1

Welcome to SQL Server Integration Services

SQL Server Integration Services (SSIS) is the anchor in a trilogy of products that make up the Microsoft SQL Server Business Intelligence (BI) platform. SSIS along with Analysis Services and Reporting Services round out a platform that clearly puts Microsoft on the map in the enterprise Business Intelligence arena. In its simplest form, SSIS is an enterprise-level extract, transform, and load (ETL) development tool. However, SSIS is not just a fancy wrapper around an import wizard. In a drag-and-drop development environment, ETL developers can snap together intricate workflows and out-of-the-box data-cleansing flows that rival custom coding and expensive third-party tools. For your edge cases, the model is easily extensible and custom components can be developed in .NET languages to simply snap into the framework. However, custom coding most likely will not even be necessary. With the latest version of SSIS, novice developers can use the embedded Visual Studio Tools for Applications (VSTA) development environment to custom code workflow tasks and data pipeline transformations in VB or C# .NET languages.

When we put together the first edition of this book, we were blown away by the new architecture and capabilities of SSIS. SSIS was a big change from the Data Transformation Services (DTS) product that it replaced and there has been much to learn. Since the first edition of SSIS, we have collectively racked up many years of experience converting older DTS packages and mindsets over to using SSIS, and trust us when we say that no one who's made the change is asking to go back. We've learned some things, too. If you run into an issue getting up and running, converting older packages, or creating new ones, we've probably run into that issue too and have a solution for you here in this book. This book is a new edition and a whole new book. Nothing was sacred in this rewrite because we really dug in to put the last few years of experience working with this product back into these pages. We think the result is worth it and this edition will make your experience with SSIS a more productive one. This chapter starts from the beginning and provides an overview of SSIS, describes where it fits within the BI product platform, and ETL development in general.

SQL Server SSIS Historical Overview

In SQL Server 7.0, Microsoft had a small team of developers work on a much understated feature of SQL Server called Data Transformation Services (DTS). DTS was the backbone of the Import/Export Wizard, and the DTS's primary purpose was to transform data from almost any OLE DB–compliant Data Source to another destination. It also had the ability to execute programs and run scripts, making workflow a minor feature.

By the time that SQL Server 2000 was released, DTS had a strong following of DBAs and maybe a few developers. Microsoft included in the release new features like the Dynamic Properties Task that enabled you to alter the package dynamically at runtime. Even though DTS utilized extensive logging along with simple and complex multiphase data pumps, usability studies still showed that developers had to create elaborate scripts to extend DTS to get what they wanted done. A typical use case was enabling DTS to load data conditionally based on the existence of a file. To accomplish this in DTS, you would have had to use the ActiveX Script Task to code a solution using the file system object in VBScript. The problem here was that DTS simply lacked some of the common components to support typical ETL processes. Although it was powerful if you knew how to write scripting code, most DBAs just didn't have this type of scripting experience (or time).

After five years, Microsoft released the much touted SQL Server 2005, and SSIS, which is no longer an understated feature like DTS. With the 2008 release, SSIS is now one of the main business intelligence (BI) platform foundations. SSIS has moved so far up in importance that it now has its own service along with the new name. This is entirely appropriate because so much has been added to SSIS. Microsoft made a huge investment in usability, adding the first set of ETL tool-based components and upping the ante again with this latest version. If what you need to do isn't readily available, you now have the full .NET library with VB and C# programming languages to add your own custom coding to message data or manage the ETL process. However, you'll be surprised how rarely you'll need to drop into a coding environment. In fact, as you dig into the toolset, you'll find that things you may have needed to hand-code in a Script Task are simply part of the out-of-the-box components.

What's New in SSIS

SSIS is now in its second edition. If you are brand new to SSIS, everything will be new, but even if you are already using SSIS each version just keeps getting better. This latest version of SSIS includes enhancements for performance and scalability, upgrades to handle new TSQL capabilities, and the addition of new components, including the long-awaited ability to use C# in the scripting tasks. We'll hit the highlights here.

The data pipeline has been overhauled so that it scales to better use the multiple, dual, and quad-core processor improvements. The Lookup Component that performs foreign key resolutions has also been redesigned to allow for persistence of lookup cache that screams when you tune them for dimension tables. Underneath SSIS now allows new TSQL extensions for multiple data manipulation language (DML) operations like the `MERGE` statement.

If you are looking for the latest toys, this version of SSIS has added new workflow components to control the cache window maintenance, to generate TSQL traces, or reset row count variables. In the Data Flows, there are new ADO Sources and Destinations to add to the OLE Sources and Destinations that were part of the first version.

Lastly, there has been a major improvement to the development environment from the previous versions with the removal of the cobbled-together Visual Basic for Applications (VBA) implementation. The VBA environment was only intended as a temporary implementation to allow custom scripting within your ETL processes, evidenced by the clunky integration and that you were limited to VB.NET only. Now the Script Tasks and Components use an embedded version of the Microsoft Visual Studio 2008 Tools for Applications (VSTA) environment that supports both VB and C# .NET programming languages. In addition, you can now add web references to your ETL processes without having to code your own wrappers to web services to make use of existing business logic or data sources. We'll touch on all of these improvements as you read through this book and explore the examples, but first let's get started.

Getting Started

Most of this book will assume that you know nothing about the past releases of SQL Server DTS and will start with a fresh look at SQL Server SSIS. After all, when you dive into the new features, you'll realize how little knowing anything about the old release actually helps you when learning this one. However, if you don't come from the world of DTS, it's hard for us not to throw in a few analogies here and there to get these folks also up to speed on SSIS. The learning curve can be considered steep at first, but once you figure out the basics, you'll be creating what would have been complex packages in DTS in no time. To get an idea of how easy SSIS is to use, look at a tool that is a staple in the ETL world, the Import and Export Wizard.

