



WHITEPAPER

SSIS Basics



Table of Contents

An Introduction to Key SSIS Functionality	1
SSIS Terminology	1
The SSIS Toolset	2
The SSIS Development Environment	2
The Moving Parts of SSIS	4
The SSIS Control Flow	9
The SSIS Data Flow	14

SSIS Basics

AN INTRODUCTION TO KEY SSIS FUNCTIONALITY

SQL Server Integration Services (SSIS) is a popular and mature tool for performing data movement and cleanup operations. In the Microsoft® ecosystem, SSIS is one of the most common extract, load, and transform (ETL) tools in use today. SSIS is powerful and configurable, yet surprisingly easy to use.

This guide walks you through the essential moving parts of SSIS. Check out the next guide in this series that focuses on how to build your first SSIS package.

SSIS TERMINOLOGY

Before we explore the machinery of SSIS, it's important to understand some of the terms you'll often hear used by SSIS professionals. Here are some of the key terms and phrases you'll need to know to be successful with SSIS.

- **Package:** The SSIS package is the central component of the SSIS code, and it's the canvas on which you'll spend most of your development time. An SSIS package is a collection of one or more operations that are invoked together.
- **Task:** A task is a single operation within a package. There are dozens of different types of tasks available in SSIS.
- **Component:** A component is part of a data pipeline, representing either a source from which data is retrieved, a destination to which data is written, or a transformation that manipulates or reshapes the data.
- **Execution:** This is the act of invoking, or running, the logic in an SSIS package. Packages can be executed from within the SQL Server Data Tools (SSDT) development environment or directly on a properly configured instance of SQL Server.
- **Deployment:** A deployment occurs when the fully developed SSIS project is pushed from the development workstation to an instance of SQL Server, where it can then be executed either manually or through a scheduling tool such as SQL Server Agent. Deployment is usually more complex than copying code from one machine to another, although SQL Server does a good job of hiding that complexity for most deployments.
- **Project:** Source code in SSIS is arranged into functional units called projects. A project can contain one package or many packages. In most cases, when deploying SSIS code, the entire project is deployed to the server.
- **Solution:** A solution is a logical grouping of related projects.
- **The SSIS runtime engine:** This is the logic that allows a package to run. When you're working with SSIS packages in SSDT, the packages will be executed using the SSIS runtime on your development machine. After the code is deployed to SQL Server, any execution run on that server will use the server's SSIS runtime.

THE SSIS TOOLSET

SSIS is licensed as part of SQL Server, and the internals of running an SSIS package can be installed as part of the normal SQL Server install. As an SSIS developer, however, you'll need a couple of additional tools installed on your workstation:

- **SSDT:** SSDT is a lighter version of Microsoft Visual Studio, configured with the extensions to create SSIS projects. When building or testing SSIS packages, SSDT is the tool in which you will spend most of your time.
- **SQL Server Management Studio (SSMS):** Although not strictly required for building SSIS packages, at some point during the development cycle you'll need SSMS to test and deploy the ETL processes you create in SSIS.

When setting up your SSIS development environment, it's important to remember that you must install the SSDT version of Visual Studio to be able to work with SSIS projects. If you already have Visual Studio installed, you'll need to download and install the SSDT components.

THE SSIS DEVELOPMENT ENVIRONMENT

If you've ever worked with Visual Studio, the SSIS development environment will look familiar. SSDT uses the same multiple-document interface (MDI) that enables you to open or close each SSIS package independently, open several packages at once, and conveniently dock them as needed.

Inside this development environment, you'll be frequently using the following assets:

- **Solution Explorer:** This window allows easy browsing of the current solution and all the files contained within it. You can also add, delete, or rename files in the current solution.
- **Properties Window:** This context-aware window shows the properties for the currently selected item.
- **SSIS Toolbox:** Like the name implies, this is your toolbox of available operations in SSIS. This list is also context-aware and shows the tools for either the control flow or data flow (more on these later in the article) based on where you're currently working.
- **SSIS Package:** This is the package you're currently working on. Although a single package is shown in Figure 1, the MDI allows you to have more than one package open at the same time.
- **Connection Manager Tray:** This is the area where any currently available connections are shown. We'll go into more details on the connection managers shortly.

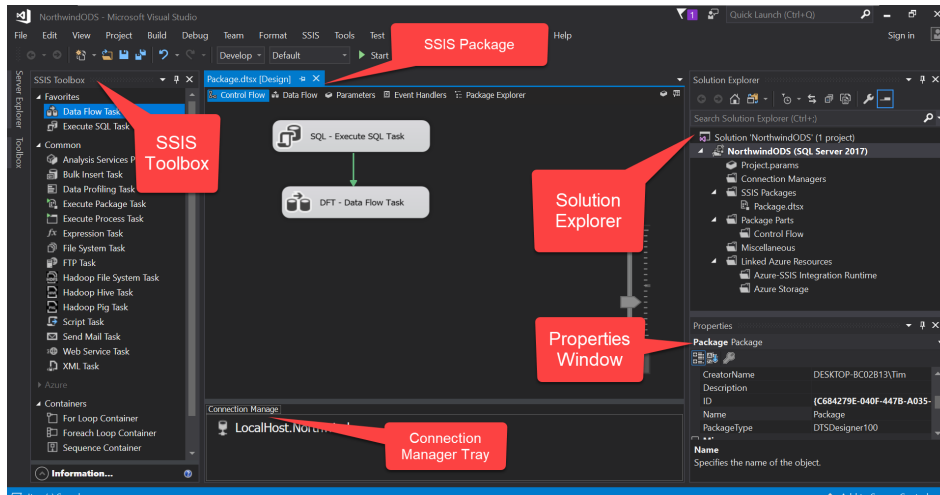


Figure 1: The SSIS Development Environment

Figure 1 shows the SSDT designer with each of these assets called out.

