**SSIS Definitions**

|  |  |
| --- | --- |
| SSIS Expression Cheat Sheet | |
| **Problems** | **Expression** |
| Create a file name with today's date | **Expression on the Flat File or File Connection Manager:**  "C:\\Project\\MyExtract" + (DT\_WSTR, 30)(DT\_DBDATE)GETDATE() + ".csv" **Expression Output Example:** C:\Project\MyExtract2009-03-20.csv |
| Use a 2 digit date  (ex. "03" for March instead of "3") | RIGHT("0" + (DT\_WSTR, 2)MONTH(GETDATE()),2) **Expression Output**: 03 (if the month is March) |
| Multiple condition if statement | **In this example, the statement determines that if the ColumnName is blank or NULL, it will be set to unknown. To make a Logical AND condition, use "&&" instead of the "||" operator.**  ISNULL(ColumnName)||TRIM(ColumnName)==""?"Unknown": ColumnName |
| Returns the first five characters from a zip code | **Derived Column Transform in the Data Flow:**  SUBSTRING(ZipCodePlus4,1,5) |
| Remove a given character from a string (ex. Remove "-" from a social security number) | **Derived Column Transform in the Data Flow:**  REPLACE(SocialSecurityNumber, "-","") |
| Uppercase data | **Derived Column Transfrom in the Data Flow:**  UPPER(ColumnName) |
| Replace NULL with another value | **Derived Column Transform in the Data flow:**  ISNULL(ColumnName)?"New Value" : ColumnName |
| Replace blanks with NULL values | **Derived Column Transform in the Data Flow:**  TRIM(ColumnName)=="" ? (DT\_STR, 4, 1252)NULL(DT\_STR, 4, 1252) : ColumnName |
| Remove any non-numeric data from a column | **Script Transform in the Data Flow Task with the code as follows (VB 2008):**  Imports System.Text.RegularExpressions  Public Overrides Sub Input()\_ProcessInputRows(ByVal Row As Input()Buffer)       If Row.ColumnName\_IsNull = False Or Row.ColumnName = "" Then             Dim pattern As String = String.Empty            Dim r As Regex = Nothing            pattern = "[^0-9]"            r = New Regex(pattern, RegexOptions.Compiled)            Row.ColumnName = Regex.Replace(Row.ColumnName, pattern, "")        End If  End Sub |
| Convert text to proper case (ex. 1st letter in each word is uppercase) | **Script Transform with the line of partial code as follows:**  Row.OutputName = StrConv(Row.InputName, VBStrConv.ProperCase) |
| Build dynamic SQL statement | **Expression on the SQLStatementSource property of Execute SQL Task:**  "SELECT Column From " + @[User::TableName] +WHERE  DateFilterColumn = '" + (DT\_WSTR,4)YEAR(@  [User::DateTimeVar]) + RIGHT("0" + (DT\_WSTR,2)MONTH(@  [User::DateTimeVar]),2) + RIGHT("0" + (DT\_WSTR,2)DAY(@  [User::DateTimeVar]),2) + "'" |
| Round to the nearest two decimal mark | **Expression on Derived Column Transform:**  ROUND(YourNumber, 2) Expression Output Example: 1.2600000 |

