1

Welcome to SQL Server Integration Services

WHAT'S IN THIS CHAPTER?

- What's new to this version of SSIS
- Exploring tools you'll be using in SSIS
- Overview of the SSIS architecture
- Considering your licensing options around BI with SQL Server

SQL Server Integration Services (SSIS) is the anchor in a growing suite of products that make up the Microsoft SQL Server Business Intelligence (BI) platform. What makes SSIS so important is without the data movement and cleansing features that SSIS brings to the table, the other SQL Server BI products can't operate. What's the point of a cube, for example, with bad or inconsistent data? In its simplest form, SSIS is an enterprise-level, in-memory ETL 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 million-dollar, third-party tools. The best thing about SSIS is that you have already paid for it when you license SQL Server.

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 was much to learn. Since the first edition of SSIS, we have collectively racked up many years of experience converting older DTS packages and mind-sets over to using it, and trust us when we say that no one who has made the change is asking to go back. We've learned some things, too.

While SQL Server 2012 was a large jump forward for SSIS, SQL Server 2014 has some very small iterative changes. When we wrote this book, we dug deeply to mine the decades of cumulative experience working with this product, adding our collective knowledge back into these pages. We hope you will agree that the result makes your experience with SSIS a more productive one. This chapter starts from the beginning by providing an overview of SSIS, describing 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 very understated feature of SQL Server called Data Transformation Services (DTS). DTS was the backbone of the Import/Export Wizard, and its primary purpose was to transform data from almost any OLE DB–compliant data source to almost any 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 had to use the ActiveX Script Task to code a solution using the file system object in VBScript. The problem with that was DTS lacked some of the common components needed to support typical ETL processes. Although it was powerful if you knew how to write scripting code, most DBAs didn't have this type of scripting experience (or time).

After five years, Microsoft released the much-touted SQL Server 2005 and SSIS, which was no longer an understated feature like DTS. With the SQL Server 2008 release, SSIS was given extra scalability features to help it appeal more to the enterprise. This is entirely appropriate because so much has been added to SSIS. Microsoft made a huge investment in usability, with simple enhancements to the toolbox that allow newer users to ramp up easier. The main focus of the newest release of SQL Server is on the management and deployment of SSIS.

WHAT'S NEW IN SSIS

The scope of the SQL Server 2014 release of SSIS resembles the scope of the SQL Server 2008 R2 release. With the last release of SQL Server 2008 R2, the Microsoft SSIS team did very incremental changes after a very large SQL Server 2008 release. In SQL Server 2012 release, Microsoft had focused on SSIS manageability, making it easier to deploy and execute. Also added in 2012 are robust new data cleansing components that help you standardize and detect data anomalies. Furthermore, improvements to the development tools will help make SSIS developers more productive and help new developers get up to speed more easily. The SQL Server 2014 release uses a newer version of Visual Studio but all in all, it will feel much like SQL Server 2012. You will find new components in SQL Server 2014 SSIS, but they will have to be downloaded from sites like CodePlex from the product team and will eventually be rolled into the core product at a future release.

TOOLS OF THE TRADE

Most of this book will assume that you know nothing about previous releases of SQL Server SSIS. Instead, it takes a fresh look at SQL Server SSIS. The learning curve can be considered steep at first, but once you figure out the basics, you'll be creating complex packages in no time. To provide an idea of how easy SSIS is to use, the following section looks at a staple tool 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 or flat file 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, but most packages you wish to keep in a BI solution should not be created with the wizard. The wizard provides a quick way to move data and perform very light transformations of data but does not create packages that use best practices. The wizard is available in all editions of SQL Server except the Local Database edition and Express. It enables you to persist the logic of the data movement into a package file. The basic concept of an import/export wizard has not changed substantially from the days of DTS. You still have the option to check all the tables you want to transfer. In addition, however, you can also encapsulate 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 "2014" ➪ Import and Export Data. The other option is to open a project in the SSIS development environment and select Project ➪ SSIS Import and Export Wizard. We cover this in detail in Chapter 2. Before we get into all the mechanics for that, see Figure 1-1 for an example of the wizard that has bulk loaded tables.