Import and Export Wizard

If you need to move data quickly from almost any OLE DB–compliant Data Source to a destination, you can use the SSIS Import and Export Wizard (shown in Figure 1-1). In fact, many SSIS packages are born this way. The wizard is a quick way to move the data, perform very light transformations of data, and all versions except Express allow you to persist the logic of the data movement into a package. The basic concept of an import/export wizard has not changed substantially from the days of DTS. You still have the option of checking all the tables you'd like to transfer. However, you also get the option now of encapsulating the entire transfer of data into a single transaction.

Where do you find the wizard? It depends. If you just need to perform a quick import or export, access the wizard directly from the Start menu by navigating to Start ⇨ Microsoft SQL Server 2008 ⇨ Import and Export Data. The other option is to open up a project in the SSIS development environment and select the menu option Project ⇨ SSIS Import and Export Wizard. We cover this in detail in Chapter 2. Before we get into all the mechanics for that, Figure 1-1 shows an example of the wizard fully loaded and ready to move some data.

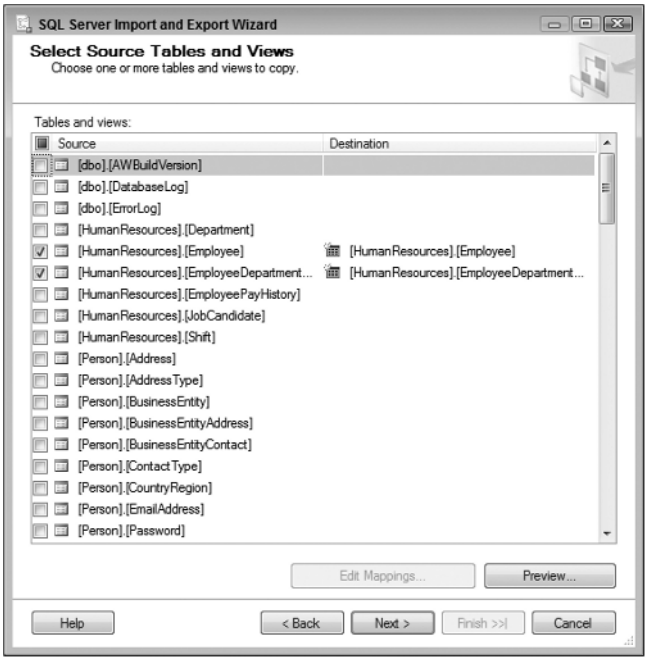

Figure 1-1

The Business Intelligence Development Studio

The Business Intelligence Development Studio (BIDS) is the central environment that you'll spend most of your time in as an SSIS developer. BIDS is just a specialized use of the familiar Visual Studio development environment that can host many different project types from Console applications to Class Libraries and Windows applications. Although you may see many project types, BIDS actually only contains project templates for Analysis Services, Integration Services, Report Server, and Report Model Projects. SSIS in particular uses a business-intelligence project type called an Integration Services project that provides development design surfaces with a completely ETL-based set of tools in the Toolbox window. Get your first look at the development environment in Figure 1-2.

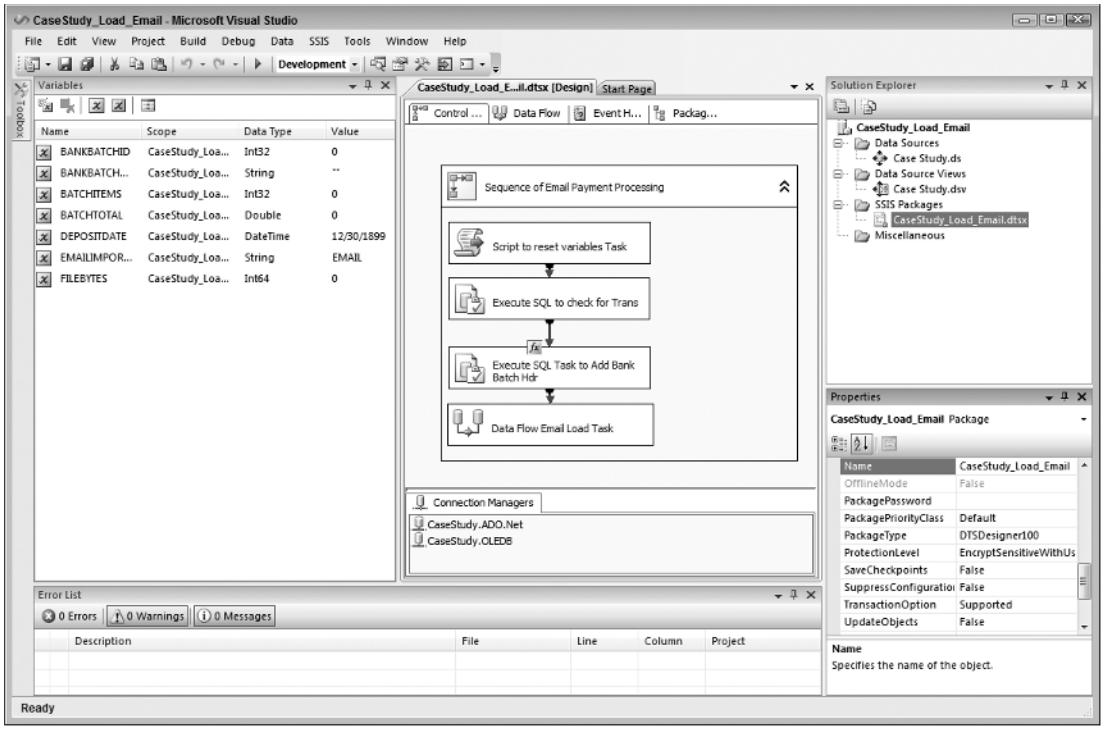

Figure 1-2

Though this development environment is similar to the legacy DTS Designer, the approach is completely different. Most importantly, this is a collaborative development environment just like any Visual Studio development effort with full source code management, version control, and multi-user project management. In fact, SSIS was not developed in the context of a SQL Server instance like the DTS Designer, and you don't get to the SSIS IDE from within a particular SQL Server instance. SSIS solutions are developed just like all other .NET development solutions, including being persisted to files — in this case, XML file-based structures. You can even develop within the BIDS environment without a connection to a SQL Server instance using the off-line mode. Once your solution is complete, it can be built and deployed to one or multiple target SQL servers. These changes are crucial to establishing the discipline and best practices of existing .NET development methodologies as you develop business intelligence solutions. We'll discuss this BIDS development interface in more detail later.

