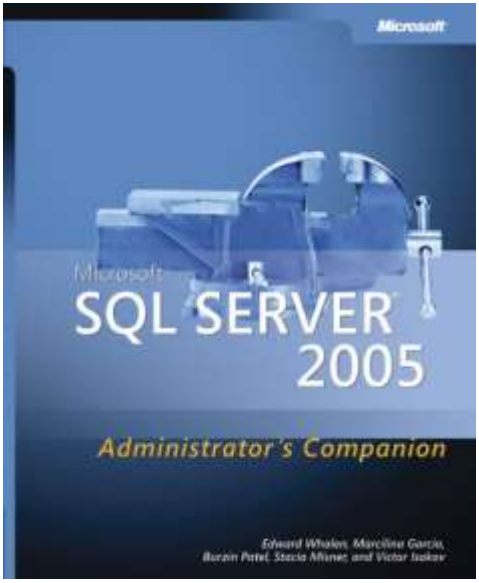

Microsoft® SQL Server™ 2005 Administrator's Companion

Edward Whalen, Marcilina Garcia, Burzin Patel, Stacia Misner, Victor Isakov

To learn more about this book, visit Microsoft Learning at
http://www.microsoft.com/MSPress/books/6793.aspx

9780735621985
Publication Date: November 2006

Microsoft Press

Table of Contents

What do you think of this book?
We want to hear from you!

Microsoft is interested in hearing your feedback about this publication so we ca
continually improve our books and learning resources for you. To participate in
online survey, please visit: *www.microsoft.com/learning/booksurvey/*

Part VII

Performance Tuning and Troubleshooting

What do you think of this book?
We want to hear from you!

Microsoft is interested in hearing your feedback about this publication so we ca
continually improve our books and learning resources for you. To participate in
online survey, please visit: *www.microsoft.com/learning/booksurvey/*

Chapter 8

Installing and Upgrading Microsoft SQL Server 2005

Now that you have a good understanding of the different editions of Microsoft SQL Server 2005 Server, the platforms on which it can be run, and capacity planning and storage configuration concepts, let's get to the next most important step: installing SQL Server 2005.

This chapter provides a detailed look at the planning necessary before installation and the step-by-step installation process using the graphical user interface and the command line. You will also learn how to upgrade to SQL Server 2005 from earlier versions, how to configure SQL Server features and services using the new SQL Server Surface Area Configuration tool, and how to uninstall SQL Server 2005 components.

Preinstallation Planning

Before installing SQL Server 2005, it is extremely important that you plan the installation process well and have all the relevant information necessary for the installation process. This will help ensure a smooth installation experience and prevent unnecessary postinstallation changes.

This section explains some of the important configuration options you need to have decided on before starting the installation. While the graphical user interface-based installation method is relatively easy and many users like to adopt a "discover-as-you-go" approach, I have found time and again that this is not the most productive approach. The time supposedly saved by not planning out the installation is spent either cancelling and restarting the installation, or debugging and resolving incorrect configuration options after the installation is complete. Both of these cases are undesirable. I highly recommend that you read the following sections to understand the various planning considerations and then decide which ones are applicable and important to your deployment.

Minimum Hardware Requirements

SQL Server 2005 has a well-defined set of minimum hardware requirements that need to be met for SQL Server 2005 installation. These requirements are listed in Table 8-1. These are only the bare minimum requirements for SQL Server 2005 installation; they do not guarantee good performance. Refer to Chapter 6, "Capacity Planning," to determine the appropriate hardware resources required for your particular deployment.

Table 8-1 Minimum Hardware Requirements

Resource	Requirement
Monitor	At least 1024 × 768 pixel resolution (SVGA) if using graphical tools
Pointing device	Microsoft mouse or compatible pointing device
DVD drive	Only required if installing from DVD media
Network card	Only required if accessing via the network
Processor	**32-bit systems:** • Processor type: Pentium III-compatible or higher • Processor speed: 600 MHz minimum **64-bit systems:** • Processor type (IA64): Itanium processor or higher • Processor type (x64): AMD Opteron, AMD Athlon 64, Intel Xenon with Intel EM64T support and Intel Pentium IV with EM64T support • Processor speed: 1 GHz minimum
Memory (RAM)	Minimum 512 MB, recommended 1 GB

During installation, the System Configuration Checker (SCC) will display an error message and terminate the installation if the system does not meet the minimum processor type requirements. SCC will issue a warning if the minimum processor speed or the recommended memory requirements are not met.

Note If you have 1 GB of memory in the system, the SQL Server 2005 installation wizard may incorrectly flag a warning stating that the current system does not meet the recommended hardware requirements. This is an anomaly in the installer. If you're sure that system does meet the minimum requirements, you can ignore this message.

The disk space requirements for the SQL Server executables and samples vary based on the components selected for installation. Table 8-2 lists the disk space utilized by the different SQL Server 2005 components.

Table 8-2 SQL Server 2005 Disk Space Requirements

Feature	Disk Space Required
Database engine, replication, and full-text search	150 MB
Analysis Services	35 KB
Reporting Services and Report Manager	40 MB
Notification Services engine, client, and rules components	5 MB
Integration Services	9 MB
Client Components	12 MB
Management Tools	70 MB
Development Tools	20 MB
SQL Server Books Online and SQL Server Mobile Books Online	15 MB
Samples and sample databases	390 MB

The maximum disk space required if all of the components and samples are selected is approximately 750 MB.

Selecting the Processor Architecture

As mentioned in Chapter 2, "SQL Server 2005 Editions, Capacity Limits, and Licensing," each SQL Server 2005 edition is available on the 32-bit (IA-32), 64-bit (IA64), and 64-bit (x64) platforms. To make sure that the software installs correctly and performs well, make sure that you install the correct executables SQL Server 2005 platform version for your operating system and hardware. While combinations such as installing the IA64 SQL Server 2005 software on a 32-bit system will simply not install and result in an error message, some combinations like 32-bit software on the x64 platform may work but not perform properly.

Installing Internet Information Services

If you plan to install Microsoft SQL Server 2005 Reporting Services, you will require Internet Information Services (IIS) 5.0 or later installed on the server before SQL Server 2005 setup is started. You can install IIS using the following steps:

1. Click Start, then select Control Panel (or select Settings and then Control Panel), and then double-click Select Add or Remove Programs in Control Panel.
2. In the left pane, click Add/Remove Windows Components.
3. Select Application Server in the Windows Components Wizard that opens, and then select Details.
4. Select the check box next to Internet Information Services (IIS) in the Application Server dialog box that appears, then click OK, and then click Next.
5. You may be prompted to insert your Windows media CD, so you may want to have this available during installation.

In general, having IIS installed on your server is not recommended unless it is absolutely required. If you do not plan to use Reporting Services on your server, I'd recommend you do not install IIS and ignore the warning messages that are displayed during the installation process.

Components to Be Installed

Unlike earlier versions of SQL Server, which required invoking separate installation processes for the different components, SQL Server 2005 has a fully integrated setup through which all the components can be installed together via a single installation process. You can select any combination of the following components for installation:

- SQL Server Database Services
- Analysis Services
- Reporting Services
- Notification Services
- Integration Services
- Workstation components, books online, and development tools

Depending on the Microsoft SQL Server components you choose to install, the following 10 services are installed:

1. **SQL Server Main** SQL Server database engine
2. **SQL Server Agent** Used for automating administrative tasks, executing jobs, alerts, and so on

3. **SQL Server Analysis Services** Provides online analytical processing (OLAP) and data mining functionality for Business Intelligence (BI) applications
4. **SQL Server Reporting Services** Manages, executes, renders, schedules, and delivers reports
5. **SQL Server Notification Services** Platform for developing and deploying applications that generate and send notifications

> **Note** When you install SQL Server Notification Services, a service is not installed by default and will not appear under Services in the Control Panel. The service is configured only when you build an application and register a service to run that application.

6. **SQL Server Integration Services** Provides management support for Integration Services package storage and execution
7. **SQL Server Full Text Search** Enables fast linguistic searches on content and properties of structured and semistructured data by using full-text indexes
8. **SQL Server Browser** Name resolution service that provides SQL Server connection information for client computers
9. **SQL Server Active Directory Helper** Publishes and manages SQL Server services in Windows Active Directory
10. **SQL Server VSS Writer** Allows backup and restore applications to operate in the Volume Shadow-copy Service (VSS) framework

I recommend that you be selective and install only the components you actually plan to use. This will limit the number of unnecessary services that run on your server and prevent them from consuming precious server resources like disk space, memory, processor, and so on.

Service Accounts

All SQL Server 2005 services require a login account to operate. The login account can be either a local service account, a domain user account, a network service account, or a local system account.

- **Local Service account** This is a built-in account that has the same level of access as members of the users group. This low-privileged access limits the damage that can be done in case the service gets compromised. This account is not effective for use with services that need to interact with other network services since it accesses network resources with no credentials.

- **Domain User account** As the name suggests, this account corresponds to an actual domain user account. This account is preferred when the service needs to interact with other services on the network.
- **Network Service account** This account is similar to the Local Service account, except that services that run as the Network Service account can access network resources with the credentials of the computer account.
- **Local System account** The Local System account is a highly privileged account and should be used very selectively. You should be careful not to confuse this account with the Local Service account. With respect to privileges, they are at opposite ends of the spectrum.

Best Practices You should always configure a service to run with the lowest effective privileges that can be used.

Table 8-3 lists the default accounts for each of the 10 SQL Server services. You can change these as required, but always consider the limitations and security exposures explained previously.

Table 8-3 SQL Server Service Default Accounts

SQL Server Service	Default Account
SQL Server	Domain User
SQL Server Agent	Domain User
SQL Server Analysis Services	Domain User
SQL Server Reporting Services	Domain User
SQL Server Notification Services	N/A
SQL Server Integration Services	Network Service
SQL Server Full-Text Search	Same account as SQL Server
SQL Server Browser	Domain User
SQL Server Active Directory Helper	Network Service
SQL Server VSS Writer	Local System

Before you start installation, make sure that all domain accounts required to configure the services during setup have been created and are available for use.

Multiple Instances and Side-by-Side Installation

Microsoft SQL Server 2005 supports multiple instances of the database engine, Analysis Services, and Reporting Services to be installed side-by-side on the same computer. Side-by-side installations are completely separate instances and not dependent on each other

in any way. You can choose to have any combination of side-by-side installs of SQL Server 7.0, SQL Server 2000, or SQL Server 2005 listed as supported in Table 8-4.

Table 8-4 Supported Side-by-Side Installations

Side-by-Side Install	SQL Server 2000 (32-bit)	SQL Server 2000 (64-bit)	SQL Server 2005 (32-bit)	SQL Server 2005 (IA64)	SQL Server 2005 (x64)
SQL Server 7.0	Yes	No	Yes	No	No
SQL Server 2000 (32-bit)	Yes	No	Yes	No	Yes
SQL Server 2000 (64-bit)	No	Yes	No	Yes	No
SQL Server 2005 (32-bit)	Yes	No	Yes	No	Yes
SQL Server 2005 (IA64)	No	Yes	No	Yes	No
SQL Server 2005 (x64)	Yes	No	Yes	No	Yes

If you already have an instance of SQL Server installed on your system, you should decide before starting the installation process whether you'd like to upgrade it (as explained later in this chapter) or install a new SQL Server 2005 instance on the side.

Licensing Mode

As explained in Chapter 2, SQL Server 2005 can be installed using a per-processor licensing model, a user client access license (user CAL) licensing model, or a device client access license (device CAL) licensing model. Before starting the installation process, you should determine the licensing model you plan to use and secure the required licenses.

Collation

A *collation* determines the rules by which character data is sorted and compared. SQL Server 2005 has two groups of collations: Windows collations and SQL collations. SQL collations are provided primarily as a compatibility option with earlier versions of SQL Server. You should use these if you plan to use replication with databases on earlier versions of SQL Server or if your application requires a specific SQL collation of an earlier SQL Server version. For all other cases, you should use the Windows collation.

> **Best Practices** You should decide on an organization-wide collation and use it for all your SQL Server 2005 servers so you're assured of consistency for all server-to-server activity.

The collation specified during the installation process becomes the SQL Server instance's default collation. This collation is used for all the system databases and any user databases that do not explicitly specify a collation.

Authentication Modes

SQL Server supports two authentication modes: Windows authentication mode and mixed mode.

- **Windows authentication mode** This authentication mode permits users to connect only by using a valid Windows user account. With Windows authentication, SQL Server validates the account credentials using information from the Windows operating system. The Windows authentication mode optionally provides password policy enforcement for validation for strong passwords, support for account lockout, and password expiration. The sa user ("sa" is short for "system administrator") is disabled when Windows authentication is selected.
- **Mixed mode** This authentication mode permits users to connect using either Windows authentication or SQL Server authentication. Users who connect through a Windows user account are validated by Windows, while users who connect using SQL Server login are validated by SQL Server. The sa user is enabled when mixed mode is selected and a password prompt appears during the installation process.

Best Practices Never use a blank or weak password for the sa account.

It is recommended that you use strong passwords for all users who will log in to SQL Server 2005. A strong password must be six or more characters long and have at least three of the following types of characters:

- Uppercase letters
- Lowercase letters
- Numbers
- Non-alphanumeric characters

Although Windows authentication is the recommended authentication mode and more secure than mixed mode, many applications require mixed mode authentication. You should evaluate your application needs and select the authentication based on that.

Security Considerations

A large part of the long-term security of your server environment is dictated by some relatively simple and inexpensive best practices you can adopt during the planning and installation phase. To make your SQL Server installation as secure as possible, the following are recommended:

- Physically secure the server and make it accessible only to authorized personnel.
- Have at least one firewall between the server and the Internet.

- Enforce strong passwords for all SQL Server accounts and enable password policies and password expiration.
- Create service accounts with least privileges.
- Run separate SQL Server services under separate Windows accounts to prevent one compromised service from being used to compromise others.
- Use NTFS instead of a FAT file system.
- Disable all unnecessary protocols, including NetBIOS and server message block (SMB), on the servers.

> **Note** Disabling the NetBIOS protocol may cause connectivity problems if you're using DHCP. You may want to check with your system administrator before disabling any protocols.

Installing SQL Server 2005

Once you've completed the preinstallation planning and have all the required information available, you are ready to install SQL Server 2005. SQL Server 2005 can be installed on your local server using either the SQL Server 2005 Installation Wizard or the command prompt installation. If you're new to SQL Server or plan to install just a couple of servers, I recommend you use the Installation Wizard. The command prompt-based installation is often slightly trickier and better suited to experienced users who need to perform multiple similar installations and want to automate the process. SQL Server also provides the option of installing just the SQL Native Access Client (SNAC) connectivity libraries on the server; this process is explained in detail later in this chapter. This is particularly useful for client systems that need to use SNAC to connect to the SQL Server 2005–based server. All of these installation methods are explained in detail in the following sections.

> **Note** Installing to a remote server, which was possible in earlier versions of SQL Server, is not supported in SQL Server 2005. To install SQL Server 2005 onto a remote server, you need to log in remotely to the server and run the setup program, or remotely execute the command prompt installation on the remote server.

Installing SQL Server 2005 Using the Installation Wizard

The SQL Server 2005 Installation Wizard is a Windows installer-based program that interactively guides you through the entire installation process. The Installation Wizard has built-in tools for performing appropriate configuration and error checking and provides meaningful warning and error messages.

The following steps explain how to install a new nonclustered SQL Server 2005 instance on your local server. If you already have an instance of SQL server installed on your server, some of the windows shown in the figures may not be presented or may be slightly different. This is because the Installation Wizard reuses the information already available on the system; for example, the Registration dialog box (step 8) will not prompt you for the PID if you've already installed the same version of SQL Server 2005 on the system before.

1. Log in to the system as Administrator or as a user who has administrator privileges on the server.

> **Note** The SQL Server 2005 Setup program can be invoked in many ways. In most cases, the program automatically starts when the SQL Server 2005 DVD media is inserted into the DVD drive or when a remote network share is mapped onto the server. If the program is not automatically loaded, you can navigate to the Servers directory and double-click the Splash.hta program. With either of these approaches, the Start dialog box, shown in Figure 8-1, appears.

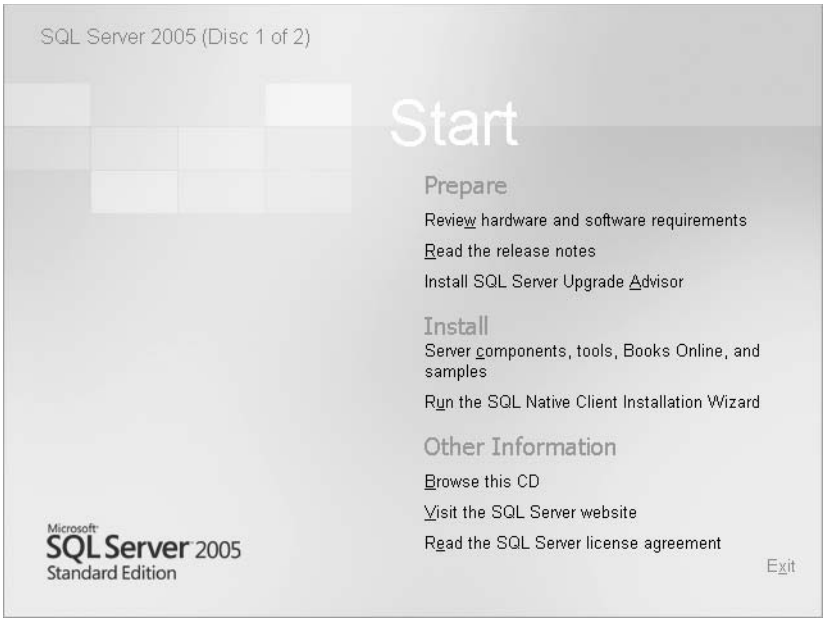

Figure 8-1 SQL Server Setup—Start window.

2. The Start window presents options to prepare and install the server as well as access other information. To install SQL Server 2005, click the "Server components, tools, Books Online, and samples" option in the Start window.
3. The End User License Agreement (EULA) window appears. Read the agreement and select the I Accept the Licensing Terms and Conditions check box. Selecting the check box will activate the Next button. Select Next.
4. The Installing Prerequisites dialog box, shown in Figure 8-2, appears, and the software components required prior to installing SQL Server 2005 are installed. Select Install. This step may take several minutes to complete.

> **Note** You may see a different list in Figure 8-2 if some of the components have already been installed via a previous install, or by some other application.

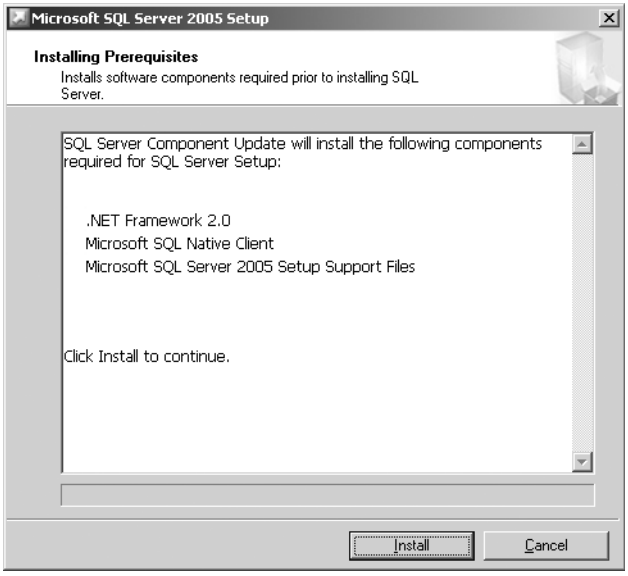

Figure 8-2 SQL Server Setup—Installing Prerequisites dialog box.

5. The Welcome page for the Installation Wizard appears. Select Next.
6. The System Configuration Check (SCC) page appears. At this point, the Installation Wizard scans the system for conditions that do not meet the minimum requirements and displays the status for each action with a message for the errors and a warning, as shown in Figure 8-3.

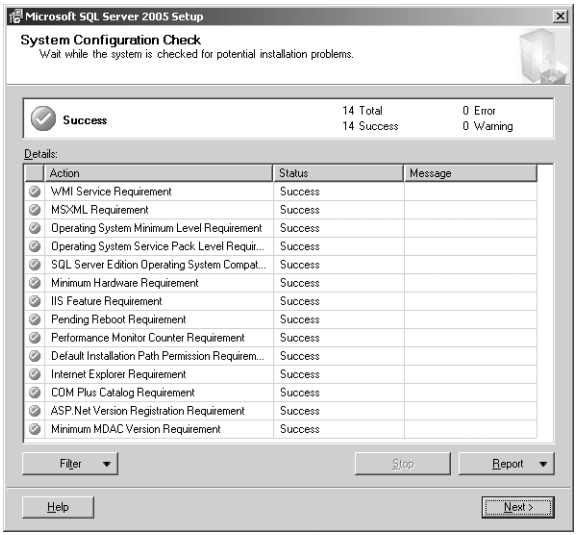

Figure 8-3 SQL Server Setup—System Configuration Check page.

7. Once the SCC has completed scanning the computer, the Filter button in the lower-left corner is activated and can be used to filter the output to Show All Actions, Show Errors, Show Successful Actions, or Show Warnings in the window. You can only view the actions that are relevant, for example if there are no errors the Show Errors option is not activated. Correspondingly, the Report button in the lower-right corner can be used to view a report in a report format, save the report to a file, copy the report to the Clipboard, or send the report as e-mail. Once SCC completes the configuration check, click Next to continue with the setup.

> **Note** If the SCC determines a pending action that must be completed before proceeding, for example a pending reboot operation, it will block the setup by not activating the Next button and force you to complete the pending actions.

8. The setup performs some additional checks that may take a few minutes and then displays the Registration Information page. On the Registration Information page, enter information in the Name, Company, and Product Key text boxes. Select Next to continue.

9. The Components To Install page displays, as shown in Figure 8-4. On this page, select the components to be installed that you identified during the preinstallation planning.

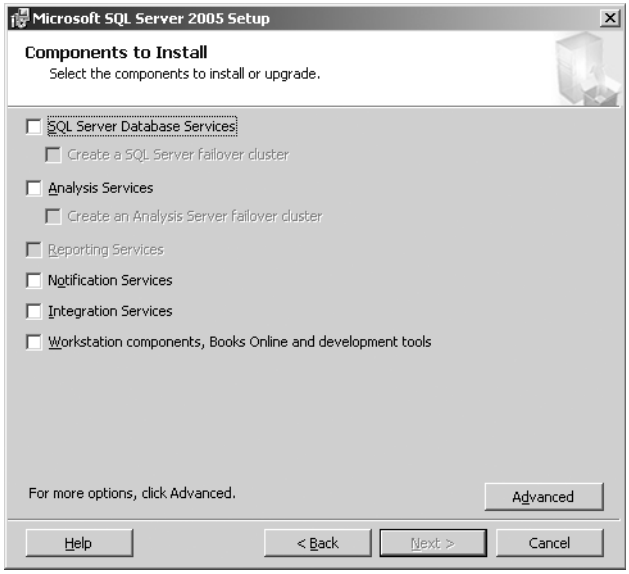

Figure 8-4 SQL Server Setup—Components To Install page.

10. To select specific subcomponents for any of the components, you can select the Advanced button on the lower-right side of the page, which will display the Feature Selection dialog box as shown in Figure 8-5.