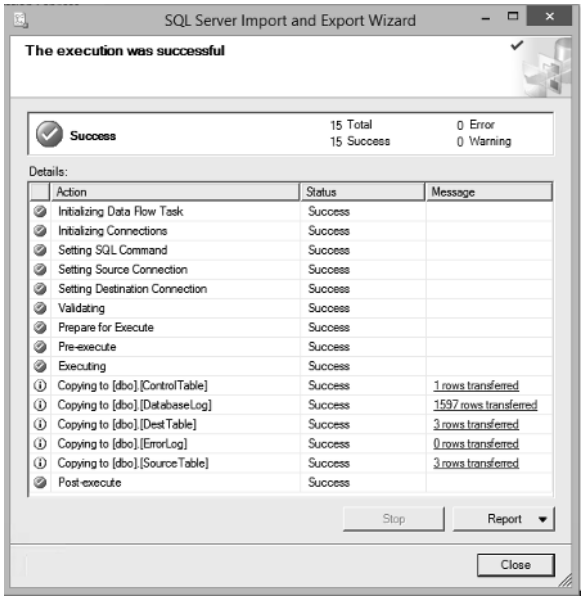

FIGURE 1-1

The SQL Server Data Tools Experience

The SQL Server Data Tools (SSDT) was previously called Business Intelligence Development Studio (BIDS) in SQL Server 2008, and it is the central environment in which you'll spend most of your time as an SSIS developer. SSDT is just a specialized use of the familiar Visual Studio development environment. In SQL Server 2014, SSDT no longer installs when you install SQL Server. Instead, you'll have to download and install the SQL Server Data Tools (Business Intelligence for Visual Studio) from the Microsoft website. At the time of this publication, SQL Server 2014 can use the Visual Studio 2012 and 2013 versions to design SSIS packages. Visual Studio can host many different project types, from Console applications to Class Libraries and Windows applications. Although you may see many project types when you create a project, SSDT actually contains project templates for only Analysis Services, Integration Services, Report Server, and variants thereof. SSIS in particular uses a BI project type called an Integration Services project (see Figure 1-2), which provides a development design surface with a completely ETL-based set of tools in the Toolbox window.

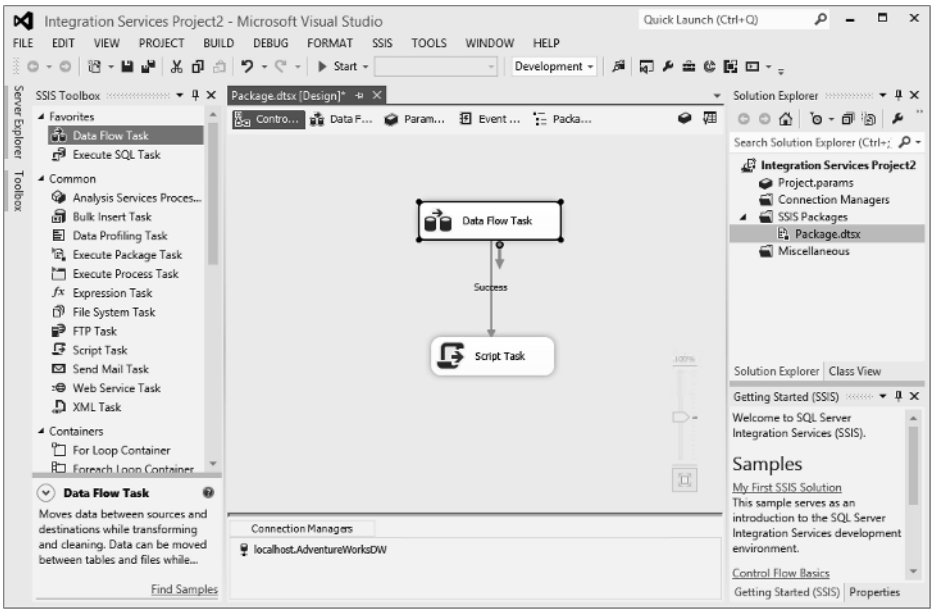

FIGURE 1-2

This development environment is similar to the legacy DTS Designer, but the approach is completely different. Most important, 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. SSIS solutions are developed just like all other .NET development solutions, including being persisted to files — in this case, XML file structures with a .DSTX file extension.