Within each SSIS package, there's a set of tabs you can use to navigate the various layers of that package. Highlighted in Figure 2 is the set of tabs for a single package.

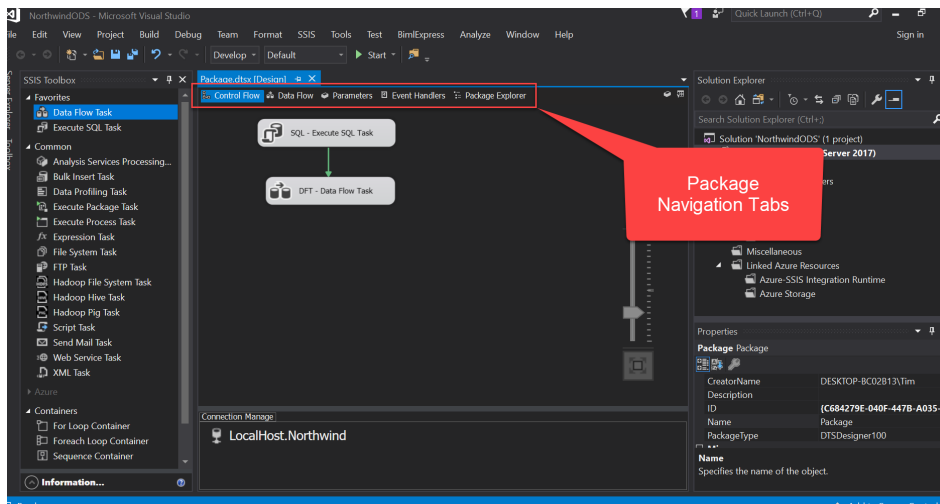


Figure 2: SSIS Package Navigation

The first four tabs listed—Control Flow, Data Flow, Parameters, and Event Handlers—are all essential parts of navigating an SSIS package. We'll dive into each of these tabs later in this article.

To start the execution of an SSIS package in SSDT, you can use one of three methods:

- With the package open and in focus in the designer, click the Start button in the menu bar at the top
- With the package open and in focus in the designer, press the F5 key
- Right-click the package in the Solution Explorer and choose the Execute Package option (note that this option works regardless of whether the package is open or in focus in the designer)

THE MOVING PARTS OF SSIS

Within each SSIS package, there are several fundamental moving parts.

Connection Managers

A connection manager defines the data structures that can be read from or written to during the execution of the package. A connection manager can refer to a relational database, a flat file, a database file (such as Microsoft Access), a web service, or a cloud structure (such as blob storage).

Once defined inside an SSIS package, a connection manager can be used by any of the tasks or components within that package. Think of the connection manager as a data gateway; it can be used to retrieve data (source), to write data (destination), and to manipulate or validate the data as it passes through the package logic (transformation). In fact, the same connection manager could be used as source, destination, and transformation within the same package.

Connection managers are unique in that they can be defined at the package level or at the project level. Package and project connections have the exact same functionality; the only difference is the scope of visibility. Package connections are usable within the package in which they are created, while project connections are accessible to any package in that project. The latter is handy when creating a project using the same connection in multiple packages.

As shown in Figure 3, both the package and project connections show up in the Connection Managers tray, and any project connections are also shown in the Solution Explorer under the Connection Managers folder. In the Connection Managers tray, any project-scoped connection will display a (project) prefix in the name.

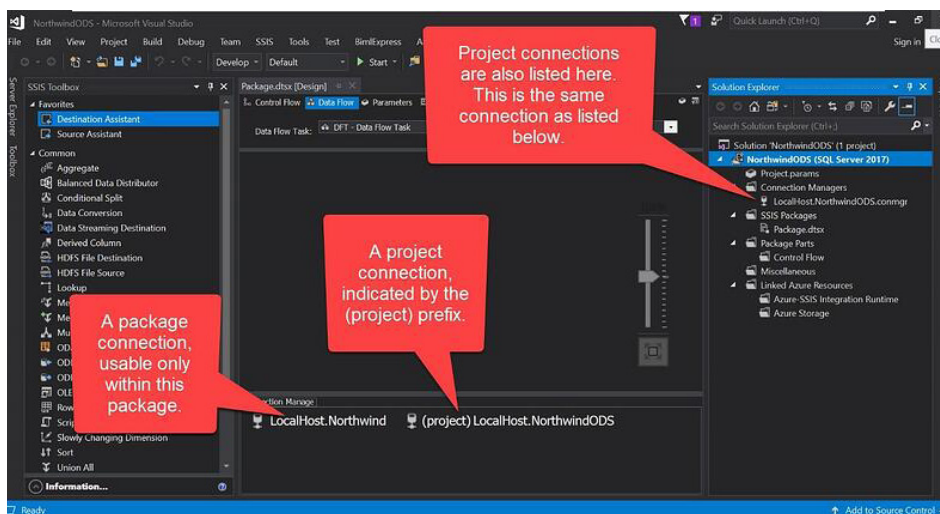


Figure 3: Package-Level and Project-Level Connection Managers

Control Flow and Data Flow

The Control Flow and Data Flow tabs represent the workspace in which most of the package logic resides. We'll go into greater detail on each of these tabs in the next section.

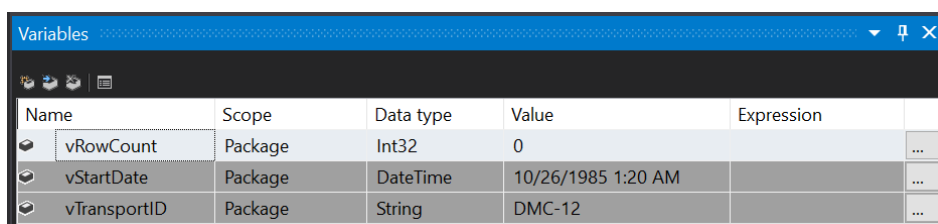
Variables

Like any good programming environment (and yes, SSIS is a programming environment), SSIS allows user-defined variables that can be statically set or dynamically manipulated during package runtime.