|  |  |
| --- | --- |
| Common SSIS Problems and Solutions | |
| **Problems** | **Solutions** |
| Loop over a list of files & load each one | **Tasks Required:** Foreach Loop, Data Flow Task  **Solution:** Configure the Foreach Loop to loop over any particular directory of files. The loop should be configured to output to a given variable. Map the given variable to a connection manager by using expressions. |
| Conditionally executing tasks | **Solution:** Double-click the precedence constraint and set the Evaluation property to Expression and Constraint. Type the condition that you want to evaluate in the Expression box. |
| Pass in variables when scheduling or running a package | **Solution:** Use the /SET command in the DTExec command line or change the Property tab in the Package Execution Utility to have the property path like: \Package.Variables[User::VariableName].Properties[Value] |
| Move and rename the file at the same time | **Tasks Required:** File System Task  **Solution:** Set the File System task to rename the file and point to the directory you'd like to move the file to. This enables you to rename and move the file in the same step. |
| Loop over an array of data in a table & perform a set of tasks for each row | **Tasks Required:** Execute SQL Task, Foreach Loop  **Solution:** Use an Execute SQL Task to load the array and send the data into an object variable. Loop over the variable in a Foreach Loop by use an ADO Enumerator. |
| Perform an incremental load of data | **Tasks Required:** 2 Execute SQL Tasks, Data Flow Task  **Solution:** Have the 1st Execute SQL Task retrieve a date from a control table of when the target table was last loaded and place that into a variable. In the Data Flow Task, create a date range on your query using the variable. Then, update the control table using a 2nd Execute SQL Task to specify when the table was last updated. |
| Perform a conditional update & insert | **Components Required:** Data Flow Task, Conditional Split, Lookup Transform or Merge Join, OLE DB Command Transform  **Solution:** Use the lookup Transform or Merge Join to detemine if the row exists on the destination and ignore a failed match. If the row yields blank on the key, then you know the row should be inserted into the target (by Conditional Split). Otherwise, the row is a duplicate or an update. Determine if the row is an update by comparing the source value to the target value in the Conditional Split. The update can be done by an OLE DB Command Transform or by loading the data into a staging table. |
| Replace blanks with NULL values | **Derived Column Transform in the Data Flow:**  TRIM(ColumnName)=="" ? (DT\_STR, 4, 1252)NULL(DT\_STR, 4, 1252) : ColumnName |
| Remove any non-numeric data from a column | **Script Transform in the Data Flow Task with the code as follows (VB 2008):**  Imports System.Text.RegularExpressions  Public Overrides Sub Input()\_ProcessInputRows(ByVal Row As Input()Buffer)       If Row.ColumnName\_IsNull = False Or Row.ColumnName = "" Then             Dim pattern As String = String.Empty            Dim r As Regex = Nothing            pattern = "[^0-9]"            r = New Regex(pattern, RegexOptions.Compiled)            Row.ColumnName = Regex.Replace(Row.ColumnName, pattern, "")        End If  End Sub |
| Convert text to proper case (ex. 1st letter in each word is uppercase) | **Script Transform with the line of partial code as follows:**  Row.OutputName = StrConv(Row.InputName, VBStrConv.ProperCase) |
| Build dynamic SQL statement | **Expression on the SQLStatementSource property of Execute SQL Task:**  "SELECT Column From " + @[User::TableName] +WHERE  DateFilterColumn = '" + (DT\_WSTR,4)YEAR(@  [User::DateTimeVar]) + RIGHT("0" + (DT\_WSTR,2)MONTH(@  [User::DateTimeVar]),2) + RIGHT("0" + (DT\_WSTR,2)DAY(@  [User::DateTimeVar]),2) + "'" |
| Calculate beginning of the previous month | **Expression on Derived Column Transform:**  (DT\_DATE) (DT\_DBDATE)DATEADD("dd" , -1\*(DAY(GETDATE()) -1), DATEADD("month", -1, GETDATE())) |
| Round to the nearest two decimal mark | **Expression on Derived Column Transform:**  Expression on Derived Column Transform: ROUND(YourNumber, 2) **Expression Output Example:** 1.2600000 |

**IMPORTANT NOTES:**

* **12. How will you migrate an SSIS package from Development to Production environment?**  
  Do not include db connections and file paths in your workflow, instead create configuration files. This will help in deploying the pkg created in DEV server to Testing and finally to the PROD environment.  
  More Info: <http://msdn.microsoft.com/en-us/library/cc966389.aspx>  
  <http://www.wpconfig.com/2010/03/26/ssis-package-configurations/>

**CHECKPOINTS:**

**EXPRESSIONS:**

**VARIABLES:**

* **Variable mapping**

**PRESEDENCE CONSTRAINT EDITOR:**

* **Allows us to set advanced presedence configurations such as:**
  + **Success**
  + **Failure**
  + **Completion**

**COLLECTION ENUMERATOR EDITOR:**

* **Provides 7 different enumerators**

**CONTROL FLOW TASKS**

**FOREACH LOOP CONTAINER:**

* Allows us to loop thru files in folders or loop thru records in a result set.

**EXECUTE SQL TASK:**

* Allows us to execute a SQL query against a database to retrieve records, perform updates and more.

SQL Server Integration Services (SSIS) is a high performance data extraction, transformation and loading (ETL) tool included with SQL Server 2005 family of products. It can be thought of as the next generation of [Data Transformation Services](http://sqlserverpedia.com/wiki/Data_Transformation_Services_(DTS)_Overview) (DTS) - the tool available with the previous versions of SQL Server. In fact many SSIS executables retain the DTS acronym because the new name wasn't introduced until later stages of the product development cycle. However, it is important to note that SSIS is a completely new product and it is developed using .NET technologies. SSIS offers superior performance and considerably improved development, debugging and logging features compared to DTS. While DTS packages have to be developed using Enterprise Manager, SSIS packages will normally be built using Business Intelligence Development Studio (BIDS).   
  
  
  
The primary purpose of SSIS is to move data from one or multiple sources into a destination, cleansing and massaging data while it is in transit. However, SSIS also offers a multitude of additional tasks that are very useful for SQL Server database administrators. For example, it includes tasks for processing Analysis Services dimensions and measure groups, tasks for checking database integrity, rebuilding indexes and more.

