

L SQL is perform other technically Server tasks tasks in Integration classified a in Welcome workflow a workflow. as Services Integration a like business Even executing (SSIS) though intelligence is a one program it’s of mainly the D

feature to most or M used a script, and powerful A for SQL T is or data a E robust it R features can loads, Services I A be way you extended. in Server to SQL can load Server use data This it to and

2005. chapter do

It

What’s describes In SQL Wizard, data making By Microsoft dynamically SQL the Server source time Server workflow and much included called that to the alter New 7.0, another SQL of DTS’s the Data Microsoft a C the in minor Server package O the architecture destination. primary Transformation in P release feature. 2000 Y had at SQL R purpose runtime. new a was small I It of G also released, features SSIS Services H was team It had Server and T also to like of the E DTS covers transform had (DTS). developers ability the had extended Dynamic the DTS a to strong basics data 2005 execute was work logging from Properties following of the on tasks.

programs backbone almost a and much SSIS

task of broke any DBAs understated and of to OLE a the help run transformation and Import/Export DB–compliant

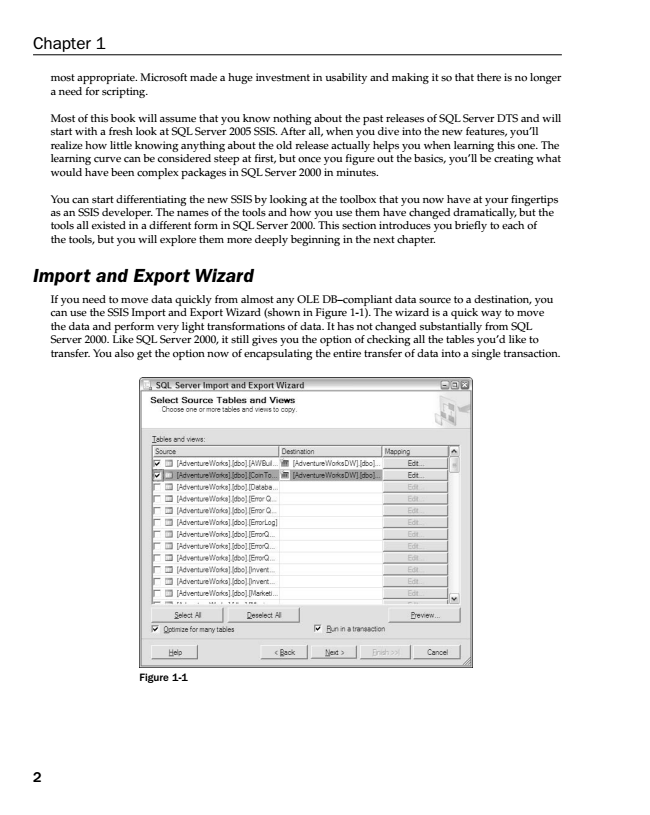
scripts,

you

developers. feature of

into many phases, called the multiphase data pump. Usability studies still showed that at this point developers had to create elaborate scripts to extend DTS to do what they wanted. For example, if you wanted DTS to conditionally load data based on the existence of a file, you would have to use the ActiveX Script task and VBScript to dynamically do this. The problem here was that most DBAs didn’t have this type of scripting experience.

After five years, Microsoft released the much touted SQL Server 2005, where DTS is no longer an understated feature, but one of the main business intelligence (BI) foundations. It’s been given so much importance now that it has its own service. DTS has also been renamed to SQL Server Integration Services (SSIS). So much has been added to SSIS that the rename of the product was



most appropriate. Microsoft made a huge investment in usability and making it so that there is no longer a need for scripting.

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 2005 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. 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 SQL Server 2000 in minutes.

You can start differentiating the new SSIS by looking at the toolbox that you now have at your fingertips as an SSIS developer. The names of the tools and how you use them have changed dramatically, but the tools all existed in a different form in SQL Server 2000. This section introduces you briefly to each of the tools, but you will explore them more deeply beginning in the next chapter.