Architecture

Microsoft has truly established SSIS in SQL Server as a major player in the extraction, transformation, and loading (ETL) market. Not only is SSIS technology a complete code rewrite from SQL Server 2000 DTS, but now it rivals other third-party ETL tools costing hundreds or thousands of dollars based on how you scale the software — and it comes free with the purchase of SQL Server 2005 and later versions. Free is great, but it can take you only so far if the feature set is minimal, or if the toolset has limited usability, scalability, or enterprise performance limitations. But SSIS is for real, satisfying these typical

ETL requirements with a new architecture that has expanded dramatically, as you can see in Figure 1-3. The SSIS architecture consists of four main components:

- The SSIS Service
- The SSIS runtime engine and the runtime executables
- The SSIS Data Flow engine and the Data Flow Components
- The SSIS clients

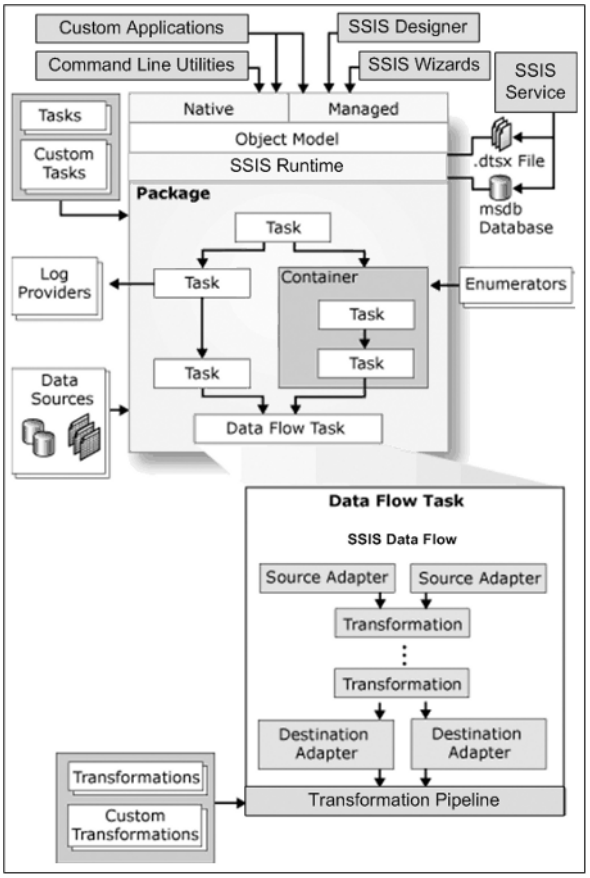

Figure 1-3

The SSIS Service handles the operational aspects of SSIS. It is a Windows service installed when you install the SSIS Component of SQL Server. It tracks the execution of packages (a collection of work items) and helps with the storage of the packages. Don't worry; we'll get to explaining what packages are shortly. The SSIS Service is turned on by default and is set to automatic. You don't need the SSIS service to be on to run SSIS packages, but if the service is stopped, the default behavior is for all the SSIS packages that are currently running to also stop. (You can configure this service differently in the service configuration if this is not the behavior you require.)

The SSIS runtime engine and its complementary programs actually run your SSIS packages. The engine saves the layout of your packages and manages the logging, debugging, configuration, connections, and transactions. Additionally, it manages handling your events when they are raised within your package. The runtime executables provide the following functionality, which you'll explore in more detail later in this chapter:

- **Containers:** Provide structure and scope to your package.
- **Tasks:** Provide the functionality to your package.
- **Event handlers:** Respond to raised events in your package.
- **Precedence constraints:** Provide ordinal relationship between various items in your package.

In Chapter 3, you'll spend a lot of time in each of these architectural sections, but the next few sections provide a nice overview for the ones that are the most important.

Packages

A core component of SSIS and DTS is the notion of a *package*. A package best parallels an executable program that maintains workflow and business logic. Essentially, a package is a collection of tasks snapped together to execute in an orderly fashion. Precedence constraints are used to connect the tasks together and manage the order in which tasks will execute, based on what happens in each task or specialized rules. The package is compiled into a .DTSX file that is actually an XML-structured file with collections of properties. Just like other .NET solutions, the file-based code is marked up using the development environment and can then be saved and compiled for deployment to a SQL Server as a file in the file system or can be saved into the msdb database metadata. The package XML structure stripped down to the basic elements looks like this:

```
<?xml version="1.0"?>
<DTS:Executable xmlns:DTS="www.microsoft.com/SqlServer/Dts"
DTS:ExecutableType="MSDTS.Package.2">
<DTS:ConnectionManager></DTS:ConnectionManager>
<DTS:PackageVariable></DTS:PackageVariable>
<DTS:Executable></DTS:Executable>
<DTS:Executable DTS:ExecutableType="DTS.Pipeline.2">
   <components>
      <component></component>
   </components>
</DTS:Executable>
<DTS:EventHandler></DTS:EventHandler>
<DTS:PrecedenceConstraint></DTS:PrecedenceConstraint>
<DTS:Executable>
```

Here you can see the package collections of connections, package variables, executables, and precedence constraints. The specific executable named `DTS.Pipeline.2` is a special task type that allows for transformation of a data stream or pipeline that we'll discover later. The point here is that the SSIS package is an XML-structured file much like .RDL files are to Reporting Services. Of course, there is much more to packages than that, but you'll explore the other elements of packages, like event handlers, later in this chapter.