SSIS supports numerous transformations that allow you to combine data originating from multiple sources, cleanse the data and give it the shape your data destination expects. Then you can import the data into a single or multiple destinations.

|  |  |  |
| --- | --- | --- |
| **Transformation** | **Description** | **Examples of when Transformation Would be Used** |
| Aggregate | Calculates aggregations such as SUM, COUNT, AVG, MIN and MAX based on the values of a given numeric column. This transformation produces additional output records. | Adding aggregated information to your output. This can be useful for adding totals and sub-totals to your output. |
| Audit | Includes auditing information, such as computer name where the package runs, package version ID, task name, etc in the data flow. | Creates advanced logs which indicate where and when the package was executed, how long it took to run the package and the outcome of execution. |
| Character Map | Performs minor manipulations on string columns. Converts all letters to uppercase, lowercase, reverse bytes, etc. | Applying string manipulations prior to loading data into the data warehouse. You can also apply the same manipulations to the data while it is being loaded into the warehouse. |
| Conditional Split | Accepts an input and determines which destination to pipe the data into based on the result of an expression. | Cleansing the data to extract specific rows from the source. If a specific column does not conform to the predefined format (perhaps it has leading spaces or zeros), move such records to the error file. |
| Copy Column | Makes a copy of a single or multiple columns which will be further transformed by subsequent tasks in the package. | Extracting columns that need to be cleansed of leading / trailing spaces, applying character map transformation to uppercase all data and then load it into the table. |
| **Data Conversion** | **Converts input columns from one data type to another.** | **Converting columns extracted from the data source to the proper data type expected by the data warehouse. Having such transformation options allows us the freedom of moving data directly from its source into the destination without having an intermediary staging database.** |
| Data Mining Query | Queries a data mining model. Includes a query builder to assist you with development of Data Mining eXpressions (DMX) prediction queries. | Evaluating the input data set against a data mining model developed with Analysis Services. |
| Derived Column | Calculates new column value based on an existing column or multiple columns. | Removing leading and trailing spaces from a column. Add title of courtesy (Mr., Mrs., Dr, etc) to the name. |
| Export Column | Exports contents of large columns (TEXT, NTEXT, IMAGE data types) into files. | Saving large strings or images into files while moving the rest of the columns into a transactional database or data warehouse. |
| Fuzzy Grouping | Finds close or exact matches between multiple rows in the data source. Adds columns to the output including the values and similarity scores. | Cleansing data by translating various versions of the same value to a common identifier. For example, "Dr", "Dr.", "doctor", "M.D." should all be considered equivalent. |
| Fuzzy Lookup | Compares values in the input data source rows to values in the lookup table. Finds the exact matches as well as those values that are similar. | Cleansing data by translating various versions of the same value to a common identifier. For example, "Dr", "Dr.", "doctor", "M.D." should all be considered equivalent. |
| Import Column | Imports contents of a file and appends to the output. Can be used to append TEXT, NTEXT and IMAGE data columns to the input obtained from a separate data source. | This transformation could be useful for web content developers. For example, suppose you offer college courses online. Normalized course meta-data, such as course\_id, name, and description is stored in a typical relational table. Unstructured course meta-data, on the other hand, is stored in XML files. You can use Import Column transformation to add XML meta-data to a text column in your course table. |
| Lookup | Joins the input data set to the reference table, view or row set created by a SQL statement to lookup corresponding values. If some rows in the input data do not have corresponding rows in the lookup table then you must redirect such rows to a different output. | Obtaining additional data columns. For example, the majority of employee demographic information might be available in a flat file, but other data such as department where each employee works, their employment start date and job grade might be available from a table in relational database. |
| Merge | Merges two sorted inputs into a single output based on the values of the key columns in each data set. Merged columns must have either identical or compatible data types. For example you can merge VARCHAR(30) and VARCHAR(50) columns. You cannot merge INT and DATETIME columns. | Combining the columns from multiple data sources into a single row set prior to populating a dimension table in a data warehouse. Using Merge transformation saves the step of having a temporary staging area. With prior versions of SQL Server you had to populate the staging area first if your data warehouse had multiple transactional data sources. |
| **Merge Join** | **Joins two sorted inputs using INNER JOIN, LEFT OUTER JOIN or FULL OUTER JOIN algorithm. You can specify columns used for joining inputs.** | **Combining the columns from multiple data sources into a single row set prior to populating a dimension table in a data warehouse. Using Merge Join transformation saves the step of having a temporary staging area. With prior versions of SQL Server you had to populate the staging area first if your data warehouse had multiple transactional data sources.   Note that Merge and Merge Join transformations can only combine two data sets at a time. However, you could use multiple Merge Join transformations to include additional data sets.** |
| Multicast | Similar to the conditional split transformation, but the entire data set is piped to multiple destinations. | Populating the relational warehouse as well as the source file with the output of a derived column transformation. |