You can even develop within the SSDT environment without a connection to a SQL Server instance using the offline mode. Once your solution is complete, it can be built and deployed to one or multiple target SQL servers. These changes from DTS to SSIS are crucial to establishing the discipline and best practices of existing software development methodologies as you develop business intelligence solutions. We'll discuss this SSDT development interface in more detail in Chapter 2.

SSIS ARCHITECTURE

Microsoft has truly established SSIS as a major player in the extraction, transformation, and loading (ETL) market. Not only is the SSIS technology a complete code rewrite from SQL Server 2000 DTS, it now rivals other third-party ETL tools that can cost hundreds of thousands of dollars depending on how you scale the software — and it is included free with the purchase of SQL Server 2014. Free always sounds great, but most free products can take you only so far if the feature set is minimal or the toolset has usability, scalability, or enterprise performance limitations. SSIS, however, is the real deal, satisfying typical ETL requirements with an architecture that has evolved dramatically from earlier incarnations. At the time of this publication, SSIS held the world speed record of loading more than 2 terabytes in a single hour.

Packages

A core component of SSIS is the notion of a *package*. A package best parallels an executable program that you can write that contains workflow and business logic. Essentially, a package is a collection of tasks snapped together to execute in an orderly fashion. A package is also a unit of execution and development, much like a .NET developer creates programs or DLL files. Precedence constraints are used to connect the tasks together and manage the order in which they execute, based on what happens in each task or based on rules defined by the package developer. The package is brought together into a .DTSX file that is actually an XML-structured file with collections of properties. Just like other .NET projects, the file-based code is marked up using the development environment and can then be saved and deployed to a SQL Server.

Don't worry; you won't have to know how to write this type of XML to create a package. That's what the designer is for. 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, and you'll explore the other elements of packages, such as event handlers, later in this chapter.

Control Flow

The brain of a package is its *Control Flow*, which orchestrates the order of execution for all its components. The components consist of tasks and containers and are controlled by *precedence constraints*, discussed later in this chapter. For example, Figure 1-3 shows three tasks that are tied together with two precedence constraints.

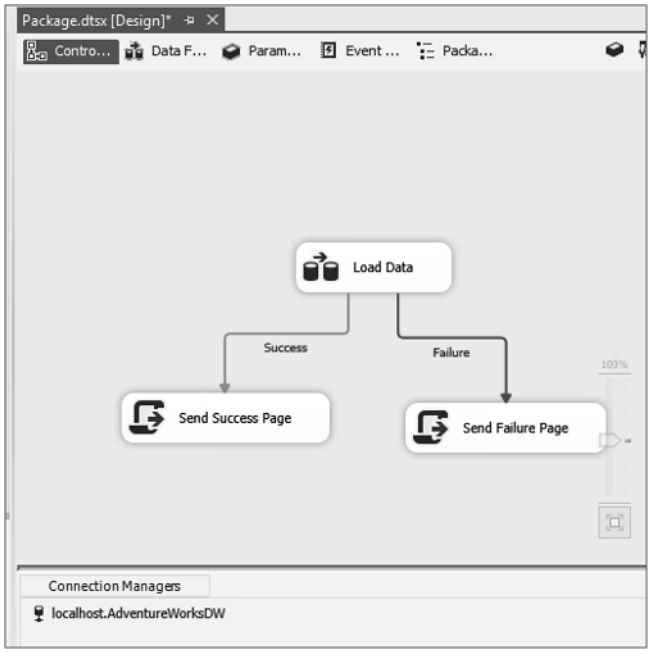

FIGURE 1-3

Tasks

A *task* can best be described as an individual unit of work. 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 develop your own tasks, but here are the current ETL tasks available to you out of the box:

- **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.
- **CDC Control Task:** Maintains and interacts with the change data capture (CDC) feature from SQL Server.
- **Data Flow Task:** This very specialized task loads and transforms data into an OLE DB and 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 task enables the examination of data; it replaces your ad hoc data profiling techniques.

- **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.
- **Expression Task:** Sets a variable to an expression at runtime.
- **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 enables you to perform .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 enables 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.

Also included are a whole set of DBA-oriented tasks that enable 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 available tasks 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. To learn more about this topic, see Chapter 19.