Tasks

A *task* can best be described as an individual unit of work. In the previous XML package snippet these are the `<DTS:Executable>` nodes. Tasks provide functionality to your package, in much the same way that a method does in a programming language. However, in SSIS, you aren't coding the methods; rather, you are dragging and dropping them onto a design surface and configuring them. You can also develop your own tasks, but here are the current ETL Tasks available to you out-of-the-box:

- **ActiveX Script Task:** Executes an ActiveX script in your SSIS package. This task is only to facilitate conversion of legacy DTS packages that use this deprecated scripting method.
- **Analysis Services Execute DDL Task:** Executes a DDL Task in Analysis Services. For example, this can create, drop, or alter a cube (Enterprise and Developer Editions only).
- **Analysis Services Processing Task:** This task processes a SQL Server Analysis Services cube, dimension, or mining model.
- **Bulk Insert Task:** Loads data into a table by using the `BULK INSERT SQL` command.
- **Data Flow Task:** This very specialized task loads and transforms data into an OLE DB, and now, optionally, an ADO.NET Destination.
- **Data Mining Query Task:** Allows you to run predictive queries against your Analysis Services data-mining models.
- **Data Profiling Task:** This exciting new task allows for the examination of data to replace your ad-hoc data profiling techniques.
- **Execute DTS 2000 Package Task:** Exposes legacy SQL Server 2000 DTS packages to your SSIS package.
- **Execute Package Task:** Allows you to execute a package from within a package, making your SSIS packages modular.
- **Execute Process Task:** Executes a program external to your package, such as one to split your extract file into many files before processing the individual files.
- **Execute SQL Task:** Executes a SQL statement or stored procedure.
- **File System Task:** This task can handle directory operations such as creating, renaming, or deleting a directory. It can also manage file operations such as moving, copying, or deleting files.
- **FTP Task:** Sends or receives files from an FTP site.
- **Message Queue Task:** Sends or receives messages from a Microsoft Message Queue (MSMQ).
- **Script Task:** This task allows you to perform more .NET-based scripting in the Visual Studio Tools for Applications programming environment.
- **Send Mail Task:** Sends a mail message through SMTP.
- **Web Service Task:** Executes a method on a Web service.
- **WMI Data Reader Task:** This task can run WQL queries against the Windows Management Instrumentation. This allows you to read the event log, get a list of applications that are installed, or determine hardware that is installed, to name a few examples.

- **WMI Event Watcher Task:** This task empowers SSIS to wait for and respond to certain WMI events that occur in the operating system.
- **XML Task:** Parses or processes an XML file. It can merge, split, or reformat an XML file.

There is also a whole set of tasks that are DBA-oriented, allowing you to create packages that can be used to maintain your SQL Server environment. These tasks perform functions such as transferring your SQL Server databases, backing up your database, or shrinking the database. Each of the tasks available to you is described in Chapter 3 in much more detail, and you will see them in other examples throughout the book. Tasks are extensible, and you can create your own custom tasks in .NET if you need a workflow item that doesn't exist or if you have a common scripting function that can benefit from reuse in your package development. We cover this topic in Chapter 18, "Programming and Extending SSIS."

Data Source Elements

The core strength of SSIS is its capability to extract data, transform it, and write it out to an alternative destination. Data sources are the conduit for these data pipelines and are represented by connections that can be used by sources or destinations once they've been defined. A data source uses connections that are OLE DB–compliant and with SSIS 2008 this now includes ADO.NET Data Sources, such as SQL Server, Oracle, DB2, or even nontraditional Data Sources, such as Analysis Services and Outlook. The data sources can be localized to a single SSIS package or shared across multiple packages in BIDS.

All the characteristics of the connection are defined in the Connection Manager. The Connection Manager dialog box options vary based on the type of connection you're trying to configure. Figure 1-4 shows you what a typical connection to SQL Server would look like.

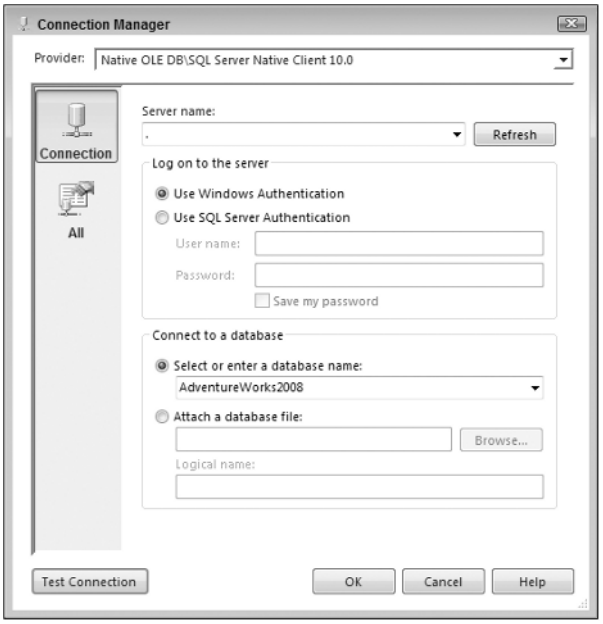

Figure 1-4

Connection Managers are used to centralize connection strings to data sources and abstract them from the SSIS packages themselves. In fact, the connections created by a Connection Manager are typically created in the registry of a machine and not stored in the package itself — although you can encrypt this information and store it. This allows you to deploy the SSIS package with a configuration file (which we'll describe later) that can set the full value of the connection properties at runtime. One nice thing is that you can even configure the connection offline and completely design an SSIS package without connecting to the server until you are ready to test. SSIS will not use the connection until you begin to instantiate it in the package. This provides the ultimate in lightweight development portability for SSIS.

Data Source Views

Data source views (DSVs) are handy abstract concepts that you can use in SSIS and other SQL Server projects. This feature allows you to create logical views of your business data. These views are a combination of tables, views, stored procedures, and queries that can be shared across your project and leveraged in Analysis Services and Report Builder projects.

