Search for
TechNet Go
TechNet Home
SQL TechCenter Home
SQL Server 2005
Technical Library
Technologies
Downloads
Events & Errors
Learning
Webcasts
Virtual Labs
Community
International TechCenters

Additional Resources
SQL Server 2005 Support Center
SQL Server 2005 Developer Center
Data Access and Storage Developer Center
SQL Server 2005 Product Evaluation

Managing and Deploying SQL Server Integration Services

Published: April 30, 2005
By Michael Otey, TECA, Inc and Denielle Otey, TECA, Inc

SQL Server Integration Services (SSIS) is an enterprise-level data integration platform. SSIS is an entirely new subsystem in SQL Server 2005. This paper tells you how to take advantage of the SSIS management and deployment features.

On This Page

Introduction

In today's businesses, decision-making processes and daily operations often depend heavily on data that is stored in a variety of data storage systems, formats, and locations. In order to turn this data into useful business information, the data typically needs to be combined, sanitized, standardized, and summarized. For instance, information may need to be converted to a different data type or heterogeneous database servers may store the necessary data using different schemas. Dissimilarities like these must be resolved before the data can be successfully loaded to a target system. In the past, the task of extracting and transforming data was primarily assigned to the database administrator (DBA). However, the need for access to critical data in a more timely manner has resulted in the need for other staff members to have these skills. This white paper introduces DBAs, database developers, Information Technology managers, and other Information Technology professionals to the management and deployment capabilities of Microsoft® SQL Server™ 2005 Integration Services (SSIS).

Microsoft SQL Server 2005 Integration Services (SSIS) is a comprehensive data integration platform that you can use to transfer, mine, transform, and consolidate your information from heterogeneous sources and load it to multiple systems. It's important to understand that SSIS is not just an Extraction, Transformation, and Loading (ETL) tool. SSIS is a complete data integration platform, providing a host of graphical development and management tools, services, programmable objects, and application programming interfaces (APIs). SSIS contains a workflow engine that supports complex logic and can be used for a wide variety of database maintenance and sophisticated data transfer operations. A great example of this can be found in the SQL Server 2005 Maintenance Plan Wizard. Beneath the surface, the Maintenance Plan Wizard is using SSIS projects to carry out all of the database maintenance operations supported by the wizard.

In this whitepaper, you'll learn how to manage and deploy SSIS projects. In the first section of this white paper, you'll get an overview of the SSIS architecture; next, you'll learn about the new SQL Server 2005 tools for managing SQL Server Integration Services. Then this paper will dive in deeper and provide you with some best practices for creating, deploying and managing SSIS packages.

Brief Overview of Integration Services

In previous versions of SQL Server, Data Transformation Services (DTS) was the primary Microsoft ETL tool. Although DTS was an extremely useful tool, it had some limitations in terms of scalability and the ease of deploying packages to different SQL Server systems. SSIS is a completely new system that was designed from the ground up. Like DTS, SSIS includes graphical tools and programmable objects, making it easier for company employees to work with it. However, SSIS has taken a new tack and has split the control flow and data flow of packages into separate components, enabling the creation of much more complex and robust packages.

The new SSIS architecture basically consists of two parts: the Data Transformation Run-time engine, which handles the control flow of a package; and the Data Flow engine or Data Transformation Pipeline engine, which manages the flow of data from data sources, through transformations, and on to destination targets. You can see an overview of the SSIS architecture in Figure 1.

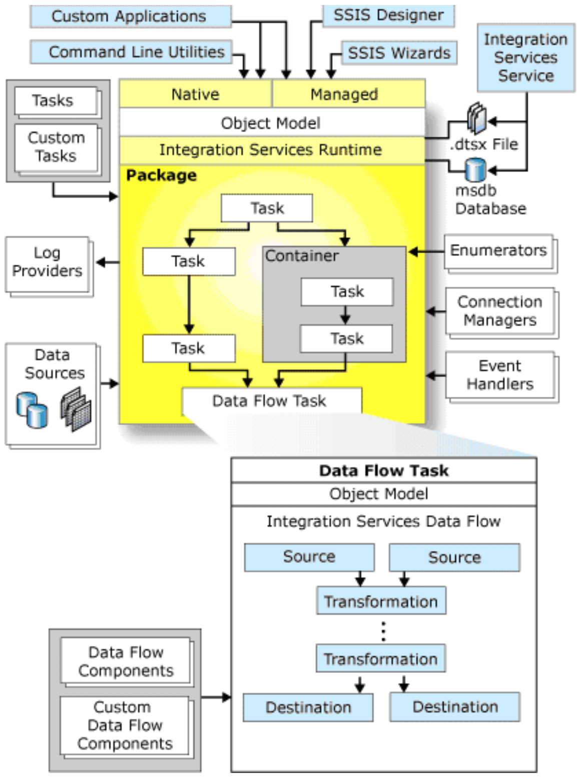


Figure 1: SQL Server Integration Services overview

The SSIS Designer is a set of graphical tools that allow you to create and manage SSIS packages by dragging objects in the user interface. SSIS wizards are graphical tools that guide you through importing data or creating simple packages. SSIS also provides command line tools for creating and managing packages. These tools are discussed in more detail in the next section.

The Data Transformation Run-time engine provides services to the packages and its contents. For instance, the Data Transformation Run-time engine handles package storage, package execution, logging, debugging, event handling, package deployment, and the management of variables, transactions, and connections.

The package is the core component of SSIS. SSIS packages are the basic unit of deployment and execution. You can create packages using the graphical tools provided in SSIS, or you can create them programmatically. A package consists of a collection of elements such as control flows and data flows, connections, variables, and event handlers. You can configure the properties of your packages to incorporate transactions or implement security, and these configuration objects are also stored within the package.

When you create a package you define three important elements of the package: the control flow, the data flow, and the precedence constraints that link different tasks in a package together. You can see the relationship of the SSIS package elements in Figure 2.

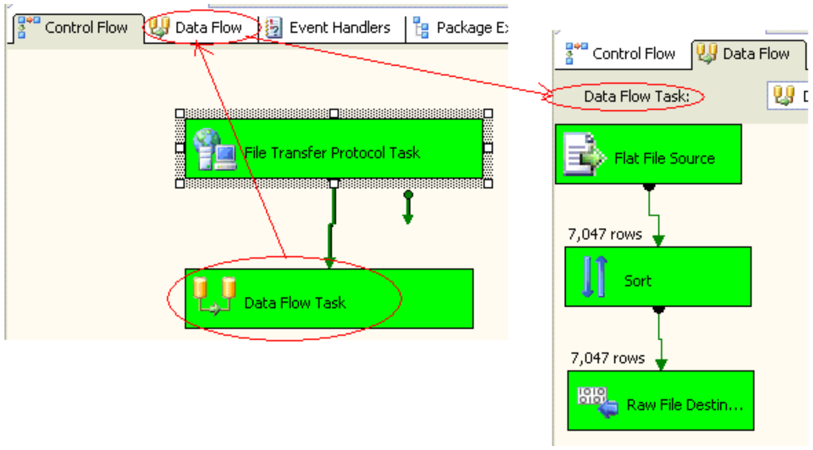


Figure 2: SSIS package elements

The control flow of the package defines the actions that are executed when the package runs. SSIS provides three types of control flow objects: containers, tasks, and precedence constraints.

- Containers provide structure to your packages. They group tasks and other containers into meaningful, logical work units.
- Tasks perform a wide variety of functions in your packages. SSIS packages contain two types of tasks; control flow tasks and data flow tasks. Control flow tasks perform a variety of workflow actions. For instance, the Execute SQL task executes SQL statements on the target database platform; the File System task can read and write from the host server's file system; the File Transfer Protocol task uses the FTP protocol to transfer files to remote systems; and the Send Mail task uses the SMTP protocol to send e-mail messages. In addition, Maintenance Plan tasks perform a set of routine database maintenance operations. Maintenance Plan tasks includes the Backup Database task, the History Cleanup task, the Rebuild Index task, and the Shrink Database task. The most important and specialized tasks are data flow tasks that move data between different data sources. Using a data flow task, you can define a source and target destination such as a flat file, Excel, OLE DB, and SQL Server.
- Precedence constraints link the items in your package into a logical flow and specify the conditions upon which the items are executed. The three default precedence constraints enable you to control package execution control flow based on the completion, the success, or the failure of a task. For instance, you might create a precedence constraint that links an Execute SQL task and a data flow task where the package will only execute the data flow task if the Execute SQL task succeeds. SSIS supports creating precedence constraints based on an evaluation operator or the execution results of a task. An example of a precedence constraint that is based on an evaluation operator might use the contents of a variable or the evaluation of an expression to determine the flow to the next task.

The data flow tasks, using the Data Transformation Pipeline engine, manage the flow of data and transformations from the data source adapters to the data destination adapters. Integration Services provides three types of data-flow elements: source adapters, transformations, and destination adapters. Like you would expect, source adapters extract data from sources such as tables, views, and files; transformations modify and summarize the data; and destination adapters load the data into the target data stores.

SSIS Development and Deployment Tools

The new SSIS productivity and management tools consist of studio environments to graphically assist you with the design, creation, debugging, and management of your data transformation solutions. SSIS also includes command line utilities that allow you to execute and manage your SSIS packages.