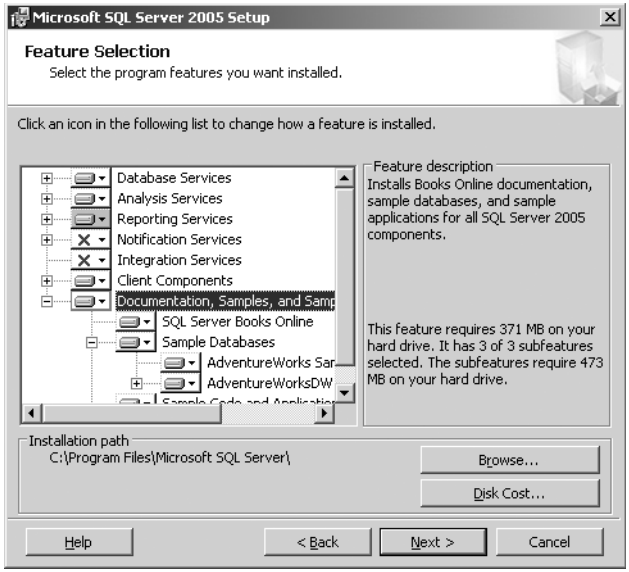

Figure 8-5 SQL Server Setup—Feature Selection dialog box.

In this dialog box, you can select the Will Be Installed On Local Hard Drive option to install the feature but not all the subcomponents of the feature, select the Entire Feature Will Be Installed On Local Hard Drive option to install the feature and all the subcomponents of the feature, or select the Entire Feature Will Be Unavailable option to not install the feature. Once you have selected the appropriate services, select Next to continue.

> **Note** The sample databases and sample code and applications are not installed by default even when the Documentation, Samples, and Sample Databases feature is selected. To install these, select the Advanced button and explicitly select them for installation, as shown in Figure 8-5, or select the Entire Feature Will Be Installed On Local Hard Drive option for the Documentation, Samples, And Sample Databases feature.

11. The Instance Name page, shown in Figure 8-6, appears. On this page, you can select the instance to be either a Default Instance or a Named Instance. If you select Named Instance, the text box in which you need to enter a valid instance name is activated. You can select the Installed Instances button in the lower right of the page to view the instances already installed on the system. If a default or named instance is already installed on the server and you select it, the setup will upgrade it and present you the option of installing additional components. This is explained in the section on upgrading to SQL Server 2005 later in this chapter. Click Next to continue.

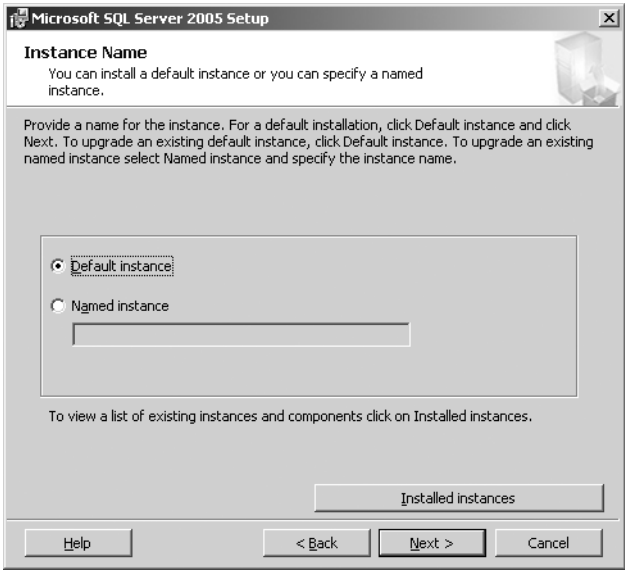

Figure 8-6 SQL Server Setup—Instance Name page.

Note A server can have only one default instance of SQL Server. This implies that if you have SQL Server 2000 installed on your server as a default instance and you do not want to upgrade it, you should install the SQL Server 2005 as a named instance.

12. The Service Account page, shown in Figure 8-7, is displayed. This page is used to specify the accounts the services use to log in. You can either specify the same account for all the services installed or select the Customize For Each Service Account check box and specify the login accounts for each service selected for installation individually. You can then select the login account to use one of the built-in system accounts (Local Service, Network Service, or Local System) by clicking on the Use The Built-in System Account radio button and selecting the appropriate account from the drop-down list, or you can specify a domain user by selecting the Use A Domain User Account radio button and entering a domain user name, password, and domain. In the Start Services At The End Of Setup section, you can select the check boxes next to the services you would like to start automatically every time the system is started. Click Next to continue.

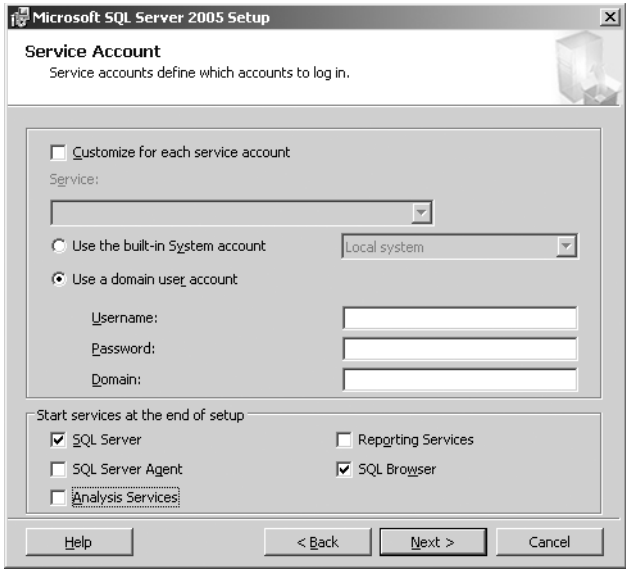

Figure 8-7 SQL Server Setup—Service Account page.

13. The Authentication Mode page, shown in Figure 8-8, appears. On this page, click the appropriate radio button to select either Windows Authentication Mode or Mixed Mode (Windows Authentication And SQL Server Authentication). If you use the mixed mode, you will need to enter and confirm the login password for the sa user. Click Next to continue.

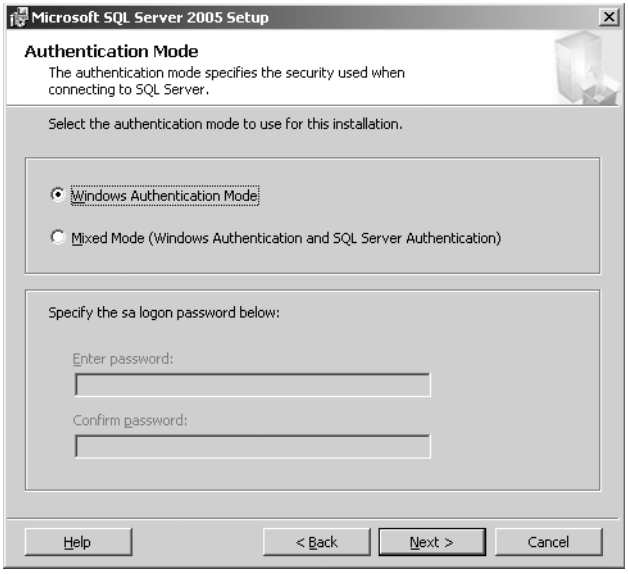

Figure 8-8 SQL Server Setup—Authentication Mode page.

14. The Collation Settings page, shown in Figure 8-9, appears. On this page you can choose to customize the collation for each individual service being installed using the Customize For Each Service Account check box, or you can use the same collation for all the services. For the collation, you can select either Collation Designator And Sort Order or SQL Collations (Used For Compatibility With Previous Versions Of SQL Server) using the radio buttons. If you are using the collation designator and sort order, select the language (for example, Latin1_General for the English language) from the drop-down list and the appropriate check boxes below. If you are using the SQL Collations, select the desired one from the scrollable list below the radio button. Click Next to continue.

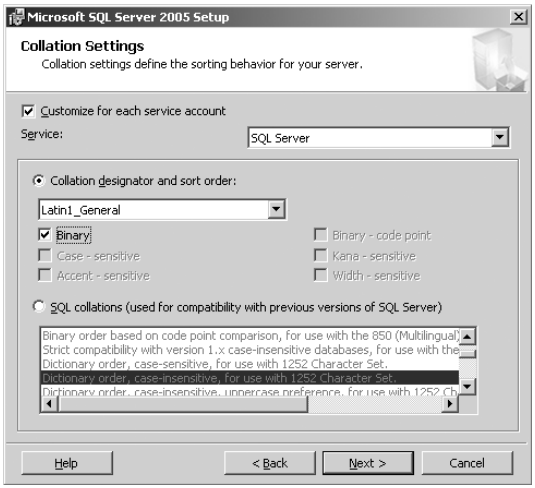

Figure 8-9 SQL Server Setup—Collation Settings page.

15. If you selected to install Reporting Services, the Report Server Installation Options page, shown in Figure 8-10, appears. You can use the radio buttons on this page to choose to Install The Default Configuration for Reporting Server or Install But Do Not Configure The Server. You can select the Details button located in the upper right of the page to view the details of the Report Server installation information. If a Secure Sockets Layer (SSL) certificate has not been installed on the server, a warning message is displayed. Since reports often contain sensitive information, it is recommended that you use SSL in most installations. Select Next to continue.

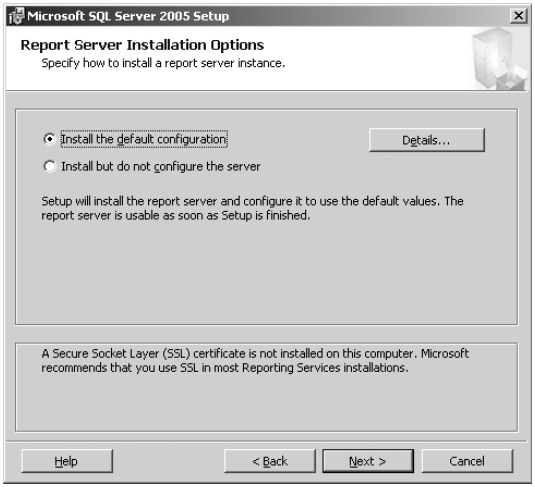

Figure 8-10 SQL Server Setup—Report Server Installation Options page.

16. The Error And Usage Report Settings page, shown in Figure 8-11, appears. On this page, you can select the two radio buttons, Automatically Send Error Reports For SQL Server 2005 To Microsoft Or Your Corporate Error Reporting Server and Usage Data For SQL Server 2005 To Microsoft, to set the desired default action. This data is collected for information purposes only, and selecting either of these options will not have any adverse effects on the performance of your system. Select Next to continue.

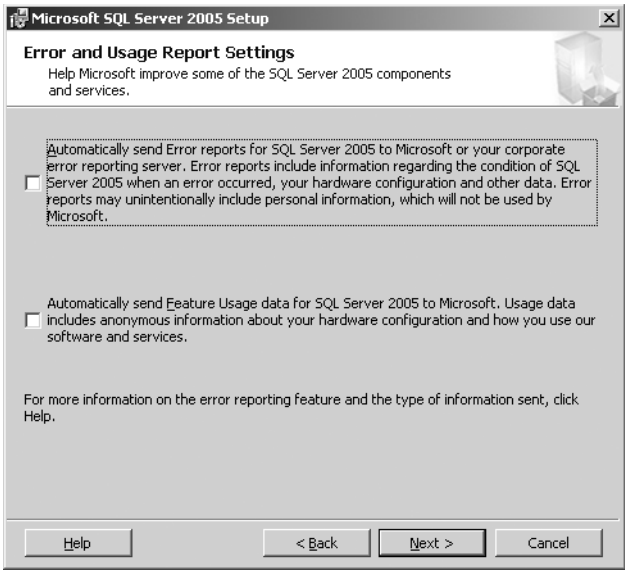

Figure 8-11 SQL Server Setup—Error And Usage Report Settings page.

17. The Ready To Install page, shown in Figure 8-12, appears. You can review the summary of features and components selected for installation. To make any changes, select the Back button and go back in the installation process until the relevant page appears. For the most part, the installation process will retain your selections so that you don't have to re-enter all of the information after backtracking through the pages. Select Install to continue.
18. The Setup Progress page, shown in Figure 8-13, appears. At this point in the installation process, the selected services are actually installed and configured on your system. This step may take a while to complete and is dependent on the speed of your processor and the disk being installed to. The page continuously updates the progress bar to reflect the installation status of the individual components and will reset for each component being installed. To view the log file for the component installation status, you can click the component name. When all of the steps are completed, select Next to continue.

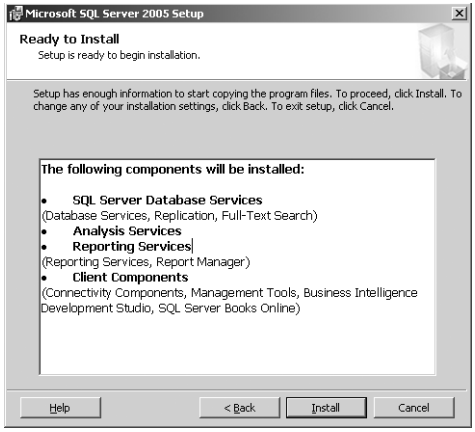

Figure 8-12 SQL Server Setup—Ready To Install page.

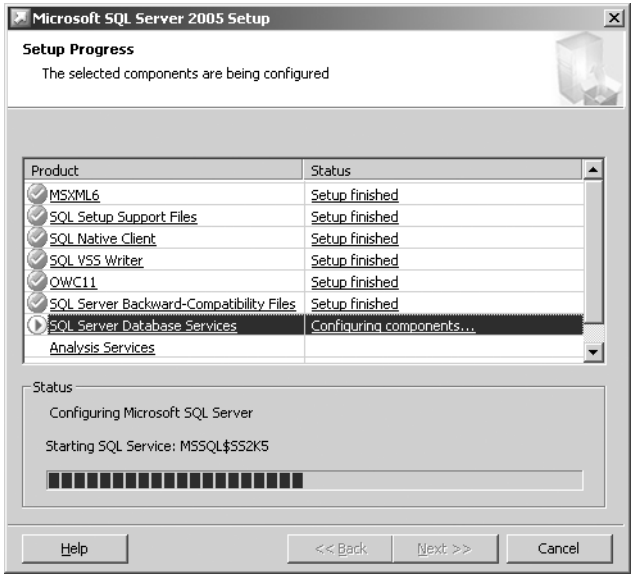

Figure 8-13 SQL Server Setup—Setup Progress page.

19. The Completing Microsoft SQL Server 2005 Setup page, shown in Figure 8-14, appears. On this page, you view the summary log. You can also select the Surface Area Configuration Tool to configure SQL Server 2005 as explained in the Surface Area Configuration section that follows. Click Finish to complete the installation.

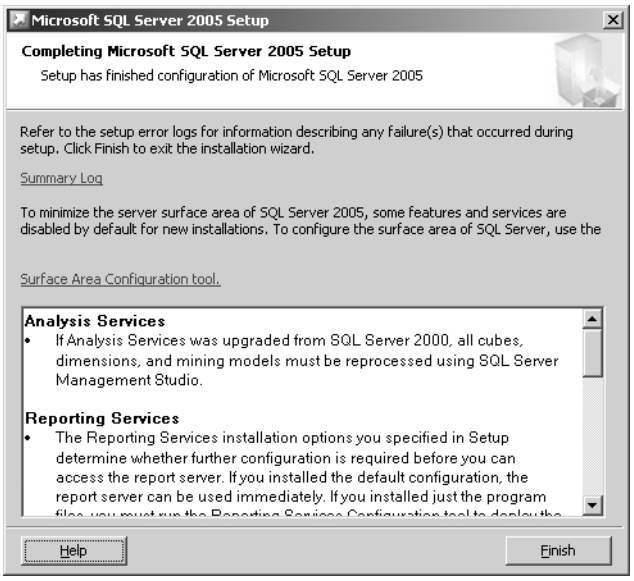

Figure 8-14 SQL Server Setup—Completing Microsoft SQL Server 2005 Setup page.

20. Restart the system if the setup prompts you to do so.

> **Note** If you need to add or remove components to a default or named instance of SQL Server 2005, you can do so by selecting Add Or Remove Programs in Control Panel, selecting the SQL Server 2005 instance you want to modify, and then clicking the Change or Remove buttons.

Installing SNAC Using the Installation Wizard

1. Log in to the system as Administrator or as a user who has administrator privileges on the server.

> **Note** The SQL Server 2005 Setup program can be invoked in many ways. In most cases, the program will start automatically when the SQL Server 2005 DVD media is inserted into the DVD drive or when a remote network share is mapped onto the server. If the program is not automatically loaded, you can navigate to the Servers directory and double-click the Splash.hta program.

2. The Start window appears, similar to what is shown in Figure 8-1. Select Run The SQL Native Client Installation Wizard.
3. The Welcome page of the wizard appears. Click Next to continue.

4. The License Agreement page appears. Read and accept the terms in the license agreement and click Next.
5. The Registration Information page appears. Enter your name and the name of your organization in the text fields and click Next.
6. The Feature Selection page, shown in Figure 8-15, appears. Select the program features you want to install and click Next.

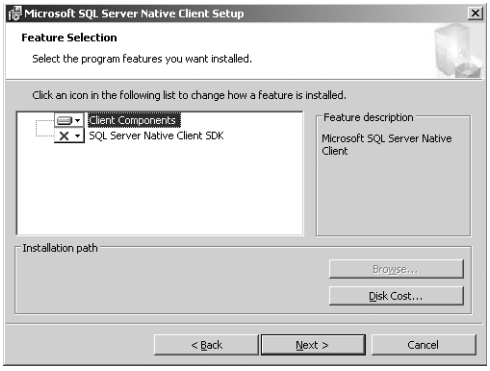

Figure 8-15 SQL Native Client Installation—Feature Selection page.

> **Note** The Client Components contain the SNAC network library files and should be selected if you are installing SNAC on a client for connectivity.

7. The Ready To Install The Program page appears. Click Install.
8. After the installation process completes, click Finish.

Installing SQL Server 2005 Using the Command Prompt

Unlike earlier versions, SQL Server 2005 does not have an unattended install recorder and playback mechanism. Instead, it ships with a powerful command prompt installation option, which can be used to install, modify, or uninstall SQL Server components and perform certain maintenance tasks. With command prompt installation, you can choose either to specify all the input parameter directly on the command line or to pass them in using a settings (.ini) file.

The syntax for a command prompt installation is shown in the following example.

```
Start /wait <DVD Drive>\Servers\setup.exe /qb INSTANCENAME=MSSQLSERVER
ADDLOCAL=SQL_Engine SQLACCOUNT=advadmin SQLPASSWORD=Pa55wD
AGTACCOUNT=advadmin AGTPASSWORD=Pa55wD SQLBROWSERACCOUNT=advadmin
SQLBROWSERPASSWORD=Pa55wD
```

In this example, the SQL Server 2005 database engine is installed as a default instance using the account advadmin and password Pa55wd.

> **Important** Since the password is clearly visible in the code, it presents a potential security risk and should be used carefully. Do not leave any references to a password such as this in an unprotected script file.

The command prompt installation can be used to customize every option in the installation process. Table 8-5 lists the command prompt installation options and gives a brief description of each.

Table 8-5 Command Prompt Installation Options

Command Prompt Option	Description
/qb	Installation is done in quiet mode with basic GUI information displayed, but no user interaction is required.
/qn	Installation is done in quiet mode with no GUI displayed.
Options	This parameter is for the Registration Information dialog box and must be specified when using a settings file.
PIDKEY	This parameter is used to specify the registration key. [Note: Do not specify the hyphens (-) that appear in the key.]
INSTALLSQLDIR	This parameter is used to specify the installation directory for the instance specific binary files.
INSTALLSQLSHAREDDIR	This parameter is used to specify the installation directory for Integration Services, Notification Services, and Workstation components.
INSTALLSQLDATADIR	This parameter is used to specify the installation directory for the SQL Server data files.
INSTALLASDATADIR	This parameter is used to specify the location for the Analysis Services data files.
ADDLOCAL	This parameter is used to specify the components to install. ADDLOCAL=ALL installs all the components. Setup fails if ADDLOCAL is not specified. (Note: Feature names are case sensitive.)
REMOVE	This parameter specifies which components to uninstall. The INSTANCENAME parameter must be used in conjunction with this parameter.

Table 8-5 Command Prompt Installation Options (continued)

Command Prompt Option	Description
INSTANCENAME	This parameter specifies the name of the instance. MSSQLSERVER is used to represent the default instance. This parameter must be specified for instance-aware components.
UPGRADE	This parameter is used to specify which product to upgrade. The INSTANCENAME parameter must be used in conjunction with this parameter.
SAVESYSDB	This parameter can be used during uninstall to specify not to delete system databases.
USESYSDB	This parameter is used to specify the root path to the system databases data directory during upgrade.
SQLACCOUNT, SQLPASSWORD, AGTACCOUNT, AGTPASSWORD, ASACCOUNT, ASPASSWORD, RSACCOUNT, RSPASSWORD	These parameters are used to specify the service accounts and passwords for the services. A service account and password need to be provided for each service selected for installation.
SQLBROWSERAUTOSTART, SQLAUTOSTART, AGTAUTOSTART, ASAUTOSTART, RSAUTOSTART	These parameters are used to specify the startup behavior of the respective service. When set to 1, the service will start automatically; when set to 0, the service must be started manually.
SECURITYMODE and SAPWD	SECURITYMODE=SQL is used to specify mixed mode authentication. SAPWD is used to specify the password for the sa account.
SQLCOLLATION and ASCOLLATION	These parameters are used to set the collations for SQL Server and Analysis Services, respectively.
REBUILDDATABASE	This parameter is used to rebuild the master database.
REINSTALLMODE	This parameter is used to repair installed components that may be corrupted.
REINSTALL	This parameter is used to specify the components to reinstall and must be specified when using REINSTALLMODE. REINSTALL parameters use the same values as ADDLOCAL parameters.
RSCONFIGURATION	This parameter is applicable only if Reporting Services or Report Manager is installed. It is used to specify whether to configure the service.
SAMPLEDATABASESERVER	This parameter is used to specify the server and instance name to which the sample databases should be attached.

Table 8-5 Command Prompt Installation Options (continued)

Command Prompt Option	Description
DISABLENETWORKPROTOCOLS	This parameter is used to set up the start-up state of the network protocols.
ERRORREPORTING	This parameter is used to configure SQL Server to send reports from any fatal errors directly to Microsoft.

More Info For a complete list of parameters and their possible values, refer to the SQL Server Setup Help by double-clicking

<DVD Drive>\Servers\Setup\help\1033\setupsql9.chm and searching for 'How to: Install SQL Server 2005 from the Command Prompt'.

In the next few sections, we will see how these parameters can be used in combination to perform a variety of operations such as installing a default instance with all the components, installing a named instance with mixed mode authentication, adding components to an existing instance, and using a settings file to pass in the installation parameters.

Installing a Default Instance

This is one of the most commonly used command prompt–based installation scenarios. The following command installs all of the SQL Server 2005 components in a default instance (MSSQLSERVER) on the local server. A Windows administrator account called advadmin with a password of Pa55wD is used for all the services.

```
start /wait <DVD Drive>\Servers\setup.exe /qb INSTANCENAME=MSSQLSERVER
ADDLOCAL=ALL SAPWD=Pa55wD SQLACCOUNT=advadmin SQLPASSWORD=Pa55wD
AGTACCOUNT=advadmin AGTPASSWORD=Pa55wD ASACCOUNT=advadmin ASPASSWORD=Pa55wD
RSACCOUNT=advadmin RSPASSWORD=Pa55wD SQLBROWSERACCOUNT=advadmin
SQLBROWSERPASSWORD=Pa55wD
```

Best Practices If you plan to store these commands as script files, you should make sure to store them in a secure location with the correct permissions, since they contain unencrypted passwords.