Variables are defined at the package level and, by default, are accessible by any task or component within the package. Each variable will have the following attributes:

- **A name:** For obvious reasons.
- **A scope:** By default, each variable is scoped at the package level. You can optionally limit the scope to a task or container. However, it's a rare situation that would require the variable scope to be changed.
- **A data type:** You'll find everything in SSIS is strongly typed, and variables are no exception. Each variable is created with a specific data type, and any values written to that variable must honor the constraints of that variable. For example, the package will fail if you try to load a value of February 31 to a DateTime variable or if you mistakenly try to assign the value "apple" to an Int32 variable. Each of the SSIS variable data types has an analog in SQL Server (e.g., Int32 corresponds to the INT in SQL Server and the string data type aligns with NVARCHAR).
- **A value:** For some data types (such as the string data type), the value can be left blank. For others (such as DateTime or any of the integer types), there must be a default value provided.
- **An expression (optional):** Variables can be set to a static value or can be configured to use an expression for more dynamic behavior.

The Variables window, shown in Figure 4, is used to create or edit variable values.



Name	Scope	Data type	Value	Expression
vRowCount	Package	Int32	0	
vStartDate	Package	DateTime	10/26/1985 1:20 AM	
vTransportID	Package	String	DMC-12	

Figure 4: The SSIS Variables Window

Variables in SSIS provide a great deal of flexibility when designing SSIS packages. Variables can be used to store operational values (such as row counts), configuration values (such as a particular file name when looping through multiple files), and can even be used to process serialized object data.

Parameters

Parameters were introduced in SSIS with the release of SQL Server 2012 to easily pass in runtime values to a package. Using SSIS parameters makes a project much more flexible by eliminating the reliance on hard-coded values.

Parameters behave much like SSIS variables. They both require a data type definition, can both be set with a default value, and can reduce administrative work by eliminating hard-coded static values within a package. There are some important differences, including:

- Parameters can be scoped at either the package level or the project level, but the scope of a variable only goes as high as the package.
- Parameters can be marked as Sensitive, in which case the value of that parameter will be encrypted when the package or project is saved. Variable values are always stored in plain text.
- Parameters can be set to Required, which will prevent the package from executing if a value is not supplied at runtime. There's no such behavior with variables.
- Parameters cannot use expressions to define their values, but variables can.
- The value of a parameter, once set, remains static for the duration of that execution and cannot be changed after the execution starts. The value of a variable can be programmatically changed at any time (and as many times as necessary) during package execution.

Parameters are incredibly powerful in that they allow you to completely decouple values that might change over time, including database connection strings, file paths, and login credentials, from the source code. By using parameters to pass in these values at runtime, you can eliminate too many unnecessary code changes. Parameterizing values such as these also makes testing easier, by allowing a separate set of test values to be passed in at runtime with no code modification.

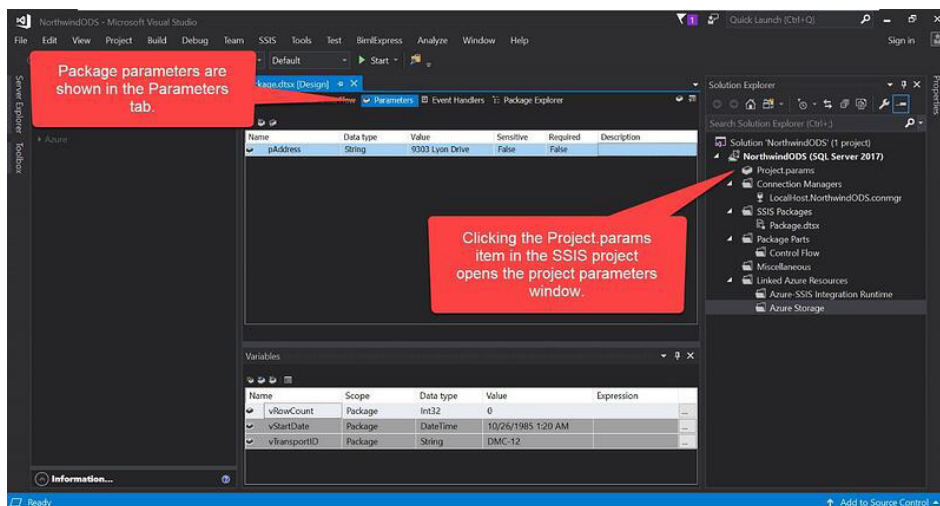


Figure 5: The Parameters Tab

To create or edit package parameters, open the package and click the Parameters tab. Project parameters can be found by opening the Project.params file in the SSIS project.

Expressions

SSIS has its own expression language for creating dynamic behaviors. Almost every task and component in SSIS can use expressions for replacing otherwise static values with a short bit of code that is interpreted at runtime.

Some of the practical uses for expressions in SSIS include:

- Replacing a static query with one that uses a customized WHERE clause
- Replacing an output file name with one that includes the current date and time in that file name
- Creating a variable that uses an expression to concatenate several other variable values together (e.g., a directory path concatenated to a relative file name)
- Substituting a hard-coded database connection string with an expression that uses a parameter, allowing the connection string to be supplied at runtime

Expressions can be used throughout SSIS. Many properties of tasks, components, and containers can be modified to substitute expression values for static text. Expressions can be used in the value expression for a variable, and an expression can reference other variables, parameters, or environment information (such as the date and time the current package began executing).

