Authentication
User name: SOLIDQUALITY\Administrator
Password:
Remember password

Figure 1-1: Start SSMS by selecting Microsoft SQL Server 2008 from the Start menu and then selecting SQL Server Management Studio. Connect to the Database Engine to manage SQL Server relational databases.

Starting the Import And Export Wizard

As its name states, the Import And Export Wizard can both import and export data. Use the following considerations to determine which part of the wizard to use:

- Importing data with the wizard lets you bring into a SQL Server table any data contained in accessible sources. Sources include other SQL Server databases, flat files, data in Microsoft Office Excel spreadsheets or Microsoft Office Access databases, and data in Oracle databases.
- Exporting data with the wizard lets you send data from SQL Server tables, views, or custom queries to flat files or database connections.

To start the Import And Export Wizard, follow these steps:

1. Through SSMS, connect to the instance of the SQL Server 2008 Database Engine that contains your source or your destination.
2. Open Object Explorer. You will find a list of various object containers under the SQL Server connection. The Databases folder shows all the databases attached to that instance of SQL Server. The System Databases subfolder contains the system databases.
3. To start the Import And Export Wizard, right-click the database that you want to use as your source or destination.
4. Click Tasks. If the database is the source of data that you want to send to a different system, select Export Data. If the database is the destination for files that currently exist outside the system, select Import Data, as Figure 1-2 shows.

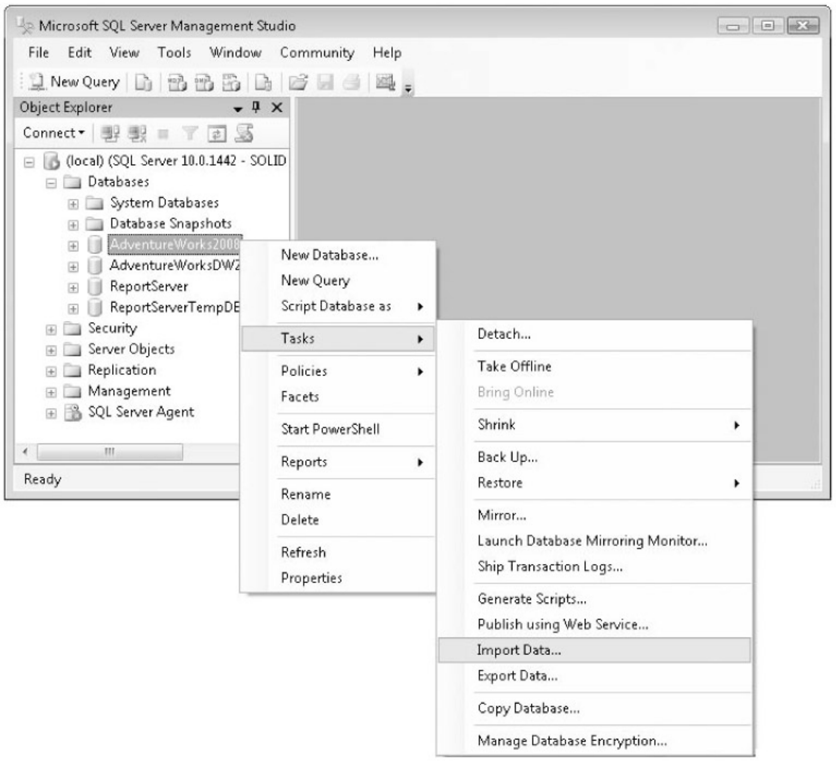


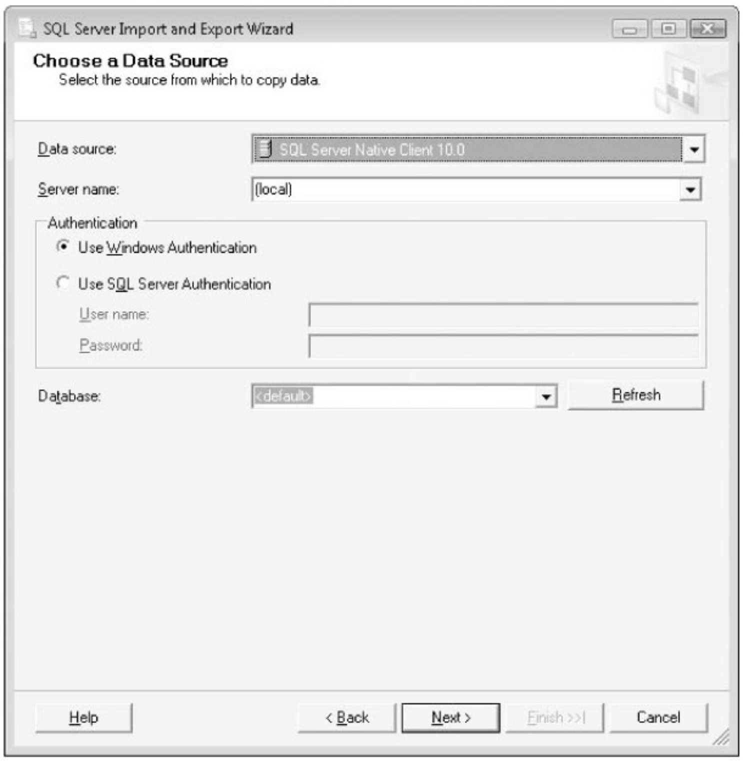
Figure 1-2: Start the Import And Export Wizard by right-clicking the database in SSMS and then clicking Tasks.

The wizard then walks you through several pages of questions, the answers to which are used to build the resulting package. The wizard pages include the following:

1. The Choose A Data Source page lets you specify where your data is coming from, such as a SQL Server database, an Excel file, a flat file, or other source. If your source is a relational database, you can also configure the security for the connection. Figure 1-3 shows the first page of the Import And Export Wizard.
2. The Choose A Destination page lets you specify where your data will be sent. You specify the destination type and, if applicable, the server name and security settings needed to access the data. If you chose Import Data in SSMS to start the wizard, the data destination settings will match those of the database you selected prior to starting the wizard.
3. If you selected a relational database source that allows custom queries, on the Specify Table Copy Or Query page,

you can choose to copy the data directly from the source to the destination or to specify a query. If you choose to specify a query, an additional page, named Provide A Source Query, enables you to manually enter the query.

4. If your source is a relational database and you do not specify a query, you can choose tables and views from your source on the Select Source Tables And Views page. If your source is a flat file or you specified a query, only the file or query is available as a choice. Also on this page, you can rename the destination table and edit the column mappings by clicking the Edit Mappings button to define column NULL settings, identity insert, and whether the table should be dropped and recreated every time.
5. Use the Save And Run Package page to execute the package immediately or save the package for later execution. If you save the package, you can later go back and edit the package by using the SSIS Designer, which is demonstrated in the rest of this chapter.

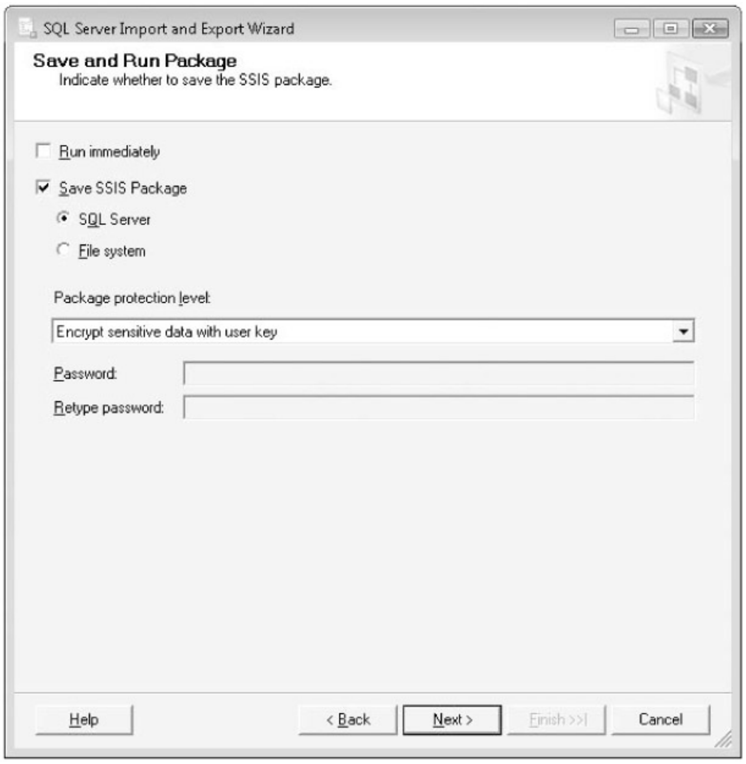

SQL Server Import and Export Wizard
Choose a Data Source
Select the source from which to copy data.
Help
< Back
Next >
Finish >>|
Cancel
Refresh

Data source: SQL Server Native Client 10.0
Server name: (local)
Authentication
(● Use Windows Authentication)
○ Use SQL Server Authentication
User name:
Password:
Database: <default>

Figure 1-3: The Import And Export Wizard first lets you choose the data source where the data will be coming from, such as a SQL Server database, an Excel spreadsheet, or a flat file.

Saving and Editing Packages Created in the Wizard

The wizard's last page lets you execute the package immediately or save it. If you choose to save the autogenerated package within an Integration Services project in BIDS, as Figure 1-4 shows, you can modify its contents later. At times, you might want to use the wizard to generate a basic package to which you can add more advanced logic that the wizard cannot generate.


SQL Server Import and Export Wizard
Save and Run Package
Indicate whether to save the SSIS package.
Help
< Back
Next >
Finish >>|
Cancel

Run immediately [ ]
Save SSIS Package [✓]
SQL Server [✓]
File system [ ]
Package protection level: Encrypt sensitive data with user key
Password:
Retype password:

Figure 1-4: The final page of the Import And Export Wizard lets you execute and/or save packages.

In general, the Import And Export Wizard provides a quick way to move data from one source to a destination, especially for a one-time use, but there are some limitations:

- You can specify only one source and one destination in the wizard.
- Advanced workflow precedence is not available through the wizard.
- The wizard does not share data sources with other packages.

You need to evaluate whether your data processing requirements enable you to use the wizard or whether you need to develop a new package from scratch in BIDS.

Creating an SSIS Project in BIDS

Although the Import And Export Wizard is useful for generating a quick package that moves data from one source to one destination, these packages are frequently only a starting point. More often than not, you will need to either develop a package that has more complicated requirements or create a set of coordinated packages. For these cases, you first need to create a new SSIS project in BIDS.

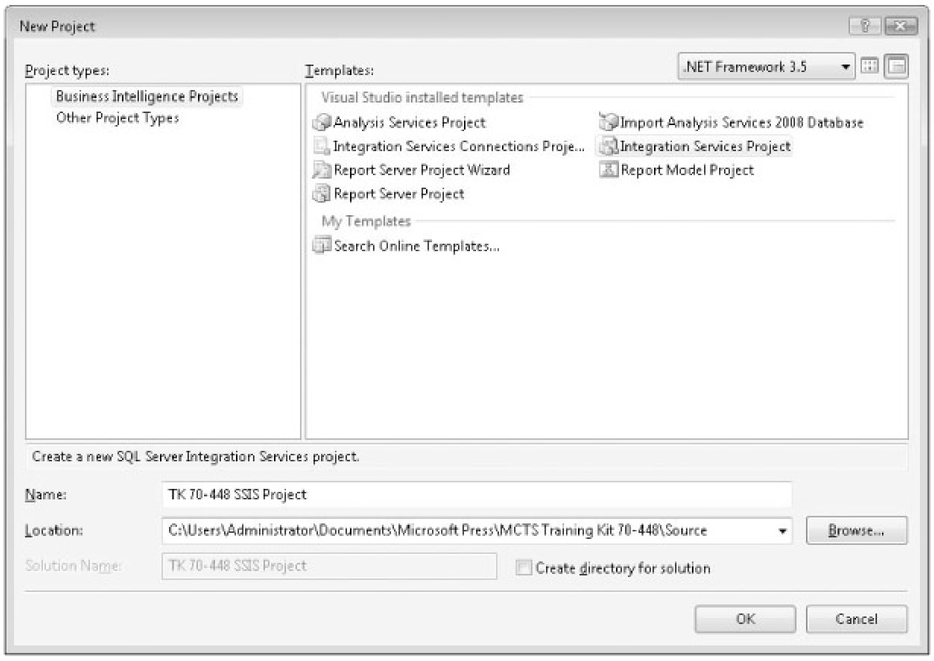
Note—Objects in BIDS Remember that any one project in BIDS can contain only objects from the same project type, such as SSIS, SQL Server Analysis Services (SSAS), or SQL Server Reporting Services (SSRS). However, a single project can be associated with projects of different types in the same solution.

All of the SQL Server BI components are generated in a similar fashion through the BIDS development tool. To launch BIDS, from the Start menu, select Microsoft SQL Server 2008 and then SQL Server Business Intelligence Development Studio. Follow these steps to create a new SSIS project:

1. In BIDS, choose New, Project from the File menu. (If you have Visual Studio 2008 installed separately from BIDS, you can simply select New Project from the File menu.) Figure 1-5 shows the resulting New Project dialog box.
2. Fill out the New Project dialog box as follows:
   a. Under Project Types, select Business Intelligence Projects.
   b. Under Templates, select Integration Services Project.
   c. Assign a name to your project in the Name box.

   d. In the Location box, either leave the default folder location for storing new projects (in the ..\Documents\Visual Studio 2008\Projects\ folder) or change to a location of your choice.

3. When you have finished, click OK to build the project. The project contains several SSIS logical object folders, which Solution Explorer displays. You use these objects in your SSIS projects to point to connections and process data. Figure 1-6 shows a new project, with the default Package.dtsx package (created with the project) in the SSIS Designer and Solution Explorer on the right.


New Project
Project types:
Business Intelligence Projects
Other Project Types
Templates:
.NET Framework 3.5
Visual Studio installed templates
Analysis Services Project
Import Analysis Services 2008 Database
Integration Services Connections Proje...
Integration Services Project
Report Server Project Wizard
Report Model Project
Report Server Project
My Templates
Search Online Templates...
Browse...
OK
Cancel

Create a new SQL Server Integration Services project.

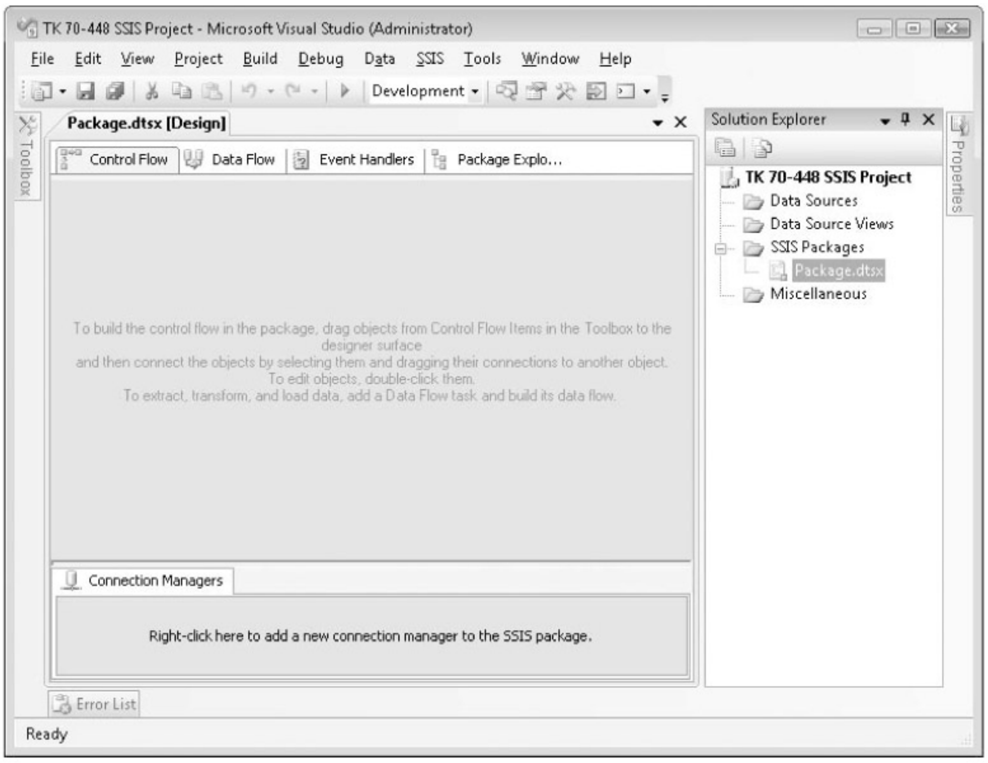
Name: TK 70-448 SSIS Project

Location: C:\Users\Administrator\Documents\Microsoft Press\MCTS Training Kit 70-448\Source

Solution Name: TK 70-448 SSIS Project

Create directory for solution

Figure 1-5: Creating a new project in BIDS begins in the New Project dialog box.



To build the control flow in the package, drag objects from Control Flow Items in the Toolbox to the designer surface and then connect the objects by selecting them and dragging their connections to another object. To edit objects, double-click them. To extract, transform, and load data, add a Data Flow task and build its data flow.

Right-click here to add a new connection manager to the SSIS package.

Figure 1-6: Creating a new project automatically creates a new SSIS package named Package.dtsx and several logical folders in Solution Explorer in BIDS.

You are now ready to configure and develop your package.

To add an existing package—such as one created by the Import And Export Wizard—to your project, right-click the SSIS Packages folder in Solution Explorer, and then click Add Existing Package. This dialog box lets you import packages from other projects or import packages that have already been deployed.

