SSIS

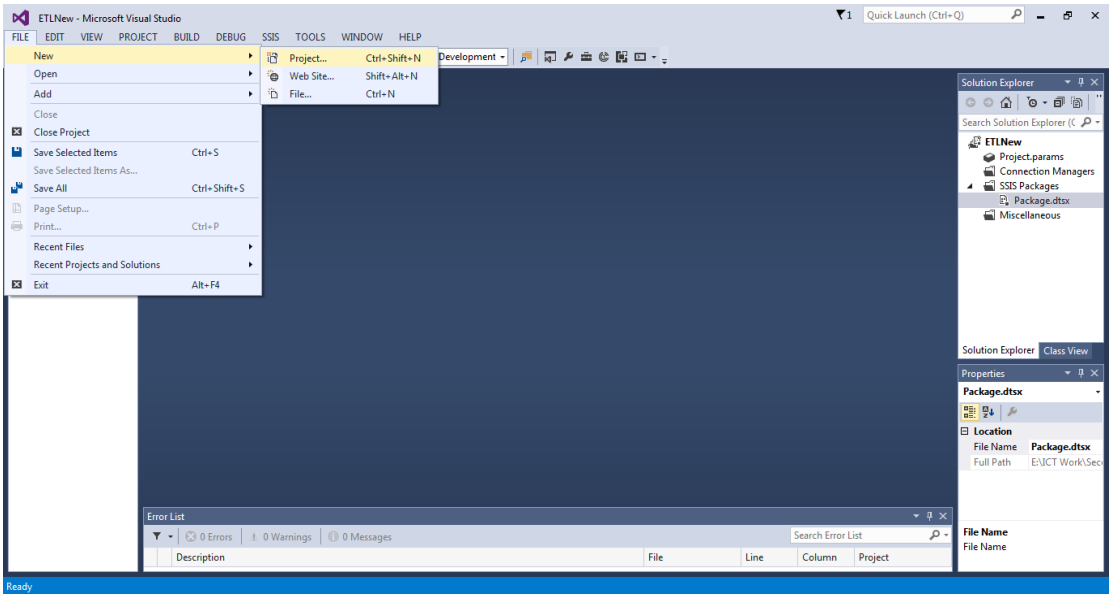
Table of Contents

1. Introduction

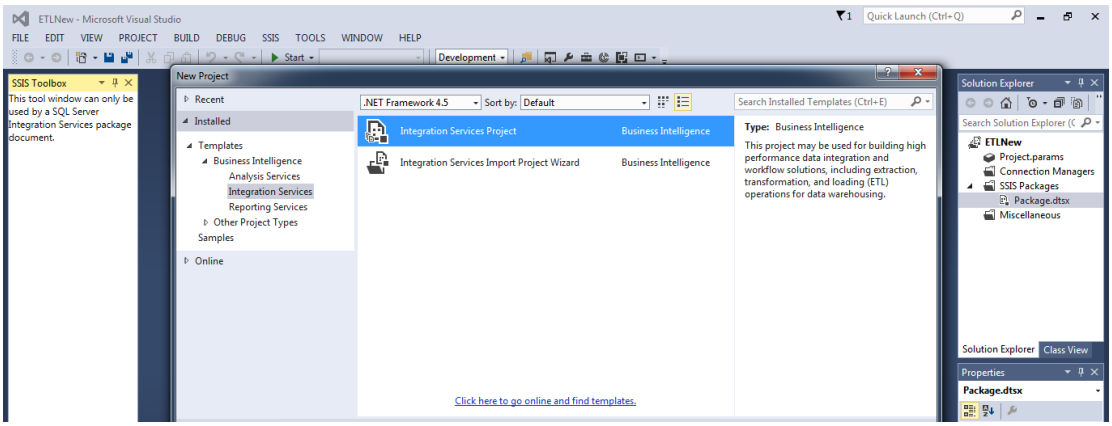
- The SQL Server Data Tools (SSDT) was previously called Business Intelligence Development Studio (BIDS) in SQL Server 2008, and it is the central environment in which you'll spend most of your time as an SSIS developer. SSDT is just a specialized use of the familiar Visual Studio development environment.
- 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.