To view the expressions available for any given task or component in SSIS, you can select the item (indicated by the arrow) and find the Expressions item in the Properties window (shown inside the rectangle), as shown in Figure 6.

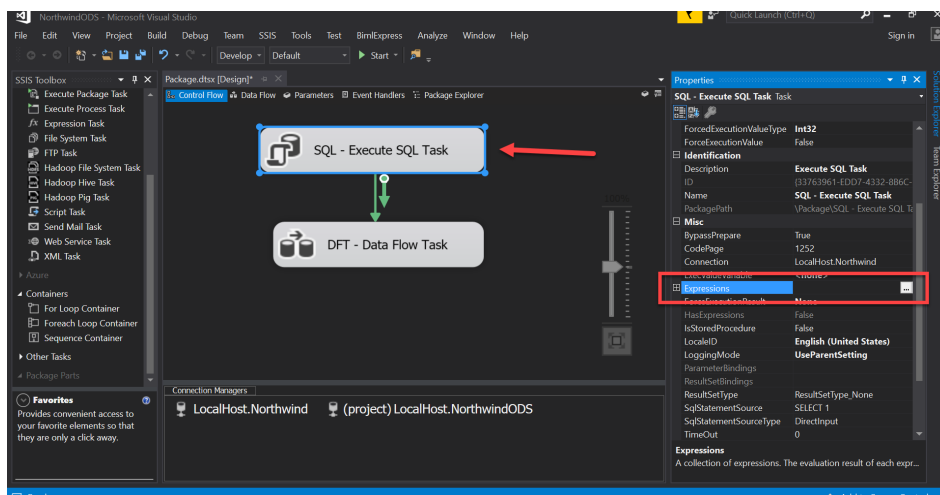


Figure 6: Expressions

After selecting the item and the property in which the expression will apply, you can use the Expression Builder window, shown in Figure 7, to design the expression. The list of variables and parameters (shown in the upper-left pane) and functions (shown in the upper-right pane) can help when building your expression. For the daring, you can code your expression entirely by hand in the Expression box.

Expressions are incredibly powerful in SSIS, but they do take a bit of getting used to. Don't feel alone if you initially struggle with the syntax of the SSIS expression language; every SSIS developer who has ever worked with it has had moments of frustration when adapting to this new lingo. Use the Expression Builder to get acquainted with the nuance of creating SSIS expressions.

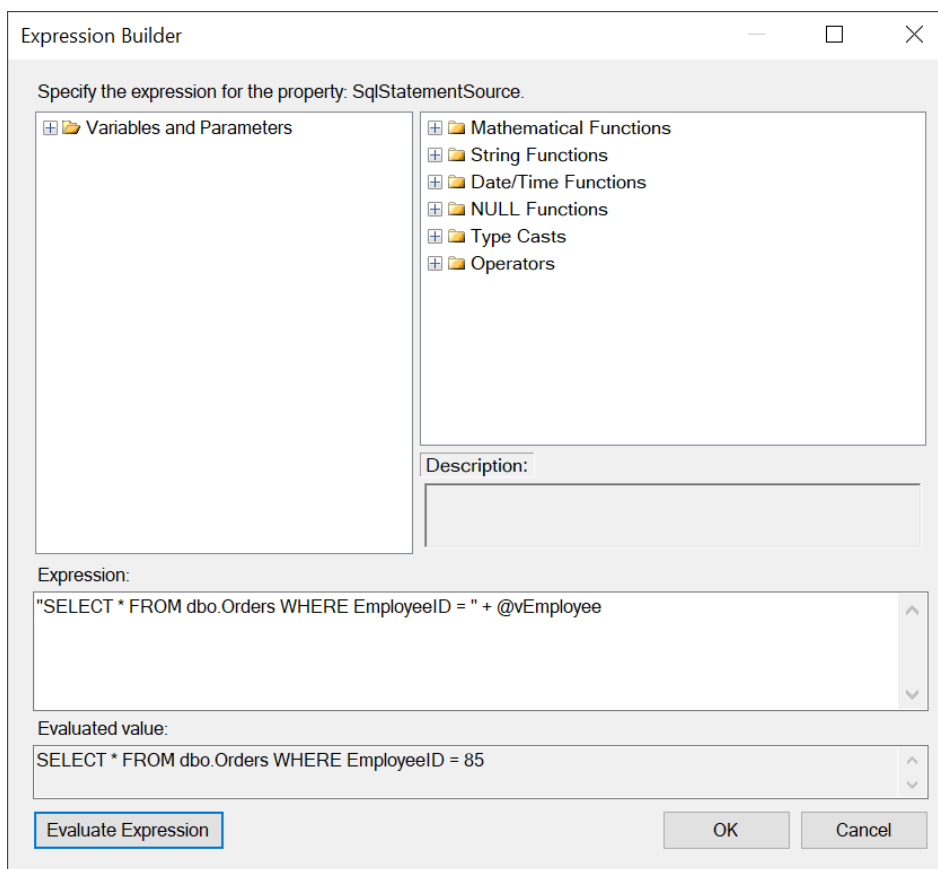


Figure 7: Expression Builder

THE SSIS CONTROL FLOW

The control flow in SSIS is the starting point at which package development begins. The control flow is the design surface onto which tasks are added. The SSIS control flow is responsible for directing which tasks will execute and in what order.

The control flow is the default view when creating or opening a package. The Control Flow tab, at the top left of an opened package, reveals the control flow surface. On a new package, this will be blank; Figure 8 shows a package with two tasks added to the control flow.

When working with the SSIS control flow, there are three different types of objects you'll use: tasks, containers, and precedence constraints.