Exam Tip When you create packages in BIDS, the package is stored in the file system with the .dtsx file extension. This .dtsx file is an XML file that contains the logic and the layout for the design you have developed in BIDS, and you can move the file to a different project, manually deploy it to different servers, or make it part of a deployment package. In Chapter 3, "Deploying and Configuring SSIS Packages," and Chapter 4, "Administering, Securing, and Executing SSIS Packages," you will work with .dtsx files during deployment and execution.

Developing Project Data Sources and Package Connections

Because the main purpose of SSIS is to move data from sources to destinations, the next most important step is to add the pointers to these sources and destinations. These pointers are called *data sources* and *connections*. Data sources are stored at the project level and are found in Solution Explorer under the logical folder named Data Sources. Connections, on the other hand, are defined within packages and are found in the Connection Managers pane at the bottom of the Control Flow or Data Flow tab. Connections can be based on project data sources or can stand alone within packages. The next sections walk you through the uses and implementation of project data sources and package connections.

Creating a Data Source

A *data source* is an SSIS project object. Data sources contain connection strings that point to files or databases, and you can reference them within one or more packages. Data sources are optional within SSIS, but they are beneficial during development if you have a large number of packages that need to use the same database or file connection. Using a data source also helps if you need to change a connection used in many packages. You simply change the data source once and then open each package in your project, which will automatically synchronize the connection string stored in the package with the data source.

Important—Project Data Sources are for Development Purposes Only Be aware that after a package is deployed to a new environment and executed outside the project, the connection string is no longer updated by the project data source. Instead, you must use package configurations to share connection strings. See Chapter 3 to find out about sharing connection strings by using package configurations.

Using data sources in your project and packages is a two-step process:

1. **Creating the data source** Within Solution Explorer, right-click the Data Sources folder, and then click New Data Source. On the Welcome page of the wizard, click Next. Figure 1-7 shows the Data Source Wizard.

   If you have made connections in the past on your server, a cached list of those connections appears in the Data Connections area, and you can choose an existing connection or click the New button to create a new connection. Figure 1-7 shows the connection page of the wizard without any cached connections.

2. **Adding the data source to a package** After you create your data source in the project, you need to add the data source to your packages.

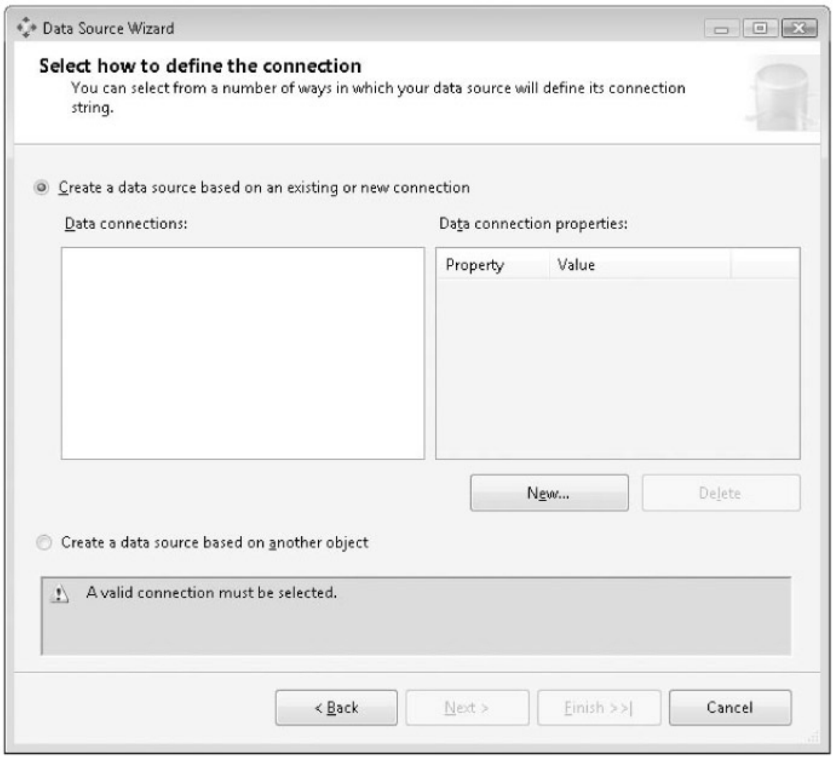
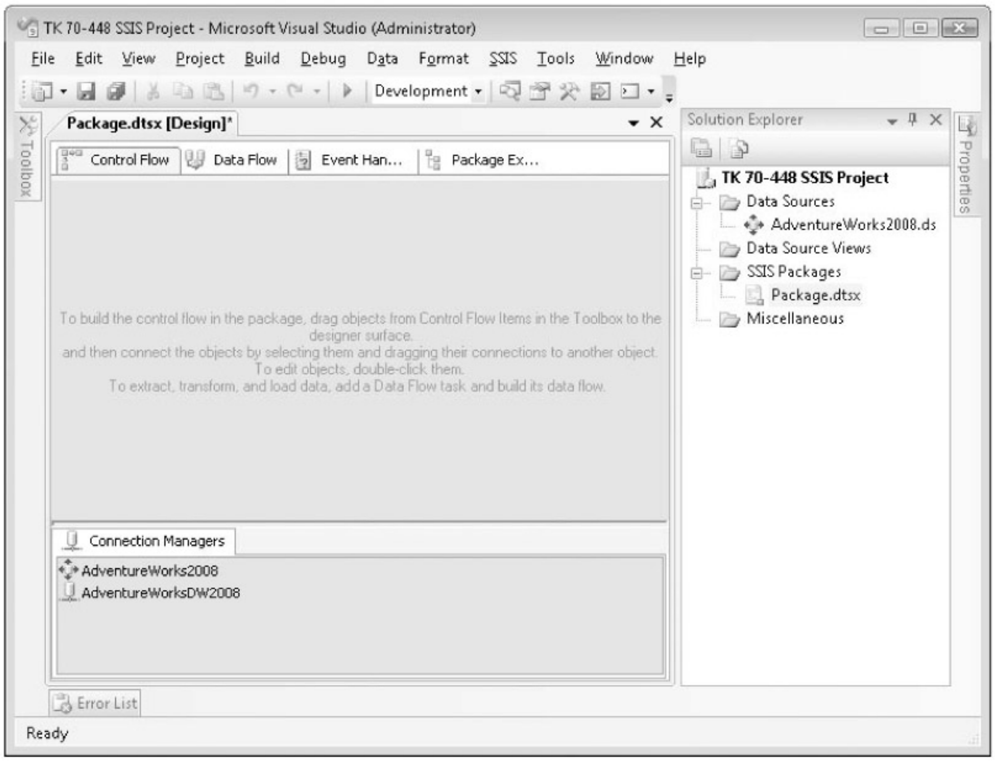

Data Source Wizard
Select how to define the connection
You can select from a number of ways in which your data source will define its connection string.
Create a data source based on an existing or new connection
Data connections:
Data connection properties:
Property
Value
New...
Delete
Create a data source based on another object
A valid connection must be selected.
< Back
Next >
Finish >>|
Cancel

Figure 1-7: The Data Source Wizard lets you create a new connection in your project that can be shared between packages. The New button starts the Connection Wizard to create a new connection.

Creating Package Connection Managers

A *package connection manager*, sometimes simply called a *package connection*, is independent of project data sources. However, package connections can reference a project data source. A package connection lets the different components in SSIS communicate with an object (such as a database, file, or server) outside the package. You can use package connections as source adapters, FTP or e-mail servers, or flat files.

If you link the package connection to the project data source, when the project data source is edited, the package connection is also updated when the package is being developed. In the BIDS design environment in Figure 1-8, Solution Explorer shows a project data source, and two package connections appear in the Connection Managers pane at the bottom of the SSIS Designer.



To build the control flow in the package, drag objects from Control Flow Items in the Toolbox to the designer surface.
and then connect the objects by selecting them and dragging their connections to another object.
To edit objects, double-click them.
To extract, transform, and load data, add a Data Flow task and build its data flow.

Figure 1-8: SSIS projects contain data sources, and packages contain connection managers. Project data sources can be linked to package connection managers, or connection managers can stand alone within a package.

Packages can be based on data sources defined in the SSIS project, or they can be stand-alone connections within a project. In Figure 1-8, the project has a data source named Adventure Works2008, which is also referenced in the package's Connection Managers pane. In this example, the package contains another connection named AdventureWorksDW2008, which does not reference a project data source. The icon used to identify a project data source matches the icon for the package connection if a package connection references a project data source.

Adding Connections in the Connection Managers Pane

To create a new connection, right-click in the Connection Managers pane at the bottom of the Control Flow tab, as Figure 1-9 shows.

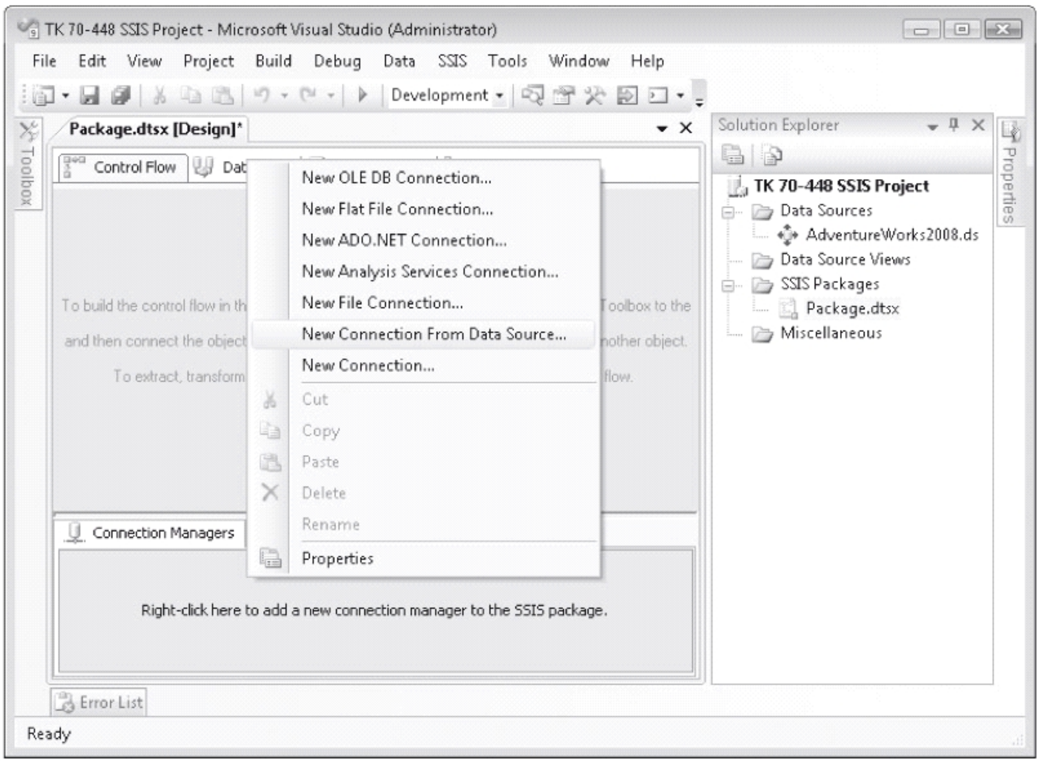

TK 70-448 SSIS Project - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Data SSIS Tools Window Help
Development
Package.dtsx [Design]*
Control Flow
New OLE DB Connection...
New Flat File Connection...
New ADO.NET Connection...
New Analysis Services Connection...
New File Connection...
New Connection From Data Source...
New Connection...
Cut
Copy
Paste
Delete
Rename
Properties
Connection Managers
Right-click here to add a new connection manager to the SSIS package.
Solution Explorer
TK 70-448 SSIS Project
Data Sources
AdventureWorks2008.ds
Data Source Views
SSIS Packages
Package.dtsx
Miscellaneous
Error List
Ready

Figure 1-9: Right-click in the Connection Managers pane to create a new connection for your package.

You can create a connection from a project data source by selecting New Connection From Data Source, as Figure 1-9 shows, or you can create a stand-alone connection within a package by selecting New Connection and then choosing another connection provider from the Connection Manager Type list. A stand-alone connection is a connection that is not shared during development with other packages using a project data source. Stand-alone connections in a package that have the same name as a connection or connections in another package can, however, be updated together at run time through package configurations.

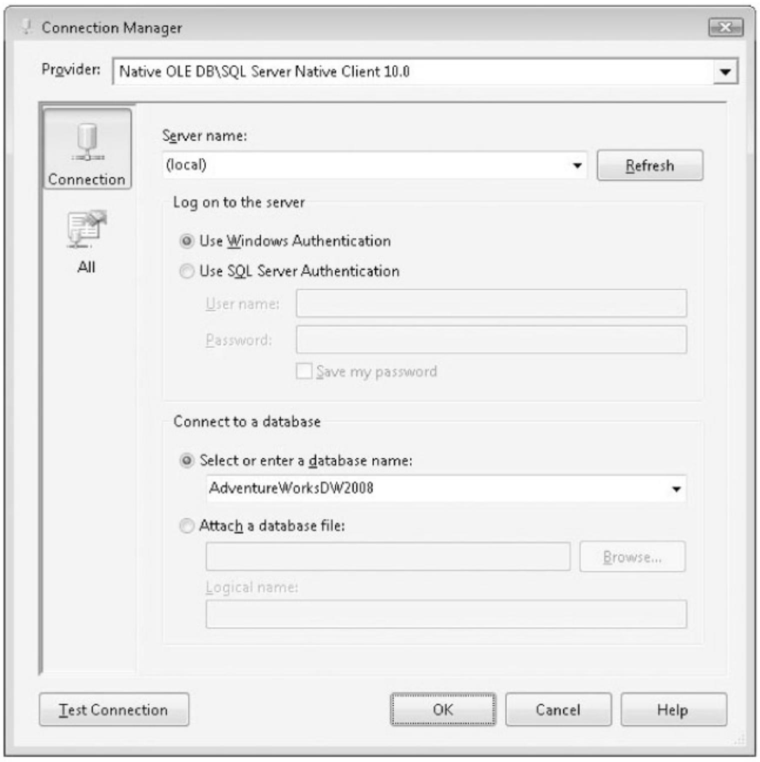
Important—Modifying a Project Data Source

When you create a package connection from a data source, that connection is updated only during development whenever the package is opened and the data source has been changed. Package connections are not updated when they are run separately from the associated SSIS project—for example, when they are run from the command line.

The first step in creating a package connection is choosing the connection type, as Figure 1-9 shows. If you select a connection based on a data source, the connection has been created. However, if you choose another type of connection, you must perform at least one more step before the connection is complete. For example, if you are connecting to a relational database, you must select either New OLE DB Connection or New ADO.NET Connection (depending on which connection provider you want to use to access an underlying database). After you select the connection type, you need to configure that connection.

When you select the connection type, SSIS prompts you to either select a connection string that has been cached on the machine you are working on or create a new cached connection. If the connection string is already cached on your machine, simply select that connection from the list to add it to the list of connections in the Connection Managers pane in your package.

If a connection string does not exist in the cache, you need to create a new connection string. For example, to define a new connection string for an OLE DB connection, in the Connection Managers pane, right-click and then click New OLE DB Connection. Next, in the Configure OLE DB Connection Manager dialog box, click New. In the Provider list, choose from the list of OLE DB providers that are installed on your machine, and then specify the database name and the connection security credentials, as Figure 1-10 shows, and then click OK. After you specify the connection options, you can choose the newly cached connection from the list, which then adds the new connection to the Connection Managers pane in the package.


Connection Manager

Provider: Native OLE DB\SQL Server Native Client 10.0

Server name: (local)

Log on to the server

Use Windows Authentication [✓]

Use SQL Server Authentication

User name:

Password:

Save my password

Connect to a database

Select or enter a database name: [✓] AdventureWorksDW2008

Attach a database file:

Logical name:

Figure 1-10: The Connection Manager dialog box lets you specify the provider, server, database, and security for the new connection.

Now that you have finished creating your package and package connections, you are ready to start developing package components. Connections are used in several package components, including control flow tasks, data flow adapters and transformations, event handlers, package logging, and package configurations—all of which are described in other chapters of this Training Kit.

Practice—Creating New Packages, Data Sources, and Connections

The following exercises familiarize you with the common tasks of creating a new project in BIDS and working with data sources and connections. All the practice files can be installed from the Practice folder on the companion CD.

EXERCISE 1 Create the Project and Packages

In this exercise, you will create a new SSIS project and then work with a couple of SSIS packages by adding data sources and connections.

1. Start SQL Server Business Intelligence Development Studio (BIDS), by clicking the Start button and then selecting All Programs, Microsoft SQL Server 2008, SQL Server Business Intelligence Development Studio.
2. Choose New, Project from the File menu. (If you have Visual Studio 2008 installed separately from BIDS, simply choose New Project from the File menu.) The New Project dialog box displays all the installed templates for Microsoft Visual Studio, including the Business Intelligence Projects templates.
3. In the New Project dialog box, confirm that Business Intelligence Projects is selected in the Project Types area, and then in the Templates area, select the Integration Services Project template.
4. Near the bottom of the New Project dialog box, in the Name box, type **TK 70-448 SSIS Project** as the name of your SSIS project.
5. In the Location box, type the path, starting with the Documents folder in your user profile: **..\Documents\Microsoft Press\MCTS Training Kit 70-448\Source\**. This is the same location where the practice exercise files for the Training Kit will be installed by default.
6. Next, clear the Create Directory For Solution check box, which stores the SSIS project in the folder you specified in step 5.