SSIS Wizards

The Import and Export Wizard is the simplest method to use for building SSIS packages. The Import and Export Wizard can be started from the following locations:

- Business Intelligence Development Studio. Open Solution Explorer, right-click the **Package** node, and select the **Import and Export** option.
- SQL Server Management Studio, Open the SQL Server node, right-click a database, and select the **Import and Export** option.
- Run DTSWizard.exe. DTSWizard.exe is a command line utility that starts the Import and Export Wizard.

The Import and Export Wizard

The Import and Export Wizard interactively guides you through the process of copying data from a variety of sources including SQL Server, flat files, Microsoft Access (.mdb), Microsoft Excel (.xls), and other OLE DB providers. Using the Import and Export Wizard, the package is created with a minimum amount of transformation facilities. You may save it to the file system as a DTSX file, and then open and edit it in the Designer, adding more complex tasks and package workflow.

The Configuration Wizard

The Configuration Wizard walks you through creating configurations that may be deployed with your packages. The Configuration Wizard is only started through the SSIS Designer.

To start the Configuration Wizard:

1.On the **SSIS** menu, select **Package Configuration.**.

2.Select **Enable PackageConfiguration**.

3.Select the **Add** option

The Configuration Wizard enables you to create SSIS configurations that dynamically change variables and object properties at run time, making your packages more flexible and easier to move from your development environment to production environments. For example, using package configurations you can pass parameters like the system name and login information to your packages at run time making them portable across multiple servers.

The Package Installer Wizard.

The Package Installer Wizard is run to install packages to the file system or to a SQL Server database. You deploy packages by creating a deployment utility using the Business Intelligence (BI) Development Studio for your SSIS project that contains the package you wish to deploy. You then run the Package Installer Wizard on the destination system, which will guide you through installing your package and editing any configuration objects

The Package Installer Wizard is started by running the DTSInstall.exe command line program. This utility is included when you build a package using the SSIS Designer.

The Migration Wizard

The Migration Wizard assists you in migrating your existing SQL Server 2000 DTS packages to SQL Server 2005. The Migration Wizard is able to automatically convert most existing SQL Server 2000 DTS packages that use the standard tasks and transformation but can not convert DTS packages that use custom tasks and transformation. To start the Migration Wizard, in the BI Development Studio, select **Project**, then select **Migrate DTS 2000 Package**.

SSIS Deployment Tool	Location
Import and Export Wizard	Start Menu | All Programs | Integration Services
Configuration Wizard	SSIS Designer | Package Designer | SSIS Package Configuration | Enable Package Configuration | Add
Package Installer	DTSInstall.exe

Migration Wizard	BI Development Studio | Project | Migrate DTS 2000 Package

Table 1: SSIS tools

SSIS Designer

The SSIS Designer is a set of graphical tools that allow you to create, execute, and debug SSIS packages. It is available in the Business Intelligence Development Studio. To start the SSIS Designer:

1.Open up the BI Development Studio.

2.Select **File**, then the **New Project** option to display the BI Development Studio New Project dialog box.

3.Select **Integration Service Project** to display the SSIS Designer.

The SSIS Designer provides a powerful drag-and-drop development surface that you can use to create robust and complex data transformation solutions containing multiple connections to heterogeneous data sources, complex workflow, data transformations, and event-driven logic. The SSIS Designer is an all new tool and even experienced DTS users will need to spend some time getting familiar with it. One quick way to quickly get some insight into the SSIS Designer is to use it to open and edit simple packages that were created by using the SSIS Import and Export Wizard. The SSIS Designer is shown in Figure 3.

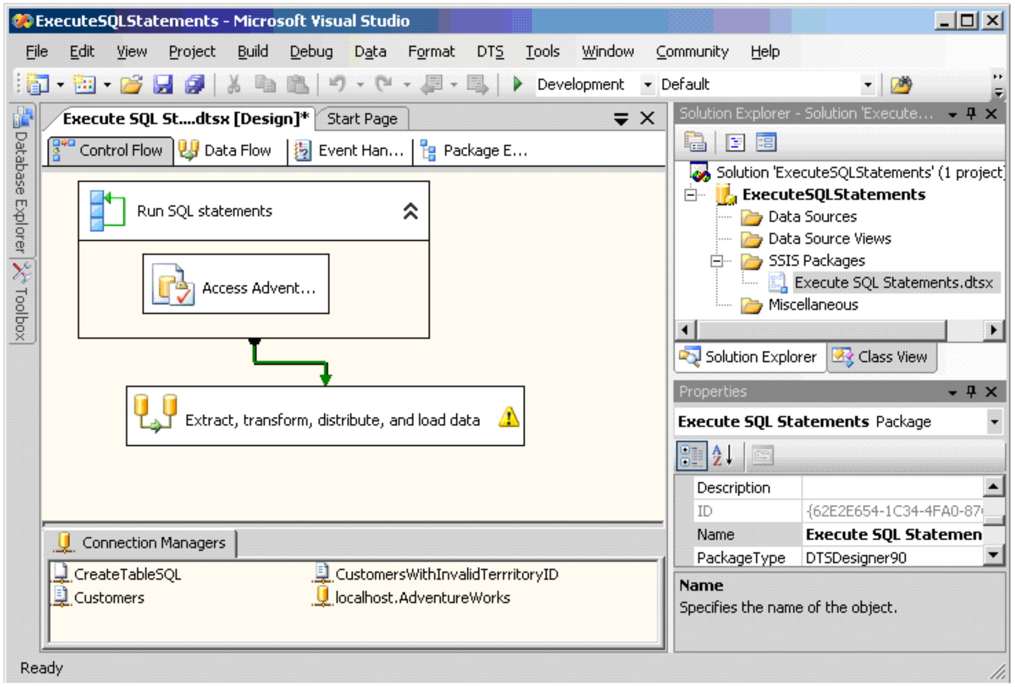

ExecuteSQLStatements - Microsoft Visual Studio
File Edit View Project Build Debug Data Format DTS Tools Window Community Help
Development
Default
Database Explorer
Toolbox
Execute SQL St....dtsx [Design]*
Start Page
Control Flow
Data Flow
Event Han...
Package E...
Run SQL statements
Access Advent...
Extract, transform, distribute, and load data
Connection Managers
CreateTableSQL
Customers
CustomersWithInvalidTerrritoryID
localhost.AdventureWorks
Solution Explorer - Solution 'Execute...
Solution 'ExecuteSQLStatements' (1 project)
ExecuteSQLStatements
Data Sources
Data Source Views
SSIS Packages
Execute SQL Statements.dtsx
Miscellaneous
Solution Explorer
Class View
Properties
Execute SQL Statements Package
Description
ID {62E2E654-1C34-4FA0-87
Name Execute SQL Statemen
PackageType DTSDesigner90
Name
Specifies the name of the object.
Ready

Figure 3: SSIS Designer

When you create a new SSIS solution in BI Development Studio, an empty package is automatically created and added to the project. The SSIS Designer's graphical user interface shows separate windows that allow you to build control flow, data flows, and event handlers for your package. The SSIS Designer enables you to add functions such as logging, implementing checkpoints for restarting of packages, incorporating transactions to manage rollback, mapping variables, and setting breakpoints for debugging. The SSIS Designer also includes a Package Explorer pane that provides a hierarchical view of the package elements.

SQL Server Management Studio

The SQL Server Management Studio is a replacement for the Enterprise Manager in SQL Server 2000. The new SQL Server Management Studio performs all of the functions that were performed by the old SQL Server Enterprise Manager. In addition, it allows you to graphically display and manage the execution of your packages. The Management Studio displays a folder view of the packages that are stored in the SQL Server database or the file system. This view is customizable permitting you to add, delete, or rename your own folders. An Import and Export feature allows you to copy packages from one storage format to another. For more information, see Managing SSIS Packages with SQL Server Management Studio in this paper.

SQL Server 2005 includes a new SSIS service, which is a Microsoft Windows® service that enables you to use the SQL Server Management Studio to manage SSIS packages and monitor running packages.

SSIS Package Management Utility (dtutil)

SSIS contains the Package Management Utility (**dtutil**) that you can run from a command line. The **dtutil** command prompt allows you to manage SSIS packages that are stored in the database or the file system. You may specify to copy, move, delete, or verify the existence of a package using the appropriate command prompt options.

SSIS Package Execution Utilities (dtexec & dtexecui)

In addition to the Package Management Utility, two package execution utilities are provided with SSIS: **dtexec** and **dtexecui**.

You run the **dtexec** utility from the command line using the appropriate command prompt option. You can use the **dtexec** tool to set all the configuration and execution features of your packages including connections, properties, variables, and logging. The new **dtexec** tool is the primary tool that you'll want to use to execute your SSIS packages from the command line or your own command shell scripts.

The **dtexecui** utility displays a graphical user interface that allows you to load and run packages. You can use this interface to interactively set the package's configuration and run-time attributes before running the package. One really handy feature that you'll find in the **dtexecui** utility is its ability to create execution commands for **dtexec** complete with the required parameters. You can then copy and paste these commands into your own SSIS package execution batch files.