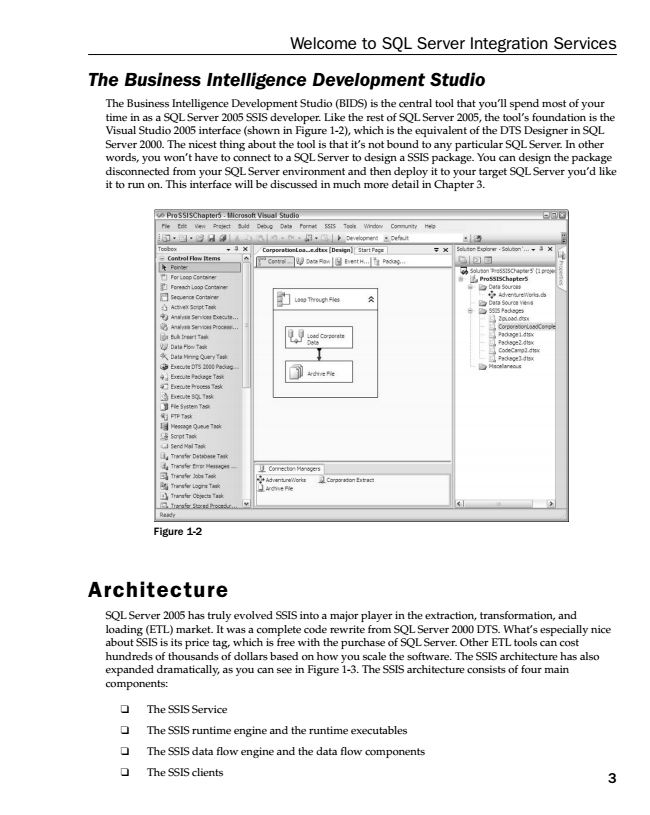
***Import and Export Wizard***

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). The wizard is a quick way to move the data and perform very light transformations of data. It has not changed substantially from SQL Server 2000. Like SQL Server 2000, it still gives you the option of checking all the tables you’d like to transfer. You also get the option now of encapsulating the entire transfer of data into a single transaction.

Figure 1-1

Chapter 1

2



***The Business Intelligence Development Studio***

The Business Intelligence Development Studio (BIDS) is the central tool that you’ll spend most of your time in as a SQL Server 2005 SSIS developer. Like the rest of SQL Server 2005, the tool’s foundation is the Visual Studio 2005 interface (shown in Figure 1-2), which is the equivalent of the DTS Designer in SQL Server 2000. The nicest thing about the tool is that it’s not bound to any particular SQL Server. In other words, you won’t have to connect to a SQL Server to design a SSIS package. You can design the package disconnected from your SQL Server environment and then deploy it to your target SQL Server you’d like it to run on. This interface will be discussed in much more detail in Chapter 3.

Figure 1-2

Architecture

SQL Server 2005 has truly evolved SSIS into a major player in the extraction, transformation, and loading (ETL) market. It was a complete code rewrite from SQL Server 2000 DTS. What’s especially nice about SSIS is its price tag, which is free with the purchase of SQL Server. Other ETL tools can cost hundreds of thousands of dollars based on how you scale the software. The SSIS architecture has also expanded dramatically, as you can see in Figure 1-3. The SSIS architecture consists of four main components:

❑ The SSIS Service

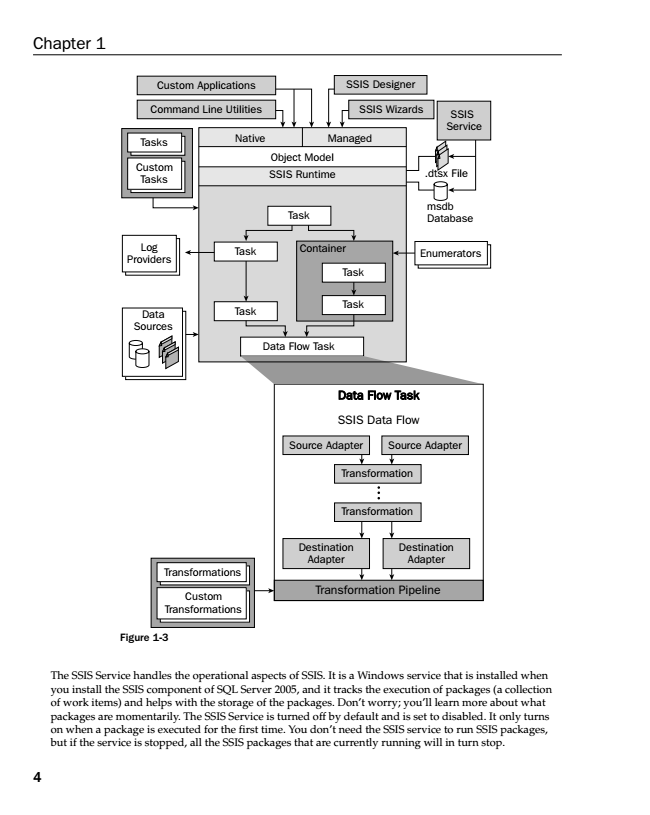
❑ The SSIS runtime engine and the runtime executables