Installing a Named Instance with Mixed Authentication

The following command installs database engine and management tools components on a named instance of SQL Server 2005 called SS2K5 with mixed authentication and the

Latin1_General_BIN collation. A Windows administrator account called advadmin with a password of Pa55wD is used for all the services.

```
start /wait <DVD Drive>\Servers\setup.exe /qb INSTANCENAME=SS2K5
ADDLOCAL=SQL_Engine,SQL_Data_Files,Client_Components,Connectivity,SQL_Tools9
0 SECURITYMODE=SQL SQLCOLLATION=Latin1_General_Bin SQLAUTOSTART=1
AGTAUTOSTART=1 SAPWD=Pa55wD SQLACCOUNT=advadmin SQLPASSWORD=Pa55wD
AGTACCOUNT=advadmin AGTPASSWORD=Pa55wD SQLBROWSERACCOUNT=advadmin
SQLBROWSERPASSWORD=Pa55wD
```

Adding Components to an Existing Instance

The command prompt installation method can also be used to add components to an existing SQL Server 2005 instance. The following command adds the Analysis Server components with the Latin1_General_Bin collation setting to an existing instance named SS2K5. Once again, a Windows administrator account called advadmin with a password of Pa55wD is used for all the services.

```
start /wait <DVD Drive>\Servers\setup.exe /qb INSTANCENAME=SS2K5
ADDLOCAL=Analysis_Server,AnalysisDataFiles ASCOLLATION=Latin1_General_Bin
SAPWD=Pa55wD ASACCOUNT=advadmin ASPASSWORD=Pa55wD
```

Installing Using a Settings (.ini) File

All the command prompt installation examples we've seen so far have specified the setup options directly on the command line. This approach works well but is not very easy to use given that the commands are usually rather long and prone to typos. Also, the commands need to be re-typed for each use and cannot be easily persisted across sessions. To circumvent these problems, SQL Server 2005 allows you to use a settings file with which you can pass in the desired command prompt options. A settings file is a text file which contains a list of setup parameters.

The following example settings file specifies the options that can be used to install all SQL Server 2005 components using the mixed mode authentication and the Latin1_General_BIN collation for both SQL Server database as well as Analysis Server.

```
[Options]
USERNAME=Mike
COMPANYNAME=Microsoft

PIDKEY=ADDYOURVALIDPIDKEYHERE


ADDLOCAL=ALL


INSTANCENAME=MSSQLSERVER
```

```
SQLBROWSERACCOUNT=advadmin
SQLBROWSERPASSWORD=Pa55wD

SQLACCOUNT=advadmin
SQLPASSWORD=Pa55wD

AGTACCOUNT=advadmin
AGTPASSWORD=Pa55wD

ASACCOUNT=advadmin
ASPASSWORD=Pa55wD

RSACCOUNT=advadmin
RSPASSWORD=Pa55wD

SQLBROWSERAUTOSTART=1
SQLAUTOSTART=1
AGTAUTOSTART=1
ASAUTOSTART=0
RSAUTOSTART=0

SECURITYMODE=SQL
SAPWD=Pa55wD

SQLCOLLATION=Latin1_General_BIN
ASCOLLATION=Latin1_General_BIN

DISABLENETWORKPROTOCOLS=2
```

Note A sample template file (Template.ini) listing all the configurable parameters is provided with the SQL Server media and can be found in the same directory as the Setup.exe program.

The settings file is specified using the /settings option of the command prompt installation. For example, the following command passes in the SqlInstall.ini file containing the setup parameters to the setup program.

```
start /wait <DVD Drive>\Servers\setup.exe /qb /settings C:\SqlInstall.ini
```

Best Practices Since the settings files contain logins, passwords, and product keys, you should always store them in a secure location with the appropriate file permissions.

Upgrading to SQL Server 2005

If you have an existing installation of SQL Server, you can choose to upgrade it to SQL Server 2005 instead of installing a new instance. SQL Server 2005 supports direct upgrade paths from SQL Server 7.0 with SP4 and SQL Server 2000 with SP3 or later versions. Table 8-7 lists the versions of SQL Server and the possible direct upgrade path to the corresponding SQL Server 2005 edition. Before upgrading from one edition to another, you should always verify that all the functionality you are currently using is supported in the edition being upgraded to.

Table 8-7 Supported Upgrade Paths to SQL Server 2005

Upgrade from	Supported Upgrade Paths
SQL Server 7.0 Enterprise Edition SP4	SQL Server 2005 Enterprise Edition
SQL Server 7.0 Developer Edition SP4	SQL Server 2005 Enterprise Edition
	SQL Server 2005 Developer Edition
SQL Server 7.0 Standard Edition SP4	SQL Server 2005 Standard Edition
	SQL Server 2005 Enterprise Edition
SQL Server 7.0 Desktop Edition SP4	SQL Server 2005 Standard Edition
	SQL Server 2005 Workgroup Edition
SQL Server 7.0 Evaluation Edition SP4	Upgrade not supported
SQL Server Desktop Engine (MSDE) 7.0 SP4	SQL Server 2005 Express Edition
SQL Server 2000 Enterprise Edition SP3 or later versions	SQL Server 2005 Enterprise Edition
SQL Server 2000 Developer Edition SP3 or later versions	SQL Server 2005 Developer Edition
SQL Server 2000 Standard Edition SP3 or later versions	SQL Server 2005 Enterprise Edition
	SQL Server 2005 Developer Edition
	SQL Server 2005 Standard Edition
SQL Server 2000 Workgroup Edition	SQL Server 2005 Enterprise Edition
	SQL Server 2005 Developer Edition
	SQL Server 2005 Standard Edition
	SQL Server 2005 Workgroup Edition

Table 8-7 Supported Upgrade Paths to SQL Server 2005 (continued)

Upgrade from	Supported Upgrade Paths
SQL Server 2000 Personal Edition SP3 or later versions	SQL Server 2005 Standard Edition SQL Server 2005 Workgroup Edition SQL Server 2005 Express Edition
SQL Server 2000 Evaluation Edition SP3 or later versions	SQL Server 2005 Evaluation Edition
SQL Server Desktop Engine (MSDE) 2000	SQL Server 2005 Workgroup Edition SQL Server 2005 Express Edition
SQL Server 2000 IA-64 (64-bit) Enterprise Edition	SQL Server 2005 IA-64 (64-bit) Enterprise Edition
SQL Server 2000 IA-64 (64-bit) Developer Edition	SQL Server 2005 IA-64 (64-bit) Enterprise Edition SQL Server 2005 IA-64 (64-bit) Developer Edition
SQL Server 2005 Developer Edition	SQL Server 2005 Enterprise Edition SQL Server 2005 Standard Edition SQL Server 2005 Workgroup Edition
SQL Server 2005 Standard Edition	SQL Server 2005 Enterprise Edition SQL Server 2005 Developer Edition
SQL Server 2005 Workgroup Edition	SQL Server 2005 Enterprise Edition SQL Server 2005 Developer Edition SQL Server 2005 Standard Edition
SQL Server 2005 Evaluation Edition	SQL Server 2005 Enterprise Edition SQL Server 2005 Developer Edition SQL Server 2005 Standard Edition SQL Server 2005 Workgroup Edition SQL Server 2005 Express Edition
SQL Server 2005 Express Edition	SQL Server 2005 Enterprise Edition SQL Server 2005 Developer Edition SQL Server 2005 Standard Edition SQL Server 2005 Workgroup Edition
SQL Server 2005 IA-64 (64-bit) Developer Edition	SQL Server 2005 IA-64 (64-bit) Enterprise Edition SQL Server 2005 IA-64 (64-bit) Standard Edition
SQL Server 2005 x64 (64-bit) Developer Edition	SQL Server 2005 x64 (64-bit) Enterprise Edition SQL Server 2005 x64 (64-bit) Standard Edition
SQL Server 2005 IA-64 (64-bit) Standard Edition	SQL Server 2005 IA-64 (64-bit) Enterprise Edition SQL Server 2005 IA-64 (64-bit) Developer Edition

Table 8-7 Supported Upgrade Paths to SQL Server 2005 (continued)

Upgrade from	Supported Upgrade Paths
SQL Server 2005 x64 (64-bit) Standard Edition	SQL Server 2005 x64 (64-bit) Enterprise Edition
	SQL Server 2005 x64 (64-bit) Developer Edition
SQL Server 2005 IA-64 (64-bit) Evaluation Edition	SQL Server 2005 IA-64 (64-bit) Enterprise Edition
	SQL Server 2005 IA-64 (64-bit) Developer Edition
	SQL Server 2005 IA-64 (64-bit) Standard Edition
SQL Server 2005 x64 (64-bit) Evaluation Edition	SQL Server 2005 x64 (64-bit) Enterprise Edition
	SQL Server 2005 x64 (64-bit) Developer Edition
	SQL Server 2005 x64 (64-bit) Standard Edition

A SQL Server 2000 32-bit instance running on the 32-bit subsystems of an x64 system cannot be upgraded to run on the 64-bit subsystem directly. If you require converting your 32-bit SQL Server instance to SQL Server 2005 (64-bit), you will need to install SQL Server 2005 on the 64-bit server as a new instance and then move the database over. You can move the databases either by backing them up from the 32-bit system and restoring them on the 64-bit system, or by detaching the databases from the 32-bit system, copying them over to the 64-bit system and attaching them to the 64-bit system. In either case, you will need to do some additional housekeeping tasks such as recreating logins, reconfiguring replication, and so forth on the new 64-bit server instance.

English-language versions of SQL Server can be upgraded to an English-language or any other localized version of SQL Server 2005. However, localized versions of SQL Server can be upgraded only to localized versions of SQL Server 2005 of the same language. In addition, SQL Server 2005 does not support cross-version instances, implying that all the components (for example, Database Engine, Analysis Services, and Reporting Services) within a single instance must be the same.

SQL Server Upgrade Advisor

SQL Server Upgrade Advisor is a stand-alone tool that can help you analyze your SQL Server 7.0 or SQL Server 2000 database for possible incompatibilities before being upgraded to SQL Server 2005 and help you proactively resolve them. Although most well-designed SQL Server databases should be seamlessly upgradeable to SQL Server 2005, there are some scenarios in which SQL Server 2005 has tightened up on checking for compliance with SQL standards and disallows certain nonstandard code constructs. The Upgrade Advisor is a great way to quickly and easily check for such cases.

The following sections explain the steps to install and use the Upgrade Advisor.

Installing SQL Server Upgrade Advisor

The SQL Server Upgrade Advisor is a stand-alone tool that must be installed via a separate installation process. To install SQL Server Upgrade Advisor:

1. Log in to the system as Administrator or a user who has administrator privileges on the server.

   > **Note** The SQL Server 2005 Setup program can be invoked in many ways. In most cases, the program will automatically start when the SQL Server 2005 DVD media is inserted into the DVD drive or when a remote network share is mapped onto the server. If the program does not load automatically, you can navigate to the Servers directory and double-click the Splash.hta program.

2. From the Start window, from the Prepare section, select Install SQL Server Upgrade Advisor.
3. On the Welcome page, click Next.
4. The License Agreement page appears. Read and accept the terms of the license agreement by selecting the radio button, and then click Next.
5. The Registration Information page appears. Enter your name and the name of your organization and click Next.
6. The Feature Selection page appears. On this page, leave the Upgrade Advisor feature selected. You can change the directory to which Upgrade Advisor will be installed by using the Browse button. You can also view the disk cost by using the Disk Cost button. Click Next to continue.
7. The Ready To Install The Program page appears. Click Install.
8. The Setup Wizard will install the Upgrade Advisor and should report a successful completion message. Click Finish to complete the installation.

Using SQL Server Upgrade Advisor

The Upgrade Advisor is built from two components: the Upgrade Advisor Analysis Wizard and the Upgrade Advisor Report Viewer.

- **Upgrade Advisor Analysis Wizard** This tool helps analyze the SQL Server 7.0 or SQL Server 2000 instance for issues that can cause the upgrade to fail or your application to falter after the upgrade. The wizard does not modify the instance in any way and can be run as many times as you like.

- **Upgrade Advisor Report Viewer** This tool is used to view the list of issues found by the Analysis Wizard.

The typical sequence of events when using the Upgrade Advisor includes executing the Update Advisor, gathering the recommendations, taking the recommended corrective actions, and rerunning Upgrade Advisor to verify the changes. While this process can be completed in a single pass, I have often found it to require a couple of iterations before all the issues are resolved.

The following steps explain the process of using the SQL Server Upgrade Advisor Analysis Wizard and viewing the report using the Upgrade Advisor Report Viewer:

1. To open SQL Server 2005 Upgrade Advisor click the Start button, and then point to All Programs, then Microsoft SQL Server 2005, and then select SQL Server 2005 Upgrade Advisor. The Upgrade Advisor Start window appears, as shown in Figure 8-16.
2. Select Launch Upgrade Advisor Analysis Wizard. The Welcome page appears. Click Next.

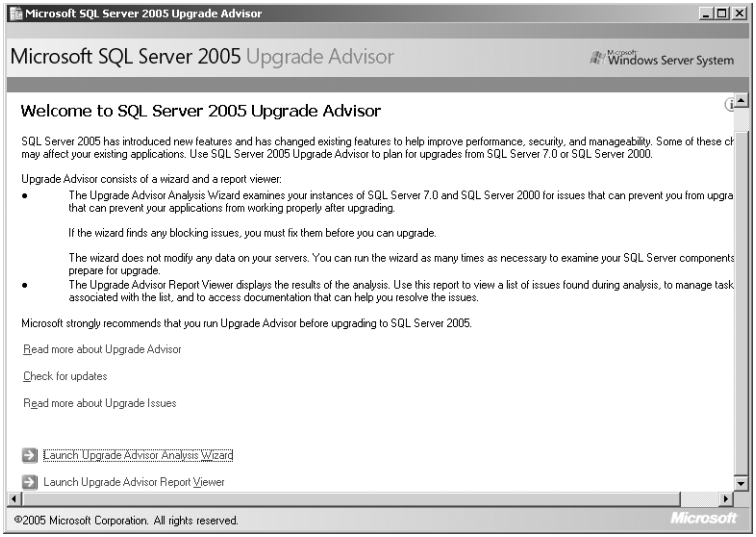

Figure 8-16 SQL Server 2005 Upgrade Advisor—Start window.

3. The SQL Server Components page appears, as shown in Figure 8-17. Enter the name of the server you want to run Upgrade Advisor against. You can choose to query the server and automatically populate the appropriate check boxes for the components by clicking on Detect, or you can choose to manually select the check boxes. Click Next to continue.

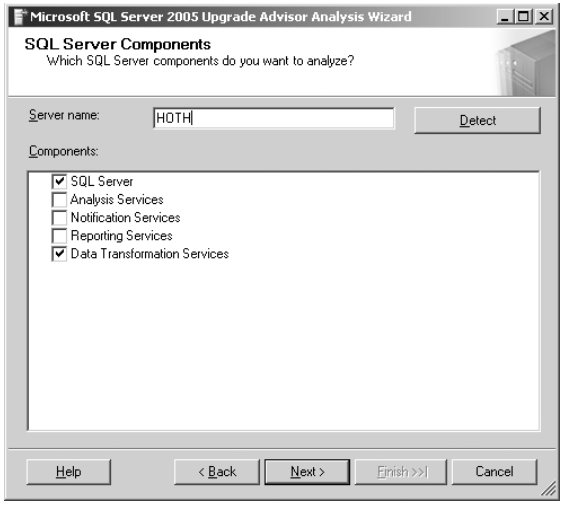

Figure 8-17 SQL Server 2005 Upgrade Advisor—SQL Server Components page.

4. The Connection Parameters page, shown in Figure 8-18, appears. Select the instance name (select MSSQLSERVER for the default instance), select the authentication mode, and enter the login credentials if using the SQL Server authentication mode. Click Next.

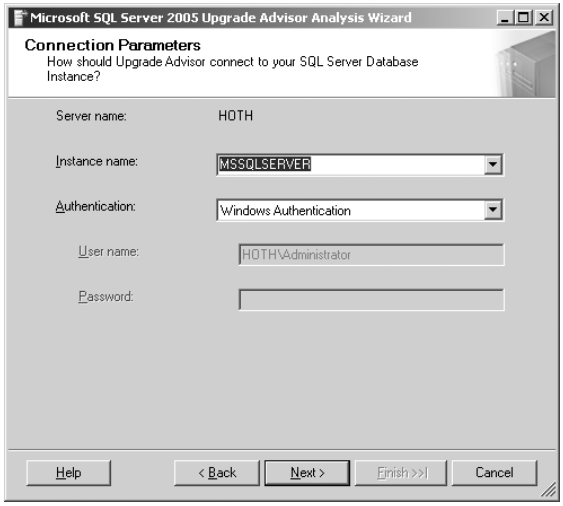

Figure 8-18 SQL Server 2005 Upgrade Advisor—Connection Parameters page.

5. The SQL Server Parameters page appears, as shown in Figure 8-19. Select the check boxes for the databases to be analyzed. Additionally, if you want to analyze trace files and SQL batch files, select the appropriate check boxes as well. Click Next.

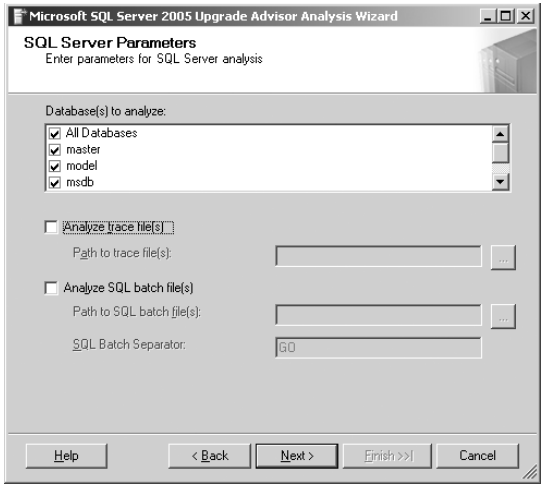

Figure 8-19 SQL Server 2005 Upgrade Advisor—SQL Server Parameters page.

6. Based on the components you selected for analysis in step 3, the appropriate component parameter page displays. Enter the requested information for each, and then click Next.
7. The Confirm Upgrade Advisor Settings page appears, as shown in Figure 8-20. Review the information and click Run to execute the analysis process. You can select the Send Reports To Microsoft check box if you want to submit your upgrade report to Microsoft. Re-executing the Upgrade Advisor process causes any previous reports to be overwritten.

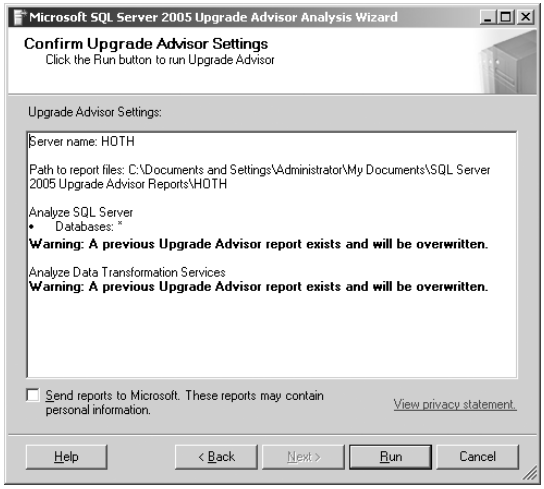

Figure 8-20 SQL Server 2005 Upgrade Advisor—Confirm Upgrade Advisor Settings page.

8. The Upgrade Advisor Progress page appears, as shown in Figure 8-21. The analysis may take several minutes to complete and is dependent on the number of components selected. Once the analysis is completed, you can select Launch Report to view the report that was generated or exit the wizard by clicking Close.

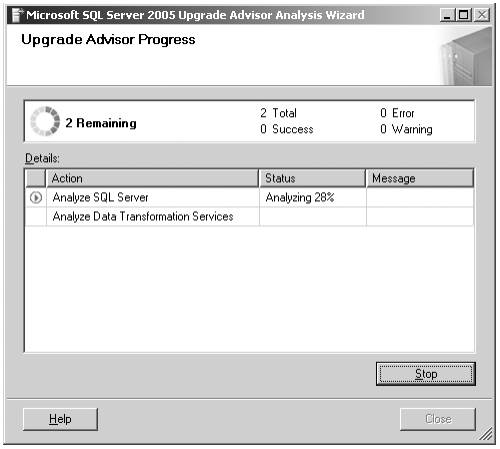

Figure 8-21 SQL Server 2005 Upgrade Advisor—Upgrade Advisor Progress page.

9. When you select Launch Report in step 8 or the Launch Upgrade Avisor Report Viewer option shown in Figure 8-16, the window shown in Figure 8-22 appears. From this window, you can choose to view all the issues for all the components together or filter the view using the Instance Or Component and Filter By drop-down lists. Clicking on the + next to a line item expands the display to show a more detailed description. You also can use the This Issue Has Been Resolved check box on each line item to mark the task resolved, which will then delete it from the current report.

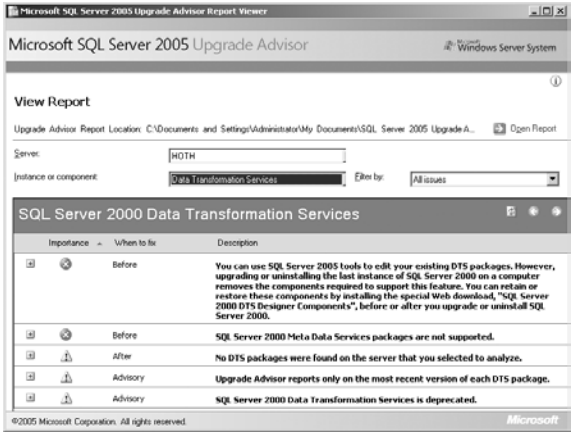

Figure 8-22 SQL Server 2005 Upgrade Advisor—View Report window.

Upgrade Procedure

The procedure to upgrade an earlier version of SQL Server to SQL Server 2005 is very similar to the procedure for a new SQL Server 2005 installation. To upgrade a version of SQL Server 7.0 or SQL Server 2000, start with steps 1 through 9 listed in the section "Installing SQL Server 2005 Using the Installation Wizard." Then follow these steps:

1. When the Instance Name page appears, select the default or named instance to upgrade. To upgrade a SQL Server default instance already installed on your system, click Default Instance, and then click Next to continue. To upgrade a SQL Server named instance already installed on your system, click Named Instance, and then enter the instance name in the text field below, or click the Installed Instances button, select an instance from the Installed Instances list, and click OK to automatically populate the instance name text field. After you have selected the instance to upgrade, click Next to continue.

   Note If you want to do an upgrade, make sure you specify the name of the existing default or named instance correctly. If the instance specified does not exist on the system, the Installation Wizard will install a new instance instead of performing an upgrade.

2. The Existing Components page, shown in Figure 8-23, appears. On this page, you can select the check boxes next to the components you want to upgrade (the list of components is based on the SQL Server instances and versions installed on your system and, therefore, may be different than what is shown in Figure 8-23). You can also view the details of the listed components by clicking on the Details button in the lower-right corner. Click Next to continue with the upgrade.