SSIS Command Line Tool	Description
Dtutil	Manage SSIS packages
Dtexec	Execute SSIS packages
dtexecui	Displays a graphical user interface to load and run SSIS packages
Dtswizard	Launch the Import/Export Wizard

Table 2: SSIS command line tools

Development Flow

Microsoft has made extracting, manipulating, and loading data easier to understand and use with the SSIS data solution platform. You start by designing a SSIS package that handles both control flow and data flow of the data. You can design packages using either the graphical BI Development Studio or you can use the Import and Export Wizard to create quick and simple packages. Once created, you can store the packages in either the file system or the SQL Server database. The **dtutil** command line utility is able to move packages between SQL Server systems and from the msdb database to the file system. You can then run a package using the SQL Agent job scheduler or by using the using the **dtexec** or **dtexecui** utilities. You can monitor and manage your packages using the SQL Server Management Studio. See Figure 4 for an illustration of the flow of these packages.

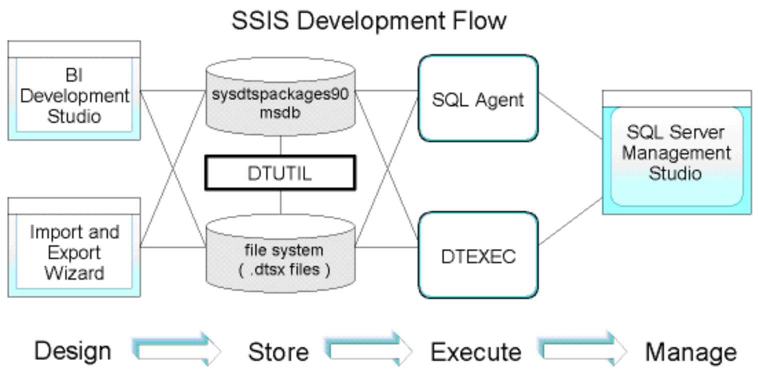


Figure 4: SSIS development flow

[Top of page](#)

Managing the SSIS Service

You can monitor the execution of SSIS packages using the SQL Server Management Studio. The SQL Server Management Studio includes a new SSIS Server node that lists saved and running SSIS packages. The SSIS management node only appears after the SSIS Service is started. The SSIS service is installed when you select the option to install SQL Server Integration Services and its purpose is to enable the management of SSIS packages. The SSIS service is normally started by default when SSIS is installed on the system. In case it isn't started, you can use the following procedure to start it manually.

To manually start the SSIS Service:

1.On the **Start** menu, click **All programs**.

2.Select **Microsoft SQL Server 2005**, then select **SQL Computer Manager**.

3.Scroll down to the **Services and Applications** section and expand the **SQL Computer Manager**. Then expand the **SQL Server 2005 Services** node.

4.Select **SSIS** Server.

5.Right-click the service entry in the right pane and select **Start** on the shortcut menu to start the service as shown in Figure 5.

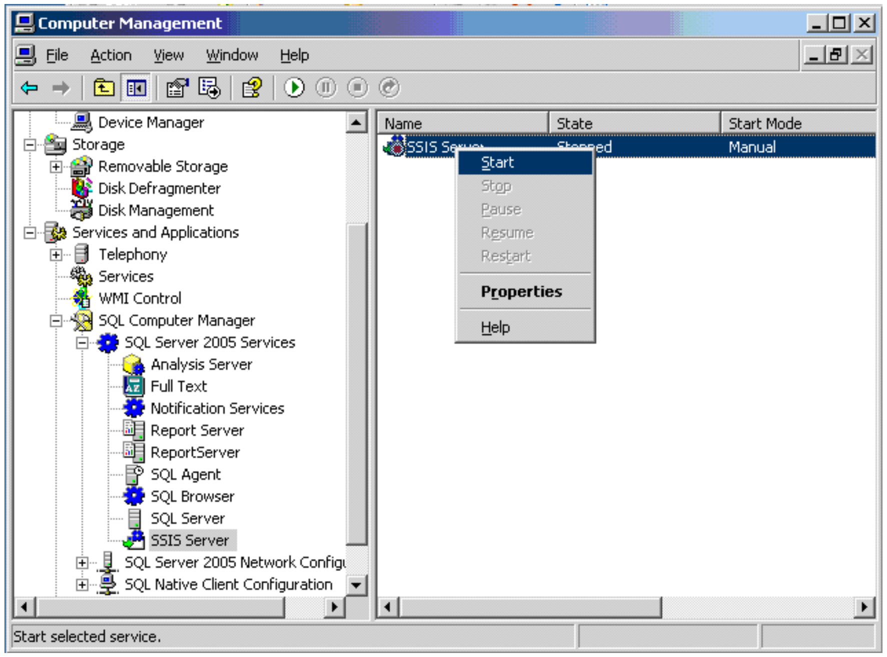


Figure 5: Starting the SSIS Server

If you want the SSIS service to always run, you can change the startup type to Automatic. This will automatically start the SSIS service whenever the server starts.

It's important to understand that the SSIS service is designed to enable the monitoring of SSIS packages; it is not necessary that it be running in order to execute a package. Likewise, stopping the SSIS service won't prohibit you from running SSIS packages. However, if the SSIS service is running, the SSIS Designer will be able to use it to cache objects that are used in the designer, enhancing the performance of the designer.

Managing SSIS Packages with SQL Server Management Studio

After the SSIS service has been started, you can use it to monitor running SSIS packages in SQL Server Management Studio. One of the key advantages to the SSIS service is the fact that it enables you to monitor packages running on both the local SQL Server as well as remote SQL Server systems that are registered in the SQL Server Management Studio. It's important to note that while the SQL Server Management Studio enables you to manage existing SSIS packages, it does not allow you to create them. Packages are created using the BI Development Studio, the Import and Export Wizard, or programmatically using the SSIS APIs.

To manage SSIS packages using the SQL Server Management Studio:

- Open the SQL Server Management Studio.
- In the Connect to Server dialog box, select **Integration Services** from the **Server Type** list.
- Supply the name of the SQL Server at the **Server Name** prompt and provide your authentication information.

SQL Server Management Studio opens and the Object Explorer displays the SSIS management information (Figure 6).

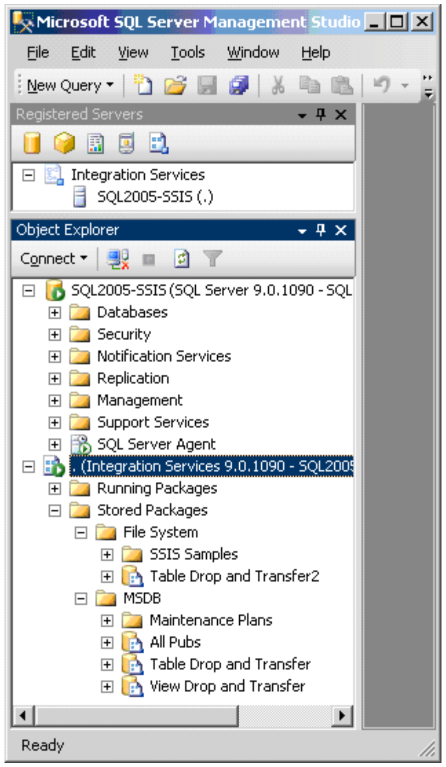


Figure 6: Managing SSIS packages with SQL Server Management Studio

By default, the Integration Services server node presents two folders for working with SSIS packages: the Running Packages folder and the Stored Packages Folder. The Running Packages folder displays the SSIS packages that are currently executing on the local server. As you might guess, the contents of this folder are constantly changing to reflect the current system activity. The contents of the Running Packages folder must be manually refreshed to keep the display updated with the current running packages.

The Stored Packages folder lists the saved SSIS packages that have been registered on the local server. By default this folder contains two subfolders: the File System Folder and the MSDB folder. The File System folder lists the SSIS packages that have been saved in the File system while the MSDB fold lists the packages that are stored in the sysdtspackages90 table in the msdb database. It's important to note that the SSIS server isn't aware of packages stored in the File System until those packages have been imported to the File System folder in the SSIS service.In addition to listing the saved SSIS packages, the SQL Server Management Studio also enables you to work with them. Right-clicking a package displays a shortcut menu that enables you to perform a number of task including:

- **New Folder**. Creates a new folder in Object Explorer for displaying packages saved in the file system or in the sysdtapackages90 table.
- **Import Package**. Imports the package from the file system to the msdb database
- **Export Package**. Exports the package from the msdb database to the file system.
- **Run Package**. Executes the package using dtexecui.
- **Delete**. Deletes the package.
- **Rename**. Renames the package.

While the SQL Server Management Studio is shipped using the default folder locations of MSDB and File System, you can freely add new folders to this structure using the **Create New Folder option**. When you create a new folder beneath the File System system folder, a new directory will be created in the file system. By default, these directories are located in the c:\Program Files\SQL Server\90\Packages directory. Importing packages to a File System folder will result in the package being copied to the like named directory in the file system. For folders that are created under the MSDB folder, a new entry is added to the sysdtspackackefolder90 table that tracks the folder structure. However, it's important to realize that the packages themselves are still stored in the msdb sysdtspackaes90 table. The **Folders** option in the SQL Server Management Studio essentially gives you a way to apply and organization structure to your packages, enabling you to group like packages together.