This is especially useful in large complex data models that are prevalent in ERP systems like Siebel or SAP. These systems have column names like ER328F2 to make the data model flexible for nearly any environment. However, this complex naming convention creates difficulties for the typical business user or developer, impeding productivity that a simple readable name would eliminate. A DSV can be used to map such columns to entities like LastPaymentDate to increase readability of the model. DSVs can also be used to map the relationships between the tables that don't necessarily exist in the physical model.

Another common use of DSVs is to segment large sections of the data model into more security- or functional-based chunks. DSVs provide an abstraction that transcends schema or even data source separation. Take, for example, a DSV from the AdventureWorks Human Resource model as shown in Figure 1-5. As you can see in this figure, not only has the DSV unified the different schemas, but a friendly name has also been assigned to rename the Birth Date column in the Employee entity to DOB.

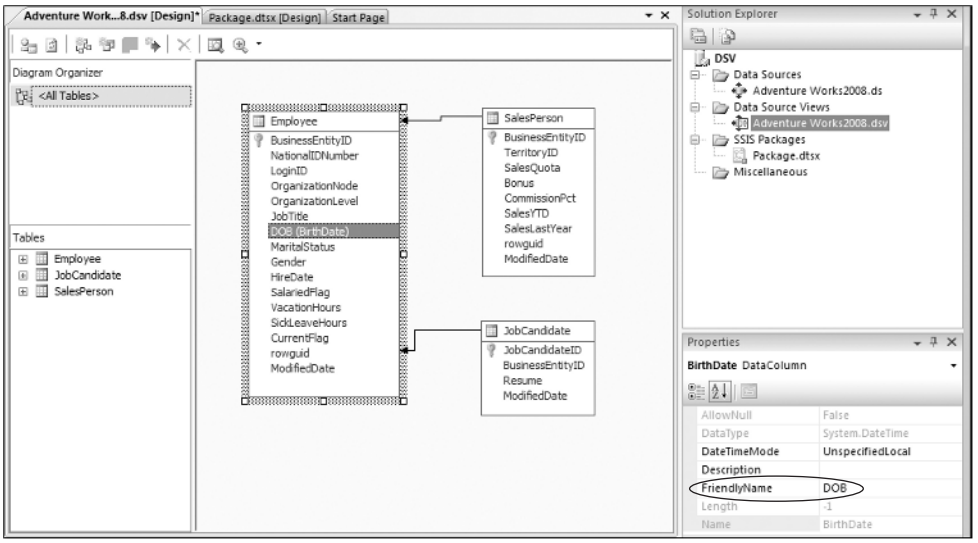

Figure 1-5

DSVs are used just like Connection Managers. However, there are a few key differences to remember when using them. Like data sources, DSVs allow you to define the connection logic once and reuse it across your SSIS packages. However, unlike connections, DSV structures are stored once, and then are disconnected from the real source. This means that the underlying structure of the DSVs may change, but the DSVs are not automatically refreshed. This can cause some problems during execution; if you were to change the Employee table in a connection to a DSV for Human Resources, the DSV would not pick up the change. On the other hand, if your model does not change often, this type of caching is a huge benefit in your development of SSIS packages. The DSVs provide the disconnected capabilities to allow development of SSIS packages to continue against cached metadata. DSVs also provide a side advantage in speeding up package development. Because DSVs are most likely defined as a subset of the actual Data Source, your SSIS connection dialog boxes will connect, realize data, and subsequently load much faster.

Precedence Constraints

Precedence constraints direct the tasks to execute in a given order. In fact, precedence constraints are the connectors that not only link tasks together but define the workflow of your SSIS package. Constraints control the execution of the two linked tasks by executing the destination task based upon the final state of the prior task and business rules that are defined using special expressions. The expression language embedded in SSIS essentially replaces the need to control workflow using script-based methodologies that enabled and disabled tasks, as was used in the DTS legacy solution. With expressions, you can direct the workflow of your SSIS package based on all manner of given conditions. We'll go into many examples of using these constraints throughout this book.

To set up a precedence constraint between two tasks, the constraint value must be set, and optionally you can set an expression. The next sections give a quick overview of the differences between the two.

Constraint Value

Constraint values define how the package will react when the prior task of two linked tasks completes an execution. The choices define whether the destination task of two linked tasks should execute based solely on how the prior task completes. There are three types of constraint values:

- **Success:** A task that's chained to another task with this constraint will execute only if the prior task completes successfully. These precedence constraints are colored green.
- **Completion:** A task that's chained to another task with this constraint will execute if the prior task completes. Whether the prior task succeeds or fails is inconsequential. These precedence constraints are colored blue.
- **Failure:** A task that's chained to another task with this constraint will execute only if the prior task fails to complete. This type of constraint is usually used to notify an operator of a failed event or write bad records to an exception queue. These precedence constraints are colored red.

Conditional Expressions

The conditional expression options that you can apply to a precedence constraint allow you to mix in a dynamically realized expression with the constraint value to determine the package workflow between two or more linked tasks. An *expression* allows you to evaluate whether certain conditions have been met

before the task is executed and the path followed. The *constraint* evaluates only the success or failure of the previous task to determine whether the next step will be executed. The SSIS developer can set the conditions by using evaluation operators. Once you create a precedence constraint, you can set the Evaluation Option property to any one of the following options:

- **Constraint:** This is the default setting and specifies that only the constraint will be followed in the workflow.
- **Expression:** This option gives you the ability to write an expression (much like VB.NET) that allows you to control the workflow based on conditions that you specify.
- **ExpressionAndConstraint:** Specifies that both the expression and the constraint must be met before proceeding.
- **ExpressionOrConstraint:** Specifies that either the expression or the constraint can be met before proceeding.

In Figure 1-6, you can see an example that contains three tasks. In this example, the package first attempts the copying of files using the File System Task. If this prior task is successful and meets the expression criteria for a good file to transfer, the package will divert to the Data Flow Task to transform the files. However, if the first step fails, a message will be sent to the user using the Send Mail Task. You can also see in the graphic a small *fx* icon above the Data Flow Task and on the precedence constraint. This is the graphical representation for a conditional expression and visually informs that this task will not execute unless an expression has also been met. The expression can check anything, such as looking at a checksum, before running the Data Flow Task.