> **NOTE** *There's a thriving ecosystem of third-party components that are available for SSIS. If you are looking for a task or Data Flow component that doesn't exist out of the box, be sure to first search online before creating your own. Some examples of these components include support for SFTP, SalesForce.com communication, SharePoint integration, and compression of files to name just a few.*

Precedence Constraints

Precedence constraints are package components that direct tasks to execute in a given order. In fact, precedence constraints are the connectors that not only link tasks together but also define the workflow of your SSIS package. A constraint controls 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 enable and disable 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. You'll look at many examples of using these constraints throughout this book.

To set up a precedence constraint between two tasks, you must set the constraint value; optionally, you can set an expression. The following sections provide a brief overview of the differences between the two.

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

- **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 or not the prior task succeeds or fails. 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. These precedence constraints are colored red.

You can also conditionally tie tasks together by writing logic on a precedence constraint. This is done by placing an SSIS expression language (resembles C#) on the precedence constraint. For example, you might specify that a task should run only at the end of each month. To do this, you would add an expression that evaluated the runtime of the package to determine if the next step should be run. Much more about writing expressions can be found in Chapter 5.

Containers

Containers are core units in the SSIS architecture for grouping tasks together logically into units of work. Besides providing visual consistency, containers enable 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:** Not a visible element that you'll find in the Toolbox, but rather 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.
- **For Loop Container:** Loops through a series of tasks 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.

Because containers are so integral to SSIS development, Chapter 6 is devoted to them. As you read through the book, you'll see many real-world examples that demonstrate how to use each of these container types for typical ETL development tasks.

Data Flow

The core strength of SSIS is its capability to extract data into the server's memory, transform it, and write it out to an alternative destination. If the Control Flow is the brains of SSIS, then the Data Flow would be its heart. The in-memory architecture is what helps SSIS scale and what makes SSIS run faster than staging data and running stored procedures. Data sources are the conduit for these data pipelines, and they 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 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 in scope to a single SSIS package or shared across multiple packages in a project.

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

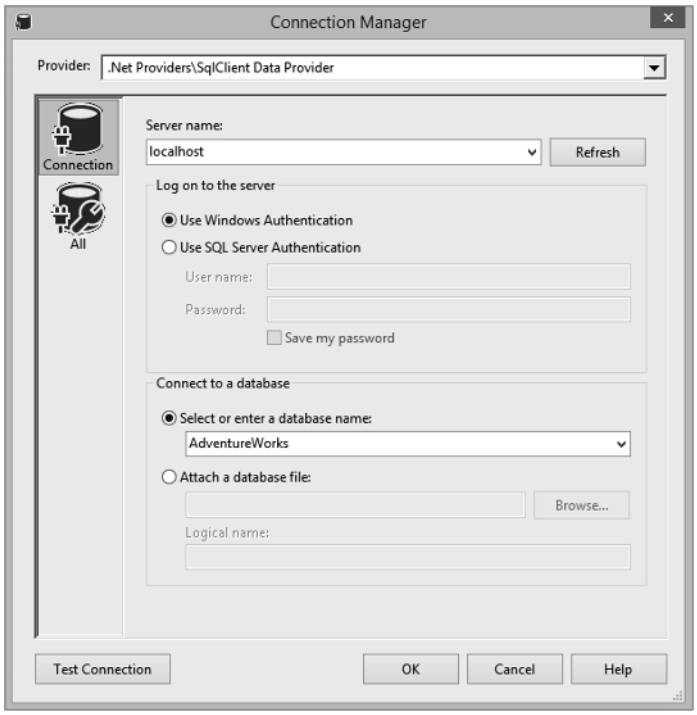

FIGURE 1-4

Connection Managers are used to centralize connection strings to data sources and to abstract them from the SSIS packages themselves. They can be shared across multiple packages in a project or isolated to a single package. Connection Managers also allow you to externalize the configuration of them at runtime by your DBA with a configuration file or parameters (which we'll describe in Chapter 22). 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.

You learned earlier that the Data Flow Task is simply another executable task in the package. The Data Flow Task is the pipeline 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 SSDT 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 the name of the Data Flow Task from the drop-down list. Just as the Control Flow handles the main workflow of the package, the Data Flow handles the transformation of data in memory. 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 1-5, a new column is derived using the Derived Column Transformation, and that new column is then available to subsequent transformations or to the destination.