Modifying the Default SSIS Package Folders

The two default folders provided by SQL Server Management Studio, the File System and MSDB folder, themselves are actually configurable. The definitions for these folders are stored in the XML file that the SSIS service reads at startup. The SSIS service retrieves the location of this file from the following registry location: HKLM\SOFTWARE\Microsoft\MSDTS\ServiceConfigFile.

To customize the SSIS startup folders you can create a new XML file that follows the required format and then point the SSIS service to that file by updating the ServiceConfigFile registry key. The following listing illustrates a sample of the SSIS service configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<DtsServiceConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<StopExecutingPackagesOnShutdown>true</StopExecutingPackagesOnShutdown>
<TopLevelFolders>
<Folder xsi:type="FileSystemFolder">
<Name>Fsystem__SQL2005-SSIS</Name>
<StorePath>C:\_work\VisualStudioProjects\DTS</StorePath>
</Folder>
<Folder xsi:type="FileSystemFolder">
<Name>Fsystem__SQL2005-SSIS MSN Money Projects</Name><StorePath>C:\_MoneyChartingRebuild\MoneyChartingRebuild2</StorePath>
</Folder>

<Folder xsi:type="FileSystemFolder">
<Name>Fsystem__SQL2005-SSISdts01 test packages</Name>
<StorePath>\\SQL2005-SSISdts01\c$\_work\testPackages</StorePath>
</Folder>
<Folder xsi:type="SqlServerFolder">
<Name>SQL__SQL2005-SSIS</Name>
<ServerName>SQL2005-SSIS</ServerName>
</Folder>
<Folder xsi:type="SqlServerFolder">
<Name>SQL__SQL2005-SSISdts01</Name>
<ServerName>SQL2005-SSISdts01</ServerName>
</Folder>