7. Click OK to have BIDS create the new SSIS project.
8. When the project is created, SSIS automatically creates a new SSIS package named Package.dtsx and opens it in the SSIS Designer. In Solution Explorer, right-click Package.dtsx, and then click Rename.
9. Rename the package by typing **MyPackage.dtsx**. BIDS might prompt you to rename the package object. If a message box appears that prompts you to rename the package object as well, click Yes. Always click Yes if you are prompted to change the package object when renaming a package because this updates the internal name of the package.
10. Click the Save button on the toolbar, and then close the package by clicking the Close button in the upper-right corner of the SSIS Designer.
11. To create a new package, right-click the SSIS Packages folder in Solution Explorer, and then click New SSIS Package. This creates a new package object named Package1.dtsx (the number depends on how many packages you have created) in the SSIS Packages folder in Solution Explorer.
12. To rename the new package, right-click the package, and then click Rename. Rename the package to **DimCustomer.dtsx** because this package will contain logic to process the customer dimension table. When prompted, click Yes to rename the package object.
13. Following the same steps, create one more package in your SSIS Project named **DimPromotion.dtsx**.

EXERCISE 2 Create Project Data Sources

In this exercise, you will create two project data sources, which will be used in your packages as the source and the destination.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project you created in Exercise 1, *TK 70-448 SSIS Project*, and then open Solution Explorer (if it is not already displayed). You can open Solution Explorer by clicking the Solution Explorer button on the Standard toolbar.
2. In Solution Explorer, right-click the Data Sources folder, and then click New Data Source. When the Welcome page of the Data Source Wizard appears, click Next.
3. On the Select How To Define The Connection page, select Create A Data Source Based On An Existing Or New Connection.
4. Click New to open the Connection Manager dialog box.
5. In the Provider drop-down list, select the Native OLE DB\SQL Server Native Client 10 provider and click OK. Type **(local)** in the Server Name field.
6. In the Select Or Enter A Database Name drop-down list, select AdventureWorks2008.
7. Click the Test Connection button, and then click OK. Click OK again to close the Connection Manager dialog box.
8. Select the (local).AdventureWorks2008 data connection in the Data Connections list, and then click Finish in the Data Source Wizard.
9. The Completing The Wizard page prompts you to enter a name for the new project data source. Type **AdventureWorks2008** in the Data Source Name box, and then click Finish. Be sure to remove the space between Adventure and Works2008.
10. Next, repeat steps 2 to 9 to create a new project data source for the (local).AdventureWorksDW2008 database, and name this data source **AdventureWorksDW2008.**
11. When you are finished creating the data sources, click the Save All button on the BIDS toolbar.

EXERCISE 3 Create New Package Connections from the Project Data Sources

In this exercise, you will add the project data sources you just created to the two packages that you have developed.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project you created in Exercise 1, *TK 70-448 SSIS Project*, and then open Solution Explorer. Edit your MyPackage.dtsx package by double-

clicking the package in Solution Explorer.

2. Locate the Connection Managers pane (at the bottom of the SSIS Designer window), right-click in the pane, and then click New Connection From Data Source.
3. In the Select Data Source dialog box, select both the AdventureWorks2008 and AdventureWorksDW2008 data sources from the list, and then click OK to accept. This puts the two project data sources into the package's Connection Managers pane.
4. Perform the same steps in the *DimCustomer.dtsx* package to add the AdventureWorks2008 and AdventureWorksDW2008 project data sources as connection managers for the package.
5. When you are finished creating the connection managers, click the Save All button on the BIDS toolbar.

Quick Check

1. You are asked to combine data from an Excel workbook and a database table and then push the results to a fixed-width flat file. Can you accomplish this task by using the Import And Export Wizard? ?

2. You need to create both SSIS packages to process your data and SSAS cubes to perform analysis. Can you create both objects in a single project? ?

3. What is the difference between a project data source and a package connection? ?

4. If a connection references a data source and the data source is changed, when will the connection be updated? ?

Answers

1. No. The Import And Export Wizard lets you work with only a single source and a single destination. To combine data merging or data cleansing tasks, you need to either create a new package specifically for that purpose or modify a package previously created by the wizard.

2. No. You cannot create both SSIS and SSAS objects in one project because BIDS does not let you combine objects used for different platforms. You need to build two separate projects in BIDS: one for the SSIS packages and another for the SSAS cubes and dimensions.

3. Both project data sources and package connections are connection strings. However, a data source resides outside the package and can be used as the connection reference for more than one package. A package connection does not have to be associated with a data source.

4. Connections are updated by their associated data sources only when the package is opened for editing in BIDS.

Lesson 2: Creating and Editing Control Flow Objects

Estimated Lesson Time: 45 Minutes

Now that you have created an SSIS project, packages, and package connections, it is time to start using SSIS features for data integration and for processing logic. This lesson and Lesson 3, "Using Data Flow Adapters and Transformations," focus on the two main components within SSIS: the control flow and the data flow. The *control flow* is the workflow engine and contains control flow tasks, containers, and precedence constraints, which manage when tasks and containers execute. The *data flow*, in contrast, is directly related to processing and transforming data from sources to destinations.

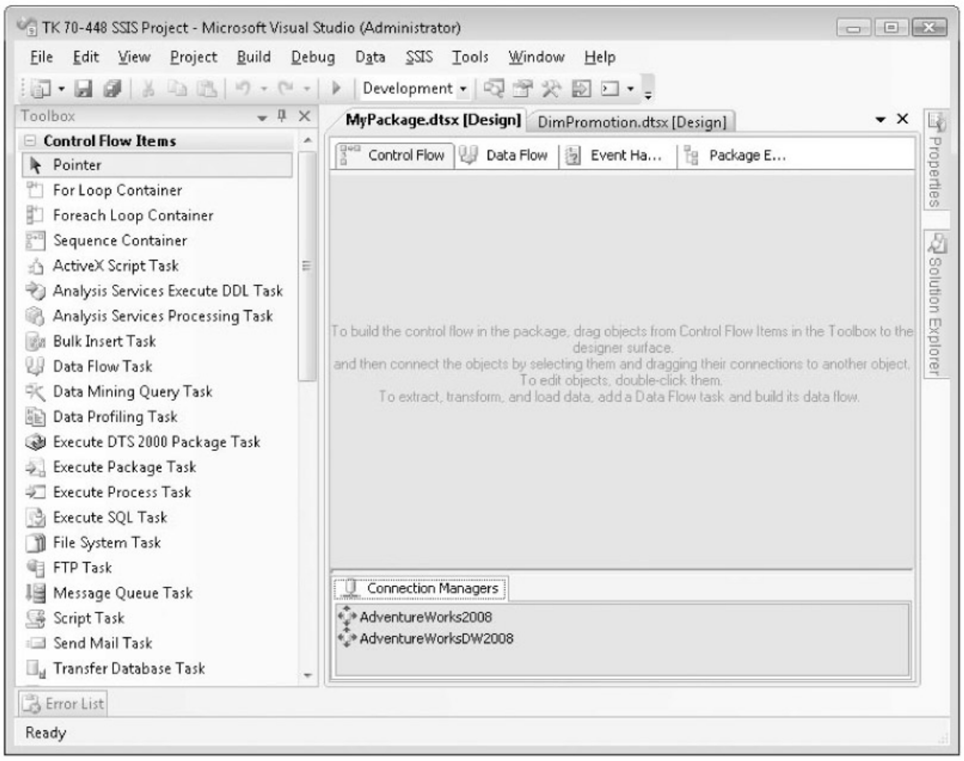
This lesson looks at defining control flow objects with the Control Flow design surface. When you have an SSIS project open within BIDS and are designing a package, the tabs across the top of the SSIS Designer let you choose an SSIS component to work with. The Control Flow design surface is the first tab and displays a workspace where you configure control flow objects. There are three primary types of control flow objects:

- **Control flow tasks** Workflow objects that perform operational-level jobs
- **Control flow containers** Provide a grouping mechanism for tasks and other containers
- **Constraints** Let you connect tasks and containers and define execution ordering and precedence

Creating Control Flow Tasks

A *control flow task* is an SSIS component that performs an operation such as sending an e-mail message, executing an SQL statement, or copying a file from an FTP server. When a control flow task is complete, it either succeeded or failed. You use the control flow to coordinate the execution of tasks in parallel or to set precedence constraints based on the tasks' completion status. See Chapter 2, "Debugging and Error Handling in SSIS," to learn more about precedence constraints.

To create a new control flow task in your package, drag the task from the toolbox to the Control Flow tab in the SSIS Designer. Figure 1-11 shows the Toolbox window when the Control Flow tab is clicked in the SSIS Designer.



To build the control flow in the package, drag objects from Control Flow Items in the Toolbox to the designer surface.
and then connect the objects by selecting them and dragging their connections to another object.
To edit objects, double-click them.
To extract, transform, and load data, add a Data Flow task and build its data flow.

Figure 1-11: Control flow objects are found in the BIDS toolbox when you click the Control Flow tab in the SSIS Designer.

After you add a task to the control flow workspace, you need to configure the task to perform the specific operation you selected. To allow configuration, every task has an editor that you can open by double-clicking the task or by right-clicking the task and then clicking Edit. Table 1-1 lists the tasks in SSIS under the Control Flow Items list in the toolbox.

Table 1-1: Common Tasks in SSIS

TASK	DESCRIPTION
ActiveX Script Task	Runs Microsoft Visual Basic Scripting Edition (VBScript) and JScript code and is included mainly for legacy support when a Data Transformation Services (DTS) package is migrated to SSIS.
Analysis Services Execute DDL Task	Runs XML for Analysis (XMLA) code against an SSAS database. XMLA is the data definition language (DDL) for SSAS; therefore, this task lets you perform common structure changes such as adding partitions to cubes.
Analysis Services Processing Task	Allows the processing of SSAS objects through an SSIS package.

Bulk Insert Task	Allows the execution of bulk copy operations for SQL Server. This task works only against a SQL Server Database Engine.
Data Flow Task	Allows data processing from sources to destinations. Lesson 3 in this chapter covers the data flow task in more detail.
Data Mining Query Task	Performs data mining queries and lets you capture the results for analysis.
Data Profiling Task	Allows the analysis of source data for patterns, missing data, candidate keys, and statistics. These results typically inform developers about what logic to include in their SSIS packages based on their data needs.
Execute DTS 2000 Package Task	Runs a DTS package within SSIS.
Execute Package Task	Runs other SSIS packages either deployed to SQL Server or in the file system.
Execute Process Task	Runs a command-line operation such as program or batch file execution.
Execute SQL Task	Runs SQL code against any underlying database connection in the SQL language of the connected database engine.
File System Task	Lets you copy, move, and delete files as well as perform other file and folder operations.
FTP Task	Sends and receives files between the file system and an FTP server and performs simple file and folder operations on the FTP server.
Message Queue Task	Integrates with Message Queuing (MSMQ) on a server running Windows to read and send messages.
Script Task	Runs Microsoft Visual Basic 2008 or Microsoft Visual C# 2008 within an SSIS package.
Send Mail Task	Sends an e-mail message through an SMTP server.
Transfer *[Object]* Task	Tasks that copy SQL Server objects from one system to another, including databases, SQL Server Agent jobs, error messages, logins, master stored procedures, and database-level objects.
Web Service Task	Lets you connect to a Web service to send or receive information.
WMI Data Reader Task	Lets you run a Windows Management Instrumentation (WMI) query against the operating system to capture server information.
WMI Event Watcher Task	Waits for a particular event before executing.
XML Task	Combines, queries, and differentiates multiple XML files on the server.

You might have noticed that there is also a list of Maintenance Plan Tasks for the control flow. These are primarily for database administrators (DBAs) who are managing SQL Server 2008 databases through the SSMS maintenance plan interface or DBAs who are creating packages in BIDS for database maintenance.

As you can see, SSIS features the ability to perform a host of different processing and integration operations. It is beyond the scope of this Training Kit and Exam 70-448 to discuss the design patterns for the components, but you will use several of these tasks in some lesson examples, and a couple of tasks are highlighted here.

Using Control Flow Containers

Your package must contain at least one task that performs a specific operation; however, most of the time, packages will have several tasks that coordinate with each other, and you need a way to organize those tasks. This is where a control flow container can help. A *control flow container* lets you group tasks together to control how tasks are run in parallel as well as ordering, logging, and transactions. Containers can also execute the tasks within them several times based on iterative requirements.

As with tasks, you find containers in the toolbox when you are working in the control flow. To use a container, you simply drag the container from the toolbox onto your control flow workspace. The screen in Figure 1-12 shows a package control flow containing a single container that holds several tasks.

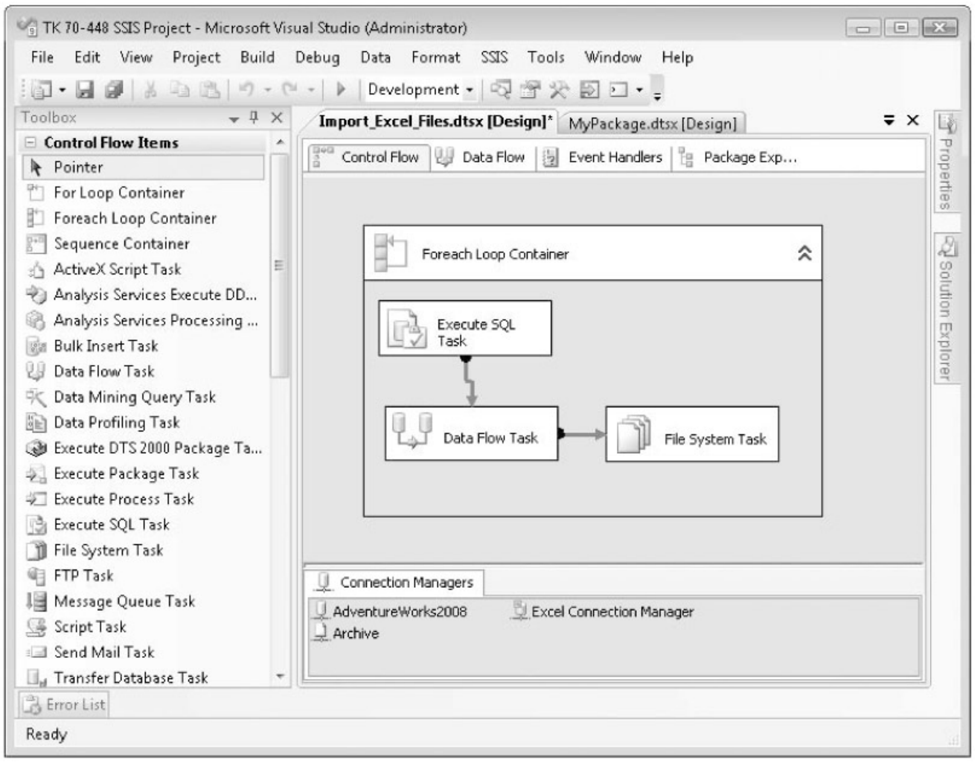


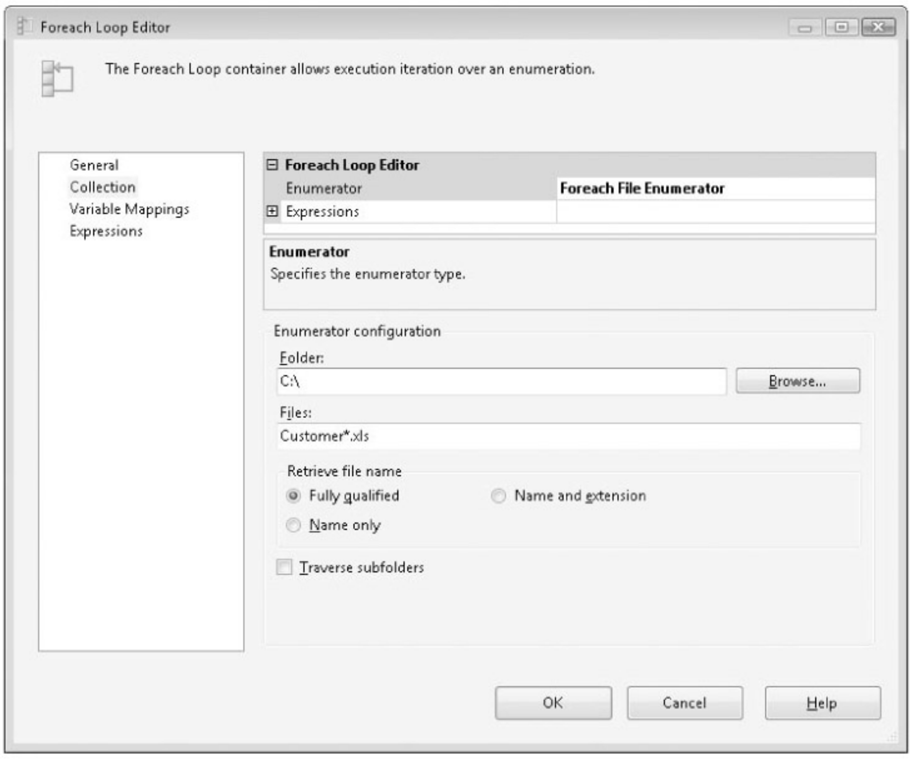
Figure 1-12: A control flow can include containers that group together tasks and subcontainers.

Additionally, you can drag task objects and other containers into your container.

There are three primary containers in SSIS: a Sequence Container, a For Loop Container, and a Foreach Loop Container.