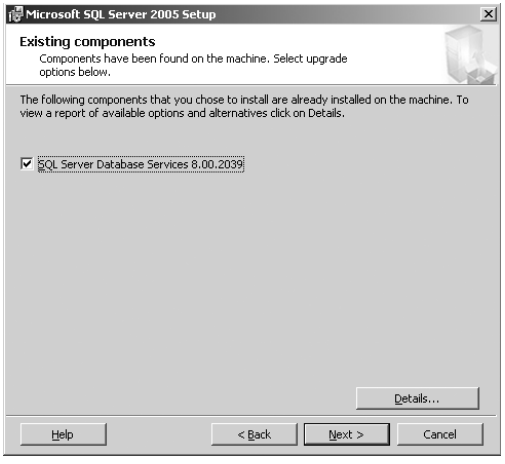

Figure 8-23 SQL Server Upgrade—Existing Components page.

Troubleshooting You should make sure that the SQL Server 2005 edition to which you're trying to upgrade is listed in Table 8-7 as a valid upgrade path. If not, the setup process will block the upgrade by graying out the component selection check boxes.

3. The Upgrade Logon Information page appears. On this page, click the appropriate radio button to select either the Windows Authentication Mode or the Mixed Mode (Windows Authentication and SQL Server Authentication). If you select to use the mixed mode, you will need to enter and confirm login password for the sa user. Click Next to continue.

4. The upgrade process will analyze the instance and then display the Error And Usage Report Settings page. On this page, you can select one of two radio buttons to Automatically Send Error Reports To Microsoft Or Your Corporate Error Reporting Server, or to Automatically Send Feature Usage Data For SQL Server 2005 To Microsoft, to set the desired default action. This data is collected for information purposes only, and selecting either of these options will not have any adverse effects on the performance of your system. Click Next to continue.

5. On the Ready To Install page, review the components selected for upgrade, and then click Install to upgrade them.

Post-Upgrade Steps

The procedure explained previously upgrades the SQL Server database executables to SQL Server 2005; however, this may not be sufficient to ensure optimal performance and functioning of your application. In addition to upgrading to SQL Server 2005, you will need to complete the following tasks manually to upgrade your individual databases and do some housekeeping tasks:

1. **Register servers** After upgrading to SQL Server 2005, you must reregister your servers.

2. **Set database compatibility to 90** After an upgrade, SQL Server 2005 automatically sets the database compatibility level for each database to the level of the previous SQL Server version. For example, if you upgrade SQL Server 2000 to SQL Server 2005, the database compatibility level for all the upgraded user databases will be set to 80 (SQL Server 2000). You should change the database compatibility level for each of your databases to SQL Server 2005 (90) by executing the *sp_dbcmptlevel* stored procure command shown below from one of the SQL Server tools like SQL Server Management Studio.

```
sp_dbcmptlevel
```

Best Practices You should always run your databases in the 90 compatibility level for SQL Server 2005 and avoid setting the compatibility level to an earlier version to permanently work around any incompatibilities you encounter after an upgrade.

3. **Execute update statistics** You should update statistics on all tables. This ensures that the statistics are current and help optimize query performance. You can use the *sp_updatestats* stored procedure to update the statistics on all user tables in your database.

4. **Update usage counters** You should run DBCC UPDATEUSAGE on all upgraded databases to correct any invalid row or page counts, for example DBCC UPDATEUSAGE ('AdventureWorks').

5. **Configure the surface area** You should enable the required SQL Server 2005 features and services using the SQL Server 2005 Surface Area Configuration tool explained later in this chapter.

6. **Repopulate full-text catalogs** The upgrade process disables full-text on all databases. If you plan to use the full-text feature, you should repopulate the catalogs. You can do this using the *sp_fulltext_catalog* stored procedure.

Reading the SQL Server 2005 Setup Log Files

SQL Server 2005 setup has a significantly enhanced logging mechanism wherein all actions performed by setup are logged in an easy-to-read format. The master log file for the setup process is named Summary.txt and is located under: %ProgramFiles%\Microsoft SQL Server\90\Setup Bootstrap\LOG\. This file contains a summary for each component being installed. The following is a typical Summary.txt log file fragment.

```
Microsoft SQL Server 2005 9.00.1399.06
==============================

OS Version      : Microsoft Windows Server 2003 family, Enterprise Edition
Service Pack 1 (Build 3790)
Time            : Thu Jan 12 22:38:12 2006
```

```
Machine         : HOTH
Product         : Microsoft SQL Server Setup Support Files (English)
Product Version : 9.00.1399.06
Install         : Successful
Log File        : C:\Program Files\Microsoft SQL Server\90\Setup
Bootstrap\LOG\Files\SQLSetup0003_HOTH_SQLSupport_1.log
------------------------------------------------------------------------------
Machine         : HOTH
Product         : Microsoft SQL Server Native Client
Product Version : 9.00.1399.06
Install         : Successful
Log File        : C:\Program Files\Microsoft SQL Server\90\Setup
Bootstrap\LOG\Files\SQLSetup0003_HOTH_SQLNCLI_1.log
------------------------------------------------------------------------------
Machine         : HOTH
Product         : Microsoft Office 2003 Web Components
Product Version : 11.0.6558.0
Install         : Successful
Log File        : C:\Program Files\Microsoft SQL Server\90\Setup
Bootstrap\LOG\Files\SQLSetup0003_HOTH_OWC11_1.log
------------------------------------------------------------------------------
…
```

You can use the Summary.txt file to examine the details of a component installation process by referring to the log file name listed on the respective Log File line. This is particularly useful when a component fails to install and the installation process needs to be debugged. The individual component log files are created in text format and stored in the %ProgramFiles%\Microsoft SQL Server\90\Setup Bootstrap\LOG\Files directory.

Uninstalling SQL Server 2005

Similar to the installation process, SQL Server 2005 can be uninstalled using either an uninstall wizard or the command prompt. The following sections explain both of these methods in detail.

Uninstalling SQL Server 2005 Using the Uninstall Wizard

1. To begin the uninstall process, click the Start button, select Control Panel (or select Settings and then Control Panel), and then in Control Panel, double-click Add Or Remove Programs.

2. In the left pane, click Add/Remove Windows Components.

3. Select the SQL Server 2005 component to uninstall. The Change and Remove buttons are then displayed. Click Remove. This starts the SQL Server 2005 Unistall Wizard.

4. The Component Selection page, shown in Figure 8-24, is displayed. On this page, you can select the installed instance and common components you'd like to uninstall. You can select the Report button to view the list of SQL Server 2005 components and features installed on your computer. The report displays the versions, the editions, any updates, and the language information for each installed component and feature. Click Next.

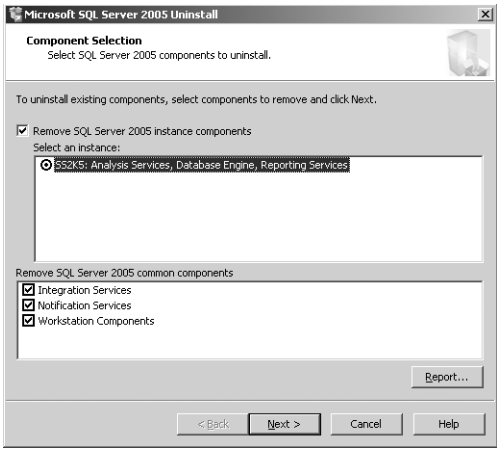

Figure 8-24 SQL Server 2005 Uninstall—Component Selection page.

5. The Confirmation page is displayed. Review the list of components and features that will be uninstalled.

6. Click Finish to uninstall the selected components. The Setup Progress window appears and displays the uninstall status for each component. When the uninstall process is completed, the window will close automatically.

Note The Add Or Remove Programs window may continue to display some of the components as installed even though they've been uninstalled. This is because the Add Or Remove Programs window does not auto-refresh. The easiest way to refresh the window is to close it and then click Add Or Remove Programs in the Control Panel again.

Uninstalling SQL Server 2005 Using the Command Prompt

As mentioned earlier, SQL Server 2005 can be uninstalled from the local server by specifying the REMOVE option in command prompt. When the option is specified with the ALL parameter, all the instance aware components are uninstalled. For example, the following command uninstalls all components from an instance called SS2K5 on the local server.

```
start /wait <DVD Drive>\Servers\setup.exe /qb REMOVE=ALL INSTANCENAME=SS2K5
```

Note To uninstall the default instance, specify INSTANCENAME=MSSQLSERVER.

The command prompt can also be used to selectively uninstall specific components of a SQL Server 2005 instance. For example, the following command uninstalls the Analysis Server components from the default instance of SQL Server on the local server in silent mode with no GUI displayed (/qn).

```
start /wait <DVD Drive>\Servers\setup.exe /qn REMOVE=
Analysis_Server,AnalysisDataFiles INSTANCENAME=MSSQLSERVER
```

Note The REMOVE option can also be used in conjunction with the ADDLOCAL option. While these two are seemingly orthogonal actions, they can be used very effectively to simplify the installation command. For example, to install all the components of SQL Server 2005 except Notification Services, you can install all the components (ADDLOCAL=ALL) and use the REMOVE option to exclude Notification Services, as shown in the following example query:

```
start /wait <DVD Drive>\Servers\setup.exe /qn ADDLOCAL=ALL
REMOVE=Notification_Services INSTANCENAME=MSSQLSERVER
```

An alternative is to specify all the components of SQL Server 2005 except Notification Services individually as comma-separated parameters to the ADDLOCAL option.

These commands do not uninstall the SQL Native Access Client (SNAC) component from the server. To uninstall SNAC, execute the command (where C is the boot drive).

```
start /wait C:\Windows\System32\msiexec /qb /X <DVD
Drive>\Servers\setup\sqlncli.msi
```

Using SQL Server Surface Area Configuration

SQL Server 2005 by default disables some features, services, and connections for new installations in order to reduce the attackable surface area and, thereby, help protect your system. This security scheme is new in SQL Server 2005 and is very different from earlier versions, which by default always enable all installed components.

The SQL Server Surface Area Configuration tool is a new configuration tool that ships with SQL Server 2005 and can be used to enable, disable, start, or stop features, services, and remote connectivity. This tool provides a single interface for managing Database Engine, Analysis Services, and Reporting Services features and can be used locally or from a remote server.

The following steps list the process used to invoke and use the SQL Server Surface Area Configuration tool:

1. Click the Start button and point to All Programs. Point to Microsoft SQL Server 2005, select Configuration Tools, and then select SQL Server Surface Area Configuration.
2. The SQL Server Surface Area Configuration start window, shown in Figure 8-25, appears. From this window, you can specify the server you want to configure by selecting the link adjacent to Configure Surface Area For Localhost. In the Select Computer dialog box that appears, select Local Computer or Remote Computer and enter the name of the remote computer in the text box if necessary. Click OK to continue.

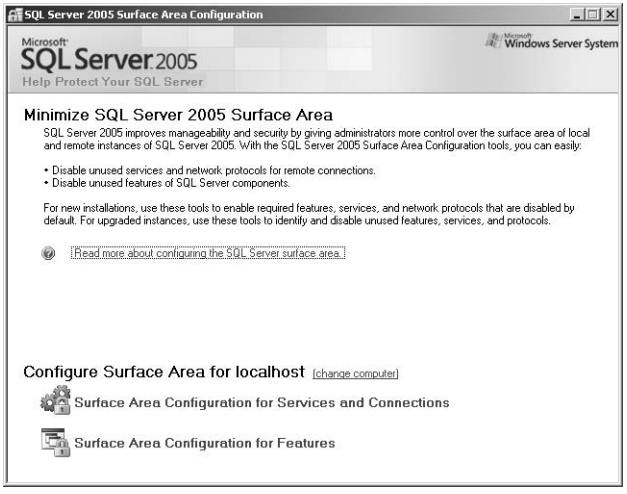

Figure 8-25 SQL Server Surface Area Configuration—Start window.

3. Click the Surface Area Configuration For Services And Connections link to enable or disable Windows services and remote connectivity or the Surface Area Configuration For Features link to enable or disable features of the Database Engine, Analysis Services, and Reporting Services.
4. Click the Surface Area Configuration for Services and Connections link to set the startup state (Automatic, Manual, or Disabled) for each of the installed services and Start, Stop, or Pause the respective service. In addition, you can use this link to manage the connectivity options by specifying whether local connections only or local and remote connections are permitted, as shown in Figure 8-26.

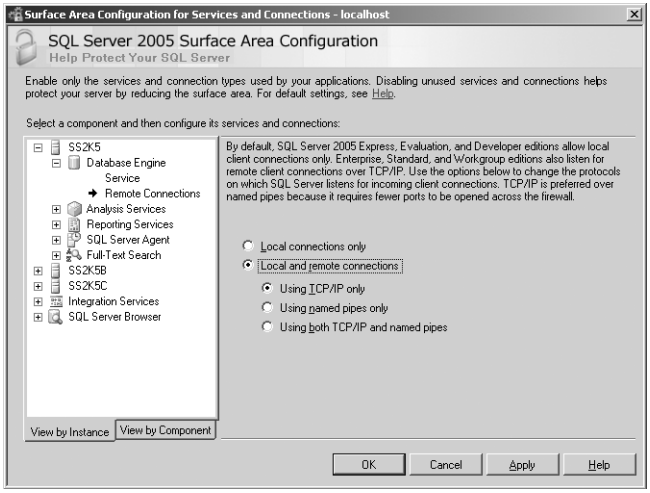

Figure 8-26 Surface Area Configuration For Services And Connections dialog box.

Real World Error While Connecting Remotely

I have found that one of the most common problems for folks using SQL Server 2005 Express, Evaluation, or Developer Editions is not being able to connect to the server from a remote system. The error message returned when trying to connect remotely is as follows:

"An error has occurred while establishing a connection to the server. When connecting to SQL Server 2005, this failure may be caused by the fact that under the default settings SQL Server does not allow remote connections. (provider: SQL Network Interfaces, error: 28 - Server doesn't support requested protocol) (Microsoft SQL Server, Error: -1)"

As mentioned earlier, SQL Server 2005 by default disables several network protocols to reduce the possible attack surface area. In line with this principle, the SQL Server 2005 Express, Evaluation, and Developer Editions disallow remote connections to the server, and that is why you cannot connect to the server remotely. You can easily remedy the situation by using the SQL Server Surface Area Configuration tool, selecting Remote Connections, and then selecting the Local And Remote Connections and Using Both TCP/IP /IP And Named Pipes options.

5. Click the Surface Area Configuration For Features link. The window shown in Figure 8-27 appears. This window provides a single interface for enabling or disabling the installed components listed in Table 8-6.

Note To configure a component, the component has to be running. If the component is not running, it will not be displayed as shown in Figure 8-27.

Table 8-6 Surface Area Configuration for Features

Component	Configurable Feature
Database Engine	Ad hoc remote queries
	CLR integration
	DAC
	Database Mail
	Native XML Web Service
	OLE Automation
	SQL Server Service Broker
	SQL Mail
	Web Assistant
	xp_cmdshell
Analysis Services	Ad hoc data mining queries
	Anonymous connections
	Linked objects
	User-Defined Functions
Reporting Services	Scheduled Events and Report Delivery
	Web Service and HTTP Access
	Windows Integrated Security

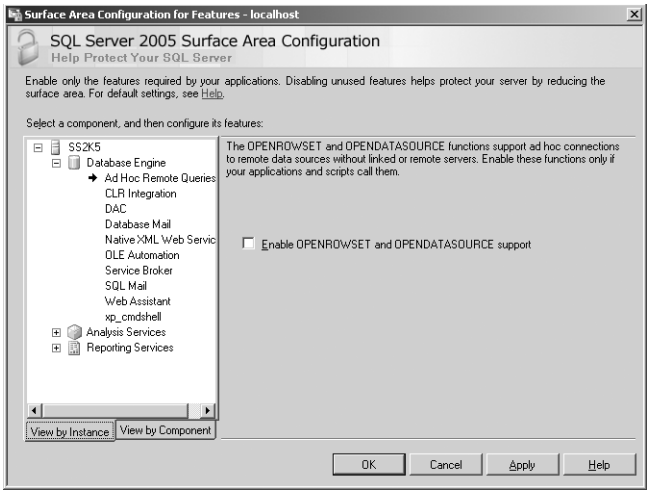

Figure 8-27 Surface Area Configuration for Features window.

Best Practices These days security is a major concern for almost all deployments. To reduce the surface area for a possible attack, be selective about the features you enable and have a policy for enabling only those that you plan to use. It is also worthwhile to use the SQL Server Surface Area Configuration tool periodically and disable features that you are no longer using.

sac Utility

The *sac utility* can be used to import or export Microsoft SQL Server 2005 surface area settings. This utility is very useful in cases where the same surface area configuration needs to be replicated on multiple servers. To configure multiple servers with the same setting, you can configure the surface area on one server using the graphical SQL Server Surface Area Configuration tool and then use the *sac* utility to export the setting to a file. This file can then be used to import the setting into remote servers using the same utility. The *sac* utility (Sac.exe) is located under the directory:

```
%Program Files%\Microsoft SQL Server\90\Shared
```

The following command can be used to export the surface area configuration settings for all default instance settings of a server named HOTH into an XML-formatted file called sacSetting.txt.

```
sac out C:\sacSettings.txt -S HOTH -U sa -P Pa55wD -I MSSQLSERVER
```

This file can then be imported into some other server using the in option. The following command imports the sacSettings.txt file into a server called NABU.

```
sac in C:\sacSettings.txt -S NABU
```

More Info The *sac* utility is very powerful and provides the flexibility of exporting and importing settings for specific services as well as Features and Network settings. The entire list of options for this utility can be found at *http://msdn2.microsoft.com/en-us/library/ms162800.aspx*.

Summary

Installing the software is the first step towards using SQL Server 2005. Although it is a relatively easy task, it is important to do preinstallation planning, select the correct installation options, and perform all of the postinstallation configuration steps to ensure an optimal installation and to avoid having to make repairs or patches postinstallation.

In this chapter, you've learned about the Installation Wizard and command prompt-based options available for installing the SQL Server 2005 components. You also learned about the new SQL Server 2005 Upgrade Advisor and how it can assist you in ensuring a smooth upgrade process, and how the SQL Server Surface Area Configuration tool and *sac* utility can be used to configure SQL Server 2005.

Chapter 21

Integration Services

Moving data between data stores can sometimes be a challenging task, especially when multiple data structures are involved or data needs to be changed before it's placed in a new location. The goal of SQL Server 2005 Integration Services, also known as Integration Services or SSIS, is not only to simplify such data transfer task but also to provide rich functionality in support of more complex extract, transform, and load (ETL) requirements for data warehousing. The goal of this chapter is to introduce you to the key concepts of Integration Services. We'll explore how to build basic packages using the new design environment and how to use some of this product's features to monitor, troubleshoot, and log package execution. We'll also take a look at the administrative tasks associated with Integration Services.

What Is Integration Services?

Integration Services is the data integration component of SQL Server 2005. Data integration activities can be as simple as moving data between data sources or as complex as consolidating large volumes of data from multiple data sources in different formats, applying rules to modify or cleanse data content, and loading the resulting data into data warehouses designed for reporting and analytical applications. Even if you're not responsible for creating and maintaining a data warehouse, you'll find the features in Integration Services quite useful for routine database administration tasks and any activities requiring you to move data in any form.

Integration Services Versus Data Transformation Services

Integration Services is the successor to Data Transformation Services (DTS), a data utility first introduced in SQL Server 7.0. DTS simplified the process of migrating data between database systems and was even used as an ETL tool in many data warehouse implementations. Despite its extensive capabilities, DTS had some limitations in performance, portability, and manageability that led Microsoft to develop an entirely new architecture for improved data integration functionality in SQL Server 2005.

Some Important Enhancements

Ask ten DBAs who have started using Integration Services which enhancements are the most important, and you're likely to get ten different lists. The official comprehensive list of enhancements can be found in the SQL Server Books Online topic, "Integration Services Enhancements." In this section, we'll limit the discussion to the features that typically get strong positive reactions from DBAs during their introduction to Integration Services.

If you were already satisfied with DTS, you're going to be happier with Integration Services. For example, during the development cycle, you can take advantage of the debugging capabilities to monitor package execution more efficiently. If you're part of a team, you can use source control, such as Microsoft Visual SourceSafe, for your data integration projects to manage collaboration within the group. Building complex workflows that included loops or conditional branches was not impossible in DTS, but it was challenging; now Integration Services provides special components to handle these workflow requirements. When you're ready to move your work from one server to another, such as migrating from development to production, you'll find deployment is much simpler in Integration Services. Of course, there are many other features, discussed later in this chapter, which you'll also appreciate.

If, on the other hand, the limitations of DTS kept you from making use of this tool, now's the time to see how Integration Services measures up to your data integration requirements. The biggest improvement is the new architecture of Integration Services replacing the DTS Data Pump with an engine optimized to process and transport large data volumes more efficiently and much faster. Another important new feature is the checkpoint support, which allows you to restart data integration processes from a specific checkpoint rather than the beginning of the package. You also have more options for responding to events during execution and for logging results during execution.

Migration from DTS to Integration Services

Several key concepts and terminology applicable to DTS still exist in Integration Services. A *package*, for example, is still an executable unit of work that encapsulates the objects that connect to data sources, perform tasks, transform data, and manage workflows. However, because the new architecture of Integration Services has replaced or eliminated

DTS objects, you need to recreate your packages in Integration Services to benefit from its scalability and performance features. To support backward compatibility, all SQL Server 2005 editions (except Express) include the SQL Server 2000 DTS run-time engine. Consequently, you can use Integration Services to develop new packages, convert those DTS packages that will benefit most from the performance enhancements of Integration Services, and then selectively convert the remaining DTS packages at your convenience.

Important If you need to edit DTS packages after upgrading the server to Integration Services but before converting the packages, download the Microsoft SQL Server 2000 DTS Designer Components available at *http://www.microsoft.com/downloads/details.aspx?familyid=D09C1D60-A13C-4479-9B91-9E8B9D835CDC&displaylang=en*. There are several items available for download. Search for SQLServer2005_dts.msi to locate the applicable download link.

To convert DTS packages, use the Package Migration Wizard. This utility does its best to create an Integration Services package that matches the original DTS package, but changes to the underlying object model and the run-time engine in Integration Services mean that any DTS object model references as well as certain tasks cannot be converted. Before you use the Package Migration Wizard, review the information in this section to understand both what it will and what it won't do.

The following list outlines the DTS tasks that are replaced by Integration Services tasks without requiring additional package modifications:

- *ActiveX*[1]
- *Bulk Insert*
- *Copy SQL Server Objects*[2]
- *Data Mining Prediction* [3]
- *Execute Package*[4]
- *Execute Process*
- *Execute SQL*
- *File Transfer Protocol*
- *Message Queue*
- *Send Mail*

1 ActiveX script converts successfully only if it contains no references to the DTS object model.

2 *Transfer SQL Server Objects* in SSIS

3 *Data Mining Query* in SSIS

4 *Execute DTS 2000 Package* in SSIS

- *Transfer Databases*
- *Transfer Error Messages*
- *Transfer Jobs*
- *Transfer Logins*
- *Transfer Master Stored Procedures*

For DTS tasks that cannot be migrated at all, described in the following list, the Package Migration Wizard inserts an *Execute DTS 2000 Package Task* to encapsulate the functionality for backward compatibility which you can replace with Integration Services tasks or leave as is:

- *Analysis Services Processing*[5]
- *Custom*
- *Data Pump*
- *Data Driven Query*
- *Dynamic Properties*
- *Parallel Data Pump*
- *Transform Data*

You need to modify the migrated package to provide replacement functionality for the deprecated DTS tasks described here:

- **Data Driven Query** Consider replacing with a *Conditional Split Transformation*, an *OLE DB Command Transformation*, or a *Slowly Changing Dimension Transformation*.
- **Dynamic Properties** Use property expressions or package configurations.
- **Transform Data** Create a *Data Flow Task* with any data flow components that replicate the transformation.