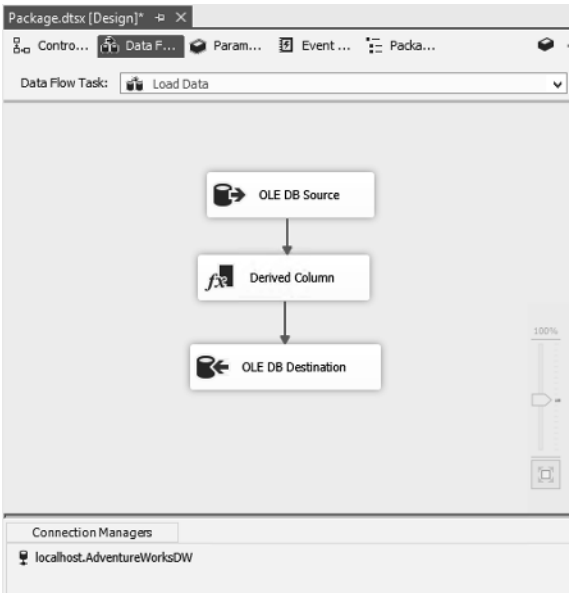

FIGURE 1-5

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.

Sources

A *source* is a component that you add to the Data Flow design surface to specify the location of the source data that will send data to components downstream. Sources are configured to use Connection Managers in order to enable the reuse of connections throughout your package. SSIS provides eight out-of-the-box sources:

- **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; it is especially quick to read by SSIS. This component is one of the only components that does not use a Connection Manager.
- **Xml Source:** Retrieves data from an XML document. This source does not use a Connection Manager to configure it.
- **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 hand-coding a connection and retrieval from a database.
- **CDC Source:** Reads data out of a table that has change data capture (CDC) enabled. Used to retrieve only rows that have changed over a duration of time.
- **ODBC Source:** Reads data out of table by using an ODBC provider instead of OLE DB. When you are given the choice between OLE DB and ODBC, it is still recommended in SSIS packages that you use OLE DB.

If the source components included in SSIS do not provide the functionality required for your solution, you can write code to connect to any data source that is accessible from a .NET application. 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 in multiple packages, you can develop one by using the SSIS .NET API and object model.

Transformations

Transformations are key components within the Data Flow that allow changes to the data within the data pipeline. You can use transformations to split, divert, and remerge data in the data pipeline. 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 easily accomplished by dropping a Sort and a Lookup Transformation onto the Data Flow design surface and configuring them.

Transformation components in the SSIS Data Flow affect data in the data pipe in memory. Because this process is done in memory, it can be much faster than loading the data into a staging environment and updating the staging system with stored procedures. Here's a complete list of transformations and their purposes:

- **Aggregate:** Aggregates data from transformation or source.
- **Audit:** Exposes auditing information from the package to the data pipe, such as when the package was run and by whom.
- **CDC Splitter:** After data has been read out of a table with CDC enabled, this transform sends data that should be inserted, updated, and deleted down different paths.

- **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.
- **DQS Cleansing:** Performs advanced data cleansing using the Data Quality Services engine.
- **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 to 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 Data Flow's total rows.
- **Pivot:** Pivots the data on a column into a more nonrelational 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 Data Flow's total rows.
- **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.

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:

- **ADO.NET Destination:** Exposes data to other external processes, such as a .NET application.
- **Data Mining Model Training:** Trains an Analysis Services mining model by passing data from the Data Flow to the destination.
- **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.
- **ODBC Destination:** Outputs data to an ODBC data connection like SQL Server, DB2, or Oracle.
- **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. It's usually used as an intermediate persistence mechanism.
- **Recordset Destination:** Writes the records to an ADO record set. Once written, to an object variable, it can be looped over a variety of ways in SSIS like a Script Task or a Foreach Loop Container.
- **SQL Server Compact Edition Destination:** Inserts data into a SQL Server running the Compact Edition of the product on a mobile device or PC.
- **SQL Server Destination:** The destination that you use to write data to SQL Server. This destination has many limitations, such as the ability to only write to the SQL Server where the SSIS package is executing. For example, if you're running a package to copy data from Server 1 to Server 2, the package must run on Server 2. This destination is there largely for backwards compatibility and should not be used.