❑ The SSIS data flow engine and the data flow components

❑ The SSIS clients

Welcome to SQL Server Integration Services

3



Chapter 1

SSIS Custom Applications

Designer

Command Line Utilities

SSIS Wizards

Object Model

Data Data Flow Flow Task Task

SSIS Data Flow

Transformation

Transformation Pipeline

SSIS Service

Managed Tasks

Native

Custom Tasks

Task

Log Providers

Data Flow Task

.dtsx File

Data Sources

SSIS Runtime

msdb Database

Task

Container

Enumerators

Task

Task

Task

Source Source Adapter

Adapter

Transformation

Destination Adapter Transformations

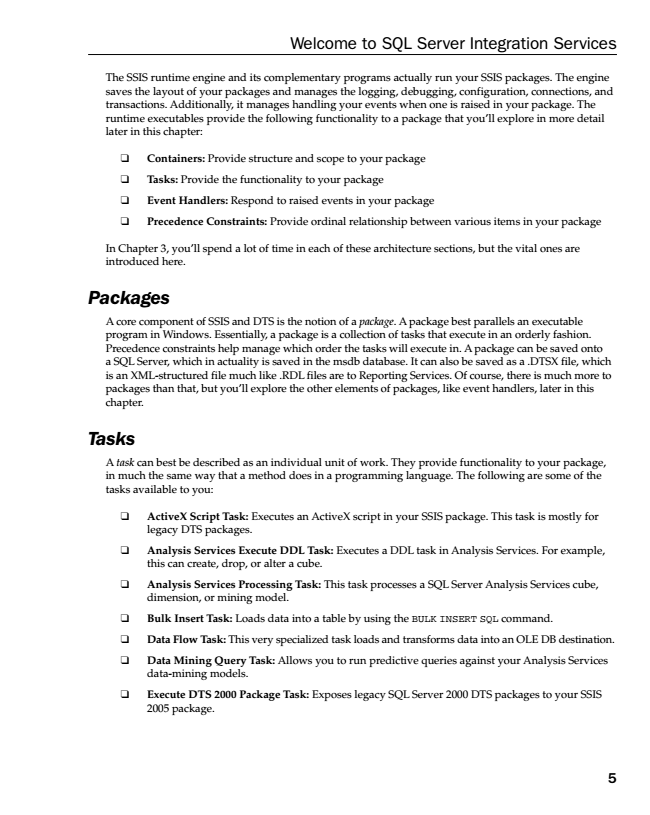
Custom Transformations

4

Destination Adapter

Figure 1-3

The SSIS Service handles the operational aspects of SSIS. It is a Windows service that is installed when you install the SSIS component of SQL Server 2005, and it tracks the execution of packages (a collection of work items) and helps with the storage of the packages. Don’t worry; you’ll learn more about what packages are momentarily. The SSIS Service is turned off by default and is set to disabled. It only turns on when a package is executed for the first time. You don’t need the SSIS service to run SSIS packages, but if the service is stopped, all the SSIS packages that are currently running will in turn stop.



Welcome to SQL Server Integration Services

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 one is raised in your package. The runtime executables provide the following functionality to a package that 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 architecture sections, but the vital ones are introduced here.

***Packages***

A core component of SSIS and DTS is the notion of a package. A package best parallels an executable program in Windows. Essentially, a package is a collection of tasks that execute in an orderly fashion. Precedence constraints help manage which order the tasks will execute in. A package can be saved onto a SQL Server, which in actuality is saved in the msdb database. It can also be saved as a .DTSX file, which 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. They provide functionality to your package, in much the same way that a method does in a programming language. The following are some of the tasks available to you:

❑ ActiveX Script Task: Executes an ActiveX script in your SSIS package. This task is mostly for

legacy DTS packages.