Several package-specific items won't migrate. For example, even though SQL Server 2005 allows you to use package passwords with the *Execute DTS 2000 Package Task*, the package password on the DTS package migrates only to the Execute DTS Package tasks and does not migrate to the Integration Services package itself. If you have a package that does not contain *Execute DTS 2000 Package Tasks* after migration, the original package password is not migrated. (In all cases, you will be prompted for the package password before you can migrate the package.) Therefore, if you want a package password on the migrated package, you must manually add the new password to the

5 The SSIS *Analysis Services Execute DDL Task* and *Analysis Services Processing Task* cannot interact with an Analysis Services 2000 database.

PackagePassword package property. Also, if you've added annotations, package logging, or error handling to your DTS package, you'll need to add them manually using Integration Services features.

There is no absolute rule you can apply to determine whether you should rebuild your DTS packages manually or try the migration. If your packages are relatively straightforward, you can get much accomplished by choosing migration. To use the Package Migration Wizard, follow these steps:

1. Click the Start button, and then point to All Programs. Point to Microsoft SQL Server 2005, and then click SQL Server Business Intelligence Development Studio.
2. On the File menu, point to New, and then click Project.
3. Ensure that the Project Type is set to Business Intelligence Projects, which is the default selection, and then click the Integration Services Project template.
4. Type a name for the project, specify a folder used for storing project-related files, and then click OK.
5. In the Solution Explorer window, right-click the SSIS Packages folder, click Migrate DTS 2000 Packages, and then click Next on the Welcome page of the wizard.

> **Note** To launch the Package Migration Wizard from SQL Server Management Studio, connect to the Database Engine, expand the Management folder and then the Legacy folder, right-click the Data Transformation Services folder, and then click Migration Wizard.

6. On the Choose Source Location page of the wizard, specify the storage location of the package. If the package is stored in SQL Server 2000, provide a server name and, if required, SQL Server credentials. If the package is stored as a Structure Storage File, select this option in the Source drop-down list and provide the location of the package.
7. Click Next to display the Choose Destination Location page, and then specify a location where the wizard will store the converted package.
8. Click Next to display the List Packages page, and then select the check box next to each package to convert.You can provide a new name for each selected package by typing the name in the Destination Package column.
9. Click Next to display the next page of the wizard. If the original package does not require a password, continue to the next step. If, on the other hand, the package requires a password, you see the Package Authentication page of the wizard. Select

the package in the Encrypted Packages list, and then click Password. Type the package owner password in the Package Password dialog box, and then click OK. Click Next to continue the wizard.

10. On the Specify A Log File page, specify a location for the log file.
11. Click Next to display the Complete The Wizard page, and then click Finish. The status of the package migration displays in the Migrating The Packages page of the wizard. Click Close.
12. The new package appears in the Solution Explorer window. Double-click the package name to open the package in the development environment.

Review the workflow to locate tasks that might not have converted successfully and tasks that have been stored in *Execute DTS 2000 Package Task*. You'll need to decide on a task-by-task basis how to resolve these migration issues.

Integration Services Fundamentals

Integration Services shares some common characteristics with DTS. These characteristics are common in name only because the underlying architecture of these objects have been redesigned. A package is still the main container of objects defining which operations, known as *tasks*, should be performed. The sequence of operations is defined by *precedence constraints*. Optionally, you can use a *container*, a new object in Integration Services, to group tasks and other containers.

Each of these objects—containers, tasks, and precedence constraints—is an element of the *control flow* architecture new to Integration Services which manages the run-time activities of each package. The Integration Services run-time engine also provides services to each package, such as connection management, transaction commitment, debugging, logging, event handling, and variable management. We'll take a closer look at these services later in this chapter.

Another key component in the Integration Services architecture is the *data flow* engine, also known as the *data pipeline*. Data flow activities are encapsulated within the *Data Flow Task*, an object for which the control flow engine merely provides operational support and to which precedence constraints are applied just like any other task. For its execution, however, the *Data Flow Task* relies on a separate engine that performs the data extractions, manipulates the data according to defined transformations, and then deposits the resulting data set into a destination.

To achieve maximum performance, the data flow engine uses buffers to manipulate data in memory. Source data—whether it's relational, structured as XML data, or stored in flat files like spreadsheets or comma-delimited text files—is converted into a table-like

structure containing columns and rows and then loaded directly into buffers without staging the data first in temporary tables. *Transformations* are data flow objects that operate on the buffered data, by sorting, merging, modifying, or enhancing the data before sending it to the next transformation or to its final destination. By avoiding the overhead of reading from and writing to disk, the processes required to move and manipulate data can operate at optimal speed. Integration Services can even provide data directly to an application by storing it in an ASP.NET DataReader. Using this method, you don't have to place the data in a persistent data store, an important step towards enabling near-real-time data delivery.

Integration Services Components Overview

Integration Services includes several components, tools, and utilities for developing, deploying, and managing packages. The following list introduces these items, many of which are explained in more detail later in this chapter:

- **Integration Services Service** Manages storage of packages in .dtsx files or in the MSDB database and monitors their execution
- **Integration Services Runtime** Saves package layout, applies configurations, executes packages, manages connections and transactions, and supports logging and debugging
- **Integration Services Runtime Executables** Includes package, containers, tasks, custom tasks, and event handlers
- **Integration Services Data Flow** Manages data sources, data destinations, transformations, and any custom components you add to a *Data Flow Task*, and provides the in-memory buffers used to transport data from source to destination
- **Business Intelligence Development Studio** Supports the development and testing of Integration Services packages; supports collaboration with source code management and version control; provides debugging tools such as breakpoints, variable watches, and data viewers; and includes the SQL Server Import and Export Wizard to jumpstart package development
- **SQL Server Management Studio** Provides access to the Execute Package utility, imports and exports packages to and from available storage modes (MSDB database or SSIS Package Store), and monitors running packages
- **SQL Server Import and Export Wizard** Copies data from one location to another
- **Package Migration Wizard** Converts SQL Server 2000 DTS packages to Integration Services packages

- **Package Installation Wizard** Deploys an Integration Services project prepared by a deployment utility
- **Execute Package Utility (dtexecui)** Provides a user interface for preparing packages for execution as well as executing packages
- **dtexec** Runs a package at the command prompt
- **dtutil** Provides package management functionality at the command prompt to copy, move, or delete packages or to confirm a package exists
- **Integration Services Object Model** Includes application programming interfaces (APIs) for customizing run-time and data flow operations and automating package maintenance and execution by loading, modifying, and executing new or existing packages programmatically

Designing Packages

Whether you're migrating a DTS package or creating a new Integration Services package, you use SQL Server Business Intelligence Development Studio. This development environment is rich with features that can initially seem overwhelming but is much easier to navigate once you understand how package objects are classified and how they fit together to achieve a data integration goal. In this section, we'll focus on commonly used components and important features of the development environment.

The Development Environment

SQL Server Business Intelligence Development Studio is the graphical interface you use to design, test, and execute a package. While Business Intelligence Development Studio appears to be a distinct application, it's actually just a shortcut to Microsoft Visual Studio 2005. If you already use Visual Studio, then the installation of Integration Services integrates the business intelligence *designers* with your existing development environment. A designer is a collection of design tools, including a workspace, toolbox, dialog boxes, and various windows, used to build a project.

Starting an Integration Services Project

When you work in Visual Studio, you work in the context of a solution and a project. A solution is simply a container for one or more projects. Each project can be a different type. For example, you can have a single solution that contains an Integration Services project that includes all of the packages used to perform the ETL processes for a data warehouse, an Analysis Services project that uses data from that data warehouse, and

a Reporting Services project with all reports used to present Analysis Services data. You organize projects into solutions to make it easier to save and retrieve files that might be related.

An Integration Services project contains four folders: Data Sources, Data Source Views, SSIS Packages, and Miscellaneous. At a minimum, you'll add one or more package files to the SSIS Packages folder, but the other folders are optional. To add a new package to the SSIS Packages folder, follow these steps:

1. Start SQL Server Business Intelligence Development Studio.
2. On the File menu, point to New, and then click Project.
3. Ensure that the Project Type is set to Business Intelligence Projects, and then click the Integration Services Project template.
4. Type a name for the project, **My Package**, change the location for the project to a folder of your choice, and confirm the Create Directory For Solution check box is selected.
5. Click OK to continue to create the package in the Integration Services package designer, as shown in Figure 21-1.

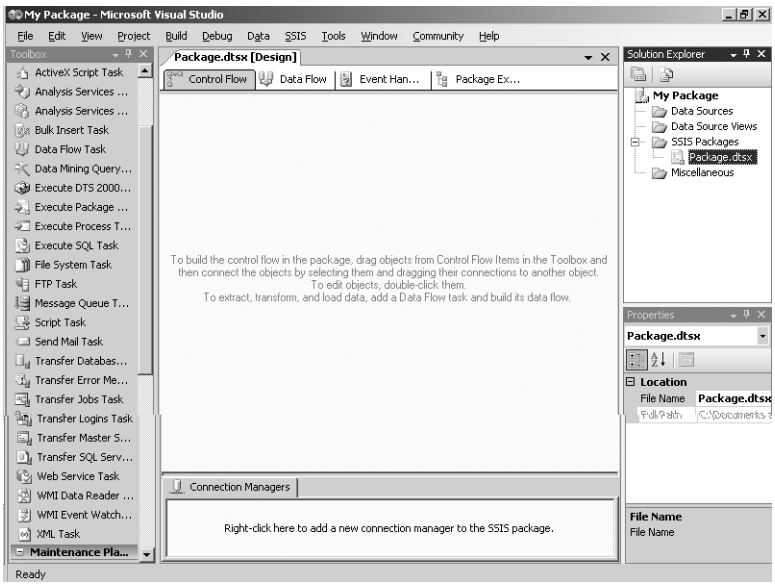

Figure 21-1 A new package in the Integration Services package designer.

The main Visual Studio windows you'll use during package development are described in the following list:

- **Solution Explorer** By default, this window displays in the upper right corner of your screen. When you add a new package to your project, you'll see a file like Package.dtsx added to the SSIS Packages folder. You can rename this file, but be sure to keep the .dtsx extension or your package won't work.
- **Properties** This window displays in the bottom right corner of your screen by default. If you select an object added to your package, you can view and modify its properties in this window.
- **Toolbox** This windows displays by default on the left side of your screen. The contents of the toolbox change according to the current workspace visible in the designer.
- **Integration Services Package Designer** The designer is the main window that represents your workspace. There are five tabs in the designer window that you use while building and testing your package: (Only four of these tabs are shown in Figure 22-1 because the fifth tab, Progress, appears only after you execute the package.)
- **Control Flow** Use this tab to manage tasks, containers, and precedence constraints in the package workflow.
- **Data Flow** Use this tab to define the flow of data between sources and destinations, as well as any transformations applied to the data during transfer.
- **Event Handlers** Use this tab to add tasks that execute only when specific events have occurred, such as a task failure.
- **Package Explorer** Use this tab to review package objects in a tree view and to access object properties.
- **Progress** Use this tab during or after executing a package within the development environment to review the status of executables, the duration of completed executables, and any errors or warnings generated during package execution. This tab appears only when you execute the package.

Using Data Sources

A *data source* is a file that contains the connection string and credentials used by Integration Services to connect to a data store. Right-click the Data Sources folder and click New Data Source to start the Data Source Wizard which steps you through building a data source. When you have many packages that extract or load data into the same data store, you can use a data source to define the connection once and reuse it many times. If you

modify the connection string in the data source, the connection managers are similarly updated. The data source is used only during development and is not saved with the package when you put it into production. Instead, the value of the connection string defined by the data source when the package is saved is stored in the connection manager.

Using Data Source Views

A *data source view* (DSV) is a file that shows selected tables or views from a data source in your project. Right-click the Data Source Views folder and click New Data Source View to start the Data Source View Wizard, which steps you through the selection of a data source and specification of the database objects you want to include in the DSV. You can add relationships between tables or add custom expressions to the DSV to create derived columns, which is handy if you don't have permissions to change the data structure at the source. When you use the DSV with a data flow component, the package designer translates the relationships and custom expressions into an SQL statement that is stored with the component.

Using the SQL Server Import and Export Wizard

The SQL Server Import and Export Wizard is designed to copy data from one database to another or to load data into SQL Server databases from flat files, spreadsheets, and OLE DB sources when an ETL solution is not required. To use the SQL Server Import and Export Wizard for building an Integration Services package, follow these steps:

1. To follow this example, you must complete the steps described in the previous example. Right-click the SSIS Packages folder, and then click SSIS Import And Export Wizard.
2. Click Next to bypass the Welcome page and to display the Choose A Data Source page.
3. Select a data source from the Data Source drop-down list. The default data source is SQL Native Client. For this example, select *AdventureWorks* in the Database drop-down list. Click Next.

   You can choose from the following additional data source options:

   - .NET Framework Data Providers (ODBC, Oracle, and SQL Server)
   - Flat File Source
   - Microsoft Access
   - Microsoft Excel
   - Microsoft OLE DB Providers (Analysis Services 9.0, Data Mining Services, OLAP Services 8.0, Oracle, and SQL Server)

> **Note** You might see additional Microsoft OLE DB Providers listed if you have applications installed on your machine that include an OLE DB provider, such as Microsoft Project 9.0, for example.

- SQLXMLOLEDB and SQLXMLOLEDB 4.0

When you select a different data source, the Choose A Data Source page changes to prompt you for options required to connect to the selected data source. For example, the .NET Framework Data Provider for ODBC requires you to provide a connection string, a Data Source Name (DSN), and the name of the ODBC driver to use.

4. On the Choose A Destination page, you select a destination for the data you're importing from the selected data source. The data destination options are the same as the types that you can choose as a data source. You can even choose a destination that doesn't exist yet. When you create a new database, you can specify options for Data File Size and Log File Size, which are explained in more detail in Chapter 10, "Creating Databases and Database Snapshots." In this example, we'll specify a new database, *MyAdventureWorks*. Click the New button, type **MyAdventureWorks** in the Name text field, and then click OK.

5. Set the authentication type applicable to your server configuration: Windows or SQL Server authentication. If you use SQL Server authentication, you must type a SQL Server username and password in the appropriate text box. Click Next.

6. On the Specify Table Copy Or Query page, you choose how you want to define which data to extract from the data source. Select Copy Data From One Or More Tables Or Views when you want all data from tables and views that you select on a subsequent page of the wizard. If you need only a subset of data, such as fewer rows or fewer columns, select the Write A Query To Specify The Data To Transfer option. For this example, leave the default option, Copy Data From One or More Tables Or Views, selected. Click Next.

> **Note** Users of SQL Server 2000 DTS notice right away that the option Copy Objects And Data Between SQL Server Databases is missing from this page of the wizard. Instead, open SQL Server Management Studio, right-click the database to copy, point to Tasks, and then click Copy Database to start the Copy Database Wizard.

7. On the Select Source Tables And Views page, you choose the tables or views to copy from the source by selecting the corresponding check boxes in the Source column, and then choose the table into which data is to be loaded by selecting

it from the drop-down list in the Destination column. Alternatively, you can type a name to create a new table in the data destination. You can resize columns or the dialog box itself to improve your view of available objects. You can also click the Preview button to confirm you will be copying the correct data. For this example, select the *[AdventureWorks].[HumanResources].[Department]* table in the Source column.

8. Click the Edit button in the Mapping column of the selected row to display the Column Mappings dialog box. Here you can apply transformations to the data during the import by specifying conversion of null values or changing the precision. If the destination table already exists, you can choose to delete rows from the table before loading the copied data or to append rows to the table.

9. If the destination table is new, click Edit SQL if you need to modify the CREATE TABLE statement. Click OK to close the Column Mappings dialog box, and then click Next to display the Complete The Wizard page. The package is saved as a .dtsx file in the folder specified for the current Integration Services project and is not executed immediately.

10. Click Finish to create and save the package, and then click the Close button to close the wizard. Your package is now visible in the package designer, as shown in Figure 21-2.

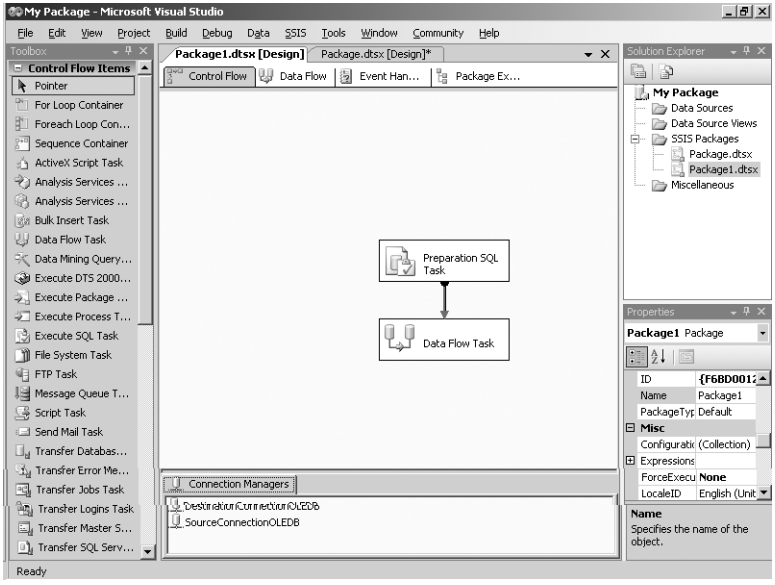

Figure 21-2 The Integration Services package designer.

By creating and saving a package to copy data from one location to another, you can repeat the data transfer quickly and easily at a later time as many times as you wish or you can make changes to this package to meet your needs more specifically.

> **Note** Alternatively, by launching the wizard in SQL Server Management Studio, you can use this tool simply to copy data without producing a package.

Adding Documentation to a Package

The Integration Services package designer includes two features that you can use to document your packages for easier reading as described here:

- **Name property** Either right-click the object to rename in the package designer, click Rename, and then type a new name for the object, or select the object in the designer, and type a name in the Name box in the Properties window. The object name is completely visible in the designer and is included in logs when you enable package logging.

  > **Note** Establish a consistent naming convention by including the type of task or transformation and a description of the object's function in the name.

- **Annotation** Right-click the designer work surface, click Add Annotation, and then type an annotation in the new text box. Annotations do not automatically wrap within the text box, so press Ctrl+Enter to force a new line, or drag the handles to resize the text box as needed. To change the font, font style, size, or color, right-click the annotations box, and click Set Text Annotation Font. Use *annotations* in a data source view to describe the tables and relationships it contains or on any package designer work surface—control flow, data flow, or event handler (components described later in this chapter)—to explain the overall objective of the package or describe the business rules being applied by package objects. Annotations are not included in package logs.

Executing a Package in Visual Studio

While your package is still in development, you can use Visual Studio to run the package. During execution, you can watch the progress of the package as each task executes, and the success or failure of each task is indicated by color. If a task is yellow, it is currently running. A green task has completed successfully, while a red task has failed. To execute a package and review execution results in Visual Studio, follow these steps:

1. To follow this example, you must complete the steps described in the previous sections. Right-click the package in the SSIS Packages folder in Solution Explorer and then click Execute Package. The results of package execution display in the package

designer, as shown in Figure 21-3. Notice the package in Figure 21-3 also illustrates renamed objects and an annotation as described in the previous section.

> **Note** There are several other ways to execute a package. You can choose Start Debugging on the Debug menu, press F5, or click the Start Debugging button on the Debug toolbar. However, if you have other packages or other projects included in your solution, Visual Studio might attempt to build a project or execute a package other than the package you are viewing.

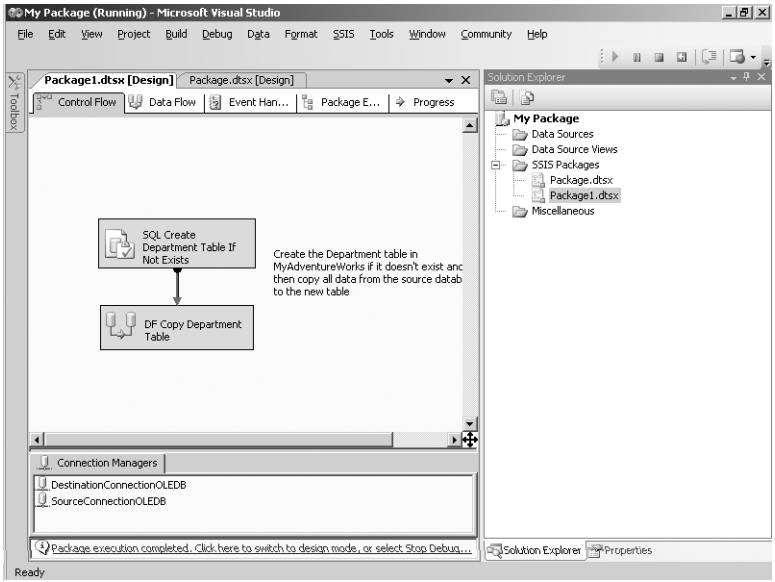

Figure 21-3 The results of package execution on control flow shown in the package designer.

2. Click the Data Flow tab of the package designer to view the number of rows copied to the destination, as shown in Figure 21-4.

3. Click the Progress tab of the Integration Services package designer to view information related to package execution, such as the start and finish time of package execution and the elapsed time for the package and each step, as shown in Figure 21-5. The Progress tab is useful for troubleshooting packages when a step fails because the error message will be available on this tab.

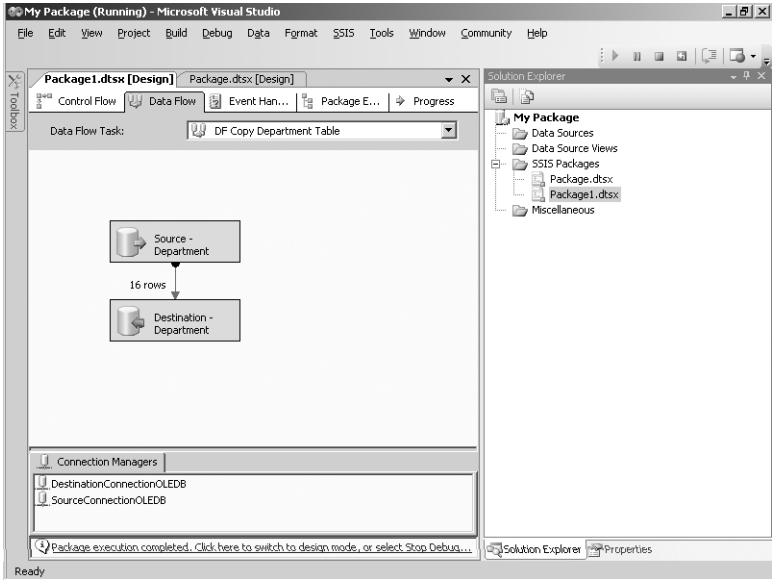

Figure 21-4 The results of package execution on data flow shown in the Integration Services package designer.