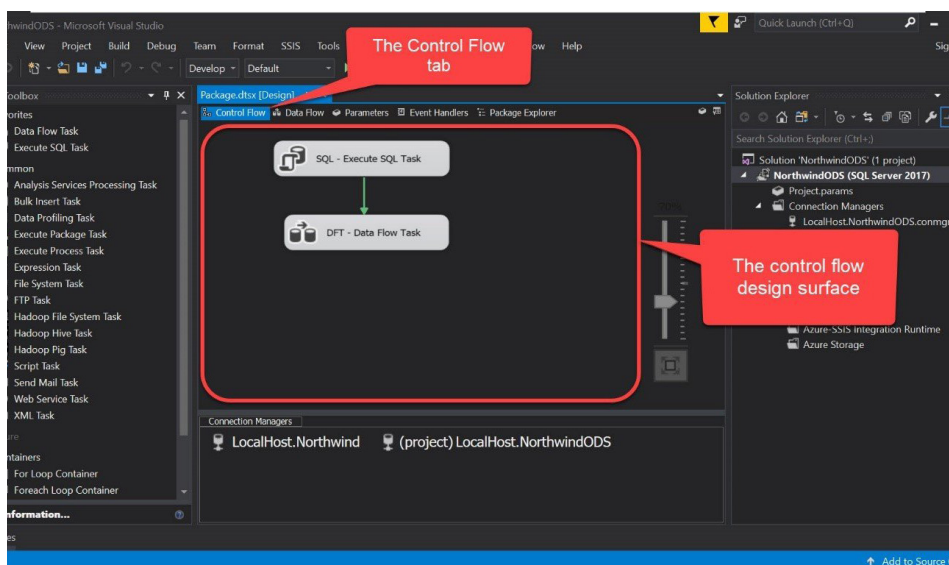


Figure 8: The SSIS Control Flow

SSIS Control Flow Tasks

Simply put, an SSIS task is a single operation in a package. A single package can contain many tasks or, in some cases, just one.

SSIS has dozens of built-in task types designed to handle almost any ETL need. When the Control Flow tab is selected, the SSIS Toolbox (docked on the left in Figure 8) lists all the available tasks. The following are some of the most used tasks:

- **Data Flow Task:** This is the most useful task in the SSIS Toolbox, as it contains dozens of data transformation components. We'll go through the Data Flow task in more detail in the next section.
- **Execute SQL Task:** This task allows the execution of SQL code against a relational database.
- **File System Task:** This task is used for interacting with the file system, including copying, moving, or deleting files and directories.

- **Execute Package Task:** Packages can invoke other packages by using the Execute Package task.
- **Execute Process Task:** This task allows you to trigger logic in an executable or batch file.
- **Script Task:** For situations requiring more flexibility than what you find in the built-in tasks, you can use the Script task to create highly customized ETL logic using C# or VB.NET.

Although there are numerous other tasks for less common operations, the tasks in this list can handle the vast majority of ETL needs.

SSIS Control Flow Containers

Included in the SSIS control flow is a set of three containers that can be used to bind together and loop through sets of tasks.

- **Sequence Container:** This container is the simplest of the container objects and is meant to logically group together two or more tasks.
- **For Loop Container:** This container will execute continuously for as long as a predefined condition remains true.
- **Foreach Loop Container:** This container loops through a discrete list of things, such as a list of files in a specified directory.

The foreach loop is especially useful when processing different but identically shaped sets of data; for example, when ingesting or exporting flat file data. Figure 9 shows an example of a foreach loop container configured to loop through each CSV file in the given directory.

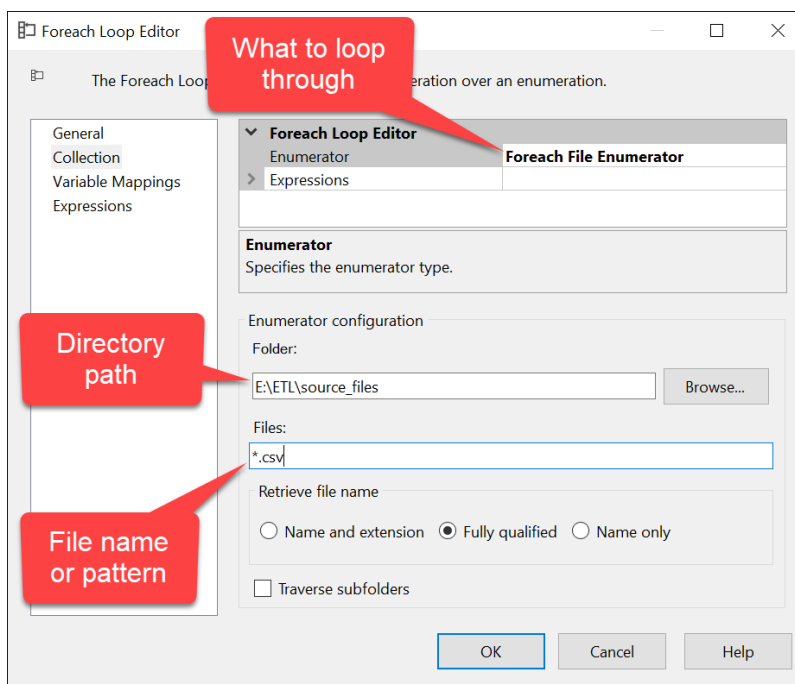


Figure 9: Foreach Loop Container for Processing Files

SSIS Control Flow Precedence Constraints

If tasks and containers are the building blocks of the SSIS control flow, then precedence constraints are the glue binding them together. Precedence constraints define which tasks are to be executed, the order in which those tasks are invoked, and whether tasks should be run serially or in parallel. If no precedence constraints are defined, all the tasks and containers in that package will be invoked at the same time.

At first glance, a precedence constraint looks like a simple connector. As Figure 10 shows, the precedent constraint indicates that the Execute SQL task will be started first, followed by the Data Flow task.