- **Sequence Container** Lets you organize subordinate tasks by grouping them together, and lets you apply transactions or assign logging to the container.
- **For Loop Container** Provides the same functionality as the Sequence Container except that it also lets you run the tasks within it multiple times based on an evaluation condition, such as looping from 1 to 10.
- **Foreach Loop Container** Also allows looping, but instead of providing a condition expression, you loop over a set of objects, such as files in a folder.

Figure 1-13 shows the Foreach Loop Editor, which you open by double-clicking the container or by right-clicking the container and then clicking Edit.


Foreach Loop Editor

The Foreach Loop container allows execution iteration over an enumeration.

General
Collection
Variable Mappings
Expressions

Foreach Loop Editor	
Enumerator	Foreach File Enumerator
Expressions	

Enumerator
Specifies the enumerator type.

Enumerator configuration

Folder: C:\

Files: Customer*.xls

Retrieve file name: [✓ Fully qualified] Name and extension, Name only

Traverse subfolders: not selected

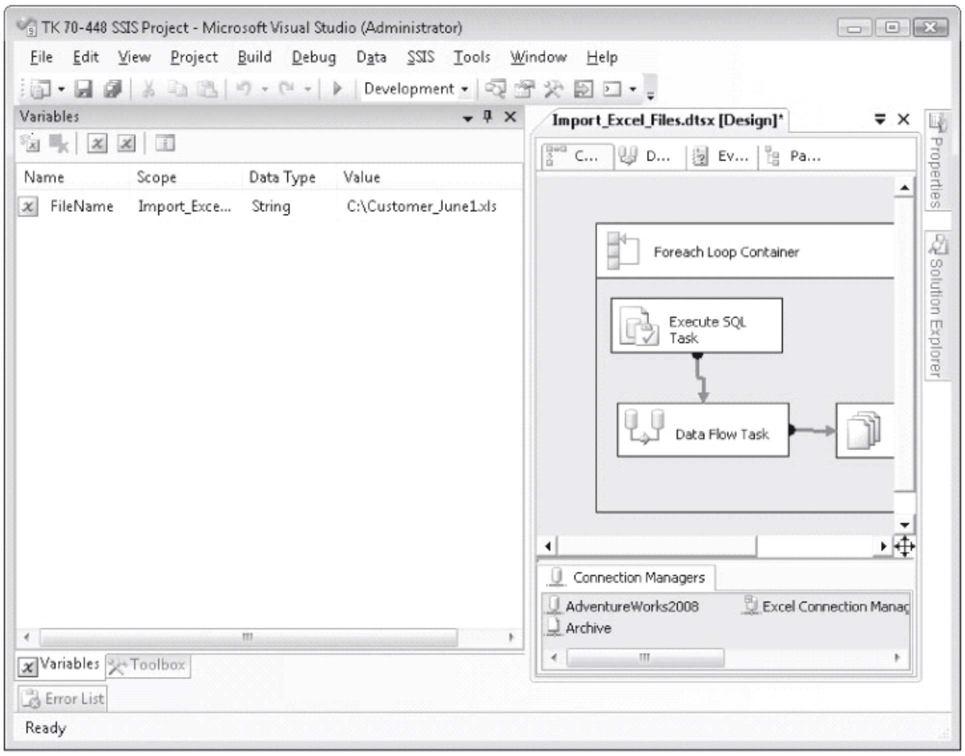
Figure 1-13: The Foreach Loop Editor lets you iterate over files in a folder and return the file names (one at a time) into a variable.

As described, the Foreach Loop Container can iterate over different types of objects, and the configuration choices let you specify the objects over which to loop and the detailed settings. Furthermore, the values of the enumerated objects can be put into user variables. For example, the Foreach Loop Container can iterate over files in a folder and return the file names into a variable.

Working with Package Variables

Variables within SSIS are a way to integrate objects by passing values between tasks and containers, accepting values from external sources, or building code dynamically that is then executed. You can also use variables for auditing and logging.

To work with variables within a package, choose Variables from the SSIS menu (when designing a package). Figure 1-14 shows the Variables window within BIDS.



Name	Scope	Data Type	Value
FileName	Import_Exce...	String	C:\Customer_June1.xls

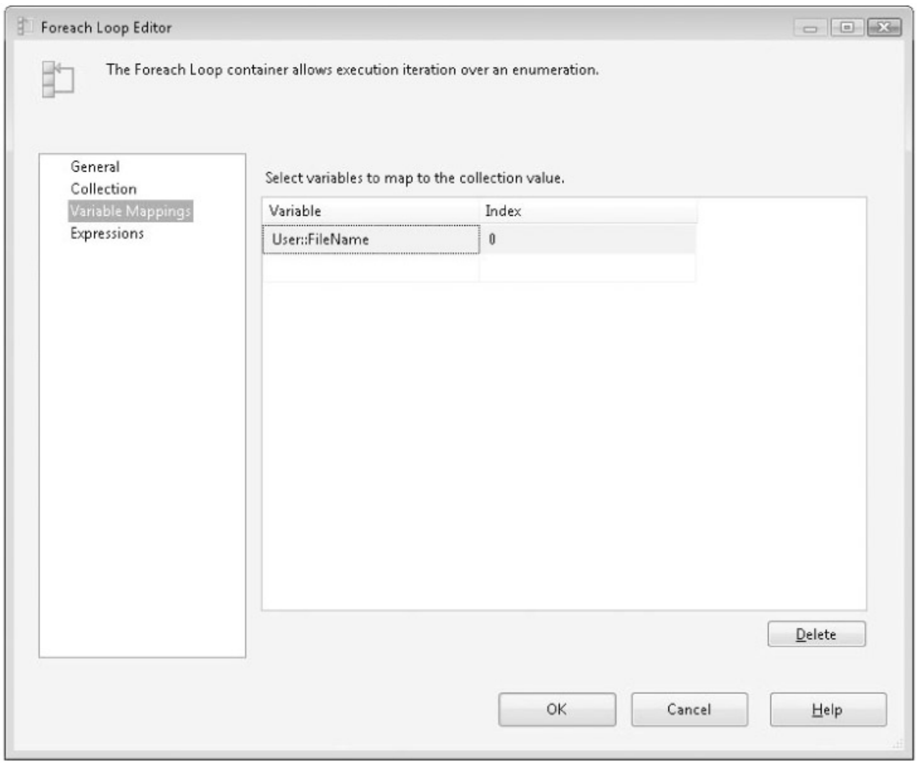
Figure 1-14: The Variables window lets you add, delete, or modify variables in a package.

At the top of the Variables window are buttons that let you create and delete variables as well as view other variables within a package. As Figure 1-14 shows, all SSIS variables are given a name, scope, data type, and value. The *scope* defines at what level within a package the variable is created. For example, if you select a Foreach Loop Container and then click the Add Variable button on the toolbar for the Variables window, the variable is scoped at that level. When no tasks or containers are selected, the variable is scoped at the entire package level.

Exam Tip Variables are viewable to a task or container only if the variables' scope is at the scope of the task or container in question, at the scope of a parent container level, or at the package scope itself. For example, if an Execute SQL Task has a variable directly assigned to it through the variable's scope, only the Execute SQL Task can see and use the variable. Other tasks or containers will not be able to reference the variable. On the other hand, if a Foreach Loop Container has a variable scoped to it, all the tasks within the Foreach Loop Container (including the container itself) can reference and use the variable. Variables are referenced as *User::[VariableName] or System::[VariableName]*.

Within SSIS, there are two types of variables: system variables and user variables.

- **System variables** System variables are not editable but can be referenced within tasks and containers. System variables are set by the package for tracking metadata such as the package name and the user that executes the package. To view all system variables, click Show System Variables (the button labeled with an X) on the Variables window toolbar.
- **User variables** You can create and define user variables for any purpose in the package. For example, the Foreach Loop Container updates user variables with values for every loop it iterates through. In Figure 1-13, shown earlier, file names for all files on the C drive that begin with the word *customer* will be mapped to a variable named *FileName*. Figure 1-15 shows the Variable Mapping tab in the Foreach Loop Editor.



The Foreach Loop container allows execution iteration over an enumeration.

Select variables to map to the collection value.

Variable	Index
User::FileName	0

Figure 1-15: The Variable Mapping tab in the Foreach Loop Editor allows the values that are looped through to update an assigned variable for each iteration in the loop.

Using the Script Task and Data Profiling Task

Although this Training Kit focuses on development and maintenance rather than design, it is worth highlighting a few key control flow tasks. The exam objective domain covers using code within a package as well as performing data profiling, so let's look at the Script Task and the Data Profiling Task.

Script Task

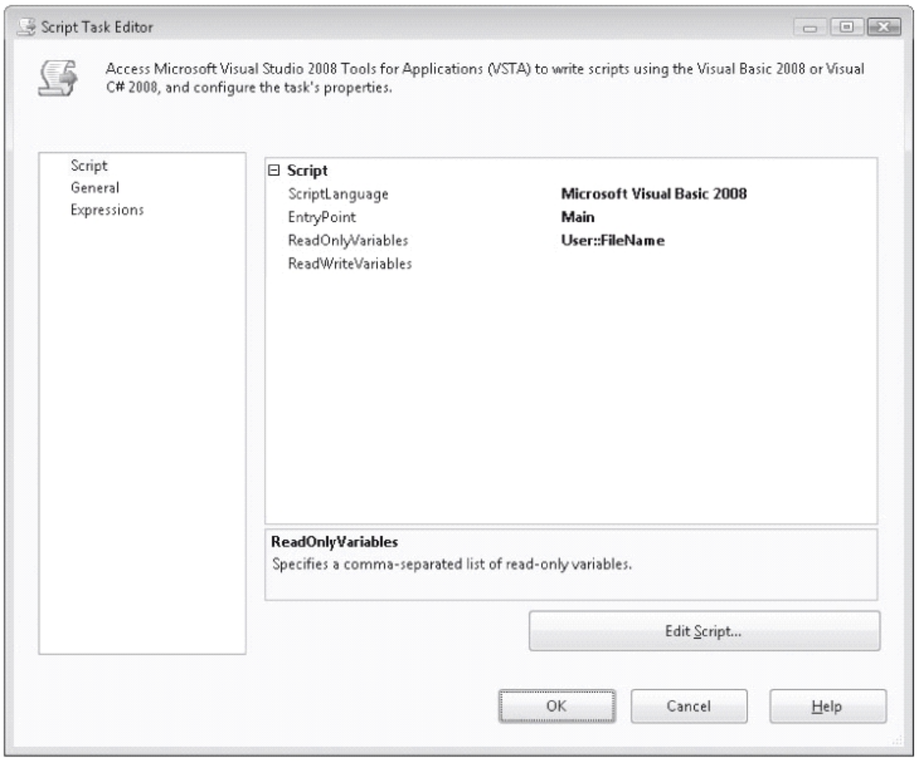
You use the Script Task within SSIS to execute VB.NET or C#.NET code. The Script Task has the following features:

- Uses the Visual Studio Tools for Applications 2.0 (VSTA) interface, which lets you run VB.NET and C#.NET code with the full host of methods and functions.
- Variables can be referenced and updated within a script.
- Connections can be referenced and updated within a script.
- SSIS breakpoints can be applied within the script's code (for the Script Task). Chapter 2 will discuss breakpoints.
- Runs in both a 32-bit environment (X86) and a 64-bit environment (X64).

Exam Tip If you want to reference SSIS variables within a Script Task, you need to include the variables in the ReadonlyVariables or ReadWriteVariables list, depending on whether you will be just accessing the variable for read purposes or updating the variable.

In the control flow example shown earlier in Figures 1-13, 1-14, and 1-15, the package is looping through Excel files and storing the Excel file path in a package variable. The package contains a connection to Excel that needs to be updated with the value of the variable, because each time the package loops, the variable needs to be updated.

Using a Script Task, you can update the Excel connection with the value of the variable. The first step is to drag a Script Task into the Foreach Loop Container from the toolbox. The script needs to be the first task that runs in the Foreach Loop Container, so place it before the Execute SQL Task and connect it to the Execute SQL Task with a precedence constraint. (Precedence constraints are covered in Chapter 2.) Figure 1-16 shows the Script Task Editor.


Script Task Editor
Access Microsoft Visual Studio 2008 Tools for Applications (VSTA) to write scripts using the Visual Basic 2008 or Visual C# 2008, and configure the task's properties.
Script
General
Expressions
Edit Script...
OK
Cancel
Help

Script
ScriptLanguage Microsoft Visual Basic 2008
EntryPoint Main
ReadOnlyVariables User::FileName
ReadWriteVariables

ReadOnlyVariables
Specifies a comma-separated list of read-only variables.

Figure 1-16: The Script Task Editor lets you select the programming language (VB.NET or C#.NET) as well as define any uses of variables within the script itself.

This example uses Microsoft Visual Basic 2008 as the *ScriptLanguage* and specifies the *User::FileName* variable in the *ReadOnlyVariables* property. To design the script, in the Script Task Editor, click Edit Script.

For this example, the script needs to update the Excel connection manager to point to the value of the variable, as the following code shows:

```
Dts.Connections("Excel Connection Manager").ConnectionString() = _
     "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + _
     Dts.Variables("FileName").Value.ToString() + _
     ";Extended Properties=""EXCEL 8.0;HDR=YES"";"
```

The reference to the connection begins with *Dts.Connections* and the reference to the variables begins with *Dts.Variables*.

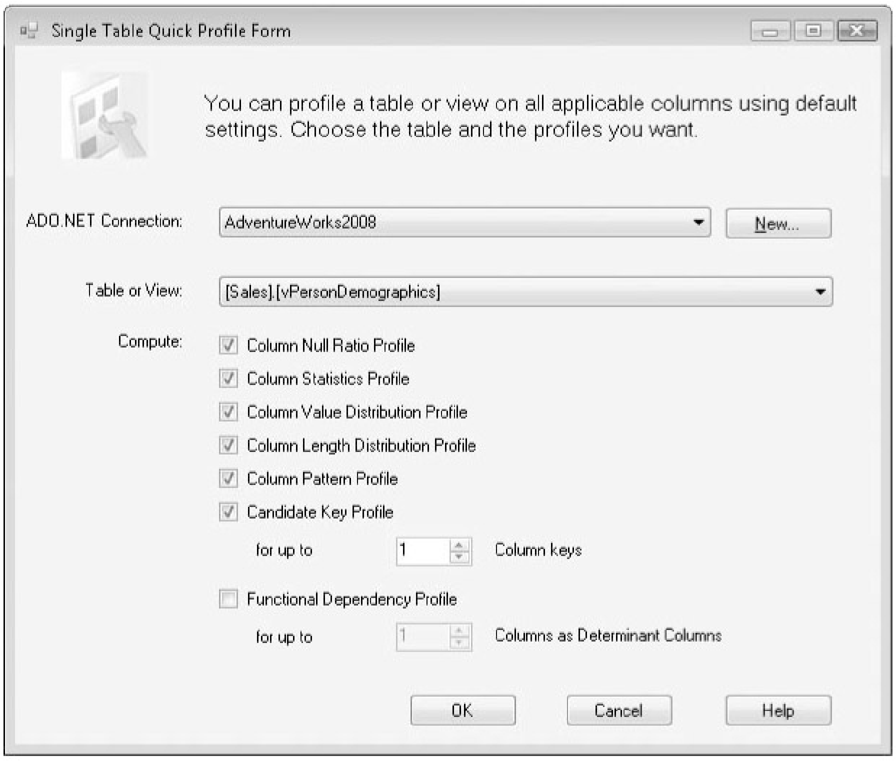
This code executes for every loop in the Foreach Loop Container and updates the Excel connection manager.

Data Profiling Task

You use the Data Profiling Task to review source data entities, to check the cleanliness and completeness of the data, and to understand how the data is organized structurally, such as possible key columns and relationships between columns.

The Data Profiling Task has two parts: the Data Profiling Task in the control flow that performs the analysis and the Data Profile Viewer to review the results.

To use the Data Profiling Task, first create an ADO.NET connection where the source tables or views reside. The Data Profiling Task requires an ADO.NET connection for sources. Next, drag the task from the toolbox onto the control flow, and then open the task to edit its properties. The easiest way to perform a data profile is to click the Quick Profile button in the Data Profiling Task Editor. Figure 1-17 shows the Single Table Quick Profile Form dialog box configured to run against the [Sales].[vPersonDemographics] view.


Single Table Quick Profile Form

You can profile a table or view on all applicable columns using default settings. Choose the table and the profiles you want.

ADO.NET Connection: AdventureWorks2008

Table or View: [Sales].[vPersonDemographics]

Compute:
- [✓] Column Null Ratio Profile
- [✓] Column Statistics Profile
- [✓] Column Value Distribution Profile
- [✓] Column Length Distribution Profile
- [✓] Column Pattern Profile
- [✓] Candidate Key Profile
  - for up to 1 Column keys
- [ ] Functional Dependency Profile
  - for up to 1 Columns as Determinant Columns

Figure 1-17: The Data Profiling Task can gather accuracy, completeness, and statistics information about the data within source tables or views.