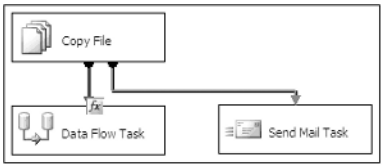

Figure 1-6

Containers

Containers are a core unit in the SSIS architecture to group tasks together logically into units of work. Besides providing visual consistency, containers allow you to define variables and event handlers (these are discussed in a moment) within the scope of the container instead of the package. There are four types of containers in SSIS:

- **Task Host Container:** The core type of container implements the basic interface to which every task implicitly belongs by default. The SSIS architecture extends variables and event handlers to the task through the Task Host Container.

The Task Host Container is not a visible element that you'll find in the Toolbox, but is an abstract concept like an interface.

- **Sequence Container:** Allows you to group tasks into logical subject areas. Within the development environment, you can then collapse or expand this container for usability. This is similar to the region concept in .NET coding.
- **For Loop Container:** Loops through a series of tasks for a given amount of time or using an iterator until a condition is met.
- **Foreach Loop Container:** Loops through a series of files or records in a data set, and then executes the tasks in the container for each record in the collection.

Containers are so integral to SSIS development that you'll find Chapter 4 is devoted to them. As you read through this book, we'll give you many real-world examples of using each of these types of containers for typical ETL development tasks.

Variables

Variables are another vital component of the SSIS architecture. In legacy DTS ETL development, global variables could be defined either by the Dynamic Property Task or by hand in the Active X Task, but they could only store static values. SSIS variables can be set to evaluate to an expression at runtime, removing much of the need to push values into the variables. However, you still can do this with the Scripting Tasks and Transforms, and as always, the configuration processes can set these variables. Variables in SSIS have become the method of exchange between many tasks and transforms, making the scoping of variables much more important. SSIS variables exist within a scope in the package. The default is to create variables at a package scope, but they can be scoped to different levels within a package as mentioned earlier in the "Containers" section.

Using variables allows you to configure a package dynamically at runtime. Without variables, each time you wanted to deploy a package from development to production, you'd have to open the package and change all the hard-coded connection settings to point to the new environment. A best practice is to set up SSIS packages using variables, so that you can just change the variables at deployment time, and anything that uses those variables will adjust.

Data Flow Elements

You learned earlier that the Data Flow Task is simply another executable task in the package. The Data Flow replaces the simple black arrow data pump that you may be familiar with from legacy DTS packages. If this is not familiar, this arrow describes what the Data Flow does, wonderfully. The Data Flow Task is the pumping mechanism that moves data from source to destination. However, in the case of SSIS, you have much more control of what happens from start to finish. In fact, you have a set of out-of-the box transformation components that you snap together to clean and manipulate the data while it is in the data pipeline.

One confusing thing for new SSIS developers is that once you drag and drop a Data Flow Task in the Control Flow, it spawns a new Data Flow design surface with its own new tab in the BIDS user interface. Each Data Flow Task has its own design surface that you can access by double-clicking the Data Flow Task or by clicking the Data Flow tab and selecting Data Flow Task from the drop-down list. Just as the Controller Flow handles the main workflow of the package, the Data Flow handles the transformation of data. Almost

anything that manipulates data falls into the Data Flow category. As data moves through each step of the Data Flow, the data changes, based on what the transform does. For example, in Figure 1-7, a new column is derived using the Derived Column Transform, and that new column is then available to subsequent transformations or to the destination.

In this section, each of the sources, destinations, and transformations are covered from an overview perspective. These areas are covered in much more detail in later chapters.

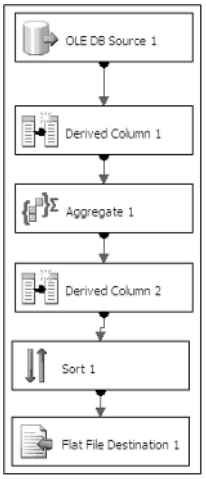

Figure 1-7

Sources

A *source* is a component that you add to the Data Flow design surface to specify the location of the source data that is to feed into the data pump. Sources are configured to use Connection Managers to allow for the ability to reuse connections throughout your package. Six sources can be used out-of-the-box with SSIS:

- **OLE DB Source:** Connects to nearly any OLE DB Data Source, such as SQL Server, Access, Oracle, or DB2, to name just a few.
- **Excel Source:** Specializes in receiving data from Excel spreadsheets. This source also makes it easy to run SQL queries against your Excel spreadsheet to narrow the scope of the data that you wish to pass through the flow.
- **Flat File Source:** Connects to a delimited or fixed-width file.
- **Raw File Source:** Produces a specialized binary file format for data that is in transit and is especially quick to read by SSIS.
- **Xml Source:** Can retrieve data from an XML document.
- **ADO.NET Source:** This source is just like the OLE DB Source but only for ADO.NET-based sources. The internal implementation uses an ADO.NET DataReader as the source. The ADO .NET connection is much like the one you see in the .NET Framework when handcoding a connection and retrieval from a database.

Sources can also be hand coded using two methods. One method is to use the Script Component to create a source stream using the existing .NET libraries. This method is more practical for single-use applications. If you need to reuse a custom source, you can develop one by extending the SSIS object model.