</TopLevelFolders>
</DtsServiceConfiguration>
```

You can see the results of using this custom SSIS service configuration file in Figure 7.

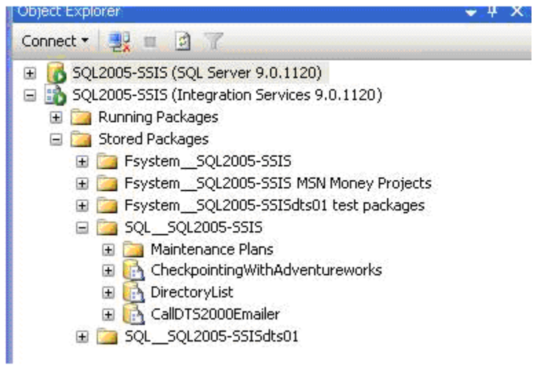

Object Explorer
Connect
SQL2005-SSIS (SQL Server 9.0.1120)
SQL2005-SSIS (Integration Services 9.0.1120)
Running Packages
Stored Packages
Fsystem__SQL2005-SSIS
Fsystem__SQL2005-SSIS MSN Money Projects
Fsystem__SQL2005-SSISdts01 test packages
SQL__SQL2005-SSIS
Maintenance Plans
CheckpointingWithAdventureworks
DirectoryList
CallDTS2000Emailer
SQL__SQL2005-SSISdts01

Figure 7: Customizing the SSIS service folders

One way you might want to use the SSIS service configuration ability is to create a common management folder structure for multiple servers. To do this, you could store the service configuration file in a central file share and point multiple servers to the shared configuration file. This would enable all of the servers to have the same SSIS folder structure.

Managing DTS 2000 Packages with SQL Server Management Studio

In addition to managing native SSIS Packages, the SQL Server Management Studio can also manage DTS packages that have been created in SQL Server 2000 and that are stored in the dtspackages table of the msdb database. To manage legacy SQL Server 2000 DTS packages using SQL Server Management Studio, open the Object Explorer using the server type of Database Server in the Connect to Server window. The Object Explorer will display a DTS 2000 Packages node like the one shown in Figure 8.

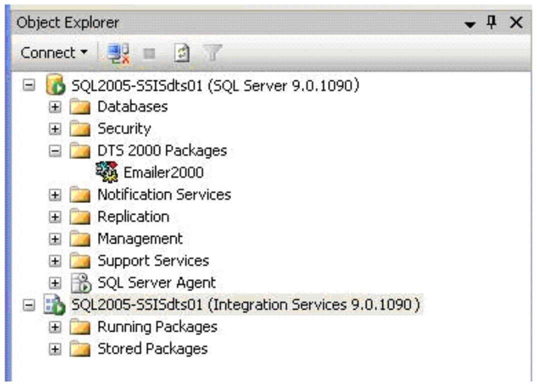


Figure 8: Managing DTS 2000 packages with SQL Server Management Studio

The DTS 2000 Packages folder lists the SQL Server 2000 packages that are in the sysdtspackages table in the msdb database. You manage the DTS by right-clicking a package which displays a shortcut menu that you can use to perform a number of tasks including:

- **Open**. Starts the SQL Server 2000 DTS Designer.
- **Migrate a package**. Opens the Migration Wizard to migrate the DTS package to an SSIS package.
- **Export**. Exports the package from the msdb database to the file system.
- **Delete**. Deletes the package from the msdb database.
- **Rename**. Renames the package.

It is important to note that in order to use the **Open** option, the SQL Server 2000 DTS Designer must be installed on the SQL Server 2005 system. The SQL Server 2000 DTS Designer will be present if an existing SQL Server installation has been upgraded to SQL Server 2005 or if the SQL Server 2000 Management Tool has been installed on the SQL Server 2005 system. More information about working with legacy DTS packages is presented in the Migrating SQL Server 2000 DTS Packages section in this paper.

Top of page

Package Manageability

The combination of the SQL Server Management Studio and the SSIS service enables you to manage the execution of SSIS packages from a system standpoint. However, by taking advantage of some of the new features in SSIS, you can build in a much more granular level of management and robustness into your SSIS packages.

Checkpoint Restart

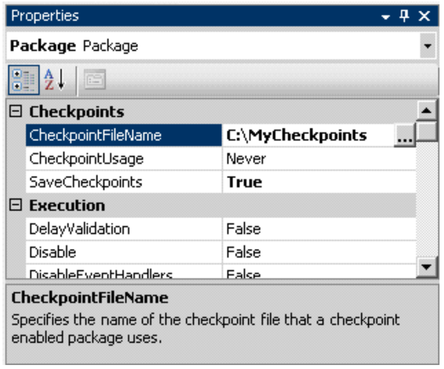
Recovering from a package failure was one of the limitations that were present in DTS in SQL Server 2000. If a package execution failed, the entire package needed to be rerun from the very beginning. This could be very time consuming for complex packages or packages that performed large data transfers. In SQL Server 2005, SSIS support for checkpoints addresses this DTS limitation. When an SSIS package is configured to use checkpoints, information about the packages execution and status is written to a checkpoint files. If a package that is using checkpoints fails, there is no need for the package to be rerun from the very beginning. Instead, the Data Transformation Run-time engine can process the checkpoint file and the package can be restarted from the container that was being processed at the point of failure. Checkpoint restart can significantly simplify the recoverability of packages that contain complex operations and can provide significant time saving for packages that contain long running tasks because the package does not need to reprocess all of the tasks prior to the checkpoint.

It's important to understand that checkpoints apply to the package's control flow not to the package's data flow. Control-flow containers are the basic unit of checkpoint restartability. When the execution of a package that uses checkpoints is resumed, it begins with the failed control flow task. All of the data flow within that task will need to be rerun from the beginning of the control flow task. In other words, if the package was interrupted while performing a data flow task, then the data flow task will be executed again from the beginning—the task will not resume from the last row transferred.

Checkpoints are enabled by setting the package's **SaveCheckpoints** property to True in the SSIS package properties. If you enable checkpoints, you also need to tell SSIS where to save the checkpoint data by assigning a file to the checkpoint in the file's **CheckpointFileName** property. In addition, the processing of the checkpoint file is controlled by the **CheckpointUsage property**. The **CheckpointUsage** property supports the following values.

- **Never**. The checkpoint file is not used and the package runs from the beginning of the package control flow.
- **Always**. The checkpoint file is always used and the package restarts from the point of the previous execution failure. The package's execution will fail if the checkpoint file is not found.
- **IfExists**. If the checkpoint file exists, the package restarts from the point of the previous execution failure; otherwise, it runs from the beginning of the package's control flow.

You can see an example of a package's checkpoint properties in Figure 9.


Properties

Package Package

Checkpoints

CheckpointFileName	C:\MyCheckpoints
CheckpointUsage	Never
SaveCheckpoints	True

Execution

DelayValidation	False
Disable	False
DisableEventHandlers	False

CheckpointFileName

Specifies the name of the checkpoint file that a checkpoint enabled package uses.

Figure 9: Enabling checkpoints

Once checkpoints have been enabled for a package, SSIS will track the package's execution by writing data into the checkpoint file. SSIS will use the data in this file to determine which control flow tasks in the package have been executed. In addition, if a package is using a log provider log, SSIS will write events to the provider detailing the package's checkpoint usage. For more information about using log providers, see SSIS Package Logging in this paper.

Package containers include two additional properties that control how the package will respond to failures: FailPackageOnFailure and FailParentOnFailure. If FailPackageOnFailure is enabled and there is an error executing the container, the package execution will be terminated and the package will be restarted from this container. If FailParentOnFailure is enabled and the package is running as a child package from a parent container and an error occurs, then the child package and the parent package will be terminated and restart will resume with the child package.

Transaction Support

To ensure that the database remains in a consistent state even in the event that a package fails, SSIS provides full support for database transactions. Like a standard database transaction, the database changes performed by a package can be committed as a unit if a package task is successful or can be rolled back as a unit if the task fails, thereby preserving database integrity. For instance, if transaction support is enabled and the package that contains multiple data flow tasks fails, all of the database update actions performed by both of the data flow tasks will be rolled back, thereby ensuring that the database is in a consistent state. When the error condition is corrected and the package is later restarted, both data flow tasks will be executed again. SSIS transactions can also make use of the Microsoft Distributed Transaction Coordinator (MS DTC) when used with connections that support MS DTC. Likewise for SQL Server connections, SSIS transactions also provide support for Multiple Active Results Sets (MARS).

Transactions can be enabled for all SSIS container types including packages, containers, the For Loop, the Foreach Loop and Sequence containers. You enable transaction support using the container's **TransactionOption** property, which can be set in the SSIS Designer's properties window or programmatically. The **TransactionOption** property supports the following values:

- **Not Supported**. The container does not start a transaction and will not join an existing transaction that was started by a parent container.
- **Supported**. The container does not start a transaction but will join an existing transaction that was started by a parent container.
- **Required**. The container starts a transaction. If an existing transaction has already been started by the parent container, the container will join it.

Transformation Error Handling

SSIS packages include the ability to transform data as it is moved from the source to the destination data adapters. To handle errors that may occur in the transformation process, SSIS provides the ability to decide on a per-column basis how to handle data that cannot be transformed. You can choose to ignore bad data for a given field or you can choose to redirect the row to a holding table for later reprocessing. Using a central database for the storage of redirected rows enables operations to centrally audit and review package transformation errors. To help identify the source packages, the error flow process itself can transform the error row adding the package name and execution identification information to the data that's written to the central error logging database.

Event Handlers

The ability to raise and handle events is another new feature found in SQL Server 2005 that can enhance the manageability of SSIS. Event handlers enable SSIS packages to programmatically respond to events that are raised at run time by containers and tasks.