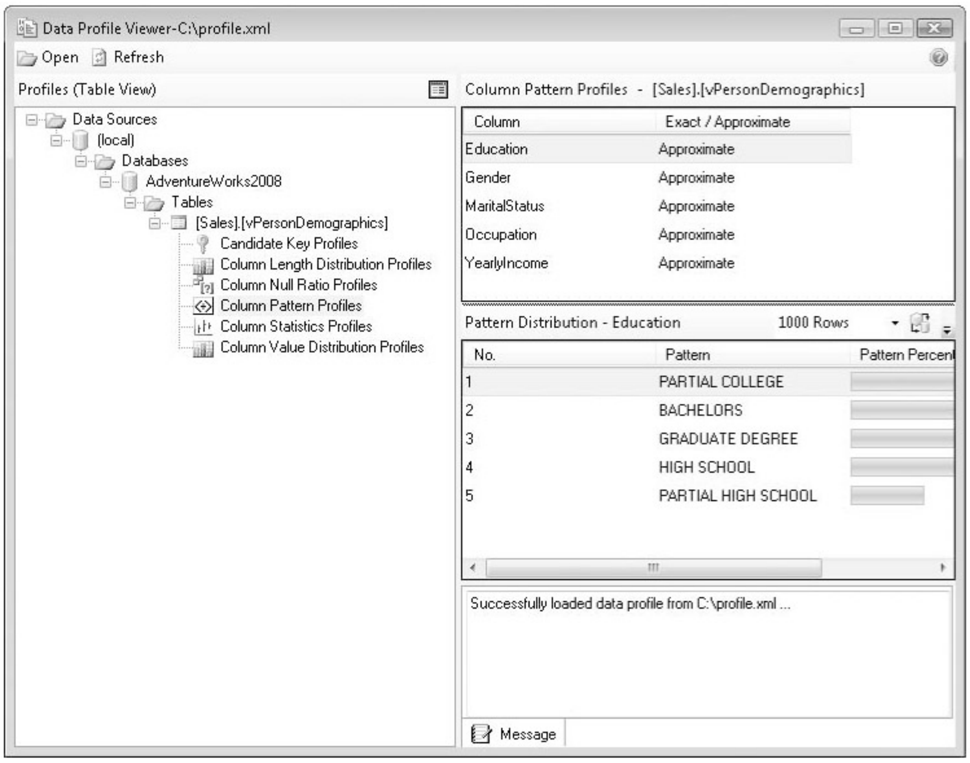
As you can see, the Data Profiling Task can analyze data in various ways, which Table 1-2 describes.

Table 1-2: Data Profiling Task Features

PROFILE	DESCRIPTION
Column Null Ratio Profile	Evaluates the column and returns the percent of NULLs in the column relative to the total number of rows in the table.
Column Statistics Profile	For numeric and datetime columns, returns the spread and averages of the values.
Column Value Distribution Profile	Identifies the uniqueness of the values in a column across all the rows for that column.
Column Length Distribution Profile	Shows the various value lengths for a text column and the percentage of all the rows that each length takes up.
Column Pattern Profile	Displays any patterns found in the column data and returns the regular expression pattern that matches the pattern.
Candidate Key Profile	Identifies one or more columns that are unique across all the rows; the percentage of uniqueness is shown.
Functional Dependency Profile	Lists any columns that have value dependencies on other columns within the table, where a value from one column is found only when the value of another column is distinct.

After you configure the Data Profiling Task through the Single Table Quick Profile Form dialog box, you need to specify the output XML file in the *Destination* property. This is where the task stores the profiling results.

To view the results, open the Data Profile Viewer. (Click Start and then select All Programs, Microsoft SQL Server 2008, Integration Services, Data Profile Viewer.) Click Open on the toolbar, and browse to the output XML file where the results are stored. Figure 1-18 shows the Data Profile Viewer.


Data Profile Viewer-C:\profile.xml
Open
Refresh
Profiles (Table View)
Data Sources
(local)
Databases
AdventureWorks2008
Tables
[Sales].[vPersonDemographics]
Candidate Key Profiles
Column Length Distribution Profiles
Column Null Ratio Profiles
Column Pattern Profiles
Column Statistics Profiles
Column Value Distribution Profiles
Column Pattern Profiles - [Sales].[vPersonDemographics]
Pattern Distribution - Education
1000 Rows
Message

Column	Exact / Approximate
Education	Approximate
Gender	Approximate
MaritalStatus	Approximate
Occupation	Approximate
YearlyIncome	Approximate

No.	Pattern	Pattern Percent
1	PARTIAL COLLEGE	
2	BACHELORS	
3	GRADUATE DEGREE	
4	HIGH SCHOOL	
5	PARTIAL HIGH SCHOOL	

Successfully loaded data profile from C:\profile.xml ...

Figure 1-18: The Data Profile Viewer displays the results of the Data Profiling Task in a graphical form that demonstrates each profile type.

The Data Profile Viewer displays each profile type. The left pane lets you navigate between the profile types and source tables that were profiled. The right pane displays the results.

Testing Package Execution in BIDS

During the development of a package, you need to test its execution to validate that your package and tasks are configured correctly.

You can execute a package from within BIDS in three primary ways:

- Choose Start Debugging from the Debug menu on the menu bar.
- Click Start Debugging (the button containing an arrow that resembles the Play button on a DVD player) on the Standard toolbar.
- Press F5 on your keyboard.

After a package is run in BIDS, a new tab named the Progress tab appears in the SSIS Designer. This tab shows the execution results and lets you troubleshoot any package errors you might find. The Progress tab is renamed as Execution Results when you are back in design view.

Practice—Creating and Editing a Control Flow Task

The following exercises will familiarize you with creating and editing a control flow task and executing the package within the design environment.

EXERCISE 1 Create a Control Flow Task and Test Package Execution

In this exercise, you will work with control flow tasks and execute packages in the SSIS Designer.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project *TK 70-448 SSIS Project* you created in Lesson 1, "Creating SSIS Packages and Data Sources," or open the completed exercise file from the companion CD, and then edit the package named MyPackage.dtsx (right-click the package in Solution Explorer, and then click Open).

2. Open the Toolbox window by selecting Toolbox from the View menu, locate the Execute SQL Task item, and drag it to the control flow workspace of your package.

3. Edit the Execute SQL Task object by double-clicking the task icon or by right-clicking the task icon and then clicking Edit.

4. Change the Connection property to use the AdventureWorks2008 connection.

5. In the SQL Statement property of the Execute SQL Task Editor dialog box, type the following code:

```
UPDATE Production.Product
SET ProductLine = 's'
WHERE ProductLine IS NULL
```

6. Click OK in the Execute SQL Task dialog box to return to the SSIS Designer. Right-click the Execute SQL Task, click Rename, and type **update ProductLine**.

7. Next, drag a Sequence Container object from the toolbox onto the control flow workspace.

8. Drag the Update ProductLine Execute SQL Task you just created into the Sequence Container so that the task is nested in the Sequence Container box.

9. To test the execution of the package, click Start Debugging on the Standard toolbar or choose Start Debugging from the Debug menu.

10. When the package execution is complete, your Sequence Container and Execute SQL Task should be green.

11. Click the Execution Results tab (named Progress while the package is executing) in the SSIS Designer to view the execution details.

12. Select the Stop button from the tool menu to stop the debugger (or choose Debug, Stop Debugging from the Debug menu).

13. Click the Save All button on the BIDS toolbar.

EXERCISE 2 Modify the DimCustomer ETL Package Control Flow

In this exercise, you will start the process of building the DimCustomer SSIS package that will handle the ETL process from the AdventureWorks2008 database to the AdventureWorksDW2008 database.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project *TK 70-448 SSIS Project* you created in Lesson 1, "Creating SSIS Packages and Data Sources," or open the completed exercise file from the companion CD, and then open the empty DimCustomer package.

2. From the toolbox, drag two Execute SQL Tasks onto the control flow workspace and then drag one Data Flow Task onto the workspace.

3. Next, connect the first Execute SQL Task to the Data Flow Task by dragging the green precedence constraint from the Execute SQL Task onto the Data Flow Task. Then connect the green precedence constraint from the Data Flow Task to the second Execute SQL Task.

4. Rename the first Execute SQL Task to **Truncate update Table**, and rename the second Execute SQL Task to **Batch updates**. Figure 1-19 shows what your resulting control flow should look like.

5. Before editing the tasks in SSIS, open SSMS, connect to the Database Engine, and create a new query against the AdventureWorksDW2008 database. Execute the following code, which you can find in the CreateCustomerUpdateTable.sql file in the ..\Source\Ch 01\ folder of the practice exercise files.

```
USE AdventureWorksDW2008
GO

CREATE TABLE [dbo].[stgDimCustomerUpdates](
  [CustomerAlternateKey] [nvarchar](15) NULL,
  [AddressLine1] [nvarchar](60) NULL,
  [AddressLine2] [nvarchar](60) NULL,
  [BirthDate] [datetime] NULL,
  [CommuteDistance] [nvarchar](15) NULL,
  [DateFirstPurchase] [datetime] NULL,
  [EmailAddress] [nvarchar](50) NULL,
  [EnglishEducation] [nvarchar](40) NULL,
  [EnglishOccupation] [nvarchar](100) NULL,
  [FirstName] [nvarchar](50) NULL,
  [Gender] [nvarchar](1) NULL,
  [GeographyKey] [int] NULL,
  [HouseOwnerFlag] [nvarchar](1) NULL,
  [LastName] [nvarchar](50) NULL,
  [MaritalStatus] [nvarchar](1) NULL,
  [MiddleName] [nvarchar](50) NULL,
  [NumberCarsOwned] [tinyint] NULL,
  [NumberChildrenAtHome] [tinyint] NULL,
  [Phone] [nvarchar](25) NULL,
  [Suffix] [nvarchar](10) NULL,
  [Title] [nvarchar](8) NULL,
  [TotalChildren] [tinyint] NULL,
  [YearlyIncome] [nvarchar](100) NULL) ON [PRIMARY]
```

6. After you have successfully created the table, switch back to the DimCustomer SSIS package and edit the Execute SQL Task named Truncate Update Table.

7. In the Execute SQL Task Editor dialog box, set the Connection property to AdventureWorksDW2008, and then enter the following SQL code in the SQLStatement property before clicking OK to save it:

   ```
   TRUNCATE TABLE dbo.stgDimCustomerUpdates
   ```

8. Edit the last Execute SQL Task, named Batch Updates, and set the Connection property to AdventureWorksDW2008.

9. In the SQLStatement property, enter the following UPDATE statement. (You can find this statement in the UpdateCustomerTable.sql file in the ..\Source\Ch 01\practice exercise folder.)

   ```
   UPDATE dbo.DimCustomer
      SET AddressLine1 = stgDimCustomerUpdates.AddressLine1
        , AddressLine2 = stgDimCustomerUpdates.AddressLine2
        , BirthDate = stgDimCustomerUpdates.BirthDate
        , CommuteDistance = stgDimCustomerUpdates.CommuteDistance
        , DateFirstPurchase = stgDimCustomerUpdates.DateFirstPurchase
        , EmailAddress = stgDimCustomerUpdates.EmailAddress
        , EnglishEducation = stgDimCustomerUpdates.EnglishEducation
        , EnglishOccupation = stgDimCustomerUpdates.EnglishOccupation
        , FirstName = stgDimCustomerUpdates.FirstName
        , Gender = stgDimCustomerUpdates.Gender
        , GeographyKey = stgDimCustomerUpdates.GeographyKey
        , HouseOwnerFlag = stgDimCustomerUpdates.HouseOwnerFlag
        , LastName = stgDimCustomerUpdates.LastName
        , MaritalStatus = stgDimCustomerUpdates.MaritalStatus
        , MiddleName = stgDimCustomerUpdates.MiddleName
        , NumberCarsOwned = stgDimCustomerUpdates.NumberCarsOwned
        , NumberChildrenAtHome = stgDimCustomerUpdates.NumberChildrenAtHome
        , Phone = stgDimCustomerUpdates.Phone
        , Suffix = stgDimCustomerUpdates.Suffix
        , Title = stgDimCustomerUpdates.Title
        , TotalChildren = stgDimCustomerUpdates.TotalChildren
     FROM dbo.DimCustomer DimCustomer
    INNER JOIN dbo.stgDimCustomerUpdates
       ON DimCustomer.CustomerAlternateKey
         = stgDimCustomerUpdates.CustomerAlternateKey
   ```

10. Click OK in the Execute SQL Task Editor dialog box, and then save the package. In the next lesson, you will complete the data flow portion of this package and then test the execution.

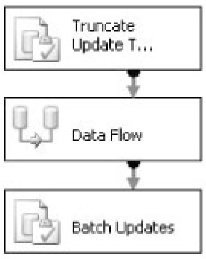


Figure 1-19: Your control flow for the DimCustomer package should contain an Execute SQL Task, followed by a Data Flow Task, followed by another Execute SQL Task.

Quick Check

1. What is the difference between a control flow task and a control flow container? ?
2. To run a stored procedure within a SQL Server database, what task would you choose? ?

Answers

1. Control flow tasks perform operations, whereas containers coordinate and group tasks. For example, a Foreach Loop Container can look through the files in a system folder, and a File System Task embedded within the container can then move the files to a new folder location.
2. The Execute SQL Task can run a stored procedure within SQL Server or any relational database for which you have an installed data provider. The syntax of the statement entered in the Execute SQL Task will be in the native language of the underlying database.

Lesson 3: Using Data Flow Adapters and Transformations

Estimated Lesson Time: 45 Minutes

Lesson 2, "Creating and Editing Control Flow Objects," showed how to use control flow tasks and containers. One of the most valuable control flow tasks is the Data Flow Task. A package can have zero, one, or more data flows. To work with the Data Flow Task, you can either drag a Data Flow Task from the Control Flow toolbox onto the workspace and then double-click it, or you can click the Data Flow tab within the SSIS Designer. After clicking the Data Flow tab, you see the Data Flow Designer, where you can use the data flow to handle and transform datasets. The Data Flow Task has three types of objects in the toolbox:

- Data flow source adapters
- Data flow transformations
- Data flow destination adapters

Figure 1-20 shows the Data Flow tab with the toolbox open, highlighting the data flow sources and some of the data flow transformations. Notice the difference between the Control Flow toolbox items and the Data Flow toolbox items.

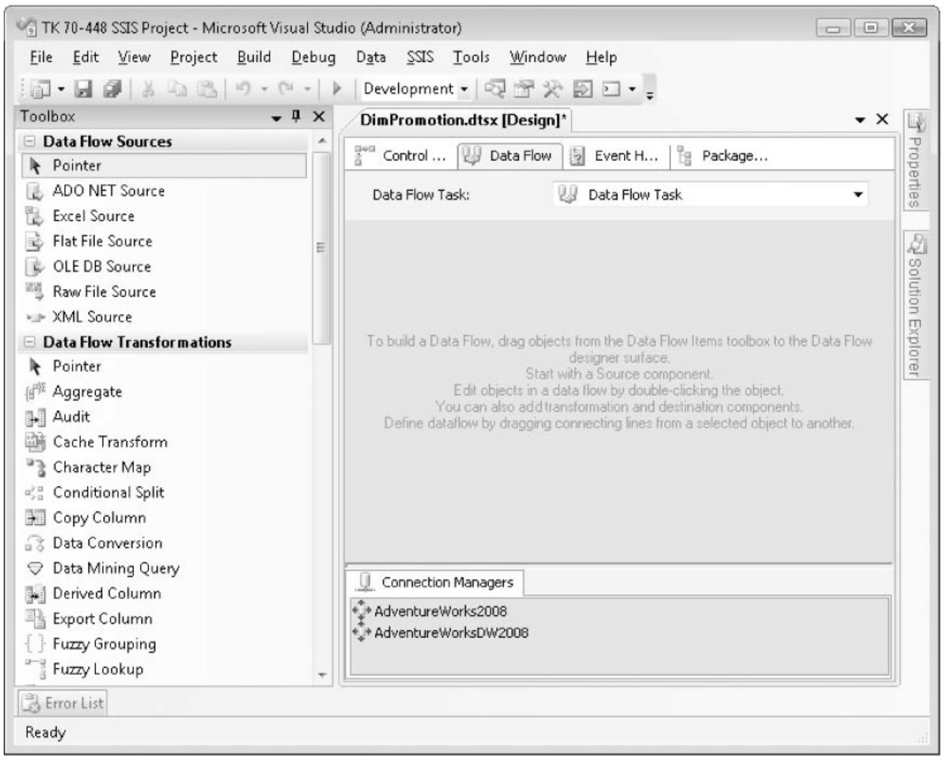

TK 70-448 SSIS Project - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Data SSIS Tools Window Help
Development
Toolbox
Data Flow Sources
Pointer
ADO NET Source
Excel Source
Flat File Source
OLE DB Source
Raw File Source
XML Source
Data Flow Transformations
Pointer
Aggregate
Audit
Cache Transform
Character Map
Conditional Split
Copy Column
Data Conversion
Data Mining Query
Derived Column
Export Column
Fuzzy Grouping
Fuzzy Lookup
DimPromotion.dtsx [Design]*
Control ...
Data Flow
Event H...
Package...
Data Flow Task:
Data Flow Task
To build a Data Flow, drag objects from the Data Flow Items toolbox to the Data Flow designer surface.
Start with a Source component.
Edit objects in a data flow by double-clicking the object.
You can also add transformation and destination components.
Define dataflow by dragging connecting lines from a selected object to another.
Connection Managers
AdventureWorks2008
AdventureWorksDW2008
Properties
Solution Explorer
Error List
Ready

Figure 1-20: When you are working in the data flow, the toolbox shows items related to data flow development, including data flow sources, data flow transformations, and data flow destinations.

In this lesson, you will look at the details of the source and destination adapters as well as the transformations.