Destinations

Inside the Data Flow, destinations consume the data after the data pipe leaves the last transformation components. The flexible architecture can send the data to nearly any OLE DB–compliant, flat-file, or ADO.NET Data Source. Like sources, destinations are also managed through the Connection Manager. The following destinations are available to you in SSIS:

- **Data Mining Model Training:** Trains an Analysis Services mining model by passing in data from the Data Flow to the destination.
- **ADO.NET Destination:** Exposes data to other external processes, such as Reporting Services or your own .NET application. It also uses the ADO.NET DataReader interface similar to the ADO .NET Source to consume the data.
- **Data Reader Destination:** Allows the ADO.NET DataReader interface to consume data similar to the ADO.NET Destination.
- **Dimension Processing:** Loads and processes an Analysis Services dimension. It can perform a full, update, or incremental refresh of the dimension.
- **Excel Destination:** Outputs data from the Data Flow to an Excel spreadsheet.
- **Flat File Destination:** Enables you to write data to a comma-delimited or fixed-width file.
- **OLE DB Destination:** Outputs data to an OLE DB data connection like SQL Server, Oracle, or Access.
- **Partition Processing:** Enables you to perform incremental, full, or update processing of an Analysis Services partition.
- **Raw File Destination:** Outputs data in a binary format that can be used later as a Raw File Source. Is usually used as an intermediate persistence mechanism.
- **Recordset Destination:** Writes the records to an ADO record set.
- **SQL Server Destination:** The destination that you use to write data to SQL Server most efficiently.
- **SQL Server Compact Edition Destination:** Inserts data into a SQL Server running on a Pocket PC.

Transformations

Transformations are key components within the Data Flow that allow changes to the data in the data pipe. You can use transformations to split, divert, and remerge data in the data pipe. Data can also be validated, cleansed, and rejected using specific rules. For example, you may want your dimension data to be sorted and validated. This can be simply accomplished by dropping a Sort and a Lookup Transformation onto the Data Flow design surface and configuring them.

Transform Components in the SSIS Data Flow affect data in the data pipe in-memory. This is not always the panacea for ETL processing, especially under high-volume data processing. However, the latest version of SSIS has changed the way the Data Flow Task breaks down the execution tree for the transforms to take full advantage of asynchronous processing and parallelism to get the most from multi-processor machines. Here's a complete list of transforms:

- **Aggregate:** Aggregates data from transform or source.
- **Audit:** Exposes auditing information from the package to the data pipe, such as when the package was run and by whom.
- **Character Map:** Makes common string data changes for you, such as changing data from lowercase to uppercase.
- **Conditional Split:** Splits the data based on certain conditions being met. For example, this transformation could be instructed to send data down a different path if the State column is equal to Florida.
- **Copy Column:** Adds a copy of a column to the transformation output. You can later transform the copy, keeping the original for auditing purposes.
- **Data Conversion:** Converts a column's data type to another data type.
- **Data Mining Query:** Performs a data-mining query against Analysis Services.
- **Derived Column:** Creates a new derived column calculated from an expression.
- **Export Column:** Exports a column from the Data Flow to the file system. For example, you can use this transformation to write a column that contains an image to a file.
- **Fuzzy Grouping:** Performs data cleansing by finding rows that are likely duplicates.
- **Fuzzy Lookup:** Matches and standardizes data based on fuzzy logic. For example, this can transform the name Jon to John.
- **Import Column:** Reads data from a file and adds it into a Data Flow.
- **Lookup:** Performs a lookup on data to be used later in a transformation. For example, you can use this transformation to look up a city based on the zip code.
- **Merge:** Merges two sorted data sets into a single data set in a Data Flow.
- **Merge Join:** Merges two data sets into a single data set using a join function.
- **Multicast:** Sends a copy of the data to an additional path in the workflow.
- **OLE DB Command:** Executes an OLE DB command for each row in the Data Flow.
- **Percentage Sampling:** Captures a sampling of the data from the Data Flow by using a percentage of the total rows in the Data Flow.
- **Pivot:** Pivots the data on a column into a more non-relational form. *Pivoting* a table means that you can slice the data in multiple ways, much like in OLAP and Excel.
- **Row Count:** Stores the row count from the Data Flow into a variable.

- **Row Sampling:** Captures a sampling of the data from the Data Flow by using a row count of the total rows in the Data Flow.
- **Script Component:** Uses a script to transform the data. For example, you can use this to apply specialized business logic to your Data Flow.
- **Slowly Changing Dimension:** Coordinates the conditional insert or update of data in a slowly changing dimension.
- **Sort:** Sorts the data in the Data Flow by a given column.
- **Term Extraction:** Looks up a noun or adjective in text data.
- **Term Lookup:** Looks up terms extracted from text and references the value from a reference table.
- **Union All:** Merges multiple data sets into a single data set.
- **Unpivot:** Unpivots the data from a non-normalized format to a relational format.

Error Handling and Logging

In SSIS, there are several places that you can control error handling, and they depend on whether you are handling task or Data Flow errors. For task errors, package events are exposed in the user interface, with each event having the possibility of its own event handler design surface. This *design surface* is yet another area where you can define workflow along with the task and Data Flow surfaces you've already learned about. The event-handler design surface in SSIS is where you can specify a series of tasks to be performed if a given event happens for a task in the task flow. There are event handlers to help you develop packages that can self-fix problems. For example, the `OnError` error handler triggers an event whenever an error occurs anywhere in scope. The scope can be the entire package or an individual container. Event handlers are represented as a workflow, much like any other workflow in SSIS. An ideal use for event handlers would be to notify an operator if any component fails inside the package. You learn much more about event handlers in Chapter 17. You can also use the precedence constraints directly on the task flow design surface to direct workflow when a proceeding task fails to complete or evaluates to an expression that forces the workflow change.

An ultimate error within a Data Flow Task can be captured and handled with an error handler, but for finer control within the data pipe itself, each transformation must be configured to define the action that should be taken if a specific error occurs while processing the data. You can define whether the entire data transformation should fail and exit upon an error, or that only the bad rows should be redirected to a failed Data Flow branch. You can also choose to ignore any errors. The error handler shown in Figure 1-8 defines that if an error occurs during the Derived Column Transformation, the error rows will be output to a new error data stream. You can then use that outputted information to write to an output log or a destination connection as seen in Figure 1-8.