❑ Analysis Services Execute DDL Task: Executes a DDL task in Analysis Services. For example,

this can create, drop, or alter a cube.

❑ 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 destination.

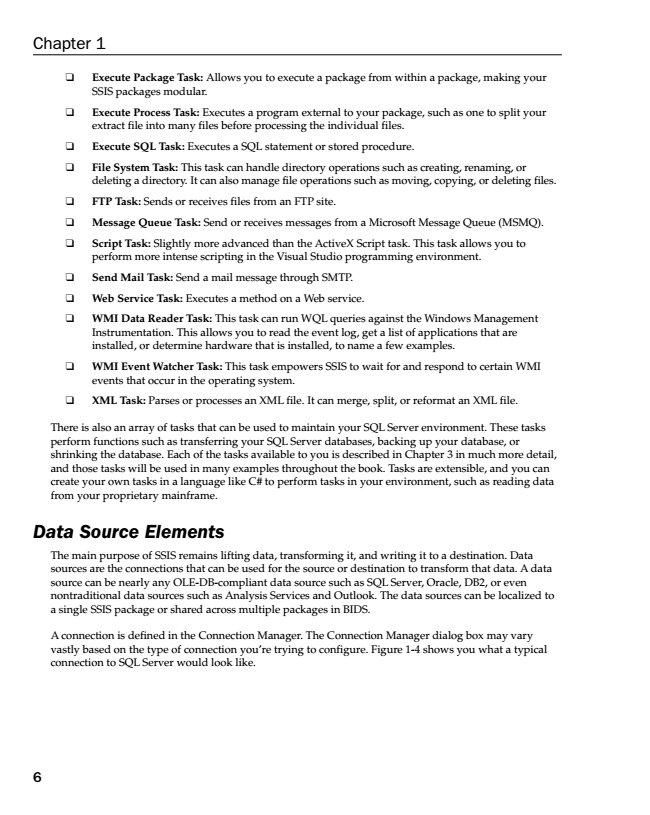
❑ Data Mining Query Task: Allows you to run predictive queries against your Analysis Services

data-mining models.

❑ Execute DTS 2000 Package Task: Exposes legacy SQL Server 2000 DTS packages to your SSIS

2005 package.

5



Chapter 1

❑ 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: Send or receives messages from a Microsoft Message Queue (MSMQ).

❑ Script Task: Slightly more advanced than the ActiveX Script task. This task allows you to

perform more intense scripting in the Visual Studio programming environment.

❑ Send Mail Task: Send 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 an array of tasks 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 those tasks will be used in many examples throughout the book. Tasks are extensible, and you can create your own tasks in a language like C# to perform tasks in your environment, such as reading data from your proprietary mainframe.

***Data Source Elements***

The main purpose of SSIS remains lifting data, transforming it, and writing it to a destination. Data sources are the connections that can be used for the source or destination to transform that data. A data source can be nearly any OLE-DB-compliant data source 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.

A connection is defined in the Connection Manager. The Connection Manager dialog box may vary vastly 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.