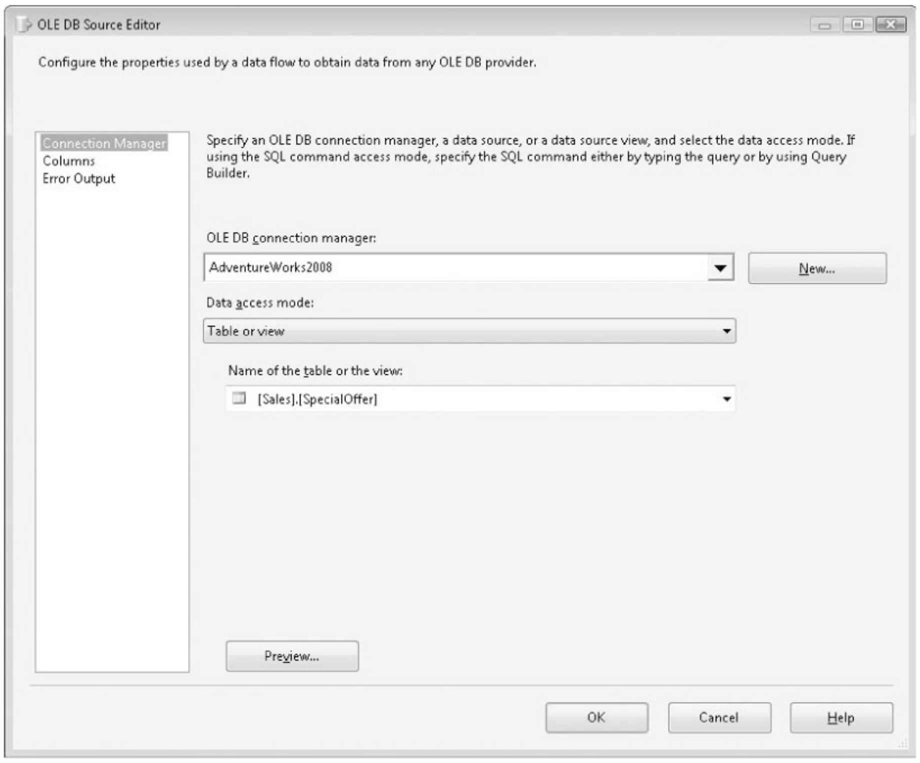
Defining Data Flow Source Adapters

Data flow source adapters use package connections, which point to the server instance or file location of the data source. (The only exception is the raw file adapter, which does not use a package connection.) A source adapter extracts data from sources and moves it into the data flow, where it will be modified and sent to a destination. You create data flow source adapters by dragging a source adapter from the Data Flow toolbox onto the Data Flow tab in the SSIS Designer. Table 1-3 describes the different data flow sources and their uses.

Table 1-3: Data Flow Sources and Their Uses

DATA FLOW SOURCE	PURPOSE
ADO.NET Source	Provides connections to tables or queries through an ADO.NET provider.
Excel Source	Allows extractions from an Excel worksheet defined in an Excel file.
Flat File Source	Connects to a delimited or fixed-width file created with different code pages.
OLE DB Source	Connects to installed OLE DB providers, such as SQL Server, Access, SSAS, and Oracle.
Raw File Source	Stores native SSIS data in a binary file type useful for data staging.
XML Source	Allows raw data to be extracted from an XML file; requires an XML schema to define data associations.

As an example, Figure 1-21 shows the OLE DB Source Editor dialog box for a package that is pulling special offer sales data from the AdventureWorks2008 database with the intention of loading it into the DimPromotions table in AdventureWorksDW2008.



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

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

OLE DB connection manager: AdventureWorks2008

Data access mode: Table or view

Name of the table or the view: [Sales].[SpecialOffer]

Figure 1-21: The OLE DB Source Editor displays that the DimPromotions package is configured to select data from the [Sales].[SpecialOffer] table.

The OLE DB source adapter is similar to the ADO.NET adapter in that a connection manager is required and either a table needs to be selected or a query needs to be written. In this example, the Data Access Mode drop-down list is set to Table Or View, and the [Sales]. [SpecialOffer] table is selected. If the Data Access Mode was set to SQL Command, you could enter a query for the source.

Creating Data Flow Destinations

Data flow destinations are similar to sources in that they use package connections. However, destinations are the endpoints in a package, defining the location to which the data should be pushed. For example, if you are sending data to an Excel file from a database table, your destination will be an Excel Destination adapter.

Exam Tip Many SSIS objects have a *ValidateExternalMetadata* property that you can set to *False* if the object being referenced (such as a table) does not exist when the package is being designed. This property is most commonly used for source or destination adapters, such as when a destination table is created during package execution.

All the source adapters (except the Data Reader source) have matching destination adapters in the SSIS data flow. And there are other destination adapters that let you send data to even more destinations. Table 1-4 lists the destination adapters in the SSIS data flow.

Table 1-4: Data Flow Destinations and Their Uses

DATA FLOW DESTINATION	PURPOSE
ADO.NET Destination	Allows insertion of data by using an ADO.NET managed provider.
Data Mining Model Training	Lets you pass data from the data flow into a data mining model in SSAS.
DataReader Destination	Lets you put data in an ADO.NET recordset that can be programmatically referenced.
Dimension Processing	Lets SSAS dimensions be processed directly from data flowing through the data flow.
Excel Destination	Used for inserting data into Excel, including Excel 2007.
Flat File Destination	Allows insertion of data to a flat file such as a comma-delimited or tab-delimited file.
OLE DB Destination	Uses the OLE DB provider to insert rows into a destination system that allows an OLE DB connection.

Partition Processing	Allows SSAS partitions to be processed directly from data flowing through the data flow.
Raw File Destination	Stores native SSIS data in a binary file type useful for data staging.
Recordset Destination	Takes the data flow data and creates a recordset in a package variable of type *object*.
SQL Server Compact Destination	Lets you send data to a mobile device running SQL Mobile.
SQL Server Destination	Provides a high-speed destination specific to SQL Server 2008 if the package is running on SQL Server.

Exam Tip You can configure the OLE DB Destination adapter to insert data from the data flow through bulk batches of data, instead of one row at a time. To use this destination-optimization technique, edit the OLE DB Destination and set the Data Access Mode to Table or View—Fast Load. When the OLE DB Destination is not configured with fast load, only one row at a time will be inserted into the destination table.

Figure 1-22 shows a simple data flow with one source and one destination. The data flow extracts records from the AdventureWorks2008 SpecialOffers table and inserts them into the AdventureWorksDW2008 DimPromotions table.

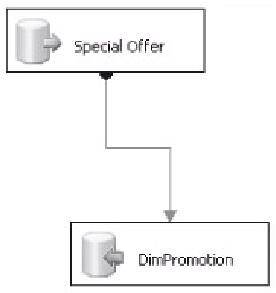
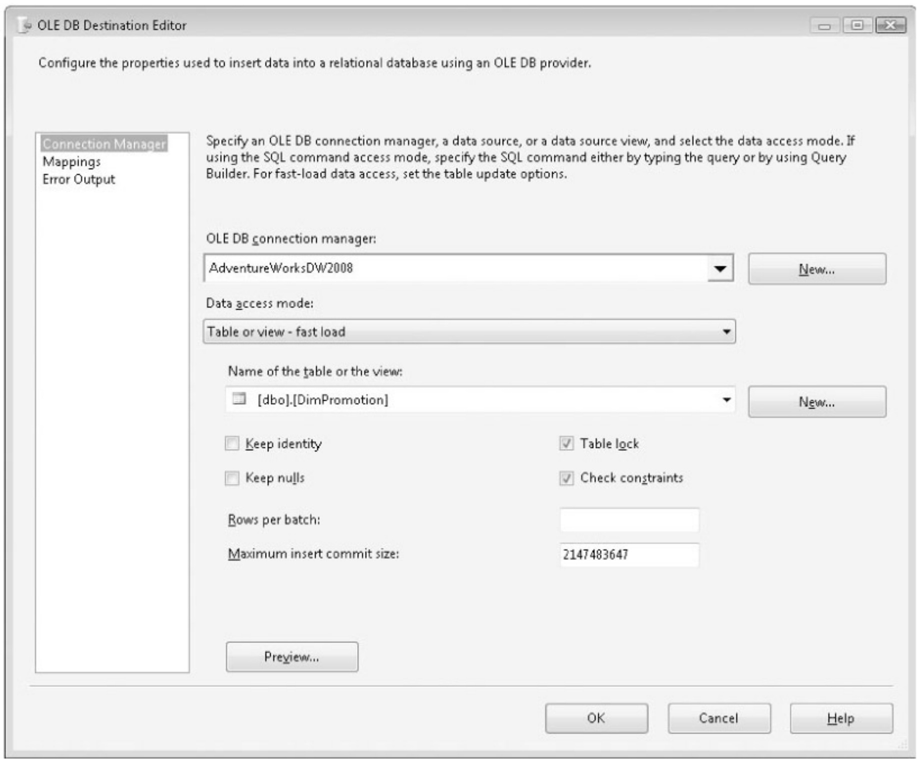


Figure 1-22: This simple data flow shows data being extracted from a source and inserted into a destination.

Like the source, the destination adapter requires configuration, both in the connection and table that the rows should be inserted into as well as in mapping the data flow columns to the destination table columns. Figure 1-23 shows the OLE DB Destination Editor for the preceding example.



Configure the properties used to insert data into a relational database using an OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder. For fast-load data access, set the table update options.

OLE DB connection manager: AdventureWorksDW2008

Data access mode: Table or view - fast load

Name of the table or the view: [dbo].[DimPromotion]

Keep identity [ ]   Table lock [✓]

Keep nulls [ ]   Check constraints [✓]

Rows per batch:

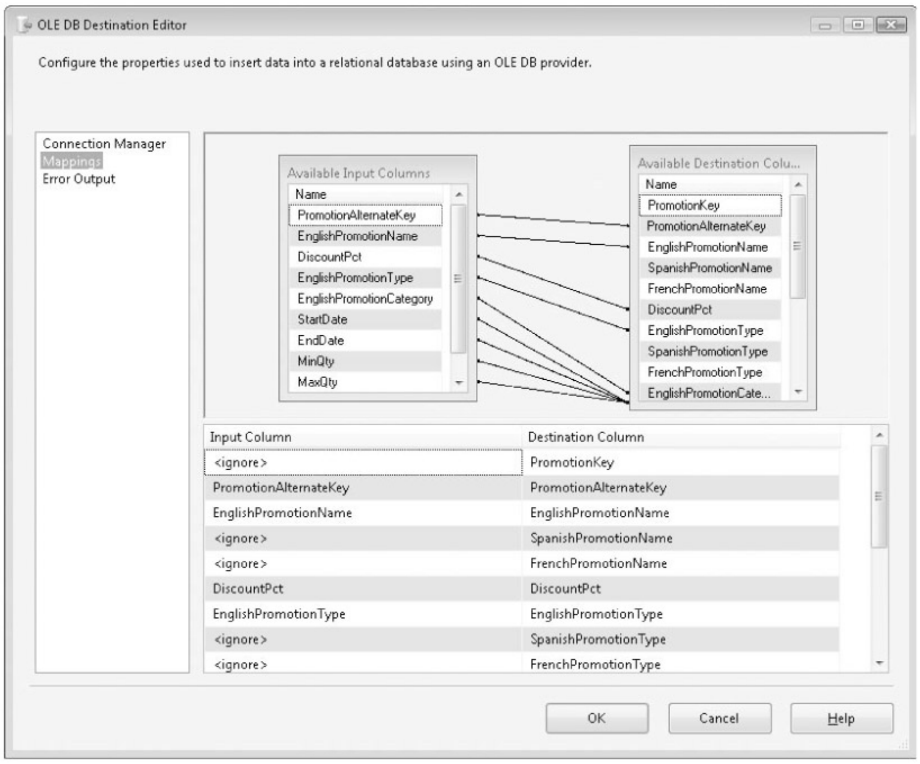
Maximum insert commit size: 2147483647

Figure 1-23: The destination adapter for the DimPromotions table is configured to connect to the

AdventureWorksDW2008 database and insert rows into the DimPromotions table by using the fast-load feature.

Notice that this OLE DB Destination uses the AdventureWorksDW2008 connection and is configured by default to use the Table Or View—Fast Load option of the Data Access Mode drop-down list. This means that records will be processed with bulk insert statements rather than one row at a time.

Figure 1-24 shows the Mappings tab of the same OLE DB Destination Editor. This is where you map columns available from the data flow to the destination columns in the destination adapter. All the destination adapters have a Mappings tab.



Configure the properties used to insert data into a relational database using an OLE DB provider.

Available Input Columns

- Name
- PromotionAlternateKey
- EnglishPromotionName
- DiscountPct
- EnglishPromotionType
- EnglishPromotionCategory
- StartDate
- EndDate
- MinQty
- MaxQty

Available Destination Colu...

- Name
- PromotionKey
- PromotionAlternateKey
- EnglishPromotionName
- SpanishPromotionName
- FrenchPromotionName
- DiscountPct
- EnglishPromotionType
- SpanishPromotionType
- FrenchPromotionType
- EnglishPromotionCate...

Input Column	Destination Column
<ignore>	PromotionKey
PromotionAlternateKey	PromotionAlternateKey
EnglishPromotionName	EnglishPromotionName
<ignore>	SpanishPromotionName
<ignore>	FrenchPromotionName
DiscountPct	DiscountPct
EnglishPromotionType	EnglishPromotionType
<ignore>	SpanishPromotionType
<ignore>	FrenchPromotionType

Figure 1-24: Each destination adapter requires you to map data from the data flow input columns to the destination columns.

Notice that not all columns are mapped. However, if one of the unmapped destination columns is marked as NOT NULL, the destination fails the package when it is run. In the section titled "Using Transformations" later in this lesson, you see how to use the Slowly Changing Dimension Transformation to handle new records and updates.

Working with Data Flow Transformations

Transformations give you the ability to modify and manipulate data in the data flow. A transformation performs an operation either on one row of data at a time or on several rows of data at once.

As with the source and destination adapters, you drag transformations from the Data Flow toolbox onto the Data Flow tab of the SSIS Designer, and edit them by right-clicking the transformation you want to change and then clicking Edit. You connect sources, transformations, and destinations through data paths, which you create by dragging the output arrow onto another component in the data flow. The green data path arrows are for rows that are successfully transformed, and the red output path arrows are for rows that failed the transformation because of an error, such as a truncation or conversion error. Figure 1-25 shows a data flow that connects a source to several transformations through data paths and onto a destination.

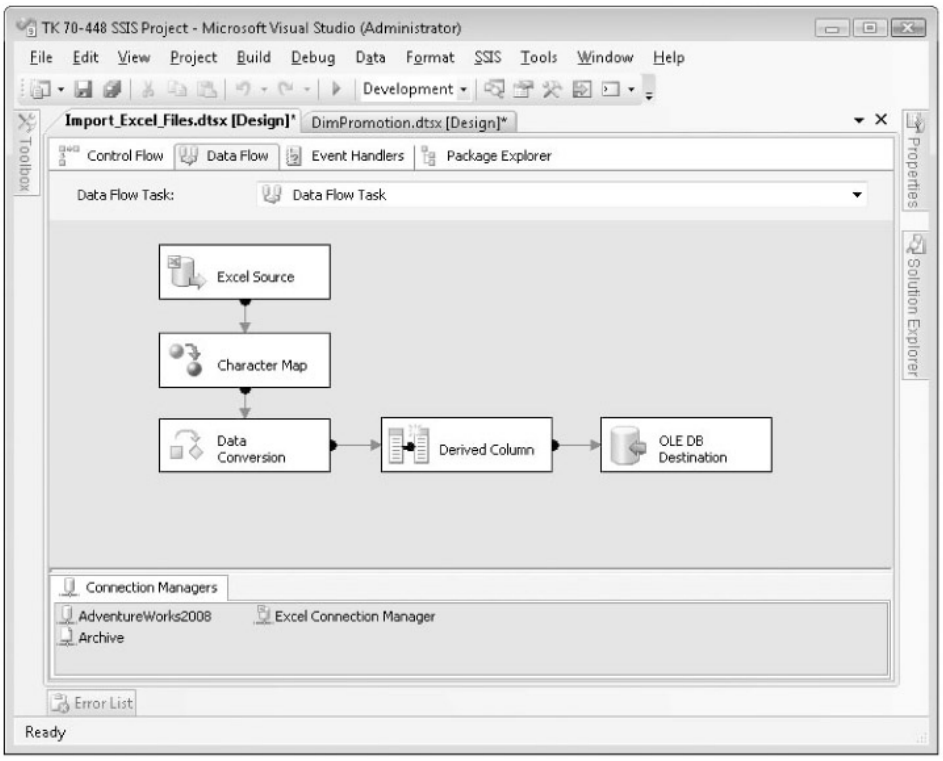


Figure 1-25: You use data paths to connect data flow transformations with sources, destinations, and other transformations.

Selecting Transformations

Transformations perform a wide variety of operations on the underlying data, and the transformation you choose depends on your data processing requirements. Some transformations operate similarly to other transformations; therefore, we can categorize them into natural groupings of like components.

Logical Row-Level Transformations

The most common and easily configured transformations perform operations on rows without needing other rows from the source. These transformations, which logically work at the row level, often perform very well. Table 1-5 lists the logical row-level transformations.

Table 1-5: Logical Row-Level Transformations

DATA FLOW TRANSFORMATION	PURPOSE
Audit	Adds additional columns to each row based on system package variables such as *ExecutionStartTime* and *PackageName*.
Cache Transform	Allows data that will be used in a Lookup Transformation to be cached and available for multiple Lookup components.
Character Map	Performs common text operations, such as Uppercase, and allows advanced linguistic bit conversion operations.
Copy Column	Duplicates column values in each row to new named columns.
Data Conversion	Creates new columns in each row based on new data types converted from other columns—for example, converting text to numeric.
Derived Column	Uses the SSIS Expression language to perform in-place calculations on existing values; alternatively, allows the addition of new columns based on expressions and calculations from other columns and variables.
Export Column	Exports binary large object (BLOB) columns, one row at a time, to a file.
Import Column	Loads binary files such as images into the pipeline; intended for a BLOB data type destination.
Row Count	Tracks the number of rows that flow through the transformation and stores the number in a package variable after the final row.