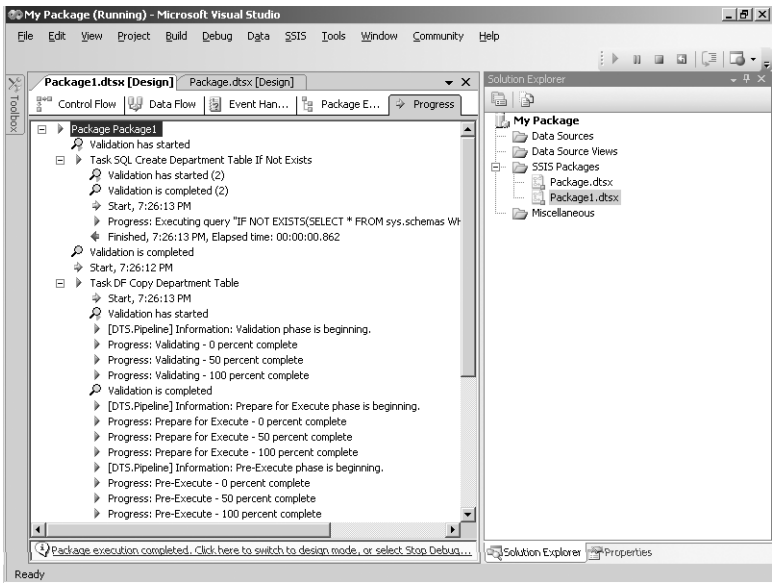

Figure 21-5 The progress information for package execution shown in the package designer.

4. In Visual Studio, the debugger remains active even when the package execution has ended. You won't be able to edit or close the package until you stop the debugger using one of the following methods: Choose Stop Debugging on the Debug menu, press Shift+F5, click the Stop Debugging button on the Debug toolbar, or, at the bottom of the package designer click the link Package Execution Completed. Click Here To Switch To Design Mode, Or Select Stop Debugging From The Debug Menu.

Control Flow Components

Now that we've created a simple package and briefly explored the development environment, let's take a closer look at the components you can use to manage the control flow of tasks in a package. As mentioned, there are three basic types of control flow components: tasks, containers, and precedence constraints. A set of tasks can optionally be organized into a logical grouping called a container. The order of operations in which each task or container is executed is determined by the sequence in which these objects are linked together by using precedence constraints.

Tasks

When you view the Control Flow tab of the Integration Services package designer, the Visual Studio Toolbox contains a wide variety of tasks you can add to your package. In the following list, tasks are grouped by function:

- **Extract, transform, and load data task**
  - ***Data Flow Task*** encapsulates a set of data flow components used to extract data from a source, optionally perform transformations, and load data into a destination
- **Execute SQL Server tasks**
  - ***Bulk Insert Task*** performs high-speed copying of data into a SQL Server database
  - ***Execute SQL Task*** runs single or multiple SQL statements or stored procedures; you must optimize queries elsewhere as Integration Services will not do it for you; you can use parameterized queries and map the results to a variable.
  - ***Transfer Database Task*** transfers a database between two SQL Server instances. Either instance can be SQL Server 2000 or SQL Server 2005
  - ***Transfer Error Messages Task*** transfers user-defined messages (with identifier greater than or equal to 50,000) between SQL Server instances

- ***Transfer Jobs Task*** transfers one or more SQL Server Agent jobs between SQL Server instances
- ***Transfer Logins Task*** transfers one or more logins (except sa) between SQL Server instances
- ***Transfer Master Stored Procedure Task*** transfers one or more user-defined stored procedures between master databases in separate SQL Server instances
- ***Transfer SQL Server Objects Task*** transfers one or more objects—tables, views, stored procedures, user-defined functions, defaults, user-defined data types, partition functions, partition schemes, schemas, assemblies, user-defined aggregates, user-defined types, or XML schema collections—between SQL Server instances

■ **Prepare data**

- ***File System Task*** creates, moves, or deletes directories or files or sets attributes on directories or files
- ***FTP Task*** downloads or uploads files or manages directories on an FTP server
- ***Web Service Task*** executes a web service method
- ***XML Task*** retrieves XML documents, applies operations using Extensible Stylesheet Language Transformations (XSLT) documents or XPATH expressions, merges documents, or validates, compares, or saves updated documents to files or variables

■ **Communicate with other processes**

- ***Execute Package Task*** runs another package as part of the current package workflow
- ***Execute DTS 2000 Package Task*** runs a package developed for SQL Server 2000
- ***Execute Process Task*** runs a command-line application or batch file
- ***Message Queue Task*** sends or receives messages between packages or applications using Microsoft Message Queuing
- ***Send Mail Task*** sends an e-mail message
- ***WMI Data Reader Task*** runs a Windows Management Instrumentation (WMI) query to return information about a computer
- ***WMI Event Watcher Task*** watches for and responds to WMI events

- **Extend package functionality with scripts**
  - ***ActiveX Script Task*** runs ActiveX script—included for backward compatibility only; new script should be developed in Script Task
  - ***Script Task*** runs custom code for functions not available in built-in tasks or transformations
- **Work with Analysis Services objects**
  - ***Analysis Services Processing Task*** processes Analysis Services cubes, dimensions, or mining models
  - ***Analysis Services Execute DDL Task*** runs an XML for Analysis command encapsulating an Analysis Services Scripting Language (ASSL) DDL statement
  - ***Data Mining Query Task*** runs a prediction query based on an Analysis Services data mining model
- **Maintain database objects**
  - ***Back Up Database Task*** backus up one or more SQL Server databases
  - ***Check Database Integrity Task*** checks the allocation and structural integrity of objects in one or more databases
  - ***Execute SQL Server Agent Job Task*** runs a SQL Server Agent Job
  - ***Execute T-SQL Statement Task*** runs a Transact-SQL statement. Use the *Execute SQL Statement Task* instead if you need to run a parameterized query or save the result to a variable
  - ***History Cleanup Task*** deletes entries in these tables: backupfile, backupfilegroup, backupmediafamily, backupmediaset, backupset, restorefile, restorefilegroup, and restorehistory
  - ***Notify Operator Task*** sends a notification message to SQL Server Agent operators
  - ***Rebuild Index Task*** rebuilds indexes in tables or views in one or more SQL Server databases
  - ***Reorganize Index Task*** reorganizes indexes in tables or views in one or more SQL Server databases
  - ***Shrink Database Task*** shrinks database and log files for one or more SQL Server databases
  - ***Update Statistics Task*** updates statistics for one or more SQL Server databases

To add the Execute SQL Task to the control flow, follow these steps:

1. To follow this example, you must complete the steps described in the previous sections, beginning with "Starting an Integration Services Project." Click the Control Flow tab if it's not already open, and locate the task to be added in the Toolbox.

   Note If the Toolbox isn't visible, choose Toolbox on the View menu.

2. Drag the task to the control flow work surface. For this example, drag the *Execute SQL Task* to the designer, as shown in Figure 21-6. The red icon that appears in the task box indicates further configuration is required. In this case, the task requires a connection.

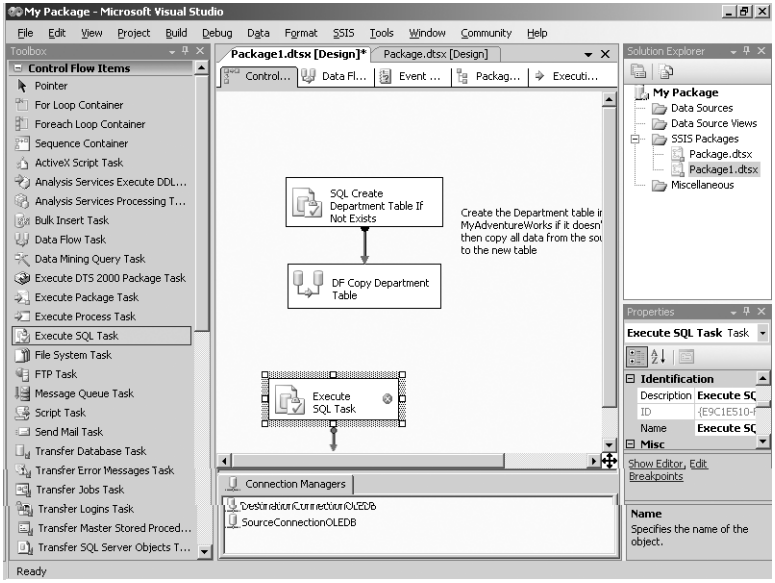

Figure 21-6 Adding a task to the control flow.

Each task has properties you configure to define its behavior when the package executes. To configure the *Execute SQL Task* properties, follow these steps:

1. Right-click *Execute SQL Task*, and then click Edit to display the Task Editor.
2. Type **Truncate Table** in the Name box, select *DestinationConnectionOLEDB* in the Connection drop-down list (or create one by selecting <New Connection> in the Connection drop-down list and completing the OLE DB Connection Manager

configuration). Type **truncate table HumanResources.Department** in the SQL-Statement box, as shown in Figure 22-7. For longer statements, you might find it easier to click the ellipsis button in the SQLStatement box and then type the statement in the Enter SQL Query dialog box.

An *Execute SQL Task* requires a connection manager, explained in more detail later in this chapter, and a SQLStatement. You can type directly in the SQLStatement box as shown in this example, you can click Build Query to open the Query Builder window, or you can click Browse to locate a SQL file and import its contents as a SQLStatement. You can use multiple SQL statements terminated by semicolons here, and you can insert a GO command between statements to create multiple batches.

Another option is to change the SQLSourceType property from the default, Direct Input, to FileConnection to specify a SQL file in the FileConnection property (which appears when you change SQLSourceType) if you want to maintain the SQL Statement externally from the package. Yet another option is to select Variable as the SQLSourceType and provide a SourceVariable that will contain one or more SQL statements at package run-time.

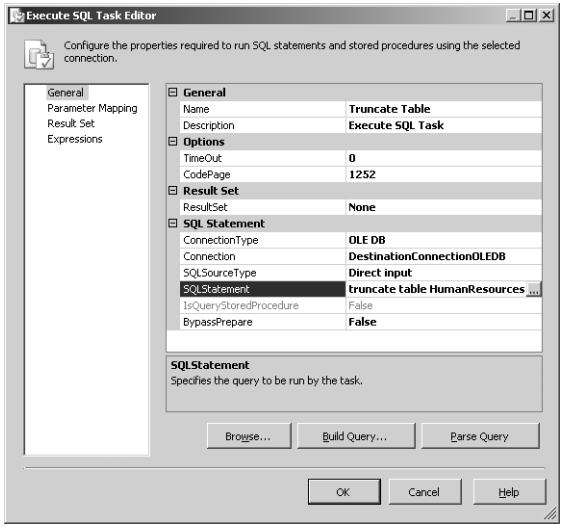

Figure 21-7 Configuring properties of an Execute SQL Task.

3. Click OK to close the editor.

Each task has its own set of properties that you configure in the task editor or by changing the property value in the Properties window. While only some tasks will

require you to specify a connection, all tasks have a page where you can associate an expression with a task property—an advanced feature that makes Integration Services very flexible.

Containers

You can group related tasks into containers. Because you can collapse or expand a container, using a container just for grouping makes it easier to navigate a package workflow that contains many tasks. Add a container to the control flow of a package just as you add a task. Drag other tasks or other containers into the newly added container. There are three different types of containers, as described here:

- ***For Loop Container*** This type of container creates a workflow that repeats the tasks in the container until the *EvalExpression* defined for the container evaluates as false. You also must assign a value to the container's InitExpression which is a starting value used for items like a counter, and is incremented according to the container's AssignExpression.
- ***ForEach Loop Container*** This type of container creates a workflow that repeats the tasks in the container for each item found in a collection, such as rows in a table, files in a folder, objects in a variable, among others. When configuring this container, you specify the type of collection and configure the location of the items in the collection.
- ***Sequence Container*** This type of container groups tasks together as a single unit in the workflow.

Precedence Constraints

Use precedence constraints to link tasks or containers, also known as *executables*, together into an order of operations. In addition to defining the sequence of operations, you define the conditions under which each subsequent step in the workflow is executed. To add a precedence constraint, click the first executable, and then drag the green arrow that appears at the base of the executable to a second executable so that the two objects are connected by the arrow, as shown in Figure 21-8.

By default, adding a precedence constraint between two executables causes the second object to execute only if the first object executes successfully. You can change the constraint to cause the second executable to run only if the first executable fails. This option is useful for handling error conditions within a package and could be as simple as sending an email to an administrator when a task fails. A third option is to define the constraint to run the second executable after the first executable runs, regardless of success or failure. To change the constraint, right-click the connector, and select Success, Failure, or Completion as necessary.

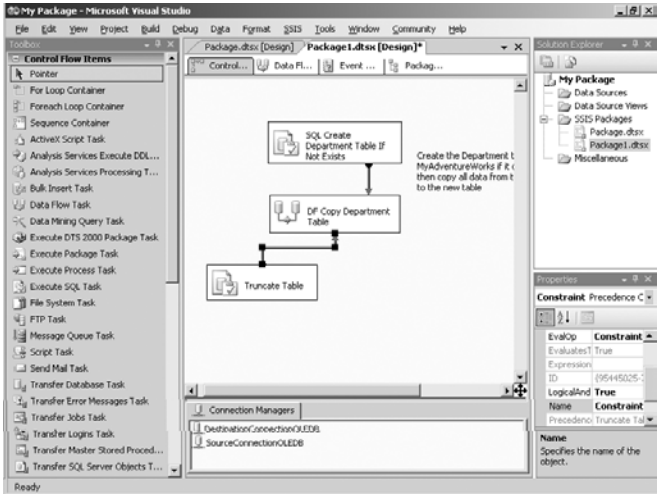

Figure 21-8 Adding a precedence constraint.

You can also create a complex precedence constraint based on the evaluation of an expression or on both the execution result of the preceding executable as well as the expression's result. For example, let's say that you want to run a second executable only if the first executable completes successfully and some condition is true, such as the existence of a file on a network share. You could use a Script Task to check for the existence of the file and set the value of a variable to True, and then require the precedence constraint to evaluate the variable's value. To add a complex precedence constraint, follow these steps:

1. To follow this example, you must complete the steps described in the previous sections, beginning with "Starting an Integration Services Project." Right-click the connector between the *Data Flow Task* and the *Execute SQL Task* and then click Edit to display the Precedence Constraint Editor dialog box, as shown in Figure 21-9.

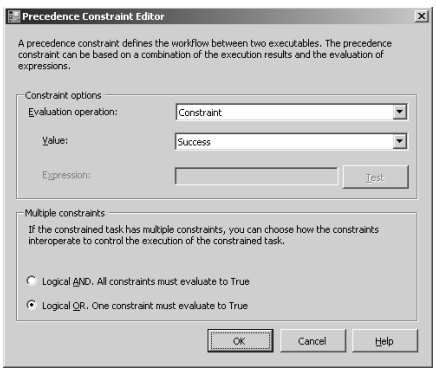

Figure 21-9 The Precedence Constraint Editor.

Possible values for Evaluation Operation are Constraint, Expression, Expression And Constraint, and Expression Or Constraint. Thus, you can specify one or the other condition, or both conditions, to control whether the next executable in the workflow executes. When you select an option that includes Constraint, you must specify a Value of Success, Failure, or Completion. When you select an option that includes Expression, you must define an expression in the editor. To continue the example requiring the existence of a file, you prefix the variable name with @; if your variable name is FileFound, use this expression: `@FileFound == True`. Your expression must also include a logical operator and a comparison value in the expression and must evaluate as a Boolean data type.

When an executable has multiple constraints placed on it—that is, when two or more worfkflow connectors lead to a single executable—be sure to specify in the Precedence Constraint Editor whether both constraints must evaluate as True for that executable to run (Logical AND), or whether either one of the constraints must be True (Logical OR). This setting is ignored if there is only one constraint on the executable.

2. Select Logical OR. One Constraint Must Evaluate To True, as shown in Figure 22-9, and then click OK.

Connection Managers

A connection manager is an Integration Services object that contains the information required to create a physical connection to data stores and the metadata describing the structure of the data. In the case of a flat file, a connection manager contains the file path, file name, and metadata identifying rows and columns. A connection manager for a relational data source contains the name of the server, the name of the database, and the credentials for authenticating access to the data. Connection managers are the bridge between package objects and physical data structures, used by tasks that require a connection (such as the Execute SQL Task), by data adapters that define sources and destinations, and by transformations that perform lookups to a reference table. To add and configure an OLE DB connection manager, follow these steps:

1. To follow this example, you must complete the steps described in the previous sections, beginning with "Starting an Integration Services Project." Right-click the Connections Manager tray at the bottom of the Control Flow, Data Flow, or Event Handlers tab of the package designer, and click New OLE DB Connection.

2. Select a defined connection in the Configure OLE DB Connection Manager dialog box, or click New to define a new connection in the Connection Manager dialog box. For this example, click New.

3. Select a provider in the Provider drop-down list and specify a server name and authentication credentials. You can type a database name or select a database in the Select Or Enter A Database Name drop-down list. For this example, type **localhost** in the Server Name box and select *AdventureWorks* in the Select Or Enter A Database Name drop-down list, as shown in Figure 22-10. (If you're working with a SQL Server on a separate server, replace localhost with the applicable server name.)

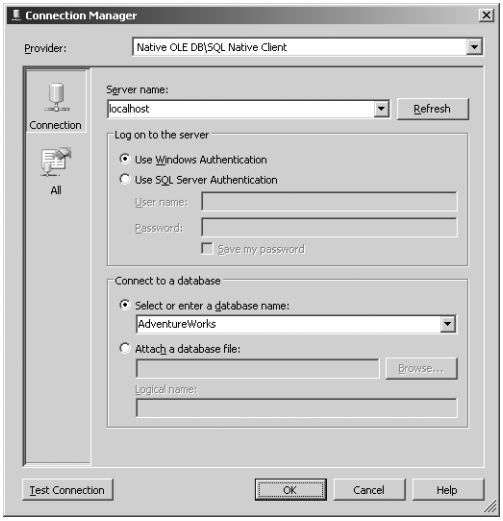

Figure 21-10 The Connection Manager dialog box.

4. Click OK twice to close each dialog box.

Data Flow Components

So far, we've seen how control flow components get things done in package, but the real power of Integration Services is the data flow engine. Now we're ready to examine the components used by the data flow engine. To work with data flow components, you first add a *Data Flow Task* to the control flow, then define one or more sources as the beginning of one or pipelines within the data flow. A single *Data Flow Task* could manage multiple pipelines if the order of operations doesn't matter. Each pipeline will run in parallel, more or less. (The degree of parallelism depends on system resources and the data volumes in each pipeline). Otherwise, you'll need to create multiple *Data Flow Tasks* and use precedence constraints to manage sequencing.

It's helpful to think of the data flow pipeline as a physical pipeline with discrete start and end points and with individual channels running through this pipeline, as shown in Figure 21-11. Each channel corresponds to a column in the row of data currently in

the pipeline. As columns are removed during transformations, the number of channels in the pipeline is reduced downstream. Conversely, as new derived columns are added to the pipeline, the number of channels downstream is increased. Both the data type and the size of the data traveling through a particular channel must stay constant unless a transformation like Data Conversion modifies the channel metadata. While the pipeline is a useful metaphor, in actuality the data is placed in a buffer and transformed in place; only certain transformations physically move data from one buffer to another.

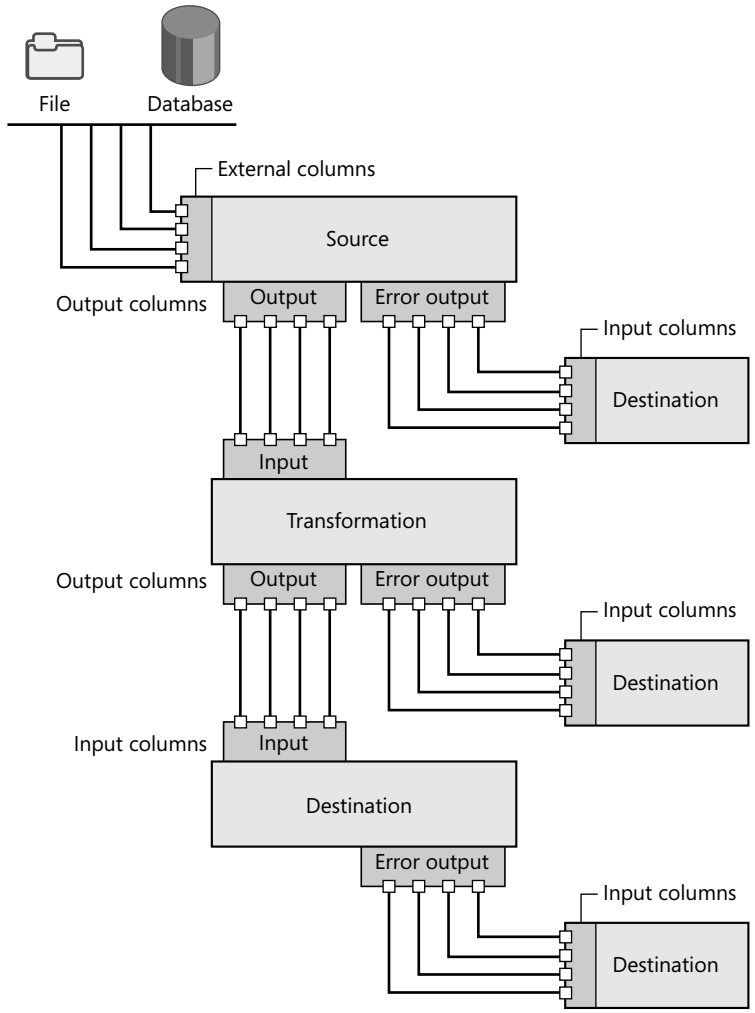

Figure 21-11 The data flow pipeline.

Data Sources and Data Destinations

Every data flow path starts with a source and usually ends with a destination. Data sources and data destinations are data flow components that sit between the physical connection to the data storage and the buffers created in memory to transport the data to another location. In most cases, the type of data storage accessed as a source can also be accessed as a destination. The direction the data flows out of or into this type of object determines whether it is a data source or a data destination. There are, however, some special data destinations for which there is no corresponding data source. As you work with these objects, you'll notice that a data source has two outputs, one for normal data flow and one for error output. By contrast, a data destination has only one input and one error output. Table 21-1 lists the available data adapters and their use as source, destination, or both.

Table 21-1 Available Data Adapters and Their Uses

Data Adapter	Source	Destination
Data Mining Model Training	—	√
DataReader	√	√
Dimension Processing	—	√
Excel	√	
Flat File	√	√
OLE DB	√	√
Partition Processing	—	√
Raw File	√	√
Recordset	—	√
Script Component	√	√
SQL Server	—	√
SQL Server Mobile	—	√
XML	√	—

If you're already responsible for accessing data in a variety of structures, you'll likely have no trouble applying what you know to the configuration of data sources and data destinations. Some of these objects, however, warrant further explanation:

- ***Data Mining Model Training*** You can automate the process of training an Analysis Services data mining model by using this destination. Before using this destination, be sure you understand the content and structural requirements of the data mining model.

- ***DataReader*** As a destination, the *DataReader* destination can expose the data in the pipeline to applications than can access the ADO.NET DataReader interface. Using this approach, for example, you can load data directly into a Reporting Services report without persisting the data in a file or relational table.
- ***Dimension Processing*** Use this destination to automate updating dimension objects in Analysis Services databases. Dimension processing is explained in Chapter 22, "Analysis Services."
- ***Partition Processing*** This destination is used to load fact table data into an Analysis Services partition, which is also explained in Chapter 22.
- ***SQL Server*** This destination type can be used only if the package executes on the same server hosting the target database in SQL Server 2005. Because this destination actually sends data to the database directly from memory without requiring a connection, performance can be up to 25 percent faster than using an OLE DB connection.