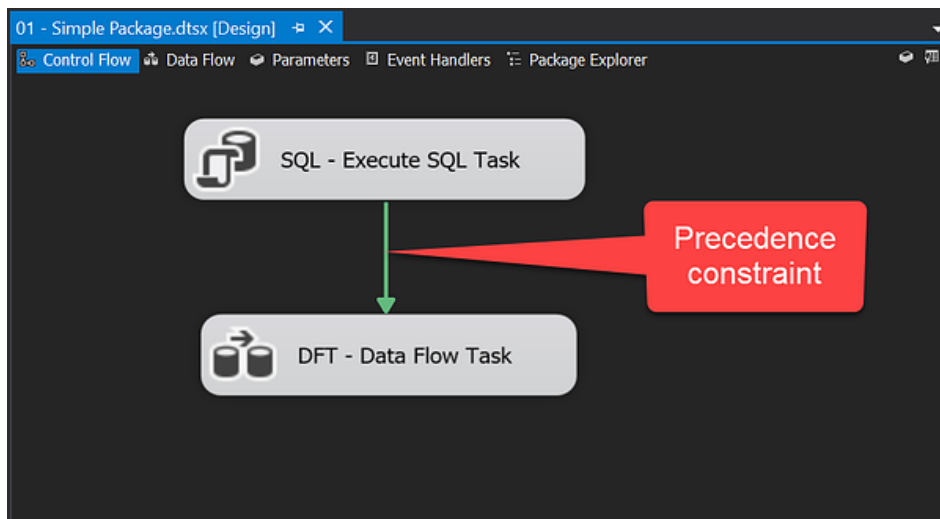


Figure 10: Simple Precedence Constraint

However, the precedence constraint can do much more than simply define the order of operations. Double-clicking the precedence constraint opens the editor for that object (shown in Figure 11), revealing more configuration options.

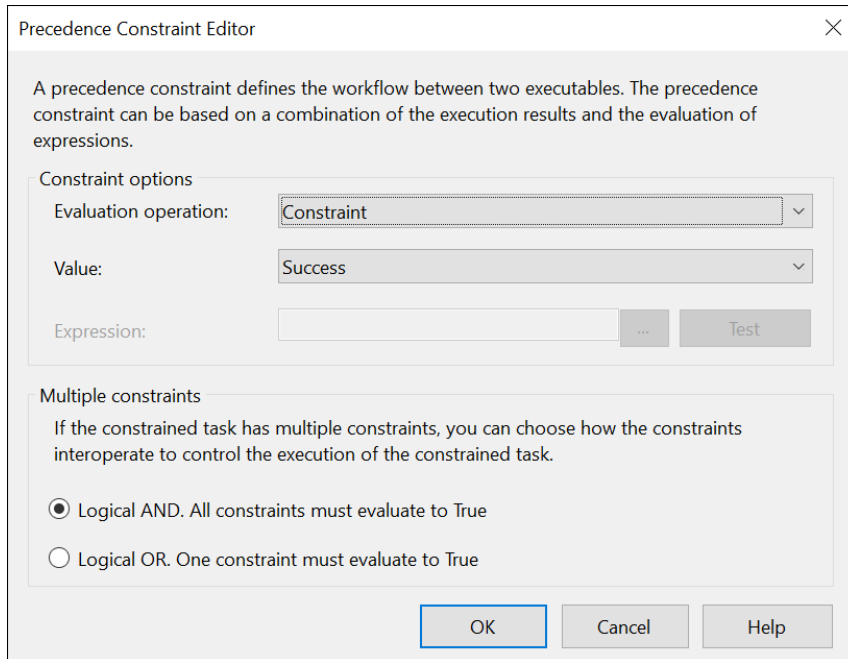


Figure 11: Precedence Constraint Editor

The configuration options available include:

- **Evaluation operation:** This option allows you to set this precedence constraint to be used solely as a constraint (the default) or to be bound with an expression.
- **Value:** This option sets the type of constraint. By default, the constraint is set to Success, meaning the downstream task will execute only after a successful execution of the upstream task. You can choose Failure or Completion if needed.
- **Expression:** If the evaluation operation is set to use an expression as part of the constraint, you'll define the expression in this field.
- **Multiple constraints behavior:** If the downstream task has multiple constraints bound to it, this selection defines whether all those constraints must be met (the default) or if only one of them must be met before that task is executed.

Figure 12 shows an example of using multiple precedence constraint for both success and failure operations.

The setup shown in Figure 12 results in the following workflow:

- DFT - Data Flow Task 1 and DFT - Data Flow Task 2 will start at the same time
- If both DFT - Data Flow Task 1 and DFT - Data Flow Task 2 complete successfully, the Sequence Container will then be started
- When the Sequence Container starts, DFT - Data Flow Task 3 and DFT - Data Flow Task 4 will start at the same time, since there's no precedence constraint between them
- If DFT - Data Flow Task 1 fails, DFT - Data Flow Task 5 will start

Using containers and precedence constraints, the workflow of the package can be as simple or as complex as needed.