At run time, events are fired by package executables like tasks, containers, and packages to signal a number of different states including: error conditions, when a task has started, when a task completes, or a change in variable status. You can extend a package's functionality by take advantage of SSIS's ability to add custom event handlers for the different package elements. Using customer event handlers your package can automatically perform a number of different management tasks including cleaning up temporary file and database objects, sending e-mail or other notifications, or creating logging information. Event handlers are created using the SSIS Designer's Event Handler tab that's found in the BI Development Studio.

Execute Package Tasks

There are times when you may want a complex package workflow broken down into separate packages or units of work. Execute Package tasks enable this behavior by providing the ability to execute one package from within another package. For example, if you have a parent package that sends e-mail, you can create a child package that does the action of connecting to the SMTP server, and simply passes the e-mail address from the parent package to the child package.

Creating smaller, more focused child packages and then calling them from multiple parent packages simplifies the development process and helps promote package reuse and manageability.

Scheduling Package Execution

You can schedule the execution of SSIS packages by using the SQL Server Agent. The SQL Server Agent is the built-in job scheduling tool that is provided with SQL Server 2005. Like the SSIS Server, the SQL Server Agent is implemented as a Windows service and that service must be running in order to support job scheduling. Also, like the SSIS Server, the SQL Server Agent service is managed using the SQL Computer Manager that's a part of the Computer Management MMC console.

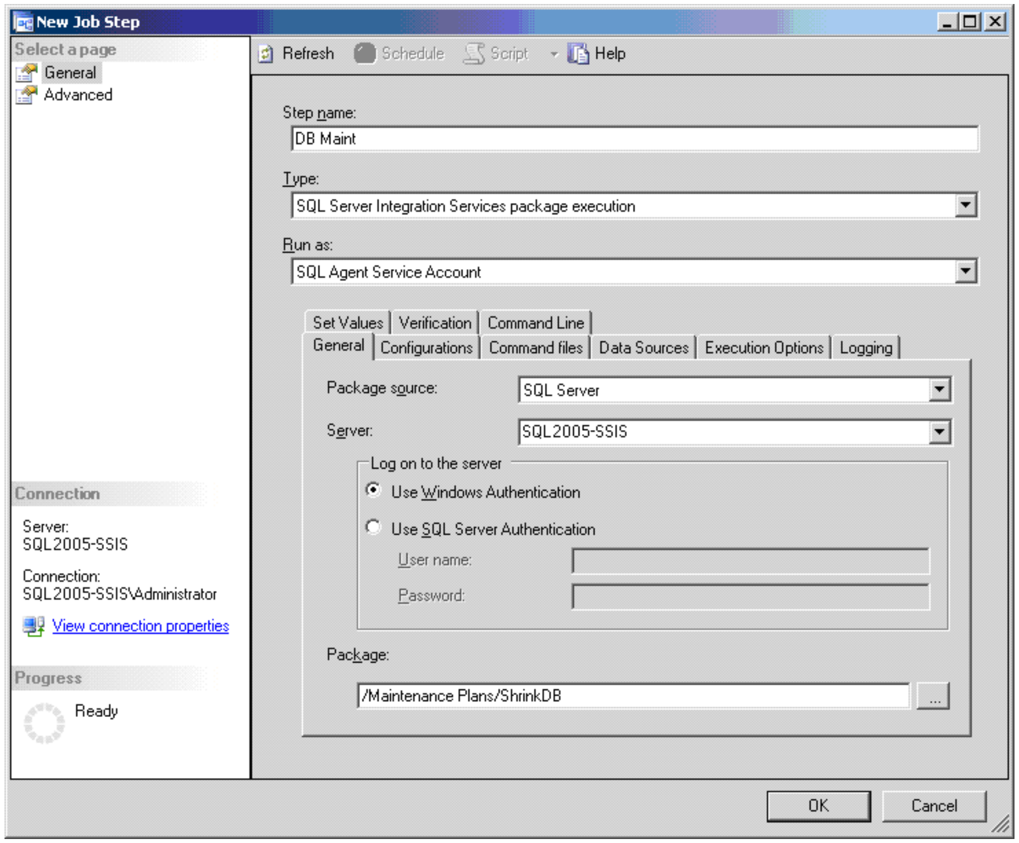
To create a new SQL Server Agent job to schedule a SSIS package using SQL Server Management Studio:

1.Open Object Explorer.

2.Expand the **SQL Server Agent** node.

3.Right-click the **Jobs** node.

4.A **SQL Server Agent** job is comprised of a series of job steps. To execute an SSIS package, you add a new Job Step to the SQL Server Agent job. To do this, select the **Steps** page and then click the **New** button to display the New Job Step dialog box shown in Figure 10.



Step name: DB Maint

Type: SQL Server Integration Services package execution

Run as: SQL Agent Service Account

Package source: SQL Server

Server: SQL2005-SSIS

(●) Use Windows Authentication

( ) Use SQL Server Authentication

User name:

Password:

Package: /Maintenance Plans/ShrinkDB

Server: SQL2005-SSIS

Connection: SQL2005-SSIS\Administrator

View connection properties

Ready

Figure 10: Scheduling SSIS packages

When you create a job step that executes an SSIS package, the SQL Server Agent enables you to specify the same run-time properties that you can use when the package is executed from the SSIS Designer or by the dtexec utility. This includes supplying configuration files, enabling checkpoints, and adding logging. If the job contains multiple packages or successive job steps, you can set up procedures between each step that control the execution of the job based on the completion, success, or failure of each job step.

Remote Package Execution

To run SSIS packages on remote SQL Server systems, you can use the SQL Server Management Studio to create a SQL Agent job on remote server. That SQL Agent job can then perform an execute Agent Task that calls the **dtexec** utility to run the SISS package on the remote system.

In addition, you can design packages that are able to execute SSIS packages on remote SQL Server systems by using the Execute SQL Server Agent task which is found in the SSIS Designer toolbox under the Maintenance Plan tasks section. When you add the Execute SQL Server Agent task to the SSIS Designer, you can set its Connection properties to point to the remote server. Then when the task is executed, the SQL Server Agent will execute a package on the remote machine.

Securing SSIS Packages

SQL Server 2005 provides several new mechanisms for securing SSIS packages—these include package encryption and the digital signing of packages. The encryption of SSIS packages enables the protection of data within the package. SSIS packages are encrypted either when they are created or when they are exported. SSIS uses the Triple Data Encryption Standard (DES) cipher algorithm with key length of 192 bits. Encryption that's based on a user key uses the Data Protection API (DPAPI) standards. Package encryption is controlled using the package's **ProtectionLevel** property. SSIS supports the following values for the **ProtectionLevel** property.

- **DontSaveSensitive**. Sensitive data like user authentication data is not saved in the package. When the package is later opened using the SSIS Designer, the sensitive data will not be present and the user will need to provide the sensitive data. As with all of the "sensitive" **ProtectionLevel** properties, only the sensitive data in the package is affected. The rest of the package is unchanged.

- **EncryptSensitiveWithUserKey**. Sensitive data is saved as a part of the package but that data is encrypted with a key that's based on the user who created or exported the package. Only the user who created the package will be able to run the package. If another user opens the package using the SSIS Designer, the previous sensitive data will remain encrypted and the SISS Designer will open the package without the sensitive data.

- **EncryptSensitiveWithPassword**. Sensitive data is saved as a part of the package but that data is encrypted with a password that the user supplies when the package is created or exported. If another user opens the package using the SSIS Designer, they must provide a password to access the encrypted data. If they do not know the password, the SISS Designer will open the package without the sensitive data.

- **EncryptAllWithPassword**. The entire contents of the package will be encrypted with a password that the user supplies when the package is created or exported. When the package is opened in the SSIS Designer, the user must provide the package's password. If the user doesn't know the password, they will not be able to access the package.

- **EncryptAllWithUserKey**. The entire contents of the package will be encrypted with a key that's based on the user key is based on the user who created or exported the package. Only the user who created the package will be able to open it.

- **ServerStorage**. No encryption is added to the package. Instead the package's contents are secured based on the database's object access security. If the ServerStorage value is used, the package must be saved to SQL Server in the sysdtspackages90 table in the msdb database. It cannot be saved to the file system.

The SSIS package security model is also extensible. There is a Sensitive attribute in the XML schema of an SSIS package that controls access to the package's properties. For example, if the Sensitive attribute is set to 1, when the user opens the package all sensitive data will be removed. Developers can incorporate this Sensitive attribute to get the same type of protection for their SSIS custom components.

If you are using the ServerStorage **ProtectionLevel** property, the method of controlling access to the packages saved in the database is by using SQL Server Database roles. By default, SQL Server 2005 provides the following roles for SSIS package management . You can find these by opening SQL Server Management Studio, then expanding the msdb database, Roles, Database Roles node:

- **db_dtsadmin**. SSIS package administrator rights.

- **db_dtsltduser**. Rights to execute only the SSIS packages the user has been given permission for.

- **db_dtsoperator**. Operation rights to SSIS packages including the ability to run as well as backup and restore packages.

You can also create your own custom database roles for SSIS Package management. Add the appropriate users to those roles and then assign those roles to your own SSIS packages.

You can enable package roles using the SQL Server Management Studio by right-clicking a saved package and then selecting the **Package Roles** option on the shortcut menu as is shown in Figure 11.

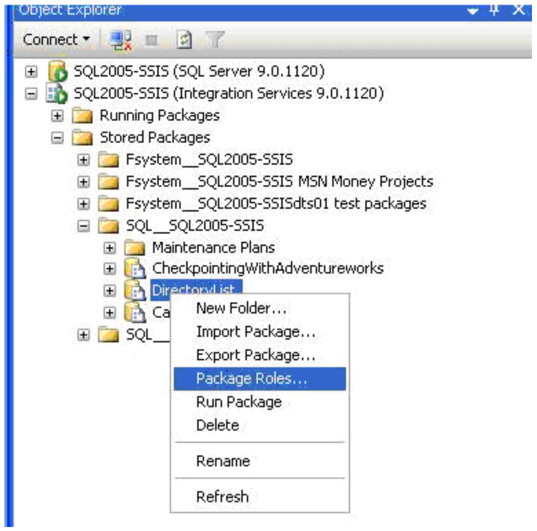


Figure 11: Adding package roles

SSIS packages can also be digitally signed. The digital signing of packages enables SQL Server to verify the authenticity of a package when it is executed. Packages can be digitally signed during the design process using the SSIS Designer. Once a package has been digitally signed, that package is read-only and can no longer be modified.

SSIS Package Logging

Logging SSIS package events provides a useful view into the tasks that are run as the package is executed. SSIS package logging creates a record that traces the execution of tasks and containers within an SSIS package.

A package author can choose on a per package and per control-flow object basis, which objects should provide log entries as well as control the details the object will post. The logging data can be sent to one or more of the several 'Log Providers' which a package author can configure per package. For example, the package can send log entries to both a text file and the Windows Event log.

The log entries can also be viewed and copied to the Windows Clipboard from the Business intelligence Studio by opening the Log Events window from the **SSIS** menu.

Figure 12 presents an overview of SSIS package logging where the logs are stored on a supporting server for central management.

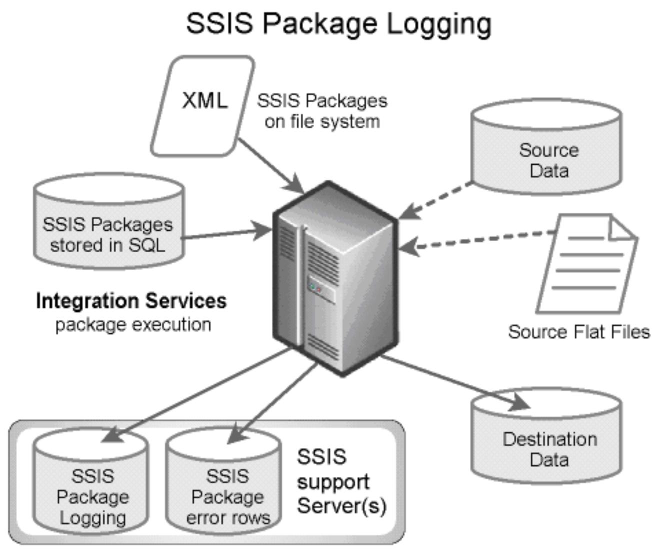


Figure 12: SSIS package logging

Table 3 lists the log providers that are included in SSIS.

Log Provider	Description
The SQL Profiler log provider	Writes log files that you can view using SQL Profiler. The default file extension is .trc.
The SQL Server log provider	Writes log entries to the sysdtslog90 table in a SQL Server 2005 database.
The Windows Event log provider	Writes entries to the application log in the Windows Event log on the local computer.