> **Note** If you're loading data to a SQL Server destination on another server, use the OLE DB destination. In the OLE DB Destination Editor, select Table Or View–Fast Load or select Table Name Or View Name Variable–Fast Load in the Data Access Mode drop-down list to get the best performance.

To start a data flow and add a data source adapter to the data flow, follow these steps:

1. To follow this example, you must complete the steps described in the previous sections, beginning with "Starting an Integration Services Project." On the Control Flow tab of the package designer, open the Toolbox window if necessary and drag the *Data Flow Task* to the design surface, and add a precedence constraint between the *Execute SQL Task* and the *Data Flow Task*.
2. Double-click the new task to open the corresponding Data Flow tab, as shown in Figure 21-12. Notice the contents of the Toolbox window change to show only sources, destinations, and transformations instead of the tasks it contains when the Control Flow tab is open.
3. Drag the *OLE DB Source* from the Toolbox onto the design surface, and then double-click the data adapter to open the OLE DB Source Editor.

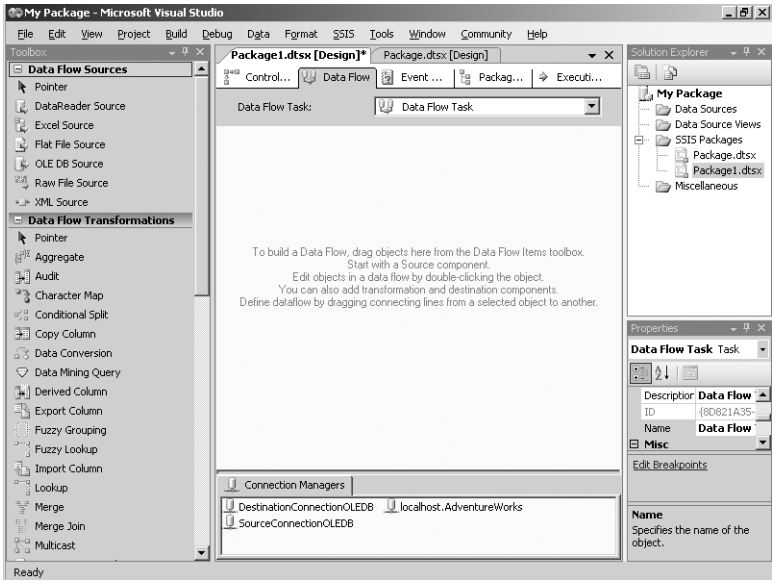

Figure 21-12 The Data Flow tab of the package designer.

4. Select a connection in the OLE DB Connection Manager drop-down list, or click the New button to configure a new connection manager to use with this data source. Specify a Data Access Mode. The default is Table Or View which you select in the Name Of The Table Or The View drop-down list. To continue the example, select the *localhost.AdventureWorks* connecton manager and select the *[HumanResources].[Department]* table, as shown in Figure 21-13.

   Instead, you could choose to get the the name of a table or a view from a variable, in which case you select Table Name Or View Name Variable as the data access mode and then specify the name of the variable in the Variable Name drop-down list. Other options include getting the name from executing a SQL command that you include in the data source adapter as SQL Command Text (when you choose the SQL Command data access mode) or getting the table or view name from a variable that contains a SQL command (when you choose the SQL Command From Variable data access mode).

5. Click Columns in the left pane of the dialog box to display the column mappings between the data store and the output columns used to send data to the next data flow component, as shown in Figure 21-14. Configuring error output will be explained later in this chapter. Click OK.

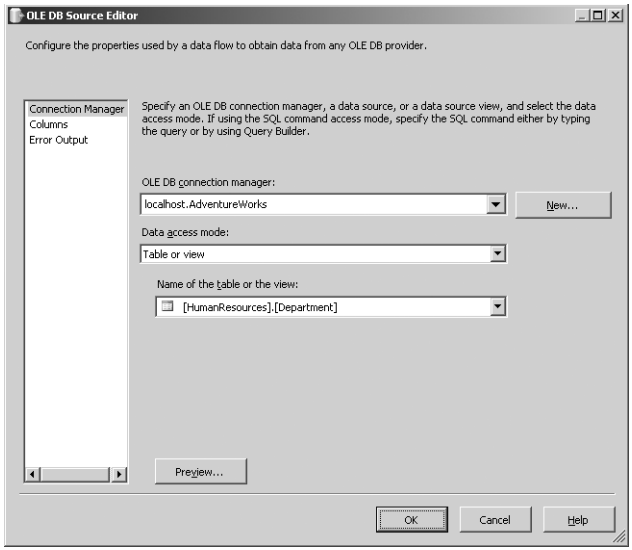

Figure 21-13 The OLE DB Source Editor dialog box.

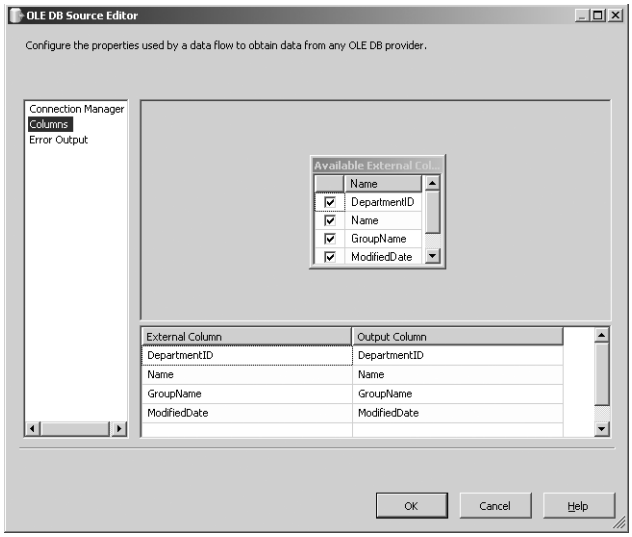

Figure 21-14 The OLE DB Source Editor's column mappings.

The External Column and Output Column mappings generate automatically.

Note You should clear the selection of any column in the Available External Columns box that won't be used downstream to prevent memory from being unnecessarily allocated to the column.

If your goal is to copy data directly from a source to a destination, your next step is to add a destination to the data flow. On the other hand, if you need to transform the data before loading it into the destination, you must add each transformation object to the data flow first because the buffer structures created for the data flow are dependent on the definition of upstream objects.

Transformations

Transformations are data flow objects that operate on data in the data flow pipeline. Most transformations have one input, one regular output, and one error output. Some transformations have multiple inputs to consolidate data from separate pipelines, and some transformations have multiple outputs to separate data into separate pipelines. Not all transformations have error output.

The type of transformations in the pipeline affect the data flow engine's management of buffers. Row transformations, like Data Conversion or Derived Column, are very efficient at reusing existing buffers. Multiple rows are received into the buffer from an upstream component in batches, data in the buffer is manipulated on a row-by-row basis, and then these rows in the buffer are operated on by the next component. This process continues until all rows from the source have been transformed.

Other transformation types must create new buffers to store output before allowing the next component access to the data, and they will consequently place the greatest demands on system resources. Transformations operating on rowsets, like Aggregate or Sort, must read all input before creating any output. Transformations that combine input, such as Merge or Union All, are somewhere in between the other transformation types, reading some input before creating output, but they do copy that output to a different buffer and thus have greater resource requirements. The available transformations and the relationship between input and output rows are described here:

- **Row transformations**
  - ***Character Map*** applies string functions, commonly *Uppercase* or *Lowercase*, to character data in a column and overwrites the column with the new string or adds a column to the output row
  - ***Copy Column*** creates new columns in the output row from existing columns in the input row for subsequent transformation

- ❑ ***Data Conversion*** applies data conversion functions to change data type, column length, or precision and scale to an input column and adds a column with the new data to the output row
- ❑ ***Derived Column*** applies expressions to an input column to create a new value for a new column in the output row or as a replacement value for the existing column
- ❑ ***Script*** runs custom code against data in the pipeline
- ❑ ***OLE DB Command*** runs a SQL statement (optionally parameterized) against each row in the pipeline

■ **Rowset transformations**

- ❑ ***Aggregate*** applies an aggregate function (for example, *Sum*, *Minimum*, *Maximum*, and so on) to grouped records and produces new output records from aggregated results; output records contain only the columns used for grouping and the aggregated values
- ❑ ***Sort*** applies an ascending or descending sort to one or more columns and optionally removes rows with duplicate values in the sort columns
- ❑ ***Percentage Sampling*** creates a random sample set of output rows by selecting a specified percentage of input rows; commonly used for data mining
- ❑ ***Row Sampling*** creates a random sample set of output rows by selecting a specified number of input rows; used primarily for testing packages with a subset of representative data
- ❑ ***Pivot*** reduces normalization in a normalized data set by pivoting on a column value, producing a smaller dataset
- ❑ ***Unpivot*** increases normalization in a denormalized data set by creating multiple output rows from a single input row based on a common column value

■ **Split and Join Transformations**

- ❑ ***Conditional Split*** separates input rows into separate output pipelines based on a Boolean expression configured for each output; a single input row is passed to the first output row which the condition is true or to a default output defined for rows which meet no conditions
- ❑ ***Multicast*** copies all input rows to two or more outputs
- ❑ ***Merge*** combines two sorted datasets with the same column structure into a single output

- ***Merge Join*** combines two sorted datasets using a FULL, LEFT, or INNER join to produce output rows with more columns than the input rows from either dataset
- ***Union All*** combines two or more datasets with the same column structure into a single output
- ***Lookup*** joins input rows with columns in a reference dataset (from a table, view, or SQL statement) that match exactly to add one or more columns to the output row

- **Data quality transformations**
  - ***Fuzzy Lookup*** finds close or exact matches between one or more columns (*DT_WSTR* and *DT_STR* data types only) in the input row and a row in a reference table; adds selected columns from the matched row and columns for fuzzy matching metrics
  - ***Fuzzy Grouping*** finds close or exact matches between input rows based on one or more columns and adds columns to the output identifying matches and similarity scores
- **Data mining transformations**
  - ***Data Mining Query*** runs a prediction query based on an Analysis Services data mining model against the input rows and creates output rows from the query result set
  - ***Term Extraction*** extracts nouns or noun phrases (or both) from an input column containing English text, places the extracted terms into a column in the output row, and adds another column for the score; multiple output rows per input row are created when multiple terms are extracted from an input column
  - ***Term Lookup*** matches text in an input column with a reference table, counts the number of occurrences of the term in the dataset, and adds a term and frequency column to the output row; as with Term Extraction, multiple output rows per input row can be created
- **Special transformations**
  - ***Export Column*** reads a file name from an input column and inserts data from another column in the same row into the specified file.
  - ***Import Column*** reads a file name from an input column and inserts data from the specified file into a new output column for the same row
  - ***Audit*** adds the value of a system variable, such as MachineName or ExecutionInstanceGUID, to a new output column

- ***Row Count*** counts the number of rows currently in the data flow and stores the value in a variable
- ***Slowly Changing Dimension*** manages inserts and updates in a data warehouse

Real World Metadata Validation

You can add one or more transformations to the design surface of a data flow, but you cannot configure a transformation until you connect it to the regular or error output of an upstream component. Because metadata about the pipeline is maintained in the pipeline connections (the red and green arrows between components) and transformations, you must add transformations in the correct sequence or you will encounter validation errors that must be resolved before you can execute the package. Similarly, if you delete columns, change a column's data type, change a string column's length, or change a numeric column's precision or scale, you will encounter errors in downstream components. Sometimes, if you double-click the first downstream component, a message displays to indicate a metadata problem exists and offers to fix the problem automatically. Click Yes, and continue this process with each subsequent component in the pipeline. You can override metadata validation by setting a transformation's ValidateExternalMetadata property to False, but property applies only during design time. Metadata problems prevent packages from executing.

Error Output Configuration

When data cannot be passed into the regular output of a data source or a transformation, you ignore the error completely, redirect the row to the component's error output, or fail the component. To configure error output, follow these steps:

1. To follow this example, you must complete the steps described in the previous sections, beginning with "Starting an Integration Services Project." Drag a destination component, such as the *OLE DB Destination* to follow this example, from the Toolbox to the data flow design surface, click the *OLE DB Source*, and then drag the red connector to the destination to complete the connection.
2. In the Configure Error Output dialog box, select a value in the Error drop-down list and in the Truncation drop-down list for a specific column. For example, select Ignore Failure in the Error drop-down list for the ModifiedDate column and select Redirect Row in the Truncation drop-down list, as shown in Figure 21-15. Click OK.

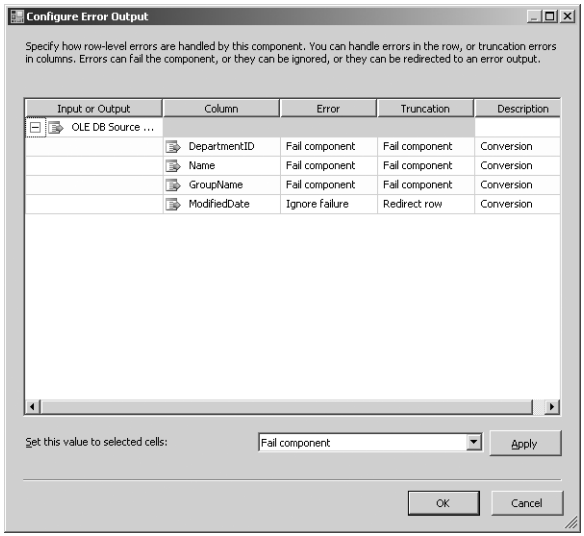

Figure 21-15 Configuring error output.

3. Double-click the destination to open the *OLE DB Destination Editor*, select a connection to a data source to which you want to send the error rows (or click New to create a new connection), select a data access mode, and provide a table name or applicable property value based on the selected data access mode. For this example, select *DestinationConnectionOLEDB* in the OLE DB Connection Manager dropdown list.

4. Click New to create and execute a script for a new table. You can modify this script by changing the default name or by changing column names. The script is generated from the metadata associated with the input to the destination. Click OK to return to the OLE DB Destination Editor, as shown in Figure 21-16. To continue this example, replace [OLE DB Destination] with **[DepartmentErrors]**. When you use the fast load option, you can set options such as keeping identity values from the source data, keep null values, lock the table for loading, check constraints during the load, and specify batch sizes.

5. Click Mappings in the left pane of the dialog box to display the column mappings between the input columns and the output columns used to send data to the destination. Click OK.

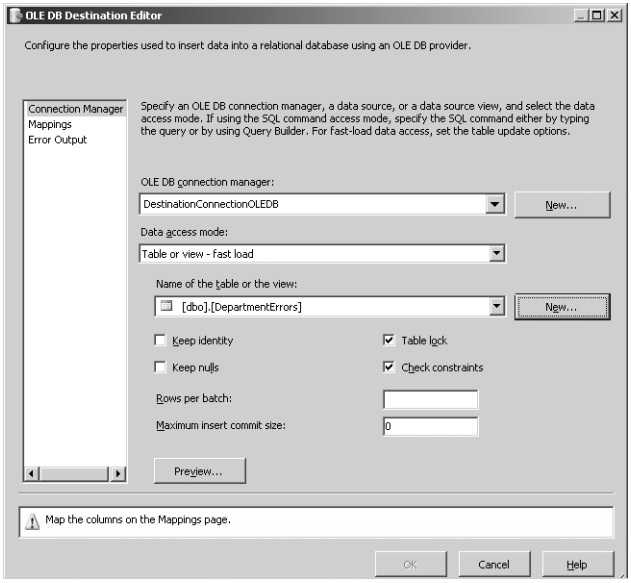

Figure 21-16 The OLE DB Destination Editor.

Debugging Tools

Sometimes you need to step through execution of a package to troubleshoot a problem or to confirm that tasks or transformations are behaving as you intend. The difference between architectures in the control flow engine and the data flow engine requires different tools for debugging control flow and data flow. They all allow you to pause the action and review the current state of information about the package.

Control Flow Breakpoints

In Control Flow, you can use *breakpoints* to pause package execution and review information in Visual Studio debug windows. You can define a breakpoint before control flow begins or after it ends, as well as before or after an executable runs within the control flow. You can even add or delete breakpoints while debugging. To add a breakpoint, follow these steps:

1. To follow this example, you must complete the steps described in the previous sections, beginning with "Starting an Integration Services Project." On the Control Flow tab of the package designer, right-click anywhere in the design surface, and

then click Edit Breakpoints. Select the Break When The Container Receives The OnPreExecute Event check box, as shown in Figure 21-17. Click OK to close the Edit Breakpoints window.

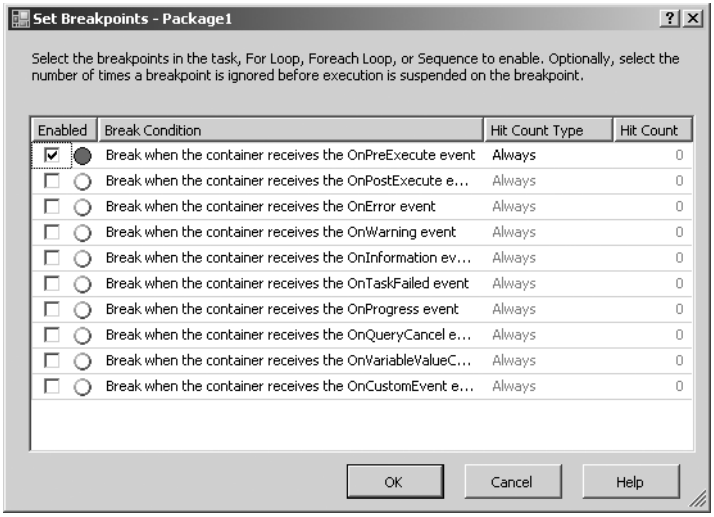

Figure 21-17 The Set Breakpoints dialog box.

You can set a breakpoint for *events* raised by executables, which are simply tasks, containers, and even packages. Events are essentially messages like "I'm starting," "I'm done now," and "I had a problem."

2. In Solution Explorer, right-click the package and click Execute Package to start debugging. The Visual Studio windows change, but nothing happens yet because your breakpoint stopped the action before it even got started. However, this pause gives you the opportunity to add another breakpoint. Right-click a component, such as the first *Data Flow Task* created by the SSIS Import and Export Wizard in a previous example, click Edit Breakpoints, and then select the Break When The Container Receives The OnPostExecuteEvent check box. This breakpoint will stop the action when this task finishes. Click OK.

3. Press F5 or click the Continue button in the Debug toolbar. Notice the breakpoint symbol on the *Data Flow Task* to indicate the point where execution has paused, as shown in Figure 21-18.

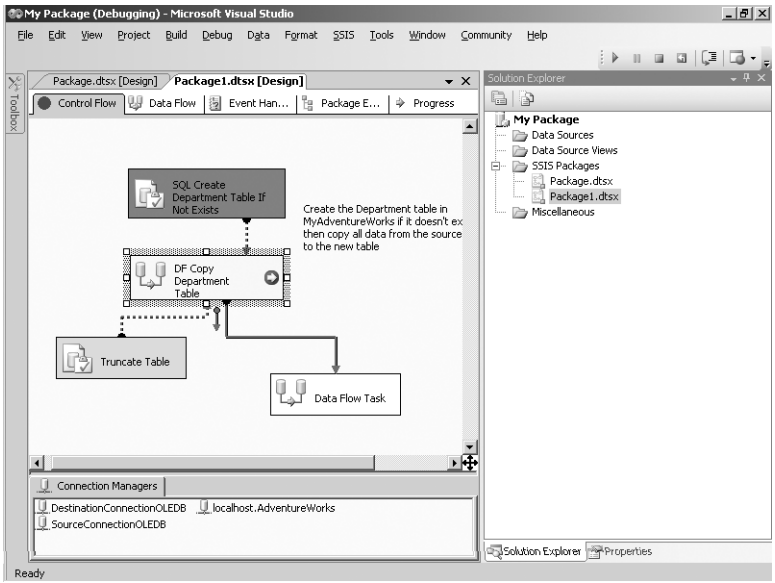

Figure 21-18 Debugging Control Flow.

Now you can go exploring. Here are some of the things you can do:

- Open the Data Flow for this task to see how many rows were transferred.
- Open the Progress tab to view warning or error messages as well as status information on completed tasks or tasks in progress.
- Open other debugging windows. Choose Output on the View menu or point to Windows on the Debug menu and then choose one of the following windows: Breakpoints, Watch, Locals, Call Stack, or Threads. Multiple debug windows can be open simultaneously.

4. Press Shift+F5 or click the Stop Debugging button in the Debug toolbar to end package execution. You don't have to run a package through all tasks before stopping.

5. Choose Delete All Breakpoints on the Debug menu when you have completed your debugging activities.

Important If you choose to delete all breakpoints, configured data viewers (discussed in the next section) are also deleted from the data flow. You can selectively clear breakpoints on the Debug menu by pointing to

Windows and choosing Breakpoints. In the Breakpoints window, you can clear the check box for individual breakpoints.

Data Flow Data Viewers

You can add a *data viewer* to the pipeline to monitor the rows passing through it. It's like having a window into your pipeline; you can see actual data rows before and after a transformation by placing a data viewer on either side of the transformation. For large volumes of data, viewing each row might be impractical, so you have other data viewers that can consolidate the information into a histogram, scatter plot, or column chart. Experiment with each viewer to discover how each best meets your needs. To add a data viewer, follow these steps:

1. To follow this example, you must complete the steps described in the previous sections, beginning with "Starting an Integration Services Project." Click the Data Flow tab of the package designer. In the Data Flow Task drop-down list at the top of the package designer, select the first *Data Flow Task* (which contains the *Source–Department* and *Destination–Department* objects) created in a previous example in the Data Flow Task drop-down list.

2. Right-click on the connector between two component objects, and then click Data Viewers.

3. In the Data Flow Path Editor, click Add to open the Configure Data Viewer dialog box, as shown in Figure 22-19. Click OK. If you want to limit the columns you see in the data viewer, click the Grid tab and remove columns from the Displayed Columns list.

4. Click OK to return to the Data Flow Path Editor. You can continue adding other data viewers to this path. Each data viewer displays in a separate window during debugging. Click OK. The Data Flow work surface now includes an icon to indicate a data viewer has been added.

5. In Solution Explorer, right-click the package and click Execute Package. The data viewer acts as a breakpoint and stops when it receives input, as shown in Figure 22-20. You might need to resize the data viewer window.

6. Scroll through the contents of the window to view columns by row. When you are finished, click the green arrow button at the top of the data viewer to continue package execution. The package will stop at any additional breakpoints in the package, or run to completion.

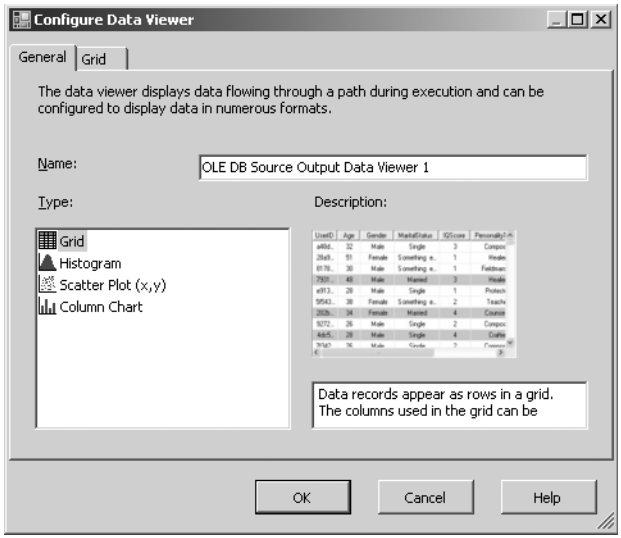

Figure 21-19 The Configure Data Viewer dialog box.