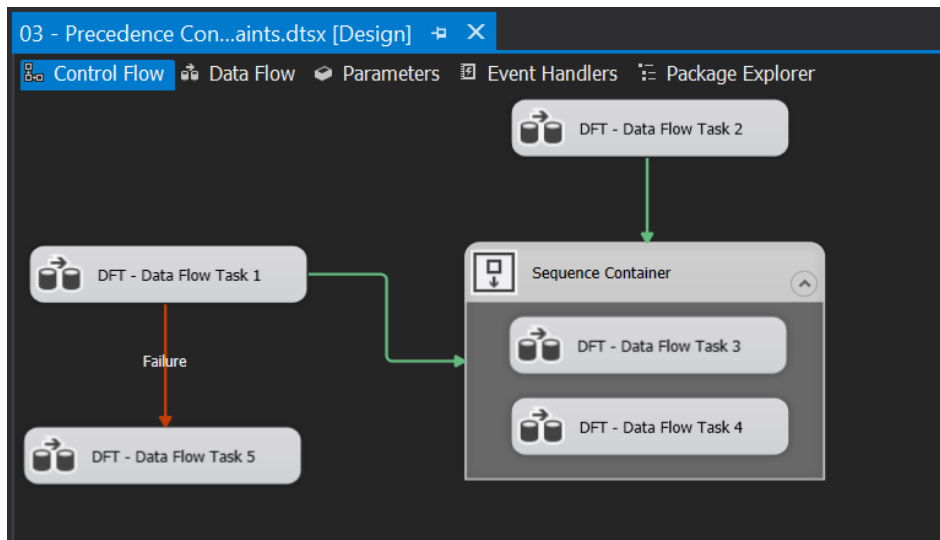


Figure 12: Precedence Constraint Example

Event Handlers

Event handlers are a special type of SSIS control flow. An event handler is designed to execute when a specific event occurs. Figure 13 shows an event handler set to execute when the executable (in this case, the package itself) encounters an error (defined as OnError, which is the default setting).

Note that, in this example, the event handler logic has not yet been set up; to create an event handler for this executable and this event, you can click the hyperlinked message in the middle of the design surface. The SSIS Toolbox for this tab shows all the tasks commonly available in the control flow.

Event handlers are useful tools but be careful that you don't overuse them. Because the event handler logic can be set up individually for every task and container (as well as for the package itself), the overuse of event handlers creates a labyrinth of logic that is difficult to maintain and debug.

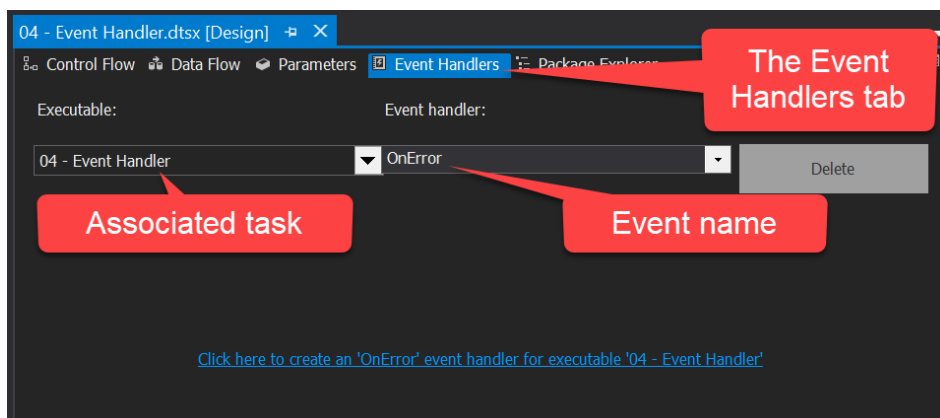


Figure 13: Event Handler

THE SSIS DATA FLOW

As mentioned earlier, the Data Flow task is one of many tasks in the SSIS Toolbox. However, this task differs significantly from the other tasks in that it has its own child elements, referred to as components, that can be connected to create end-to-end data flow operations.

The data flow can be accessed through the Data Flow tab in the package editor. If a package contains multiple data flows, you can use the Data Flow Task drop-down menu, shown in Figure 14, to select the data flow to edit.

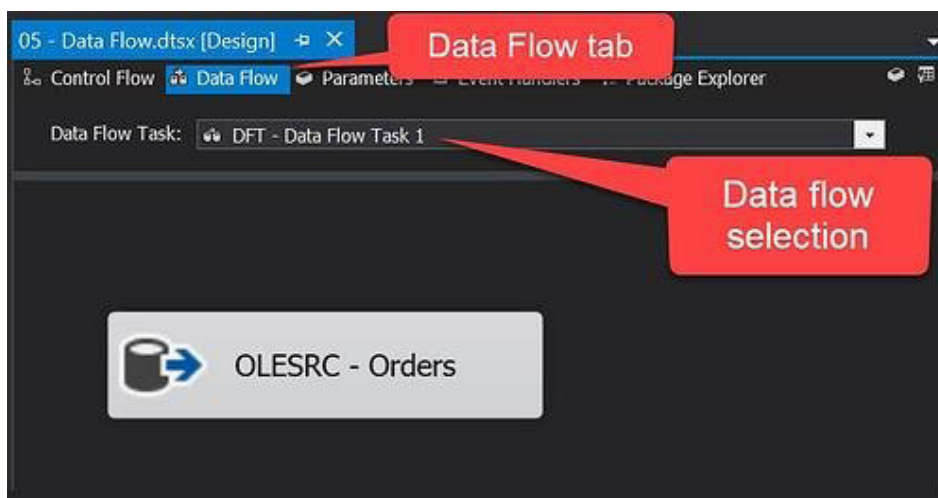


Figure 14: Data Flow Layout

When switching from the Control Flow tab to the Data Flow tab, you'll notice the list of items in the SSIS Toolbox changes, showing only the data flow components when the Data Flow tab is selected. The following are a few of the many components available:

- **OLE DB Source and OLE DB Destination:** Used for retrieving data from or writing data to relational data stores using the Microsoft OLE DB driver
- **ODBC Source and ODBC Destination:** Used for retrieving data from or writing data to an ODBC connection
- **Flat File Source and Flat File Destination:** Used for reading from or writing to flat files
- **Lookup:** Used for comparing one set of data to another for validation or key retrieval
- **Derived Column:** Useful for performing lightweight cleansing operations, data type changes, or combining or splitting string values
- **Merge Join:** Much like a join operation in the database engine, this component takes two streams of input and performs an inner, outer, or left join
- **Union All:** Brings together two or more inputs of data into a single stream, similar to the UNION ALL operator in T-SQL

- **Conditional Split:** Used for logically separating one set of data into two or more sets
- **Sort:** Performs a sort operation against a set of data in the data flow
- **Script Component:** A versatile scripting environment for creating customized transformation logic in C# or VB.NET

Each of the SSIS data flow components is classified into one of three categories: sources, destinations, and transformations. Most data flows consist of at least one source and one destination, and optionally contain one or more transformations. Figure 15 shows an example of a simple data flow design with an OLE DB source connecting to a SQL Server table, using a derived column transformation to trim the whitespace from each character field and then writing the data to a flat file destination.