6

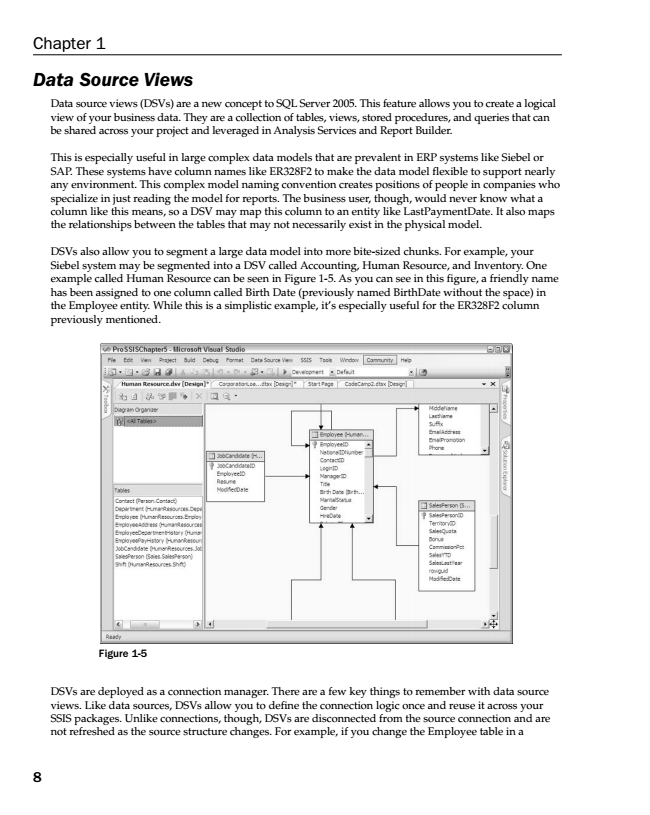


Figure 1-4

You can configure the connection completely offline, and the SSIS package will not use it until you begin to instantiate it in the package. The nice thing about this is that you can develop in an airport and then connect as needed.

Welcome to SQL Server Integration Services

7



***Data Source Views***

Data source views (DSVs) are a new concept to SQL Server 2005. This feature allows you to create a logical view of your business data. They are a collection of tables, views, stored procedures, and queries that can be shared across your project and leveraged in Analysis Services and Report Builder.

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 to support nearly any environment. This complex model naming convention creates positions of people in companies who specialize in just reading the model for reports. The business user, though, would never know what a column like this means, so a DSV may map this column to an entity like LastPaymentDate. It also maps the relationships between the tables that may not necessarily exist in the physical model.

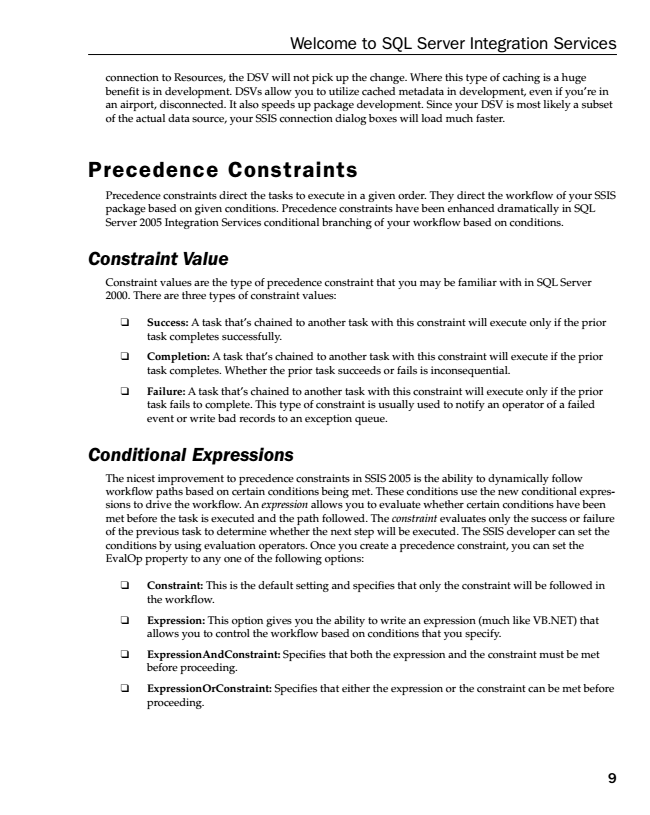
DSVs also allow you to segment a large data model into more bite-sized chunks. For example, your Siebel system may be segmented into a DSV called Accounting, Human Resource, and Inventory. One example called Human Resource can be seen in Figure 1-5. As you can see in this figure, a friendly name has been assigned to one column called Birth Date (previously named BirthDate without the space) in the Employee entity. While this is a simplistic example, it’s especially useful for the ER328F2 column previously mentioned.

Figure 1-5

DSVs are deployed as a connection manager. There are a few key things to remember with data source views. Like data sources, DSVs allow you to define the connection logic once and reuse it across your SSIS packages. Unlike connections, though, DSVs are disconnected from the source connection and are not refreshed as the source structure changes. For example, if you change the Employee table in a

Chapter 1

8



Welcome to SQL Server Integration Services

connection to Resources, the DSV will not pick up the change. Where this type of caching is a huge benefit is in development. DSVs allow you to utilize cached metadata in development, even if you’re in an airport, disconnected. It also speeds up package development. Since your DSV is most likely a subset of the actual data source, your SSIS connection dialog boxes will load much faster.