Some common uses of these transformations include performing mathematical calculations, converting a text value to a numeric or decimal data type, and replacing NULLs with other values. Because the Import Column and Export Column Transformations work with large binary data types, these two transformations carry an increased workload.

Multi-Input or Multi-Output Transformations

Multi-input and multi-output transformations can work with more than one data input or can generate more than one output, respectively. These transformations provide the capability to combine or branch data and give the data flow the overall ability to process data from one or more sources to one or more destinations. Table 1-6 lists the multi-input and multi-output transformations.

Table 1-6: Multi-Input and Multi-Output Transformations

DATA FLOW TRANSFORMATION	PURPOSE
Conditional Split	Routes or filters data based on Boolean expressions to one or more outputs, from which each row can be sent out only one output path.
Lookup	Allows matching between pipeline column values to external database tables; additional columns can be added to the data flow from the external table.
Merge	Combines the rows of two similar sorted inputs, one on top of the other, based on a defined sort key.
Merge Join	Joins the rows of two sorted inputs based on a defined join column(s), adding columns from each source.
Multicast	Generates one or more identical outputs, from which every row is sent out every output.
Union All	Combines one or more similar inputs, stacking rows one on top of another, based on matching columns.

As Table 1-6 describes, the Merge and Merge Join Transformations require sorted inputs. When these components are used in a data flow, the transformation waits for rows from either input, based on the defined sort order, to preserve the sorted output or match across the sorted rows; this means that rows might not immediately be sent out the output path.

Exam Tip When trying to determine which transformation to use that brings more than one data source together, remember that the Merge Join Transformation brings two sorted sources together and matching rows together with either an Inner Join, a Full outer Join, or a Left outer Join. Merge Join can match more than one row across the join columns. This behavior is different from that of the Lookup Transformation, which brings back only a single match across the join columns of the Lookup table.

The Union All Transformation does not join rows together but rather brings each row separately from the sources, stacking the rows together. The number of rows in the output of Union All is the combined row counts of all the inputs. The Merge Transformation is similar to Union All, except that the sources have to be sorted and the sort position is preserved.

Multi-Row Transformations

Some transformations perform work based on criteria from multiple input rows or generate multiple output rows from a single input row. These transformations can be more intensive in operation and memory overhead, but they provide valuable functions to meet business requirements. Table 1-7 lists the multi-row transformations.

Table 1-7: Multi-Row Transformations

DATA FLOW TRANSFORMATION	PURPOSE
Aggregate	Associates records based on defined groupings and generates aggregations such as *SUM, MAX, MIN*, and *COUNT*.
Percent Sampling	Filters the input rows by allowing only a defined percent to be passed to the output path.
Pivot	Takes multiple input rows and pivots the rows to generate an output with more columns based on the original row values.
Row Sampling	Outputs a fixed number of rows, sampling the data from the entire input, no matter how much larger than the defined output the input is.
Sort	Orders the input based on defined sort columns and sort direction and allows the removal of

	duplicates across the sort columns.
Unpivot	Takes a single row and outputs multiple rows, moving column values to the new row based on defined columns.

In the cases of the Sort, Aggregate, and Row Sampling Transformations, all the input rows are blocked, allowing the transformations to perform the work before sending rows down the output path. These transformations often require more server resources, memory, and processor capacity than do other transformations.

Advanced Data-Preparation Transformations

The final grouping of transformations lets you perform advanced operations on rows in the data flow pipeline. Table 1-8 lists these advanced data-preparation transformations.

Table 1-8: Advanced Data-Preparation Transformations

DATA FLOW TRANSFORMATION	PURPOSE
OLE DB Command	Performs database operations such as updates and deletes, one row at a time, based on mapped parameters from input rows.
Slowly Changing Dimension	Processes dimension changes, including tracking dimension history and updating dimension values. The Slowly Changing Dimension Transformation handles these common dimension change types: Historical Attributes, Fixed Attributes, and Changing Attributes.
Data Mining Query	Applies input rows against a data mining model for prediction.
Fuzzy Grouping	Associates column values with a set of rows based on similarity, for data cleansing.
Fuzzy Lookup	Joins a data flow input to a reference table based on column similarity. The Similarity Threshold setting specifies the closeness of allowed matches—a high setting means that matching values are closer in similarity.
Script Component	Provides VB.NET scripting capabilities against rows, columns, inputs, and outputs in the data flow pipeline.
Term Extraction	Analyzes text input columns for English nouns and noun phrases.
Term Lookup	Analyzes text input columns against a user-defined set of words for association.

Real World

Erik Veerman

Most customers I've worked with have numerous systems that host all kinds of data—from sales transaction systems, to human resources, to custom business apps. Many of these systems even run on different database platforms such as SQL Server, Oracle, DB2, or legacy systems that aren't even sold any more. The complexity of data within organizations extends beyond just the enterprise relational database, often including Excel files and departmental Access applications. Navigating through these systems can be difficult, to say the least.

SSIS delivers real benefits in these situations because it helps you efficiently consolidate data.

In one customer engagement I was involved with, our task was to simplify a complicated process that pulled in five different data sources. Two of them were in SQL Server, one was in Oracle, another was in Excel, and the last was a large binary flat file created from an IBM AS/400 system.

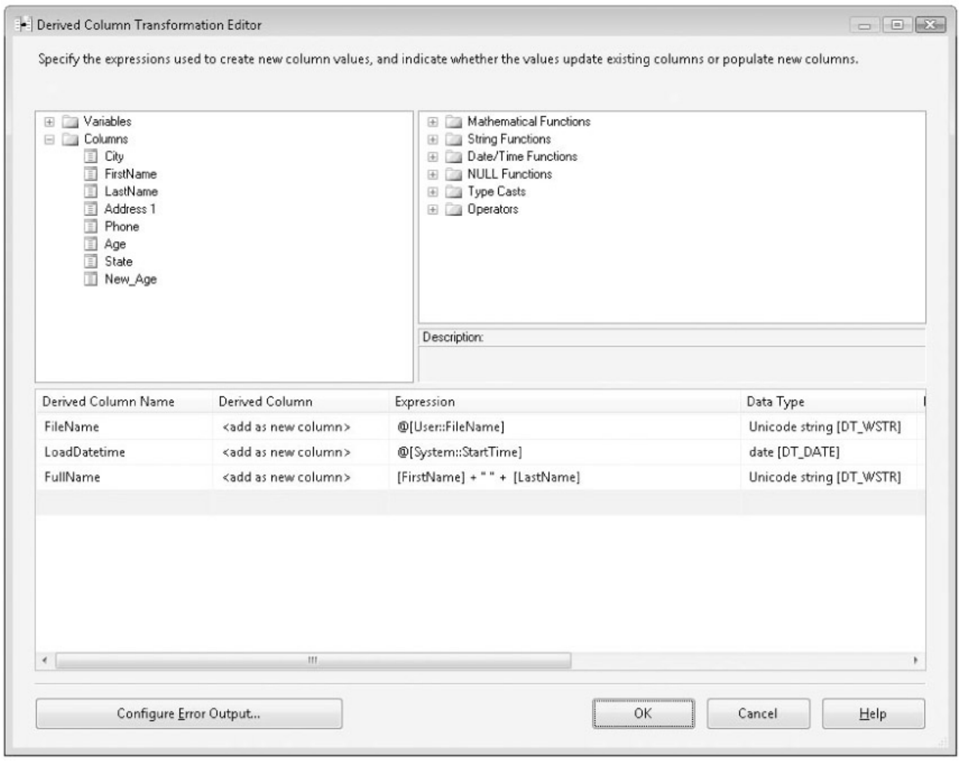
Before we redesigned the processing of this data, the operation required a nightly job that ran for 7.5 hours. The job included a batch process to convert the AS/400 binary file to ASCII, and the job pulled the Oracle and Excel data into a staging environment inside SQL Server and then through a rather large stored procedure. Custom logic joined data (numbering in the millions of rows) across servers through linked servers and staged the data into about 15 staging tables before the finished product was produced. Sound familiar?

The redesign in SSIS reduced a lot of the complexities because we could extract data from these sources directly into the data flow in SSIS and join different sources together. We also were able to convert the complicated T-SQL logic involving the staging tables to a series of transformations and tremendously reduce the overall disk I/O by going from 15 to 3 staging tables.

The net result was three SSIS packages that ran together in 25 minutes. What a time gain. In addition, using SSIS reduced hardware overhead and management of the old process, allowing the customer's IT professionals to do much more with time they didn't think they could ever recoup.

Using Transformations

Each transformation has an editor window to define the way the operation is applied to the data. For example, the Derived Column Transformation specifies an expression that generates a new column in the data flow or replaces an existing column. To open the Transformation Editor, either double-click the transformation or right-click the transformation and then click Edit. Figure 1-26 shows the Derived Column Transformation Editor.



Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

- Variables
- Columns
  - City
  - FirstName
  - LastName
  - Address 1
  - Phone
  - Age
  - State
  - New_Age

- Mathematical Functions
- String Functions
- Date/Time Functions
- NULL Functions
- Type Casts
- Operators

Description:

Derived Column Name	Derived Column	Expression	Data Type
FileName	<add as new column>	@[User::FileName]	Unicode string [DT_WSTR]
LoadDatetime	<add as new column>	@[System::StartTime]	date [DT_DATE]
FullName	<add as new column>	[FirstName] + " " + [LastName]	Unicode string [DT_WSTR]

Figure 1-26: The Derived Column Transformation Editor specifies how the data is manipulated as it flows through a transformation.

In the Derived Column example in Figure 1-26, one of the new columns added to the data flow is named FullName, which is based on the concatenation of the FirstName column and the LastName column using the following SSIS expression:

```
[FirstName] + " " + [LastName]
```

Other transformations contain similar functionality. Each transformation has an editor specific to the chosen operation.

The next example uses the Slowly Changing Dimension Transformation in the DimPromotion package to identify new records versus updated records. Figure 1-27 shows the data flow that results.

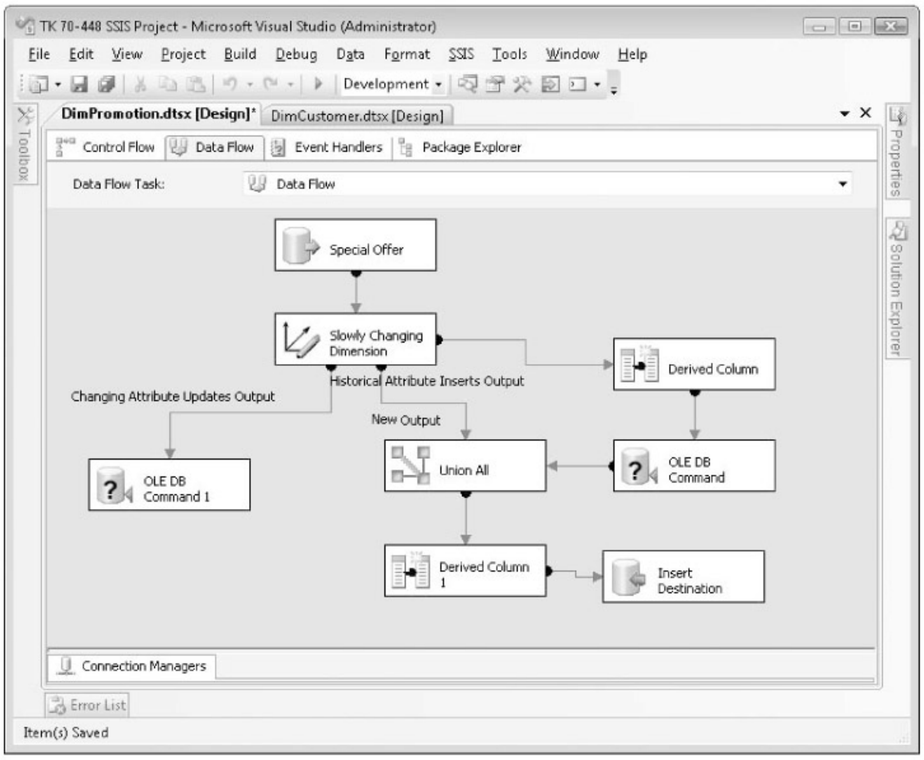
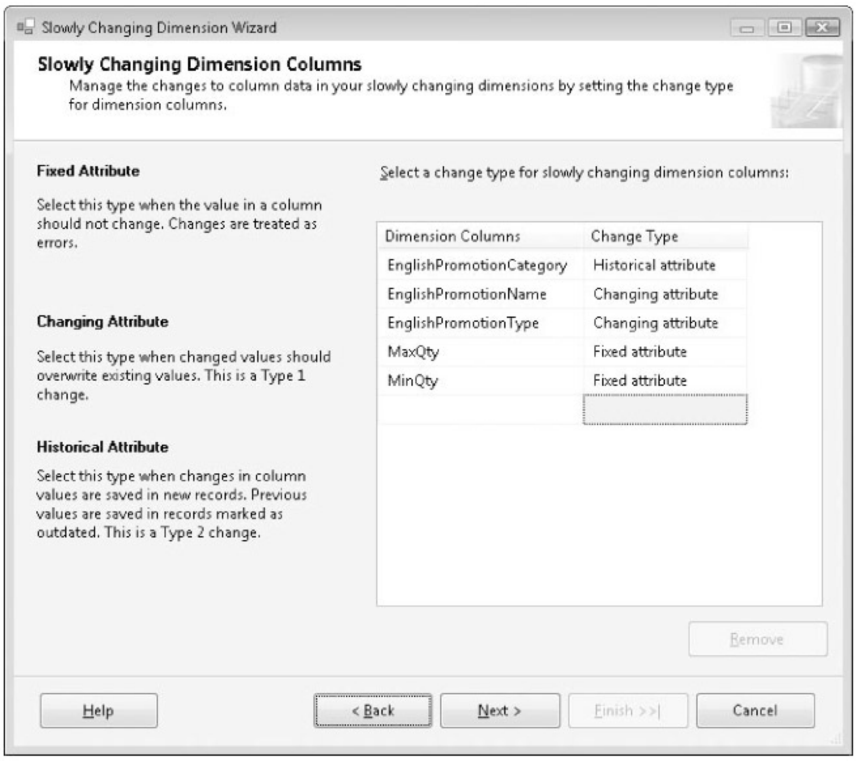

TK 70-448 SSIS Project - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Data Format SSIS Tools Window Help
Development
DimPromotion.dtsx [Design]*
DimCustomer.dtsx [Design]
Control Flow
Data Flow
Event Handlers
Package Explorer
Data Flow Task:
Data Flow
Special Offer
Slowly Changing Dimension
Derived Column
Historical Attribute Inserts Output
Changing Attribute Updates Output
New Output
OLE DB Command 1
Union All
OLE DB Command
Derived Column 1
Insert Destination
Connection Managers
Error List
Item(s) Saved
Toolbox
Properties
Solution Explorer

Figure 1-27: The Slowly Changing Dimension Transformation sends rows out multiple outputs depending on whether there is a new record or a change and on what kind of change.

Figure 1-27 shows the output of the Slowly Changing Dimension Transformation. All the output transformations and destinations were created by the Slowly Changing Dimension Wizard, which built the rest of the data flow.

Figure 1-28 shows the Slowly Changing Dimension Columns page of the wizard, which defines which dimension columns should cause what kind of change to the output. The options are Fixed Attribute, which means the change should not happen; Changing Attribute, which means that an update happens; or Historical Attribute, which means that the change creates a new record.


Slowly Changing Dimension Wizard

Slowly Changing Dimension Columns

Manage the changes to column data in your slowly changing dimensions by setting the change type for dimension columns.

Fixed Attribute

Select this type when the value in a column should not change. Changes are treated as errors.

Changing Attribute

Select this type when changed values should overwrite existing values. This is a Type 1 change.

Historical Attribute

Select this type when changes in column values are saved in new records. Previous values are saved in records marked as outdated. This is a Type 2 change.

Select a change type for slowly changing dimension columns:

Dimension Columns	Change Type
EnglishPromotionCategory	Historical attribute
EnglishPromotionName	Changing attribute
EnglishPromotionType	Changing attribute
MaxQty	Fixed attribute
MinQty	Fixed attribute

Figure 1-28: The Slowly Changing Dimension Wizard lets you define what kind of output should be created depending on the kind of change that occurs.

A detailed review of all the SSIS transformations is outside the scope of this Training Kit, but you can find information about them in the References section at the end of this book.

Practice—Creating Simple and Complex Data Flows

These exercises walk you through creating data flows that include sources, destinations, and one or more transformations. You will begin with a rather simple data flow but will then build a more complex data flow.