| **OLEDB Command** | **Runs a SQL command for each input data row. Normally your SQL statement will include a parameter (denoted by the question mark), for example: UPDATE employee\_source SET has\_been\_loaded=1 WHERE employee\_id=?** | **Setting the value of a column with BIT data type (perhaps called "has\_been\_loaded") to 1 after the data row has been loaded into the warehouse. This way the subsequent loads will only attempt importing the rows that haven't made it to the warehouse as of yet.** |
| Percentage Sampling | Loads only a subset of your data, defined as the percentage of all rows in the data source. Note that rows are chosen randomly. | Limiting the data set during development phases of your project. Your data sources might contain billions of rows. Processing cubes against the entire data set can be prohibitively lengthy.   If you're simply trying to ensure that your warehouse functions properly and data values on transactional reports match the values obtained from your Analysis Services cubes you might wish to only load a subset of data into your cubes. |
| Pivot | Pivots the normalized data set by certain column to create a more easily readable output. Similar to PIVOT command in Transact-SQL. You can think of this transformation as converting rows into columns. For example if your input rows have customer, account number and account balance columns the output will have the customer and one column for each account. | Creating a row set that displays the table data in a more user-friendly format. The data set could be consumed by a web service or could be distributed to users through email. |
| Row count | Counts the number of transformed rows and store in a variable. | Determining the total size of your data set. You could also execute a different set of tasks based on the number of rows you have transformed. For example, if you increase the number of rows in your fact table by 5% you could perform no maintenance. If you increase the size of the table by 50% you might wish to rebuild the clustered index. |
| Row sampling | Loads only a subset of your data, defined as the number of rows. Note that rows are chosen randomly. | Limiting the data set during development phases of your project. Your data warehouse might contain billions of rows. Processing cubes against the entire data set can be prohibitively lengthy.   If you're simply trying to ensure that your warehouse functions properly and data values on transactional reports match the values obtained from your Analysis Services cubes you might wish to only load a subset of data into your cubes. |
| Script Component | Every data flow consists of three main components: source, destination and transformation. Script Component allows you to write transformations for otherwise un-supported source and destination file formats. Script component also allows you to perform transformations not directly available through the built-in transformation algorithms. | Custom transformations can call functions in managed assemblies, including .NET framework. This type of transformation can be used when the data source (or destination) file format cannot be managed by typical connection managers. For example, some log files might not have tabular data structures. At times you might also need to parse strings one character at a time to import only the needed data elements.    Much like Script Task the Script Component transformation must be written using Visual Basic .NET. |
| Slowly Changing Dimension | Maintains historical values of the dimension members when new members are introduced. | Useful for maintaining dimension tables in a data warehouse when maintaining historical dimension member values is necessary. |
| Sort | Sorts input by column values. You can sort the input by multiple columns in either ascending or descending order. The transformation also allows you to specify the precedence of columns used for sorting. This transformation could also discard the rows with duplicate sort values. | Ordering the data prior to loading it into a data warehouse. This could be useful if you're ordering your dimension by member name values as opposed to sorting by member keys.   You can also use Sort transformation prior to feeding the data as the input to the Merge Join or Merge transformation. |
| Term Extraction | Extracts terms (nouns and noun phrases) from the input text into the transformation output column. | Processing large text data and extracting main concepts. For example, you could extract the primary terms used in this section of SQLServerPedia by feeding the Term Extraction transformation the text column containing the entire section. |
| Term Lookup | Extracts terms from the input column with TEXT data type and match them with same or similar terms found in the lookup table. Each term found in the lookup table is scanned for in the input column. If the term is found the transformation returns the value as well as the number of times it occurs in the row. You can configure this transformation to perform case-sensitive search. | Analyzing large textual data for specific terms. For example, suppose you accept email feedback for latest version of your software. You might not have time to read through every single email messages that comes to the generic inbox. Instead you could use this task to look for specific terms of interest. |
| Union ALL | Combines multiple inputs into a single output. Rows are sorted in the order they're added to the transformation. You can ignore some columns from each output, but each output column must be mapped to at least one input column. | Import data from multiple disparate data sources into a single destination. For example, you could extract data from mail system, text file, Excel spreadsheet and Access database and populate a SQL Server table.   Unlike Merge and Merge Join transformations Union ALL can accept more than two inputs. |
| Unpivot | Opposite of Pivot transformation, Unpivot coverts columns into rows. It normalizes the input data set that has many duplicate values in multiple columns by creating multiple rows that have the same value in a single column.     For example if your input has a customer name and a separate column for checking and savings' accounts Unpivot can transform it into a row set that has customer, account and account balance columns. | Massaging a semi-structured input data file and convert it into a normalized input prior to loading data into a warehouse. |