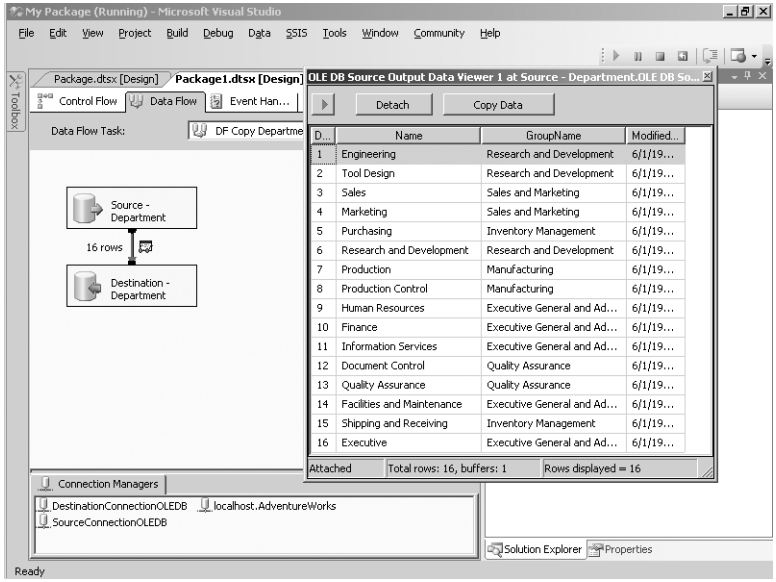

Figure 21-20 Debugging with the grid data viewer.

Logging

As shown in the previous section, you have several options for monitoring the progress of package execution during the development cycle, but these options are useful only when you are executing the package manually. Once you place a package into production, however, you need a different way to monitor the results. Integration Services provides a variety of formats for logging and allows you to configure the level of detail to be logged at the package, container, and task levels. You can configure logging to one or more providers, as described here:

- **Text** Saves as CSV file with .log extension; easy to view package results using this format
- **SQL Server Profiler** Writes trace file with .trc extensions; useful for monitoring package execution time over a period of time
- **SQL Server** Writes data to a sysdtslog90 table in a specified database; easy to query with SQL
- **Event Log** Writes to the Application Log in the Windows Event log; good for use in conjunction with Microsoft Operations Manager
- **XML File** Writes data to file with .xml extension; with XSLT, easy to share as web page and to consolidate log files from multiple sources

To configure logging, follow these steps:

1. To follow this example, you must complete the steps described in the previous sections, beginning with "Starting an Integration Services Project." In the package designer, choose Logging on the SSIS menu. Select a log provider, such as SSIS Log Provider for SQL Server, in the Provider Type drop-down list. Click Add.
2. Next, associate the log provider with one or more executables. For this example, select the Package1 check box and then select the newly added log in the Select The Logs To Use For The Container. By default, all child executables are selected.
3. To configure the log provider, select a connection manager in the Configuration drop-down list, such as DestinationConnectionOLEDB, as shown in Figure 21-21.
4. Click the Details tab and select the events to be logged. For this example, select OnError and OnPostExecute, as shown in Figure 22-22. If you click Advanced, you can select or clear specific columns to be logged by event, such as Computer, Operator, MessageText, and others. All available columns are selected by default. Click OK.

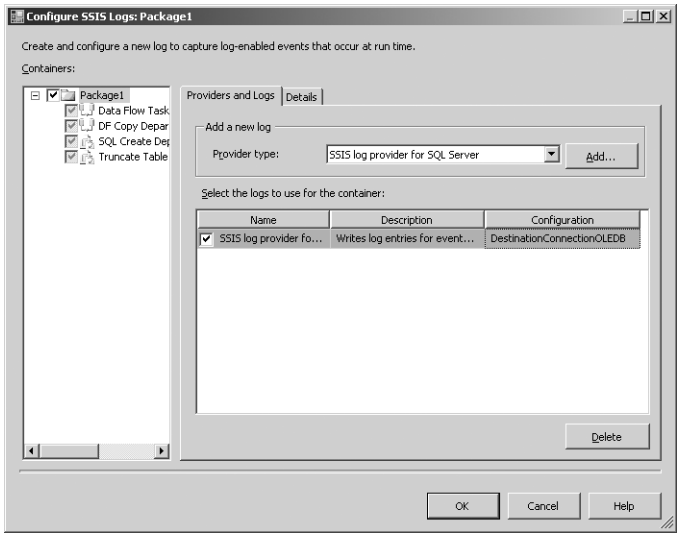

Figure 21-21 Configuring package logging.

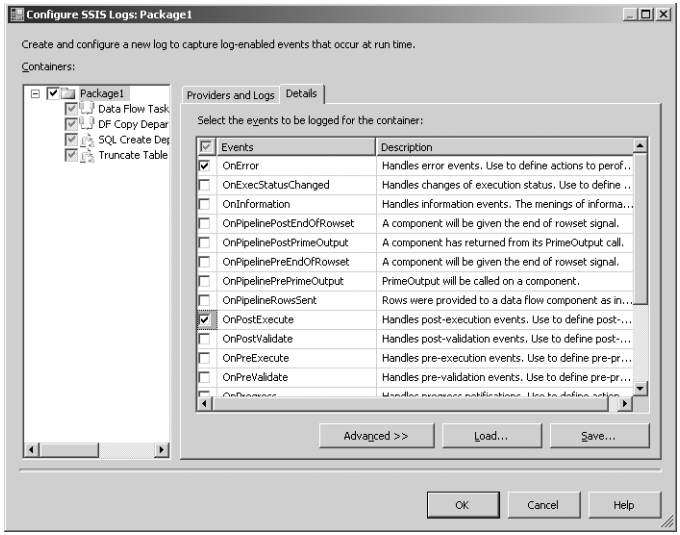

Figure 21-22 Selecting events to log.

5. Run the package, and then use SQL Server Management Studio to view the contents of the sysdtslog90 in the target database specified in the connection you associated with the log provider.

Important Keep in mind that logging adds overhead to package execution and can tax system resources.

Advanced Integration Services Features

There are so many features available in Integration Services, but there is so little space to discuss all of them in detail. In this section, we'll review a few other features you should be familiar with but which you may not need to implement right away if your data integration needs are still relatively simple.

Variables

Variables are the means by which you can add infinite flexibility to your packages. The subject of variables was introduced previously in this chapter in the context of passing a variable value as a parameter in a SQL statement used in an Execute SQL Task. Variables can also be used in expressions, most commonly for changing an executable's properties at run-time. There are two kinds of variables available in Integration Services, as described in the following list:

- **System variables** Defined by Integration Services on creation of a package, container, task, or event handler. You cannot add or update a system variable. You can access the value of a variable only at run-time. For example, you can read the MachineName and then change properties of other tasks to modify package behavior when executing on a specific machine.
- **User variables** Defined by the package developer. You can use this type of variable to dynamically set values and control processes during package execution. You can assign a literal value to the variable or derive a value by evaluating an expression. As explained in a previous example, you can test for the existence of a file, store the result as True or False in a variable, and then test the variable in a precedence constraint before allowing an executable to run.

Note Variable names are case sensitive in Integration Services.

Event Handlers

An event handler is a special type of container—think of it as a mini-package that has its own tasks and containers. An event handler executes only in response to the event for which it is configured. The following list provides a few examples of how you might use an event handler.

- Delete a file on a network share after its presence has been detected and has caused an executable to run.
- Determine if the computer has enough memory to run a package.
- Send an e-mail message to you when an error causes a component to fail.

Checkpoints

Using *checkpoints* allows you to restart a package restart from a point of failure instead of the beginning of package. This capability allows you to save a lot of time, particularly when you're extracting large volumes of data from another server or processing Analysis Services objects. As the package executes, information about its progress is written to a checkpoint file. If the package fails, the checkpoint file is read by Integration Services to determine the point at which the package should be restarted.

An important concept to understand, however, is the relationship between nested containers and checkpoints if a package is restarted, for example, if you use a loop container that in turn holds other containers. If failure occurs before all loops of the parent container are complete, then the loop starts at the beginning even if tasks in the child containers completed successfully before the failure.

Deploying Packages

Once you've developed and tested your package, what's next? You probably don't want to run the package manually in Visual Studio on a regular basis. In this section, we'll review package configuration options, deployment and security considerations when moving packages into production, and tasks related to package management after deployment.

Package Configuration

Package configurations allow you to customize how a package runs in different circumstances. For example, when you run the package on a development server, you might want to connect to sample databases for extracting and loading data, but when you run the package on a production server, you want to connect instead to real data sources. You aren't limited to changing connection information; by using variables and expressions liberally when configuring properties for executables and data flow components, you can manage run-time values externally. These values can be stored as environment variables, registry entries, or SQL Server tables, or passed from a parent package to a child package as a variable. While the possibilities seem limitless, the

most common reason for using package configuration is to change the connection string when moving the package to another server. To define package configuration, follow these steps:

1. To follow this example, you must complete the steps described in the previous sections, beginning with "Starting an Integration Services Project." In the package designer, choose Package Configurations on the SSIS menu. Select the Enable Package Configurations check box, and then click Add to start the Package Configuration Wizard. Click Next on the Welcome page of the wizard to display the Select Configuration Type page

2. Select an item in the Configuration Type drop-down list, such as XML Configuration File. Your selection changes the boxes in this page of the wizard that define the location of the configuration information. In this example, we'll use an XML file because it's easy to view and change later. Type **MyConfig.dtsConfig** in the Configuration File Name box, as shown in Figure 21-23. Click Next.

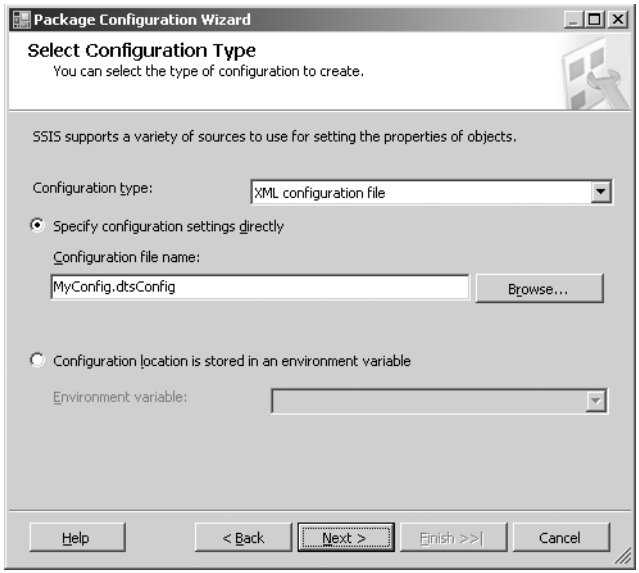

Figure 21-23 The Select Configuration Type page of the Package Configuration Wizard.

3. On the Select Properties To Export page of the wizard, as shown in Figure 21-24, you can select as many properties of executables, connection managers, log providers, the package itself, and package variables as you want to manage externally in the configuration file. For this example, scroll up to locate the Connection Managers folder, expand the SourceConnectionOLEDB and the Properties folders, and

then select the ConnectionString property. The current attributes associated with the selected property display in the dialog box. Click Next.

4. Type a Configuration Name such as **Source ConnectionString**, and then click Finish. The configuration is now listed in the Package Configurations Organize, as shown in Figure 21-25. You can create multiple configurations for a single package. When you later deploy the package, you can control which configurations to apply at run-time. Click Close.

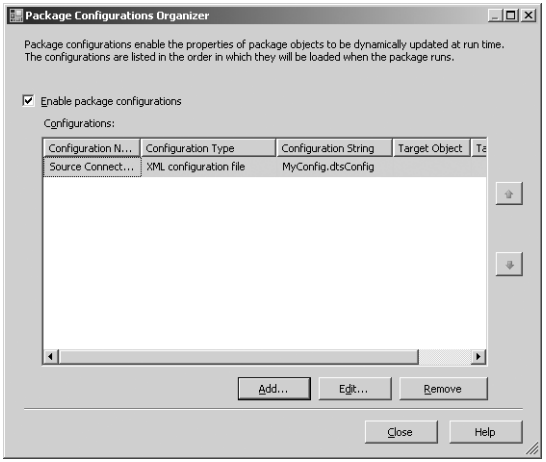

Figure 21-24 The Package Configurations Organizer.

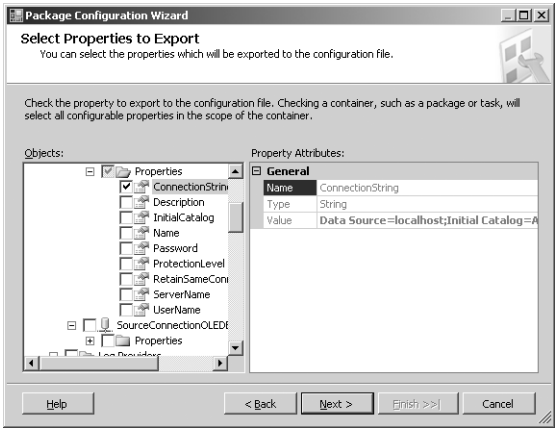

Figure 21-25 The Select Properties To Export page of the Package Configuration Wizard.

Now that a package configuration exists, you can open the configuration container, which in this case is an XML file, and edit the contents directly. For example, you can change the value of the connection string to have the package connect to a different server when you use this configuration the next time the package executes.

Package Deployment

Integration Services makes it very easy to deploy your package to an Integration Services server. Use Visual Studio's Build command to create a manifest file and a set of package and configuration files from your Integration Services project which you can then move to the new server. After moving the files, you launch the Package Installation Wizard by double-clicking the manifest file. The wizard steps you through the process of selecting a target server and a storage method for your package. If you have a package configuration, the wizard also allows you to change configured values before the package is installed on the server. To deploy a package, follow these steps:

1. To follow this example, you must complete the steps described in the previous sections, beginning with "Starting an Integration Services Project." Right-click the project name in the Solution Explorer window and then click Properties. In the Property Pages dialog box for your project, click Deployment Utility. Set the CreateDeploymentUtility property to True. If you want to allow package configuration changes on deployment, be sure the value of the AllowConfigurationChanges property is True. Also, take note of the DeploymentOutputPath. This is the folder where your utility will be stored. Click OK.
2. Right-click the project in the Solution Explorer window again, but this time click Build. If the deployment utility is built successfully, the status bar at the bottom of the screen will display Build Succeeded.
3. Using Windows Explorer, navigate to the solution folder for your project and then navigate to the folder specified by DeploymentOutputPath in Package Properties. Here you will find the copies of your packages (the .dtsx files), a copy of the configuration file (with extension .dtsConfig), and a special SSISDeploymentManifest file. This file details the files to be deployed together to the new server.
4. To install your packages on a new server, copy the contents of the Deployment folder to a location on the other server. For this example, you can just leave the files where they are and continue as if the files were relocated. Double-click the SSISDeploymentManifest file to start the Package Installation Wizard, and then click Next on the Welcome page.
5. On the Deploy SSIS Packages page of the wizard, select SQL Server Deployment, which is the recommended option. Regardless of the deployment option you

choose, you can optionally validate your package after deployment to ensure it still works after being copied and transferred. Click Next.

6. Provide a target server name and credentials if you're using SQL Server authentication. You can optionally choose to apply SQL Server security by selecting Rely On Server Storage For Encryption on this page. Click Next.

7. Note the default folder to which the package is installed. Click Next twice to start installation.

8. On the Configure Packages page of the wizard, you can change the value of properties you included in the package configuration file. For this example, you can keep the current value. Click Next, and then click Finish to complete the wizard.

9. To confirm the package is installed on the server, open SQL Server Management Studio. Choose Connect Object Explorer on the File menu, select Integration Services in the Server Type drop-down list, change the Server Name if necessary, and then click Connect.

10. Expand the Stored Packages and MSDB folders to view a list of the installed packages. If you previously selected the file system for installation, your packages will be visible in the File System folder.

Package Security

There are several ways to use Integration Services security to protect sensitive data, such as passwords and property values defined in a package, and data produced by the package, such as configurations, checkpoints, and logs. The key points about security are outlined in this section.

Protecting Data

Integration Services considers some component properties to be sensitive. That is, the values supplied for those properties should be treated as confidential, such as a connection string or a password. Whether you provide this value directly to the property or by using a variable to pass in a value at run-time, Integration Services will detect the components with sensitive properties and then suppress or encrypt the values based on the protection level defined for the package. You can use different protection levels for each environment in which the package exists. For example, on your development computer, you might use a different level of protection than you configure for a package deployed to the server. To set the protection level of a package in Visual Studio, click anywhere on the Control Flow tab, select the package in the drop-down list at the top

of the Properties window. Change the Protection Level to an appropriate value, as described in the following here:

- **DontSaveSensitive** Marks sensitive information when the package is saved and replaces the information with blanks if a different user opens the package
- **EncryptSensitiveWithUserKey** Encrypts sensitive information with a key based on the user and replaces the information with blanks if a different user opens the package; package execution fails for a different user
- **EncryptSensitiveWithPassword** Encrypts sensitive information with the PackagePassword value and replaces the information with blanks if a different user opens the package and cannot supply the password; any user providing the password can execute the package.
- **EncryptAllWithPassword** Saves the PackagePassword value with the encrypted package; any user providing the password can open the package in the package designer or execute the package using dtexec
- **EncryptAllWithUserKey** Encrypts the package with a key based on the user creating or exporting the package; only that user can open the package in the package designer or execute the package using the dtexec utility
- **ServerStorage** Uses SQL Server database roles to protect the package, only when the package is saved to the SQL Server MSDB database

The content of a package isn't the only information you might need to protect. You also may need to protect configuration files, checkpoint files, and log files. You should use the SQL Server Configuration Type for package configurations and the SQL Server log provider for logging so you can apply database security to this information. If you must save any information to a file, then configure operating system security on the folder where the configuration, checkpoint, or log file is stored.

Using Digital Signatures

A package that has been digitally signed and later modified cannot be executed. You must update the digital signature if you later need to change the package. You won't be able to sign a package without a digital certificate available on your computer. Once you have a certificate, choose Digital Signing from the SSIS menu in the package designer, click Sign, and select the certificate. When the package is deployed to the server, you can specify an execution option to require validation of the signature.

Protecting Packages

Packages installed in the MSDB database on a SQL Server 2005 instance can be associated with a reader role and a writer role. In SQL Server Management Studio, connect to

Integration Services, right-click a package in the MSDB folder, and click Package Roles. As shown in Figure 22-26, you can select one or more database-level roles for each package role. In addition, be aware the Windows administrators on the server can view and stop all packages currently running while SQL Server administrators can view and stop only those packages they started.

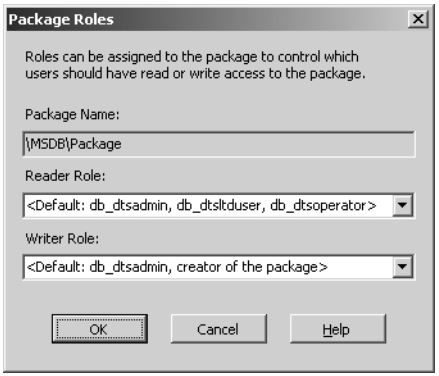

Figure 21-26 The Package Roles dialog box.

Package Execution

To manually start package execution, you can use the dtexec command prompt utility, but this method requires that you know the syntax for the command, which can be tricky if you want to specify additional execution options such as a change in connection string or property value. This utility gives you one more way to manage package execution without using configuration files. However, it's easiest to build the command and arguments for the dtexec utility by using the Execute Package Utility (also known as the dtexecui utility), which provides a graphical interface for specifying run-time options.

To start this utility in SQL Server Management Studio, you connect to Integration Services, right-click a package in either the File System or MSDB folder, and click Run Package. You can also run dtexecui from the command-line. In the Execute Package Utility dialog box, you navigate through each page and select execution options. For example, click Configurations to select a different configuration file. This file doesn't have to be stored on the same server, but it does need to be in a location accessible by Integration Services during package execution. When you have finished selecting options, click Command Line to view the command line string generated, as shown in Figure 22-27. If you select different options in the Execute Package Utility dialog box, the command line string will not match this example.

Note You can copy the command line string and append it to the dtexec command to create a job step in a SQL Server Agent job when you want the package to run on a regular schedule.

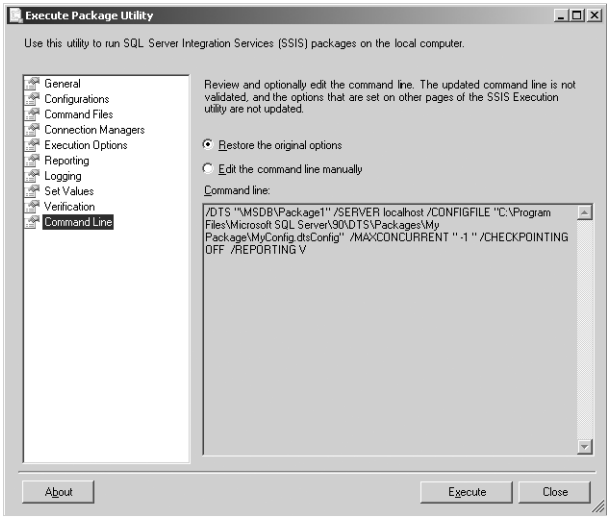

Figure 21-27 The Command Line page of the Execute Package Utility.

Package Management

There are several places where you can store packages for use by Integration Services. You can register a package with the Integration Services service by storing it in either the MSDB database or the SQL Server Package Store, but you can also reference a package file on the file system without registering it. In that case, you simply have to supply a path and package name. In SQL Server Management Studio, you can review the registered packages in the Stored Packages folder list, which is further divided into two folders, File System and MSDB. The File System folder contains packages that have been placed in the <SQL Server install directory>\90\DTS\Packages folder during package deployment. The MSDB folder lists packages that are stored in the sysdtspackages90 table of the *MSDB* database. The usual method of storing packages in this location is to use the package deployment utility, but you can also use the Import Package command (by right-clicking on the File System or MSDB folder) to locate, register, and store packages in the respective storage mode.

Monitoring Packages

Use the Running Packages folder to view a list of packages currently executing. When you click this folder in the Object Explorer window and then click the Report button in the Summary page toolbar, general information about running packages displays, such as the amount of time a package has been running. Refresh the folder to view the most recent information because it does not refresh automatically. If you click a specific package on the Summary page, you can view additional information about the package such as its version and description. To stop a package, right-click the package and then click Stop.

Summary

In this chapter, you've learned a lot about Integration Services, from fundamental concepts to designing and managing packages. Far from being a complete reference for Integration Services, this chapter introduced you to the most important things you need to know to start building basic packages and to put packages into production. Some advanced features were also briefly discussed so you can consider their usefulness for more complex data integration projects you tackle once you gain more experience. As data continues to proliferate in organizations, there will be plenty of opportunities to expand your data integration skills and take full advantage of the flexibility that Integration Services offers.