EXERCISE 1 Create a Simple Data Flow

In this exercise, you will develop a simple data flow that contains a source adapter, an Aggregate Transformation, and a destination adapter.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project *TK 70-448 SSIS Project* you created in Lesson 2, "Creating and Editing Control Flow Objects," or open the completed exercise file from the companion CD, and then open the *MyPackage.dtsx* package for editing.
2. On the Control Flow tab of the SSIS Designer, drag Data Flow Task from the toolbox into the Sequence Container object. The Sequence Container object should now include an Execute SQL Task named Update ProductLine and a Data Flow Task object.
3. Drag the output arrow from the Update ProductLine Task onto the Data Flow Task object. The output arrow is green, which means it represents a precedence constraint; see Chapter 3 for more information about precedence constraints.
4. Click the Data Flow tab at the top of the SSIS Designer.
5. In the toolbox, drag OLE DB Source, located under the Data Flow Sources group, onto the data flow workspace. Right-click the OLE DB Source item and then click Edit to open the OLE DB Source Editor dialog box.
6. Select AdventureWorks2008 in the OLE DB Connection Manager list and then click OK.
7. From the Data Access Mode drop-down list, select SQL Command.
8. In the SQL Command text box, type the following query (available in the SQLCommandQuery.sql file in the ..\Source\Ch 01\ folder for the practice exercises):
```
SELECT SH.OrderDate, SD.LineTotal, P.ProductLine
FROM Sales.SalesOrderHeader SH
INNER JOIN Sales.SalesOrderDetail SD
ON SH.SalesOrderID = SD.SalesOrderID
INNER JOIN Production.Product P
ON SD.ProductID = P.ProductID
```
9. Click the Columns tab on the left, and then verify that the OrderDate, LineTotal, and ProductLine columns are shown as available columns in the source adapter.
10. Click OK in the OLE DB Source Editor dialog box.
11. From the Data Flow toolbox, drag an Aggregate Transformation onto the Data Flow design surface, just below the OLE DB Source adapter.
12. Link the OLE DB Source output to the Aggregate Transformation by dragging the green output arrow onto the Aggregate Transformation.
13. Edit the Aggregate Transformation by double-clicking it or by right-clicking it and then clicking Edit.
    a. In the Aggregate Transformation Editor dialog box, select OrderDate from the Input Column drop-down list, and then verify that the default operation Group By is selected for the new row.
    b. Add a second Input Column row by selecting the LineTotal column from the drop-down list. For the Operation column of the newly added LineTotal, select Sum from the list. And last, type **SubTotal** in the Output Alias

column for the LineTotal row.

c. Add a third Input Column row by selecting the ProductLine column from the list.

d. Verify that the default operation Group By is selected for the new row.

e. Click OK in the Aggregate Transformation Editor dialog box.

14. In the Data Flow toolbox, navigate to the Data Flow Destinations grouping of objects, and then drag the OLE DB Destination object onto the Data Flow design surface.

15. Connect the output of the Aggregate Transformation to the new OLE DB Destination object by dragging the output arrow from the Aggregate Transformation onto the OLE DB Destination adapter.

16. Right-click the OLE DB Destination adapter, and then click Edit to display the OLE DB Destination Adapter Editor dialog box.
    a. In the OLE DB Destination Adapter Editor dialog box, verify that the OLE DB Connection Manager drop-down list is set to AdventureWorks2008.

    b. Click the New button next to the Name Of The Table Or The View drop-down list.

    c. In the Create Table dialog box, change the name of the new table to **Sales_Summary**. The CREATE TABLE code listed in the window should look like the following:

```
CREATE TABLE [Sales_Summary] (
    [OrderDate] DATETIME,
    [SubTotal] NUMERIC (38,6),
    [ProductLine] NVARCHAR(2)
) ON [PRIMARY]
```

    d. Click OK in the Create Table dialog box.

    e. On the Mappings tab of the OLE Destination Editor dialog box, ensure that the columns are all mapped from source to destination.

    f. Click OK to save your settings.

Figure 1-29 shows the completed data flow, with the source, aggregate, and destination components.

17. Right-click the Data Flow design surface, and then click Execute Task. Observe the execution of the data flow to confirm successful completion of this exercise.

18. Click the Stop button on the toolbar to stop the debugger (or choose Debug, Stop Debugging from the Debug menu).

19. Click the Save All button on the BIDS toolbar.

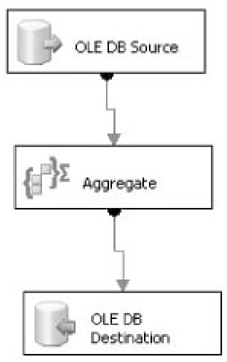


Figure 1-29: The data flow for this exercise contains an OLE DB Source adapter, an Aggregate Transformation, and an OLE DB Destination adapter.

EXERCISE 2 Create a Data Flow Destination

In this exercise, you will create a data flow that loads new records into the DimCustomer table of the AdventureWorksDW2008 database and that performs updates for existing records.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project TK 70-448 SSIS Project, and then open the DimCustomer.dtsx package for editing. Your DimCustomer.dtsx package should contain an Execute SQL Task named Truncate Update Table, a Data Flow Task object, and a second Execute SQL Task named Batch Updates that was created in Lesson 2.

2. Click the Data Flow tab at the top of the SSIS Designer to navigate to the Data Flow design surface.

3. Drag an OLE DB Source adapter from the toolbox onto the design surface. Rename the OLE DB Source adapter **Customer Source**. Edit the source adapter and set the following properties as shown:

OLE DB Connection Manager	AdventureWorks2008
Data Access Mode	SQL Command
SQL Command Text (Code available in the CustomerSourceQuery.sql practice exercise file in the ..\Source\Ch 01\ folder.)	`select convert(nvarchar(15),SC.AccountNumber) as CustomerAlternateKey, C.Title, C.FirstName, C.MiddleName, C.LastName, C.Suffix, C.EmailAddress, C.AddressLine1, C.AddressLine2, D.BirthDate, D.MaritalStatus, D.YearlyIncome, D.DateFirstPurchase, D.Gender, D.TotalChildren, D.NumberChildrenAtHome, D.Education, D.Occupation, D.HomeOwnerFlag, D.NumberCarsOwned from Sales.vIndividualCustomer C inner join Sales.Customer SC on C.BusinessEntityID = SC.PersonID inner join Sales.vPersonDemographics D on C.BusinessEntityID = D.BusinessEntityID`

4. Drag a second OLE DB Source adapter from the toolbox onto the Data Flow design surface, rename it to **Customer Dim**, and then edit it. Edit the source adapter and set the following properties as shown:

OLE DB Connection Manager	AdventureWorksDW2008
Data Access Mode	Table or view
Name Of The Table Or View	[dbo].[DimCustomer]

5. For this next set, you will be sorting the data from the sources on the business key. First, drag two Sort Transformations from the Data Flow toolbox onto the Data Flow design surface, and then connect the output arrow for the Customer Source adapter to the first Sort Transformation and the Customer Dim to the second Sort Transformation, as Figure 1-30 shows.

6. Edit the first Sort Transformation and select the check box on the left side of the CustomerAlternateKey column in the Available Input Columns. Click OK to save the transformation.

7. Edit the second Sort Transformation and select the check box on the left side of the CustomerAlternateKey column in the Available Input Columns. Click OK to save the transformation.

8. From the Data Flow toolbox, drag a Merge Join Transformation to the design surface, and then connect the output arrow from the first Sort Transformation (originating from Customer Source) to the Merge Join Transformation. When prompted with the Input Output Selection dialog box, choose Merge Join Left Input from the Input drop-down list, and then click OK.

9. Also connect the output arrow of the second Sort Transformation (originating from Customer Dim) to the Merge Join Transformation.

10. Edit the Merge Join Transformation to display the Merge Join Transformation Editor dialog box.

   a. Change the Join Type drop-down list setting to Left Outer Join, which will retrieve all the rows from the originating Customer Source query (the left source of the Merge Join Transformation) and any matching rows from the right side (which is from the *dbo.DimCustomer* source).

   b. To return all the columns from the Customer Source query, select the check box immediately to the left of the Name column header in the left Sort list. Doing this will select all the check boxes for every column that is the desired result.

   c. In the right list of columns from Customer Dim, select only the check box next to the CustomerAlternateKey column.

   d. Scroll down the Output Columns list at the bottom of the Merge Join Transformation Editor dialog box to the very bottom, and for the CustomerAlternateKey column, change the Output Alias value to **Dim_CustomerAlternateKey**.

   e. Click OK to save the changes to the Merge Join Transformation.

11. From the Data Flow toolbox, drag a Conditional Split Transformation onto the Data Flow design surface, and then connect the output arrow from the Merge Join Transformation to the Conditional Split Transformation.

12. Edit the Conditional Split Transformation to display the Conditional Split Transformation Editor dialog box.

   a. Create a new output by typing **New Records** in the Output Name box for the first row of the output list.

   b. In the same row of the output list, type the following code in the Condition field:
   ```
   ISNULL([Dim_CustomerAlternateKey]) == TRUE
   ```

   c. In the Default Output Name box, change the value from Conditional Split Default Output to **updated Records**.

   d. Click OK to save your changes in the Conditional Split Transformation Editor dialog box.

13. From the Data Flow toolbox, drag an OLE DB Destination adapter to the Data Flow design surface (be sure not to drag the similar source adapter but rather the destination adapter), and then change its name to **DimCustomer Table**.

14. Drag the output arrow of the Conditional Split Transformation onto this new OLE DB Destination adapter. When prompted in the Input Output Selection dialog box, select New Records from the Output drop-down list, and then click OK.

15. Right-click the DimCustomer Table Destination adapter that you just created and click Edit to display the OLE DB Destination Editor dialog box. Set the following properties in the OLE DB Destination Editor dialog box:

OLE DB Connection Manager	AdventureWorksDW2008
Data Access Mode	Table Or View—Fast Load
Name Of The Table Or View	[dbo].[DimCustomer]

   a. While you are still in the OLE DB Destination Editor dialog box, click the Mappings tab in the left area of the dialog box. This automatically maps the columns from the data flow to the DimCustomer table based on column name and data type.

   b. Not all columns will be mapped. From the Available Input Columns list, locate the Education column and drag it on top of the EnglishEducation column of the Available Destination Columns list. Do the same for Occupation to EnglishOccupation and HomeOwnerFlag to HouseOwnerFlag.

   c. Click OK to save your changes in the OLE DB Destination Editor dialog box.

16. Add a second OLE DB Destination adapter to the Data Flow design surface, and then connect another output arrow from the Conditional Split Transformation to the new OLE DB Destination adapter. Rename the destination adapter **DimCustomer Update Table**.

17. Edit the DimCustomer Update Table destination adapter you just created to display the OLE DB Destination Editor dialog box. Set the following properties in the OLE DB Destination Editor dialog box:

OLE DB Connection Manager	AdventureWorksDW2008
Data Access Mode	Table Or View—Fast Load
Name Of The Table Or View	[dbo].[stgDimCustomerUpdates] (This table was created in Lesson 2.)

a. While you are still in the OLE DB Destination Editor dialog box, click the Mappings tab. This automatically maps the columns from the data flow to the DimCustomer table based on column name and data type.

b. Not all columns will be mapped. From the Available Input Columns list, locate the Education column and drag it on top of the EnglishEducation column of the Available Destination Columns list. Do the same for Occupation to EnglishOccupation and HomeOwnerFlag to HouseOwnerFlag.

c. Click OK to save your changes in the OLE DB Destination Editor dialog box.

Your data flow should now resemble the one shown in Figure 1-31. You can find the completed exercises in the ..\Source\Ch 01\ folder of the Training Kit materials.

18. Confirm the correct development of your package by executing the package in BIDS.

19. Choose Debug, Stop Debugging from the Debug menu to stop the debugger, and then click the Save All button on the BIDS toolbar.

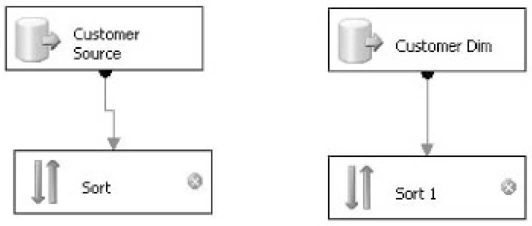


Figure 1-30: The initial data flow for this exercise contains two OLE DB Source adapters and two Sort Transformations.

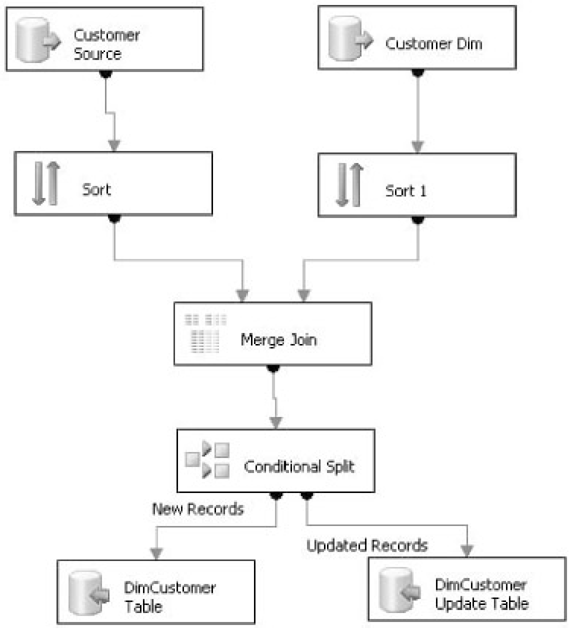


Figure 1-31: The final data flow for this exercise contains several sources and destinations, with transformation logic to handle inserts and to stage updates for the DimCustomer table.

Quick Check

1. How would you use SSIS to import a file from an FTP server to a SQL Server table? ?
2. You need to migrate a user-created Access database to SQL Server, but the Data Flow toolbox does not contain an Access source adapter. How do you import this data into SQL Server? ?
3. The Multicast Transformation and the Conditional Split Transformation both can have multiple outputs. Which transformation would you use if you needed to send rows matching certain conditions out one output and rows matching different conditions out another? ?
4. Describe the transformations you could use to combine data from two different database tables that exist on two different servers. ?

Answers

1. First, you would use an FTP Task to copy the file to the machine on which SSIS is installed. You can then import the file into a SQL Server table by using a Data Flow Task configured with a Flat File Source adapter and either a SQL Server Destination adapter or an OLE DB Destination adapter.
2. Although not listed in the toolbox, Access is one of the many database sources and destinations that SSIS works with. To extract data from Access, you first need to make a package connection to the Microsoft Jet OLE DB Provider. You can then use the OLE DB Source adapter to select the table or perform a custom query.
3. The Conditional Split Transformation lets you define expressions against which the rows from the source are evaluated. For every row, the expressions are evaluated in order, and a row is sent out the first output when the matching expression evaluates to True. Therefore, any single row can go out only one output. With a Multicast Transformation, on the other hand, all rows go out every output.
4. To combine data from two different database tables that exist on two different servers, you could use the Merge Join Transformation, which combines datasets by joining the rows across a set of common keys. This transformation allows an inner join, a left outer join, or a full outer join. You could also use a Lookup Transformation to associate data from two sources. The Lookup can cache a table in memory and, through matching columns, can return new columns to the data flow.

Case Scenario: Creating an ETL Solution

The business development department of Adventure Works has requested that you implement a data mart that it can use to analyze reseller sales against salesperson sales targets. Your first task is to create a series of SSIS packages that move data from the source Enterprise Resource Planning (ERP) system to a data mart database that contains fact tables and dimension tables.

1. How would you work within BIDS to create SSIS project structures, packages, project data sources, and package connections? ?
2. What transformations would you use, and how would you implement the data flow that loads dimension tables? ?
3. What transformations would you use, and how would you implement the data flow that loads fact tables? ?

Answers

1. The best practice for creating a set of packages that all work together is to create a single SSIS project within BIDS. You would then create multiple packages within the project, one for each of the different dimensions and fact tables. Because all the packages would use the same source connection string to the ERP system and the same destination connection string to the data mart, you should create two data sources within the project: one for the source and one for the destination. Last, each package would need to reference the project data sources, so within each package, you

would create package connections based on the project data sources.

2. The SSIS data flow contains a Slowly Changing Dimension Transformation that can handle changes and new records in the dimension tables. Before using the Slowly Changing Dimension Transformation, you need to create a source adapter to the ERP system that pulls the data to be compared with the dimension tables. You might need to use a transformation to clean and correct any data anomalies, and after those steps are complete, you can connect the data to the Slowly Changing Dimension Transformation. Ideally, you would create a separate SSIS package for each dimension package so that they could be reused for different groups of package executions.

3. Fact tables contain surrogate keys that reference the dimension tables but also contain the business keys from the source. So as you are pulling the reseller sales data and the sales quotas, you can use a Lookup Transformation to get the surrogate keys from dimension tables by joining across the business keys. You can then insert new rows into the fact tables by using the surrogate keys and measures. As you do when using dimensions, you will typically have a separate package for each fact table that needs processing.

Chapter Summary

- Creating SSIS packages involves working with BIDS and creating a new SSIS project.
- The main object in an SSIS project is a package, which contains the business logic to manage workflows and process data.
- Within a package, the control flow lets you create tasks and containers, which provide the ability to run process-oriented operations.
- The Data Flow Task is the second core object (behind the control flow) in an SSIS package, enabling data-processing operations.
- The data flow uses source adapters, destination adapters, and transformations.

Notes

Chapter 1 - Developing SSIS Packages (Lesson 1: Creating SSIS Packages and Data Sources)

$Package Creation Wizard limitations: In general, the Import And Export Wizard provides a quick way to move data from one source to a destination, especially for a one-time use, but there are some limitations: You can specify only one source and one destination in the wizard. Advanced workflow precedence is not available through the wizard. The wizard does not share data sources with other packages. *** $Objects in BIDS --> Remember that any one project in BIDS can contain only objects from the same project type, such as SSIS, SQL Server Analysis Services (SSAS), or SQL Server Reporting Services (SSRS). However, a single project can be associated with projects of different types in the same solution. ** Be aware that after a package is deployed to a new environment and executed outside the project, the connection string is no longer updated by the project data source ** When you create a package connection from a data source, that connection is updated only during development whenever the package is opened and the data source has been changed. Package connections are not updated when they are run separately from the associated SSIS project—for example, when they are run from the command line. **