This section lists various tasks available with SSIS and gives a brief description of each. We will also examine how SSIS tasks compare with tasks that were available in DTS with SQL Server 2000:

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Description** | **Counterpart in SQL Server 2000 DTS** | **New with SSIS** |
| Active X Script | Executes a script. This task is deprecated with SSIS and is intended solely for backward compatibility with DTS. For all new development you must use Script task. | Active X Script | N |
| Analysis Service Processing | Processes Analysis Services dimensions, measure groups, cubes databases. | Analysis Service Processing | N |
| Bulk Insert | Bulk loads data into SQL Server tables. | Bulk Insert | N |
| Execute SQL | Executes Transact-SQL statements or stored procedures. | Execute SQL | N |
| Execute Process | Executes a command line process; for example, a set of commands in a batch file. | Execute Process | N |
| File Transfer Protocol | Either uploads or downloads files using FTP. | File Transfer Protocol | N |
| Message Queue | Sends and receives string, variable and data file messages while executing the SSIS package. | Message Queue | N |
| Send Mail | Sends an email message. | Send Mail | N |
| Transfer Database | Copies database from one server / instance to another. | Transfer Database | N |
| Transfer Error Messages | Copies user defined error messages from one SQL Server instance to another. | Transfer Error Messages | N |
| Transfer Jobs | Transfers job definitions from one SQL Server instance to another. | Transfer Jobs | N |
| Transfer Logins | Transfers logins from one SQL Server instance to another. | Transfer Logins | N |
| Transfer SQL Server Object | Transfers any object from one SQL Server instance to another. | Copy SQL Server Object | N |
| Data Mining Query | Executes data mining prediction queries against Analysis Services. | Data Mining Prediction | N |
| Execute DTS 2000 Package | This task is deprecated and is intended solely for backward compatibility. Used specifically for executing DTS 2000 packages. | Execute Package | N |
| Transfer Master Stored Procedures | Transfers user-defined stored procedures which reside in master database from one instance to another. | Transfer Master Stored Procedures | N |
| **Data Flow Task** | **Transforms and/or transfer data from one data source to another. This is the most important task in SSIS.** | Transform Data | No, but it has changed significantly. |
| Backup Database | Backups one or multiple databases on a given instance of SQL Server. | None | Y |
| **Check Database Integrity** | **Checks integrity of a single or multiple SQL Server databases with DBCC CHECKDB statement.** | None | Y |
| Execute SQL Server Agent job | Executes a job on a given instance of SQL Server. | None | Y |
| Execute T-SQL Statement | Identical to Execute SQL task. | Execute SQL | N |
| Execute Package | Executes another SSIS package. | None | Y |
| History Cleanup | Deletes the maintenance plan history, job history and backup history for a given instance of SQL Server. | None | Y |
| Maintenance Cleanup | Deletes old backup files and maintenance plan report files. | None | Y |
| Notify Operator | Notifies SQL Server operator through email, pager or net send. | None | Y |
| Rebuild Index | Rebuilds indexes on one or multiple SQL Server tables and views. | None | Y |
| Reorganize Index | Reorganizes indexes on one or multiple SQL Server tables and views. | None | Y |
| Shrink Database | Executes DBCC SHRINKFILE on one or multiple SQL Server databases. | None | Y |
| Update Statistics | Executes UPDATE STATISTICS statement against chosen tables / views within one or multiple SQL Server databases. | None | Y |
| Analysis Services Execute DDL | Executes XML for Analysis (XMLA) statements to create, alter, drop or process Analysis Services objects. | None | Y |
| File System | Copies, moves, deletes or renames files and directories. This task is useful for reading all files within a given directory. | None | Y |
| Web Service | Executes a web method. | None | Y |
| XML | Applies XML operation to a previously retrieved XML document. | None | Y |
| WMI Data Reader | Reads Windows Management Instrumentation (WMI) data about the system. | None | Y |
| WMI Event Watcher | Queries and responds to WMI events. | None | Y |

**SSIS** is a platform for [data integration](http://en.wikipedia.org/wiki/Data_integration) and [workflow applications](http://en.wikipedia.org/wiki/Workflow_application). It features a fast and flexible [data warehousing](http://en.wikipedia.org/wiki/Data_warehouse) tool used for data extraction, transformation, and loading ([ETL](http://en.wikipedia.org/wiki/Extract,_transform,_load)). The tool may also be used to automate maintenance of SQL Server databases and updates to multidimensional [cube data](http://en.wikipedia.org/wiki/OLAP_cube).

First released with Microsoft SQL Server 2005, SSIS replaced [Data Transformation Services](http://en.wikipedia.org/wiki/Data_Transformation_Services), which had been a feature of SQL Server since Version 7.0. Unlike DTS, which was included in all versions, SSIS is only available in the "Standard" and "Enterprise" editions.

The SSIS Import/Export [Wizard](http://en.wikipedia.org/wiki/Wizard_(software)) lets the user create packages that move data from a single data source to a destination with no transformations. The Wizard can quickly move data from a variety of source types to a variety of destination types, including text files and other SQL Server instances.

Developers tasked with creating or maintaining SSIS packages use a visual development tool based on [Microsoft Visual Studio](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio) called the SQL Server [Business Intelligence Development Studio](http://en.wikipedia.org/wiki/Business_Intelligence_Development_Studio) (BIDS). It allows users to edit SSIS packages using a drag-and-drop user interface. A scripting environment in which to write programming code is also available in the tool. A package holds a variety of elements that define a workflow. Upon package execution, the tool provides color-coded, real-time monitoring.

Connections

A connection includes the information necessary to connect to a particular data source. Tasks can reference the connection by its name, allowing the details of the connection to be changed or configured at run time.

Tasks

A task is an atomic work unit that performs some action. There are a couple of dozen tasks that ship in the box, ranging from the file system task (which can copy or move files) to the data transformation task. The data transformation task actually copies data; it implements the ETL features of the product.

Precedence constraints

Tasks are linked by precedence constraints. The precedence constraint preceding a particular task must be met before that task executes. The run time supports executing tasks in parallel if their precedence constraints so allow. Constraints may otherwise allow different paths of execution depending on the success or failure of other tasks. **Together with the tasks, precedence constraints comprise the workflow of the package.**

Event handlers

A workflow can be designed for a number of events in the different scopes where they might occur. In this way, tasks may be executed in response to happenings within the package —such as cleaning up after errors.