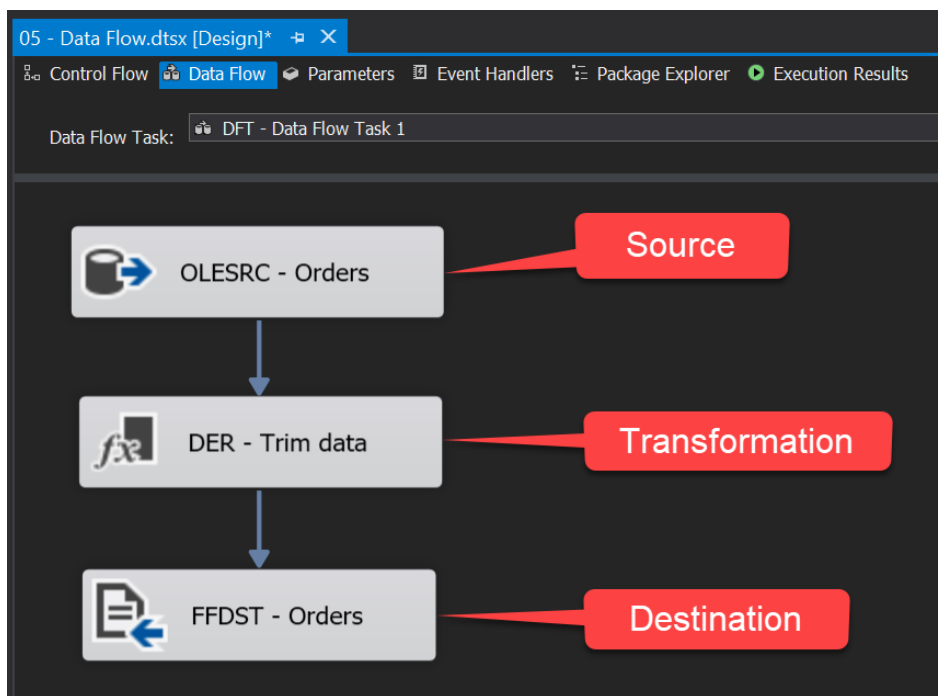


Figure 15: Simple Data Flow Layout

Not every Data Flow task will be this simple, of course. Complex problems require more complex solutions, and it's not uncommon for there to be a dozen or more transformations between the source and destination. Using a transformation such as the conditional split or the lookup transformation allows data to be split into multiple outputs, which can add even more complexity to the data flow. The bottom line is that a Data Flow task can be as simple or as complex as needed.

ABOUT SOLARWINDS

SolarWinds (NYSE:SWI) is a leading provider of powerful and affordable IT management software. Our products give organizations worldwide—regardless of type, size, or complexity—the power to monitor and manage their IT services, infrastructures, and applications; whether on-premises, in the cloud, or via hybrid models. We continuously engage with technology professionals—IT service and operations professionals, DevOps professionals, and managed services providers (MSPs)—to understand the challenges they face in maintaining high-performing and highly available IT infrastructures and applications. The insights we gain from them, in places like our **THWACK** community, allow us to solve well-understood IT management challenges in the ways technology professionals want them solved. Our focus on the user and commitment to excellence in end-to-end hybrid IT management has established SolarWinds as a worldwide leader in solutions for network and IT service management, application performance, and managed services. Learn more today at www.solarwinds.com.



For additional information, please contact SolarWinds at 866.530.8100 or email sales@solarwinds.com.
To locate an international reseller near you, visit http://www.solarwinds.com/partners/reseller_locator.aspx

© 2021 SolarWinds Worldwide, LLC. All rights reserved

The SolarWinds, SolarWinds & Design, Orion, and THWACK trademarks are the exclusive property of SolarWinds Worldwide, LLC or its affiliates, are registered with the U.S. Patent and Trademark Office, and may be registered or pending registration in other countries. All other SolarWinds trademarks, service marks, and logos may be common law marks or are registered or pending registration. All other trademarks mentioned herein are used for identification purposes only and are trademarks of (and may be registered trademarks) of their respective companies.

This document may not be reproduced by any means nor modified, decompiled, disassembled, published or distributed, in whole or in part, or translated to any electronic medium or other means without the prior written consent of SolarWinds. All right, title, and interest in and to the software, services, and documentation are and shall remain the exclusive property of SolarWinds, its affiliates, and/or its respective licensors.

SOLARWINDS DISCLAIMS ALL WARRANTIES, CONDITIONS, OR OTHER TERMS, EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, ON THE DOCUMENTATION, INCLUDING WITHOUT LIMITATION NONINFRINGEMENT, ACCURACY, COMPLETENESS, OR USEFULNESS OF ANY INFORMATION CONTAINED HEREIN. IN NO EVENT SHALL SOLARWINDS, ITS SUPPLIERS, NOR ITS LICENSORS BE LIABLE FOR ANY DAMAGES, WHETHER ARISING IN TORT, CONTRACT OR ANY OTHER LEGAL THEORY, EVEN IF SOLARWINDS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.