Variables

Variables are another vital component of the SSIS architecture. SSIS variables can be set to evaluate to an expression at runtime. You can also set variables to be set in the Control Flow with either a Script Task or an Expression Task. Variables in SSIS have become the method of exchange between many tasks and transformations, making the scoping of variables much more important. By default, SSIS variables exist within a package scope, but they can be scoped to different levels within a package as mentioned earlier in the "Containers" section.

Parameters

Parameters behave much like variables but with a few main exceptions. Parameters, like variables, can make a package dynamic. The largest difference between them is that parameters can be set outside the package easily and can be designated as values that must be passed in for the package to start, much like a stored procedure input parameter. Parameters replace the capabilities of Configurations in previous releases of SQL Server.

Error Handling and Logging

In SSIS, you can control error handling in several places, depending on whether you are handling task or Data Flow errors. For task errors, package events are exposed in the user interface, and each event can have its own event-handler design surface. This design surface is yet another area where you can define workflow, in addition to the Control Flow and Data Flow surfaces you've already learned about. Using the event-handler design surface in SSIS, you can specify a series of tasks to be performed if a given event happens for a task in the task flow.

Some event handlers can 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 task or container. Event handlers are represented as a workflow, much like the Control Flow workflow in SSIS. An ideal use for an event handler would be to notify an operator if any component fails inside the package. (You will learn much more about event handlers in Chapter 18.) You can also use the precedence constraints directly on the task flow design surface to direct workflow when a task fails to complete or it evaluates to an expression that forces the workflow to change.

Logging has also been improved in SSIS in this latest release. Logging is now enabled by default, and packages are easier to troubleshoot. More than a dozen events can be simply selected within each task or package for logging. You can also choose to enable partial logging for one task and enable much more detailed logging for another task, such as billing. 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. You'll see some examples of this in Chapter 18.

EDITIONS OF SQL SERVER

The available features in SSIS and SQL Server vary according to what edition of SQL Server you're using. Of course, the more high-end the edition of SQL Server, the more features are available. In order from high-end to low-end, the following is a partial list of SQL Server editions:

- **SQL Server Enterprise Edition:** This edition of SQL Server is for large enterprises that need high 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 the amount of RAM that the OS can handle. Microsoft will also continue to support Developer Edition, which enables developers to create SQL Server solutions at a much reduced price. Ultimately, if you're trying to scale your solution into terabytes of data storage in SQL Server then Enterprise Edition is the right choice for you.
- **SQL Server Business Intelligence Edition:** This edition includes all the features of Standard Edition and also includes additional data cleansing features like Data Quality Services, which helps you create business rules that SSIS consumes. It also has many SharePoint integration features outside of SSIS and some scalability features. You can scale all the BI features other than the database engine to the OS maximum of cores with this edition.
- **SQL Server Standard Edition:** This edition of SQL Server now offers even more value than before. For example, you can create a highly available system in Standard Edition by using clustering, database mirroring, and integrated 64-bit support. Like Enterprise Edition in SQL Server 2012, 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.

As for SSIS, you'll have to use at least the Standard Edition to receive the bulk of the SSIS features. In the Express Edition, only the Import and Export Wizard is available. BI Edition gives you access to items like Data Quality Services and the DQS Cleansing Transformation. You need to upgrade to the higher editions in order to see some features in SSIS. For example, the following advanced transformations are available only with Enterprise Edition:

- Analysis Services Partition Processing Destination
- Analysis Services Dimension Processing Destination
- CDC Source, Destination, Splitter Transformation, and CDC Control Task
- 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. Along those same lines, 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 capabilities of the SQL Server Integration Services (SSIS) platform. You looked at where SSIS fits into the business intelligence (BI) platform for SQL Server, and then dove into an overview of the SSIS architecture.

Within the architecture, we stayed up at 20,000 feet to ensure that you have a good understanding of how SSIS works and the core parts of the architecture. You learned 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 contain 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.

The next chapter describes some of the tools and wizards you have at your disposal to expedite tasks in SSIS. Chapter 3 dives deeply into the various tasks in the SSIS Toolbox that you can use to create SSIS workflows. In Chapter 4, you'll learn about Data Flow Task and examine the data components that are available for use within the Data Flow pipeline to perform the transformations in ETL.