Variables

Tasks may reference variables to store results, make decisions, or affect their configuration.

Parameters (SQL Server 2012 Integration Services)

Parameters allow you to assign values to properties withint packages at the time of package execution. You can project parameters and package parameters. In general, if you are deploying a package using the package deployment model, you should use configurations instead of parameters.

A package may be saved to a file or to a store with a hierarchical namespace within a SQL Server instance. In either case, the package content is persisted in [XML](http://en.wikipedia.org/wiki/XML).

Once completed, the designer also allows the user to start the package's execution. Once started, the package may be readily debugged or monitored.

**Features of the data flow task**

SSIS provides the following built-in transformations:

* Conditional Split
* Multicast
* Union-All, Merge, and Merge Join
* Sort
* Fuzzy Grouping
* Lookup and Fuzzy Lookup
* Percentage Sampling and Row Sampling Transformation.
* Copy/Map, Data Conversion, and Derived Column
* Aggregation
* Data Mining Model Training, Data Mining Query, Partition Processing, and Dimension Processing
* Pivot and Unpivot
* Slowly Changing Dimension
* Script Component
* Audit
* Cache Transform
* Export and Import Column
* OLE DB Command
* Row Count
* Term Extraction
* Term Lookup
* Row Sampling

The Conditional Split transformation is used to speed up the query on the source table based on a particular condition. It is similar to the "if..else" construct in the **C** language.

**Other included tools**

Aside from the Import/Export Wizard and the designer, the product includes a few other notable tools.

DTEXEC executes a package from the command line wherever it may be stored. Before running the package, the tool may be instructed to apply configuration information, which will allow the same package to be reused with slightly different parameters, including different connection strings for its endpoints.

DTUTIL provides the ability to manage packages, again from the [command prompt](http://en.wikipedia.org/wiki/Command_prompt). The tool can copy or move a package from a file into the server store, or back out again. Among a few other sundry functions, it can be used to delete, rename, encrypt, or decrypt packages.

The value of SSIS is as a workflow engine to move data from one spot to another with maybe some limited transformation and conditional branching along the way. If your packages contain a lot of script then your team is using SSIS for the wrong tasks or isn't comfortable with SQL or has bought into the hype. SSIS packages are very difficult to debug. Script components are an absolute nightmare and should be used only for formatting, looping, or as a last resort.

1. Keep your packages simple, sql tasks and data flow tasks.
2. Do as much work as possible outside of SSIS, preferably in SQL
3. Keep your variables in a single global scope
4. Keep your SQL in variables or store procedures, never in-line
5. Keep your variable values in a configuration store, preferably a SQL database

Integration Services tools. Those concepts include the following:

* Packages
* Control Flow
* Data Flow
* Connection Managers
* Package Configurations
* Property Expressions
* Data Sources and Data Source Views

A [**package**](http://www.microsoft.com/technet/prodtechnol/sql/2005/technologies/ssisvcs.mspx) presents a unit of work that addresses a business requirement. The package is the Integration Services object that you save, manage, or run. In SQL Server 2005 Integration Services introduces the concepts of control flow and data flow in packages. **A control flow consists of the tasks and containers. The tasks perform specific types of work such as executing SQL statements or sending email messages, and the containers define repeating subsets of the control flow or group subsets of the control flow to make the package easier to manage**. The tasks and containers are usually connected by precedence constraints that specify the sequence in which tasks and containers are executed and the conditions that must be satisfied to run the next task or container in the control flow. **A** [**data flow**](http://searchsystemschannel.techtarget.com/feature/%20http:/msdn2.microsoft.com/en-us/library/ms140080.aspx) **consists of sources that extract data, transformations that modify data, and the destinations that load the data into data stores.**

To connect to the data stores, a package uses **connection managers**. The connection managers are defined when you create the package. From the definition, the Integration Services runtime creates a connection at run time. When you construct a package, you configure properties of the connection managers, control flow and data flow items in the package, as well as the package itself. Frequently, a package must be configured differently for each environment to which you deploy it. For example, the connection string of connection managers may require updating to specify a different server, the location, the location of the data sources it accesses may change, and so forth. Integration Services provides package configurations to support this common scenario. **Package configurations make it possible to dynamically update properties at run time. A configuration is a name/value pair that maps a property and a value. The configurations are stored outside the package in XML files, Database Engine tables, variables, or Registry entries. When the package is run, the value from the configuration replaces the value of the mapped-to property within the package. The values of the properties are not changed permanently.**

You can set property values of packages and package objects in two different ways: directly by setting the value of each property, or indirectly by using property expressions. An expression, mapped to a property, is called a **property expression**. You build property expressions by using the operators and functions that the Integration Services expression language provides and variables. When the package is validated, which occurs when you save the package, the evaluation results of the property expressions replace the original values of properties.