The XML File log provider	Writes log files to an XML file. The default file extension is .xml.
The Text File log provider	Writes log entries to ASCII text files using a comma-separated value (CSV) format. The default file extension is .log.

Table 3: Log providers

The SQL Profiler Log Provider

The SQL Profiler log provider enables you to create logs that can be opened using the SQL Profiler GUI. Using the SQL Profiler to combine log data from the system's performance monitors gives you a powerful analysis tool for analyzing the effect of package execution on system performance and troubleshooting problem situations such as unexpectedly long package or task run times.

The SQL Server Log Provider

Using the SQL Server log provider enables you to capture SSIS package run-time data in the sysdtslog90 table of the database. You can later query the data using SQL statements. Writing the SSIS log data to SQL Server also enables you to centrally store all of the log data generated by multiple SSIS packages running from one or more systems. Using the SQL Server log provider is a good choice if you want to easily report and monitor the operations of multiple SSIS packages.

The Windows Event Log Provider

The Windows Event log provider is a good option for logging if you are using operations management software such as Microsoft Operations Manager (MOM) to monitor your servers. Using MOM and the SQL Server Management Pack you can set up alerts and execute other actions based on log entries from your packages.

The XML File Log Provider

The XML File log provider is a good logging choice if you want the ability to browse the log visually. You can write XSLT transformations that format the log data to display as a Web page. XML is also an advantageous format to use if you need to share package results with outside individuals, or if you want to consolidate log files from a variety of data sources, including SQL Server.

The Text File Log Provider

Using the Text File log provider enables you to create logs that are easy to work with. Text files are easy to view and they are easily transportable, which makes them especially useful during the basic testing phase of a package.

Custom Log Provider

You can create your own custom log provider to obtain information to accommodate your business needs. Using a custom log provider allows you to integrate SSIS logs within your applications.

To facilitate operations management, SSIS packages always write some basic logging information to the Windows event log, even for packages that do not incorporate logging. SSIS packages write events for package initiation and package completion and they can be identified using either the SQLISPackage or SQLISService event sources.

SQL Server DTS Backward Compatibility

If you are currently using SQL Server 2000 DTS packages in your business, you will likely want to continue to use many of your present package functions when you upgrade to SSIS. SSIS offers several options for running or migrating your DTS packages, including using the Migration Wizard to migrate simple packages, manually performing package migration, or running your existing packages using the Execute Package task.

When you opt to keep your current DTS packages as is, at least for the short-term, your existing DTS packages are run from a SSIS package that calls the Execute SQL Server 2000 DTS Package task. On a server where SQL Server 2000 Enterprise Manager was previously installed, you may edit your DTS package in the SQL Server 2005 SSIS editor. When you select the **Edit Package** button in the task editor, the SQL Server 2000 DTS editor for the package is loaded. This approach allows you to upgrade to SQL Server 2005, while still running your original DTS packages. You may then gradually migrate or rewrite your DTS packages to SSIS packages to take advantage of the new features of SSIS.

Migrating SQL Server 2000 DTS Packages

Although SSIS is able to continue running your DTS packages, you will benefit by updating your packages to include the new abilities of SSIS packages. Because SSIS is all new in its design and object model, some DTS packages migrate easily using the Migration Wizard. Other DTS packages will receive the full benefits of SSIS only if they are manually migrated or entirely rewritten.

Using the Migration Wizard

The Migration Wizard is best suited for simple packages. DTS packages that contain tasks that have a one-for-one correlation to SSIS tasks typically migrate straight to SSIS packages without requiring any other effort. These tasks are listed in Table 4.

Task	Description
Execute SQL task	Runs SQL statements or stored procedures from a package
Bulk Insert task	Copies large amounts of data into a table or view
File Transfer Protocol task	Downloads/uploads files and manages directories
Execute Process task	Runs an application or batch file
Send Mail task	Sends an e-mail message
Copy SQL Server Objects task	Copy a table between two servers
Execute Package task	Launch another package

Table 4: SSIS simple tasks

In addition to these seven DTS tasks that have a direct migration path to SSIS, there are three additional tasks that will migrate to an updated SSIS package but may not work as expected after migration. These tasks are the ActiveX Script task, Dynamic Properties task, and Analysis Services DTS Processing task. These tasks generally interact with the SQL Server 2000 DTS object model and as the SSIS object model is newly designed, you will probably need new development of these task components.

Several other tasks such as Custom tasks, Copy Database Wizard tasks, Data Driven Query tasks, Data Pump tasks, Parallel Data Pump tasks, and Transform Data tasks, can not be migrated using the Migration Wizard. You need to rewrite them using native SSIS components or you can execute the existing DTS package using the Execute DTS 2000 Package task.

You can start the Migration Wizard in multiple ways. In the SQL Server Management Studio, right-click a package listed under the **DTS 2000 Package** node and click **Migrate** on the shortcut menu. Alternatively, in the BI Development Studio, you select the **DTS Packages** folder in the Solution Explorer pane and click the **Migrate DTS 2000** Package option on the menu.

The Migration Wizard will guide you through the steps required to migrate your DTS package. You will be able to select your original DTS package, designate a location to store the migrated package, assign a log file to capture the particulars of the wizard changes during the migration, and finish by reviewing the migration summary. It's important to note that the Migration Wizard creates a log file that records the migration results. When the migration completes, you can review this log file for any warnings or errors that occurred during the migration process. After the migration finishes, close the Migration Wizard and open the migrated SSIS package in the BI Development Studio editor to correct any errors or add elements to your new package.

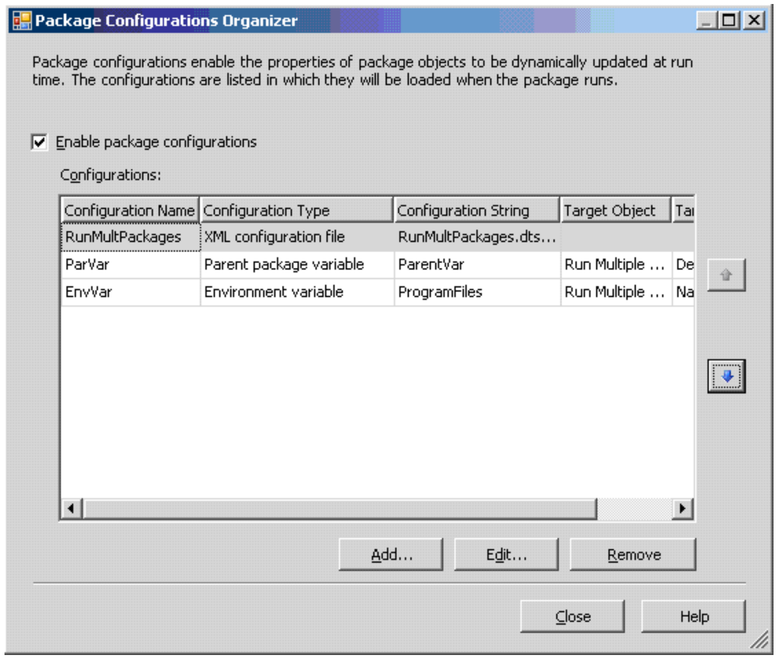
Creating Package Configurations

Package configurations allow you to supply data to your SSIS packages at run time. For instance, a common use of configuration is to enable the server name and user login information to be dynamically applied at run time. When an SSIS package is setup to use configurations, it will automatically load the configuration information at run time. Two of the most common uses for configurations are to pass in variables and connection information to the package at run time. In the case of variables, the **Value** property is assigned with the value that will be used at run time. Similarly, to dynamically set the connection properties of an OLE DB connection, use a separate configuration to assign a value to each of the Connection Manager's properties: ConnectionSting, ServerName, and InitialCatalog. When the package is used at run time, each of the configurations' values will be used to build the connection string.

SSIS package configurations are created by using the Package Organizer that's started from the BI Development Studio.

To create a package configuration:

1.Select the **DTS Package Configurations** option in the BI Development Studio. This will start the Package Configurations Organizer tool shown in Figure 13.



Package configurations enable the properties of package objects to be dynamically updated at run time. The configurations are listed in which they will be loaded when the package runs.

☑ Enable package configurations

Configurations:

Configuration Name	Configuration Type	Configuration String	Target Object	Ta
RunMultPackages	XML configuration file	RunMultPackages.dts...		
ParVar	Parent package variable	ParentVar	Run Multiple ...	De
EnvVar	Environment variable	ProgramFiles	Run Multiple ...	Na

Figure 13: SSIS Package Configurations Organizer

2.When the Package Configurations Organizer is displayed, click the **Enable package configurations** link.

3.Click the **Add** button to start the SSIS Configuration Wizard.

4.The Configuration Wizard steps you through creating a package configuration. The first step is to decide on the configuration type. Configuration data can be loaded into your packages from the following locations:

- The registry
- Environment variables
- A parent package
- XML files
- SQL Server

You can create multiple configurations for a single package. Each configuration is applied to the package in the order that they are shown in the Package Organizer. Use the directional buttons to move a configuration up or down in the list. For example, in corporate and department scenarios, the corporation might have a configuration that applies to all packages. The corporation's configuration file would be applied at the top level and then each department's unique configuration file would be applied afterward.

You can also create a single configuration object that you apply to multiple packages. For example, if you need to move multiple packages to several systems where the only difference in the package properties is the server name, you can create one configuration object that has the type value of environment variable and include this configuration with each of the packages. When you choose an XML configuration type, an XML configuration file is created with the extension of .dtsConfig.