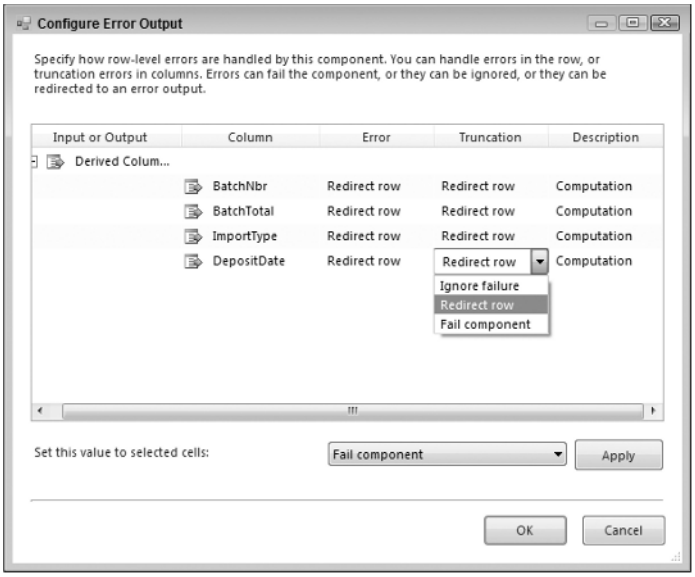

Figure 1-8

The On Failure error data stream can be seen in Figure 1-9 as a red line connecting the transform Derived Column Task to a Script Component Destination. The green lines show the normal happy path for the Data Flow through the data pipeline.

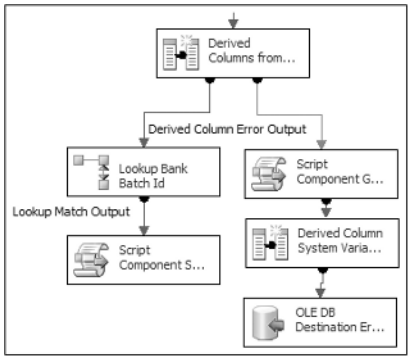

Figure 1-9

Logging has also been improved in SSIS. It is now at a much finer detail than what was available in the legacy DTS. More than a dozen events can be simply selected within each task or package for logging. You can enable partial logging for one task and enable much more detailed logging for billing tasks. Some of the examples of events that can be monitored are: `OnError`, `OnPostValidate`, `OnProgress`, and `OnWarning`, to name just a few. The logs can be written to nearly any connection: SQL Profiler, text files, SQL Server, the Windows Event log, or an XML file. We go through some examples of this in Chapter 17.

Editions of SQL Server

The available features in SSIS and SQL Server vary widely based on what edition of SQL Server you're using. As you can imagine, the more high-end the edition of SQL Server, the more features are available. In order from high-end to low-end, the following SQL Server editions are available:

- **SQL Server Enterprise Edition:** This edition of SQL Server is for large enterprises that need higher availability and more advanced features in SQL Server and business intelligence. For example, there is no limit on processors or RAM in this edition. You're bound only by the number of processors and amount of RAM that the OS can handle. Microsoft will also continue to support Developer Edition, which lets developers develop SQL Server solutions at a much reduced price.
- **SQL Server Standard Edition:** This edition of SQL Server now has a lot more value than ever before. For example, you can now create a highly available system in Standard Edition by using clustering, database mirroring, and integrated 64-bit support. These features were available only in Enterprise Edition in SQL Server 2000 and caused many businesses to purchase Enterprise Edition when Standard Edition was probably sufficient for them. Like Enterprise Edition in SQL Server 2005, it also offers unlimited RAM. Thus, you can scale it as high as your physical hardware and OS will allow. However, there is a cap of four processors with this edition.
- **SQL Server Workgroup Edition:** This new edition is designed for small and medium-sized businesses that need a database server with limited business intelligence and Reporting Services. Workgroup Edition supports up to two processors with unlimited database size. In SQL Server 2008 Workgroup Edition, the limit is 3 GB of RAM.
- **SQL Server 2008 Compact Edition:** This version was formally called the Express Edition and is the equivalent of Desktop Edition (MSDE) in SQL Server 2000 but with several enhancements. For example, MSDE never offered any type of management tool, and this is now included. Also included are the Import and Export Wizard, and a series of other enhancements. This remains a free addition of SQL Server for small applications. It has a database size limit of 4 GB. Most important, the query governor has been removed from this edition, allowing for more people to query the instance at the same time.

As for SSIS, you'll have to use at least the Standard Edition to receive the bulk of the SSIS features. In the Express and Workgroup Editions, only the Import and Export Wizard is available to you. You'll have to upgrade to the Enterprise or Developer Edition to see some features in SSIS. For example, the following advanced transformations are available only with the Enterprise Edition:

- Analysis Services Partition Processing Destination
- Analysis Services Dimension Processing Destination
- Data Mining Training Destination
- Data Mining Query Component
- Fuzzy Grouping
- Fuzzy Lookup
- Term Extraction
- Term Lookup

Half of these transformations are used in servicing Analysis Services. To continue that theme, one task is available only in Enterprise Edition — the Data Mining Query Task.

Summary

In this chapter, you were introduced to the historical legacy and the exciting new capabilities of the SQL Server Integration Services (SSIS) platform. We looked at where SSIS fits into the Business Intelligence roadmap for SQL Server, and then dove into an overview of the SSIS architecture. Within the architecture we stayed up at 20,000 feet to make sure you have a good understanding of how SSIS works and the core parts of the architecture. We talked about the core components of tasks, Data Flows, transformations, event handlers, containers, and variables — all crucial concepts that you'll be dealing with daily in SSIS. Packages are executable programs in SSIS that are a collection of tasks. Tasks are individual units of work that are chained together with precedence constraints. Lastly, transformations are the Data Flow items that change the data to the form you request, such as sorting the data.

In the next chapter, you look at some of the tools and wizards you have at your disposal to expedite tasks in SSIS. In Chapter 3, we do a deep dive into the various tasks in the Toolbox menu that you can use to create SSIS workflows, then move on to containers in the following chapter. In Chapter 4, we circle back into the Data Flow Task and examine the data components that are available to use within the data pipeline to perform the transform in ETL.