**A** [**data source**](http://msdn2.microsoft.com/en-us/library/ms156450.aspx) **is a connection reference that you create and save outside a package, and then use as a source when adding new connection managers to a package. A data source represents a simple connection to a data store and therefore makes all tables and views in the data store available to the package.** A [**data source view**](http://searchsystemschannel.techtarget.com/feature/%20http:/msdn2.microsoft.com/en-us/library/ms170400.aspx) is built on a data source. It can contain only selected database objects and it can be extended with calculated columns that are populated by custom expressions, new relationships between tables, and queries. You can also apply a filter to a data source view to specify a subset of the data selected. In Integration Services, data sources and data source views are saved within the package definitions of the packages in which they are used.

**Checkpointing and Transaction Management:**

If a job fails part way through, SSIS records (in a local XML file) the point of failure, as well as all of the job variable values. This allows an operator to investigate the cause of failure (typically by examining logs), take corrective action, and then restart the job from point of failure. In the world of ETL, this scenario is not uncommon. On top of checkpointing, SSIS also makes it possible to contain multiple steps as part of a single transaction, so that if any step within the transaction fails, previous steps are rolled back. This is accomplished through tight integration with Microsoft's Distributed Transaction Controller - a component of the Windows operating system. I have experimented with this feature, and have got it to work successfully, although at first it was less than straightforward to configure. That said, I've never had a need for this feature. Also, this transaction management (from what I can tell) only works at the job level, and not at the transformation level, which is where it's more needed.

**Consolidated Logging:**

Consolidated logging, and ability to log to Windows Event Log. All ETL tools produce a single log. Most ETL tools allow for varying levels of verbosity, all from a single setting. SSIS goes one step further and nicely integrates with the Windows Event Viewer which if you're a Microsoft shop is HUGE benefit. As I mentioned earlier, logging was a sore spot for PowerShell. So the difference between SSIS and PowerShell is day-and-night.

**Consolidated Error Handling:**

Typically when a job fails (for any reason), you want your ETL job to fire off an e-mail to technical support. SSIS makes this very easy to do, as there is a single consolidated Error Handling job which you can develop for. Both PowerShell and Pentaho DI appear to lack this feature.

**C# and VB.NET scripting steps:**

Most ETL tools include a scripting step. This makes it possible to perform arbitrary transformations, as well as doing other tasks like adding or removing rows from the pipeline, or even adding and removing columns. SSIS allows you to script in either VB.NET or C# (C# is only supported in SSIS 2008 or later) which if you're already a .NET developer is a major benefit. Furthermore, it's significantly easier for that same code to me made into a standalone data flow step component (which gets you back to the visual metaphor). My only warning with SSIS's scripting step, is that I can easily see developers overly relying on it to perform the majority of their transformations. As I keep saying: the visual metaphor is the most powerful concept behind ETL, and dramatically lowers post-implementation costs, such as support costs, and the cost of impact analysis.

**Built-in text analytics steps:**

This is one of the first things I noticed about SSIS. Basically there are two Data Flow (i.e. transformation) steps which support basic text analytics. The first is a **keyword extractor. Namely, the step can be configured to extract relevant keywords from documents. These keywords can then be used to classify documents, making them easier to search against and report on. The second text analytic step allows you to search a given set of text against a list of keywords retrieved from a configurable database query. This step is very useful for doing things like sentiment analysis (i.e. determining if a block of text includes words like "cool" or "fun" versus words like "sucks" or "fail"), but it could also be used to flag sensitive information, such as people's names. if I had to choose, I prefer SSIS's text analytics functionality.**

Packages can include additional objects that provide advanced features or extend existing functionality, such as event handlers, configurations, logging, and variables.

**Event Handlers**

An event handler is a workflow that runs in response to the events raised by a package, task, or container. For example, you could use an event handler to check disk space when a pre-execution event occurs or if an error occurs, and send an e-mail message that reports the available space or error information to an [administrator](http://msdn.microsoft.com/en-us/library/ms141134(v=sql.105).aspx). An event handler is constructed like a package, with a control flow and optional data flows. Event handlers can be added to individual tasks or containers in the package. For more information, see [Integration Services Event Handlers](http://msdn.microsoft.com/en-us/library/ms140223(v=sql.105).aspx) and [Adding Package Event Handlers](http://msdn.microsoft.com/en-us/library/ms140011(v=sql.105).aspx).

**Configurations**

A configuration is a set of property-value pairs that defines the properties of the package and its tasks, containers, variables, connections, and event handlers when the package runs. Using configurations makes it possible to update properties without modifying the package. When the package is run, the configuration information is loaded, updating the values of properties. For example, a configuration can update the connection string of connection.

The configuration is saved and then deployed with the package when the package is installed on a different computer. The values in the configuration can be updated when the package is installed to support the package in a different environment. For more information, see [Creating Package Configurations](http://msdn.microsoft.com/en-us/library/ms141132(v=sql.105).aspx).

**Logging and Log Providers**

A log is a collection of information about the package that is collected when the package runs. For example, a log can provide the start and finish time for a package run. A log provider defines the destination type and the format that the package and its containers and tasks can use to log run-time information. The logs are associated with a package, but the tasks and containers in the package can log information to any package log. Integration Services includes a variety of built-in log providers for logging. For example, Integration Services includes log providers for SQL Server and text files. You can also create custom log providers and use them for logging. For more information, see [Integration Services Log Providers](http://msdn.microsoft.com/en-us/library/ms140246(v=sql.105).aspx).