5.Next, you select the package's properties or variables that will have their values set by the configuration when the package is run. XML file configurations and SQL Server configurations support the selection of multiple properties in a single configuration object. The other configuration types allows for one configurable property per configuration.

6.After the wizard screens are completed, the new configuration is added to the configuration list in the Package Configuration Organizer dialog box. If you want to modify a configuration, click **Edit** to rerun the Configuration Wizard and select different objects and different properties.

Top of page

Deploying SSIS Packages

Moving DTS packages between servers was one of the major challenges with SQL Server 2000 DTS. To move your packages either from your development environment to your production systems, or to take a package from a one production system and move it to another, typically required manually editing the packages to make sure that all of the connection objects were pointed to the right server and databases. Needless to say, manually configuring these packages was a time consuming and potentially error-prone endeavor. Because SQL Server Integration Services can create package configuration objects, the old manual configuration of packages is now obsolete. Package configurations allow you to dynamically change variables and object properties at run time, making package deployment flexible and much easier to manage.

You can deploy SSIS packages in one of four ways:

- Use the Deployment Utility in BI Development Studio.
- Use the import and export package features in SQL Server Management Studio.
- Save a copy of the package in the file system.
- Run the **dtutil** command line utility.

The Deployment Utility allows you to deploy SSIS packages and package dependencies such as package configurations and the DTSDeploymentManifest.xml file. After building the deployment utility for your project, you run the SSIS Package Installer Wizard to install the packages to the file system or to an instance of SQL Server 2005. You can see an overview of the deployment process in Figure 14.

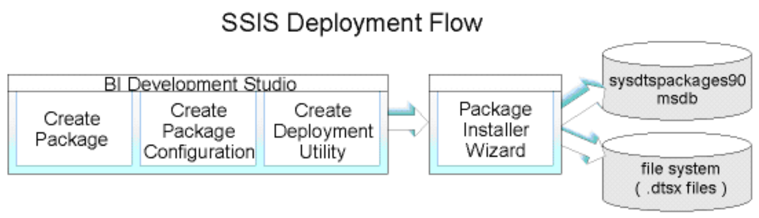
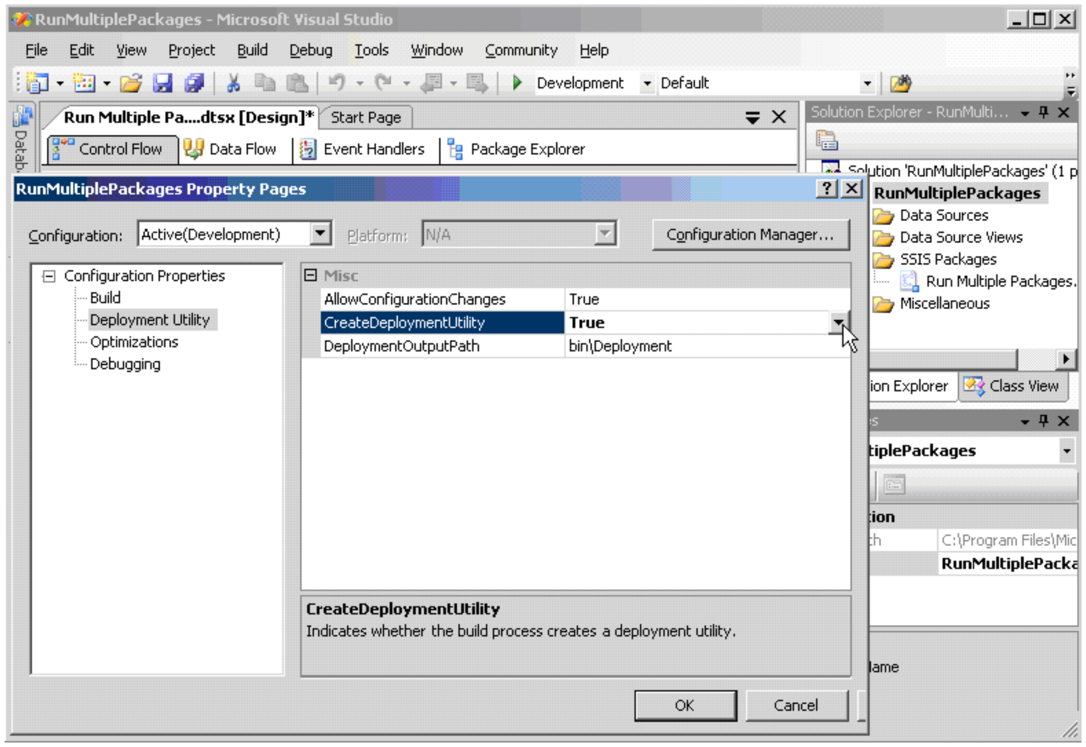


Figure 14: SSIS deployment flow with the Deployment Utility

Create the Package Deployment Utility

SSIS contains a handy feature called the Package Deployment Utility that allows you to assemble your SSIS packages, package configurations, and supporting files into a deployment folder and build an executable setup file to install your packages.

After you've designed your package in the BI Development Studio and added any configurations to your project, right-click the project properties in the Solution Explorer pane. Then select the **Properties** option to display the Property Pages dialog box as shown in Figure 15.



Configuration: Active(Development)   Platform: N/A

AllowConfigurationChanges	True
CreateDeploymentUtility	**True**
DeploymentOutputPath	bin\Deployment

CreateDeploymentUtility
Indicates whether the build process creates a deployment utility.

Figure 15: Package Deployment Utility Property

In the Property Pages dialog box, select the **Deployment Utility** option and set the property values. Table 5 lists the Deployment Utility properties.

Property	Description
AllowConfigurationChange	A value that specifies whether configurations can be updated during deployment. The default value of this property is True.
CreateDeploymentUtility	A value that specifies whether a package deployment utility is created when the project is built. The default value of this property is False. The property must be True to create a deployment utility.
DeploymentOutputPath	The location, relative to the SSIS project, of the files the project deployment uses.

Table 5: Package Deployment Properties

To create a Deployment Utility, set the CreateDeploymentUtility option to True on the project property page. Then build your project by selecting the Build Solution option on the BI Development Studio menu. Building the project creates the file, DTSDeploymentManifest.xml, and copies the project packages, along with the DTSInstall.exe, to the bin/Deployment folder, or to the location specified in the DeploymentOutputPath property. The DTSDeploymentManifest.xml file lists the packages and the package configurations in the project. DTSInstall.exe is the application that runs the Package Installer Wizard.

Using the Package Installer Wizard

To deploy the SSIS project, build a package deployment utility then run the package installation executable program, DTSInstall.exe. This DTSIntall.exe program is automatically copied to your deployment folder.

To use the Package Installer Wizard:

- To start the Package Installer Wizard that will guide you through the installation process, run the **DTSInstall.exe** program.

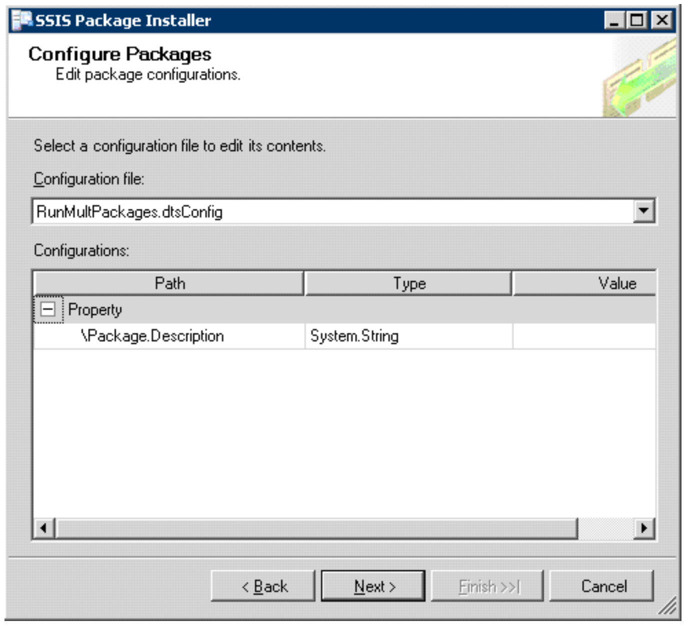
  Select a deployment type: **file system deployment** or **SQL Server deployment**.

  The **file system deployment** option installs your packages in the file system as .dtsx files. You can later open and edit .dtsx files using the SSIS Designer. If you've used custom folders to store your SSIS file system packages then you will want to select the appropriate target folder.

  The **SQL Server deployment** option installs your packages in the sysdtspackages90 table in SQL Server 2005's msdb database. This option also copies any of the package's dependency files such as an XML Configuration file, to a folder on the file system that you designate.

  Using either of these deployment options will register the package and display it in the Stored Packages folder of SQL Server Management Studio.

- Next, the wizard prompts you for an installation folder location for file system installs, or target SQL Server for SQL Server deployments. SQL Server deployments will also prompt you for the file system folder to copy the packages and dependency files. For packages that contain configurations, you may optionally edit the updateable configuration values. The Package Installer Wizard will display a prompt similar to the one shown Figure 16.


SSIS Package Installer
Configure Packages
Edit package configurations.
Select a configuration file to edit its contents.
Configuration file:
RunMultPackages.dtsConfig
Configurations:
< Back
Next >
Finish >>|
Cancel

Path	Type	Value
Property		
\Package.Description	System.String	

Figure 16: Package Installer Wizard

- Running the Package Installer Wizard will prompt you to specify the package's **ProtectionLevel** property. For more information on the package's **ProtectionLevel** property, see Securing SSIS Packages in this paper.

Manual Package Deployment

In addition to using the tools that are built into the BI Development Studio for package deployment, you can also deploy packages manually. The initial steps of package creation are the same for both types of deployment. You would typically make the package using the SSIS Designer in the BI Development Studio, which results in the creation of a .dtsx package file.

Once the package file has been built, you can copy the package file to the destination system manually or you can incorporate a file copy routine as a part of a custom deployment script. If you manually deploy SSIS packages, then you need to explicitly include any required supporting files in your deployment script. If you want the package to be installed to SQL Server or you want the package in the file system to be visible to the SSIS service, then you'll want to include the **dtutil** tool as a part of your deployment processes. **Dtutil** can copy the package to either the file system or to the SQL Server msdb database. Once the package has been moved to its destination, you can execute the package using the **dtexec** or **dtexecui** utilities.

SSIS packages can also be run on systems where SQL Server is not installed. However, the .NET Framework and the SSIS runtime must be installed in order to execute the packages on systems that do not have SQL Server 2005 installed.

Top of page

Summary

New in SQL Server 2005, SQL Server Integration Services is an enterprise-level data integration platform. SSIS contains a workflow development engine that addresses a variety of different tasks from data transfer to database maintenance operations. The new management and deployment features in SSIS are major improvements over the same features of its predecessor, Data Transformation Services. SSIS enables you to create configurations for packages that allow the packages to flexibly adapt to different run-time conditions. The SSIS service enables you to manage package storage as well as monitor package execution. The collection of tools included with SQL Server 2005 enables you to perform package management and deployment both from the integrated development environment as well as from the command line.

Top of page

Additional Resources

You can lean more about SQL Server 2005's Integration Services at:

http://msdn.microsoft.com/SQL/sqlwarehouse/SSIS/default.aspx

Michael Otey is Technical Director for Windows IT Pro Magazine and Senior Technical Editor for SQL Server Magazine. He is also President of TECA, Inc., a software development and consulting company that specializes in interoperability and database applications. Michael has worked with various version of DB2 since 1983. Michael is the author of the SQL Server 2005 New Features Guide published by Osborne McGraw Hill.

Denielle Otey is the Vice President of TECA, Inc., as well as a software consultant who has extensive experience designing, implementing, testing, and debugging software in C, VC++, VB, and Visual Studio .NET. She also is the developer of several SQL Server utilities, she has developed and delivered DB2 applications through TECA, Inc,, and she is co-author of ADO.NET The Complete Reference published by Osborne McGraw Hill.

This article was developed in partnership with A23 Consulting.

Top of page