Precedence Constraints

Precedence constraints direct the tasks to execute in a given order. They direct the workflow of your SSIS package based on given conditions. Precedence constraints have been enhanced dramatically in SQL Server 2005 Integration Services conditional branching of your workflow based on conditions.

***Constraint Value***

Constraint values are the type of precedence constraint that you may be familiar with in SQL Server 2000. 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.

❑ 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.

❑ 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.

***Conditional Expressions***

The nicest improvement to precedence constraints in SSIS 2005 is the ability to dynamically follow workflow paths based on certain conditions being met. These conditions use the new conditional expres- sions to drive the workflow. 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 EvalOp 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.

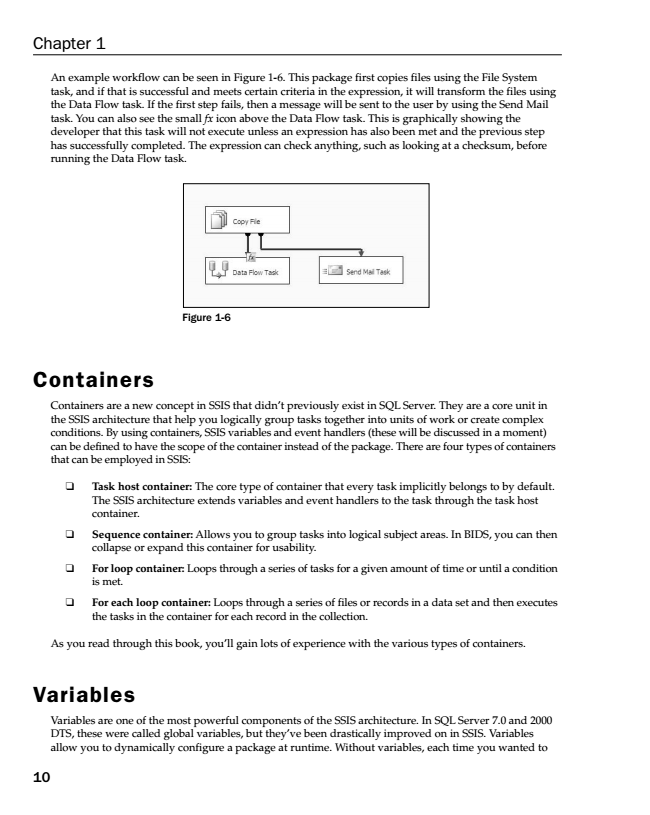
❑ ExpressionAndConstraint: Specifies that both the expression and the constraint must be met

before proceeding.

❑ ExpressionOrConstraint: Specifies that either the expression or the constraint can be met before

proceeding.

9



An example workflow can be seen in Figure 1-6. This package first copies files using the File System task, and if that is successful and meets certain criteria in the expression, it will transform the files using the Data Flow task. If the first step fails, then a message will be sent to the user by using the Send Mail task. You can also see the small fx icon above the Data Flow task. This is graphically showing the developer that this task will not execute unless an expression has also been met and the previous step has successfully completed. The expression can check anything, such as looking at a checksum, before running the Data Flow task.

Figure 1-6

Containers

Containers are a new concept in SSIS that didn’t previously exist in SQL Server. They are a core unit in the SSIS architecture that help you logically group tasks together into units of work or create complex conditions. By using containers, SSIS variables and event handlers (these will be discussed in a moment) can be defined to have the scope of the container instead of the package. There are four types of containers that can be employed in SSIS:

❑ Task host container: The core type of container that every task implicitly belongs to by default.

The SSIS architecture extends variables and event handlers to the task through the task host container.

❑ Sequence container: Allows you to group tasks into logical subject areas. In BIDS, you can then

collapse or expand this container for usability.

❑ For loop container: Loops through a series of tasks for a given amount of time or until a condition

is met.

❑ For each 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.

As you read through this book, you’ll gain lots of experience with the various types of containers.

Variables

Variables are one of the most powerful components of the SSIS architecture. In SQL Server 7.0 and 2000 DTS, these were called global variables, but they’ve been drastically improved on in SSIS. Variables allow you to dynamically configure a package at runtime. Without variables, each time you wanted to

Chapter 1

10