2. Create SSIS Project

Run SQL Server Data Tools→ File→ New→ Project



From Business Intelligence →Integration Services → Integration Services Project



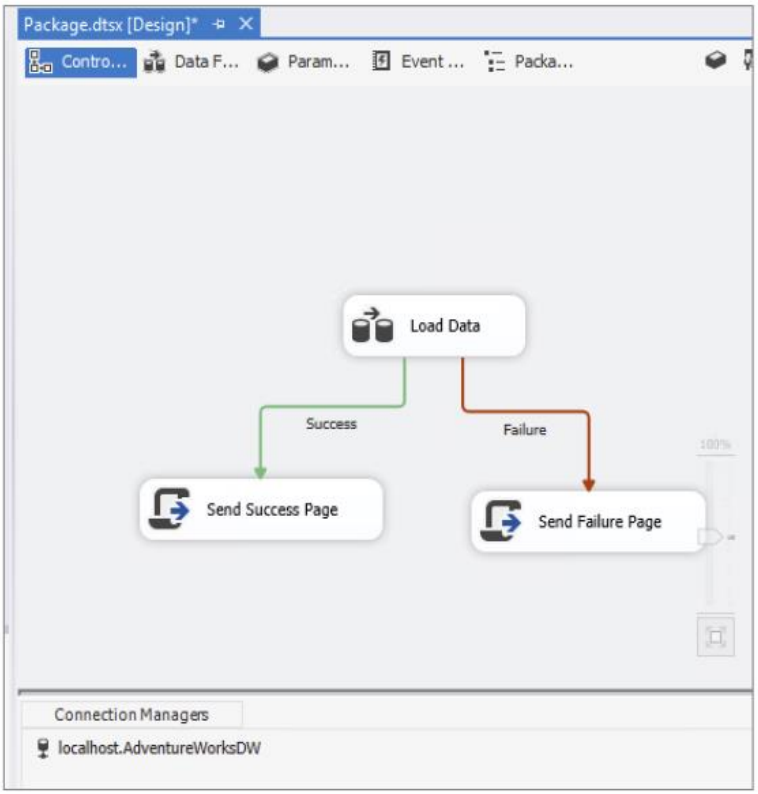
You can set the name, location and solution name from the menu:

3. SSIS Architecture

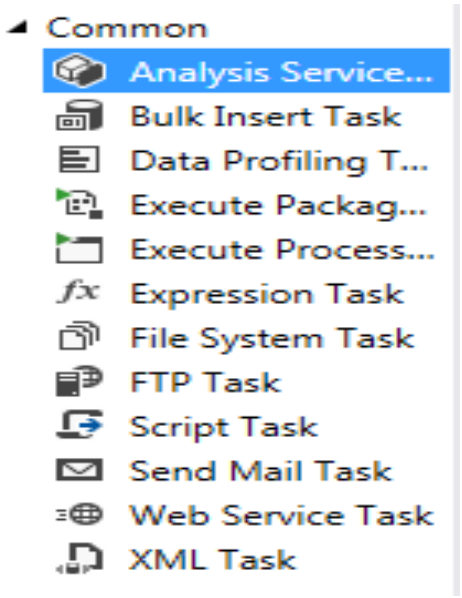
- Microsoft has truly established SSIS as a major player in the extraction, transformation, and loading (ETL) market. Not only is the SSIS technology a complete code rewrite from SQL Server 2000 DTS, it now rivals other third-party ETL tools that can cost hundreds of thousands of dollars depending on 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.

- **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.



- **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.



- *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.
-

- **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.
- To set up a precedence constraint between two tasks, you must set the constraint value; optionally, you can set an expression.
- 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.

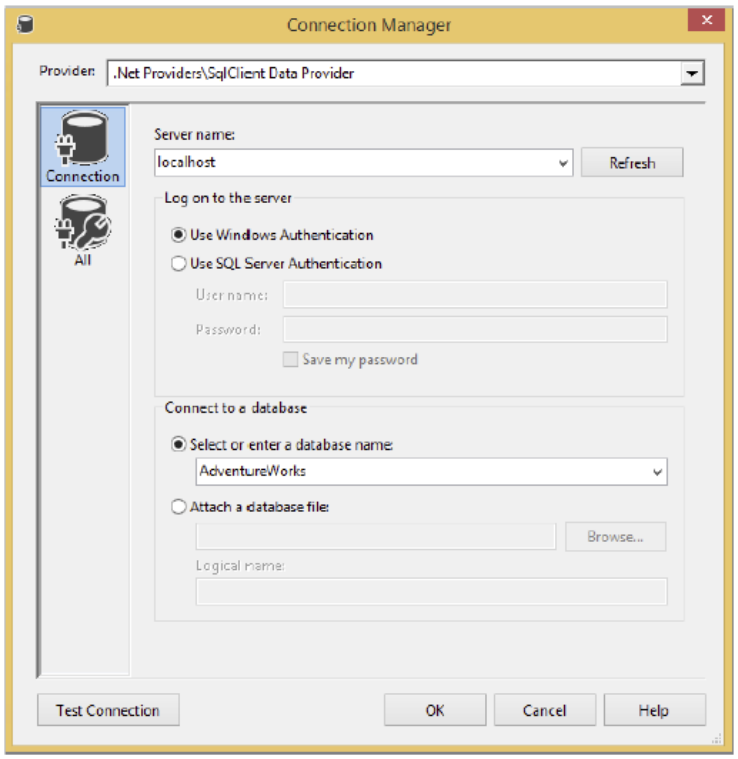
- **Containers**
  - Containers are core units in the SSIS architecture for grouping tasks together logically into units of work. Besides providing visual consistency, containers enable you to define variables and event handlers (these are discussed in a moment) within the scope of the container, instead 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.
  - *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.

- **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.



- **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 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.

- **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 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 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 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.