**Variables**

Integration Services supports system variables and user-defined variables. The system variables provide useful information about package objects at run time, and user-defined variables support custom scenarios in packages. Both types of variables can be used in expressions, scripts, and configurations.

The package-level variables include the pre-defined system variables available to a package and the user-defined variables with package scope. For more information, see [Integration Services Variables](http://msdn.microsoft.com/en-us/library/ms141085(v=sql.105).aspx).

Packages are frequently used as templates from which to build packages that share basic functionality. You build the basic package and then copy it, or you can designate the package is a template. For example, a package that downloads and copies files and then extracts the data may include the FTP and File System tasks in a Foreach Loop that enumerates files in a folder. It may also include [Flat File](http://msdn.microsoft.com/en-us/library/ms141134(v=sql.105).aspx) connection managers to access the data, and Flat File sources to exact the data. The destination of the data varies, and the destination is added to each new package after it is copied from the basic package. You can also create packages and then use them as templates for the new packages that you add to an Integration Services project. For more information, see [Creating a Package in Business Intelligence Development Studio](http://msdn.microsoft.com/en-us/library/ms345180(v=sql.105).aspx).

When a package is first created, either programmatically or by using SSIS Designer, a GUID is added to its ID property and a name to its Name property. If you create a new package by copying an existing package or by using a template package, the name and the GUID are copied as well. This can be a problem if you use logging, because the GUID and the name of the package are written to the logs to identify the package to which the logged information belongs. Therefore, you should update the name and the GUID of the new packages to help differentiate them from the package from which they were copied and from each other in the log data.

To change the package GUID, you regenerate a GUID in the ID property in the Properties window in Business Intelligence Development Studio. To change the package name, you can update the value of the Name property in the Properties window. You can also use the **dtutil** command prompt, or update the GUID and name programmatically. For more information, see [Setting Package Properties](http://msdn.microsoft.com/en-us/library/ms137749(v=sql.105).aspx) and [dtutil Utility (SSIS Tool)](http://msdn.microsoft.com/en-us/library/ms162820(v=sql.105).aspx).

**STAIRWAY TO HEAVEN NOTES**

**LOOKUP TOOL**

A Lookup Transformation does exactly what the name implies: it looks in another table, view, or query for a match to the rows flowing through the transformation. There are few key concepts here and we will point them out as we configure the Lookup, but the general idea is "go to this other table, view, or query, and see if you find a match on this (or these) column(s). If you find a match, bring back this (or these) other column(s)." It sounds relatively simple - and it is. But there are a couple quirks.

Quirk #1: If no match is found between the column(s) in the data flow and the Lookup table, view, or query; the default Lookup Transformation configuration makes the transformation fail.

Quirk #2: If there is more than one match found in the lookup table, the Lookup Transformation returns only the very first match it finds.

I describe these quirks as a vicious top 1 join. Vicious, because the operation fails if there's no match found. Top 1 JOIN because the Transform returns only the first match found when joining the rows in the Data Flow to the rows in the Lookup table, view, or query.

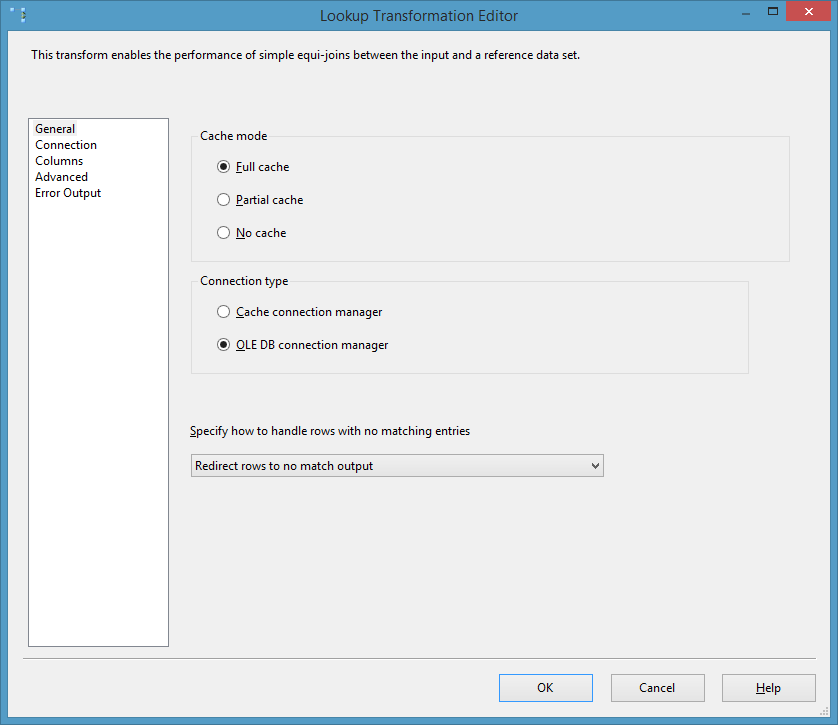
The Cache Mode property controls when and how the actual lookup operation will occur

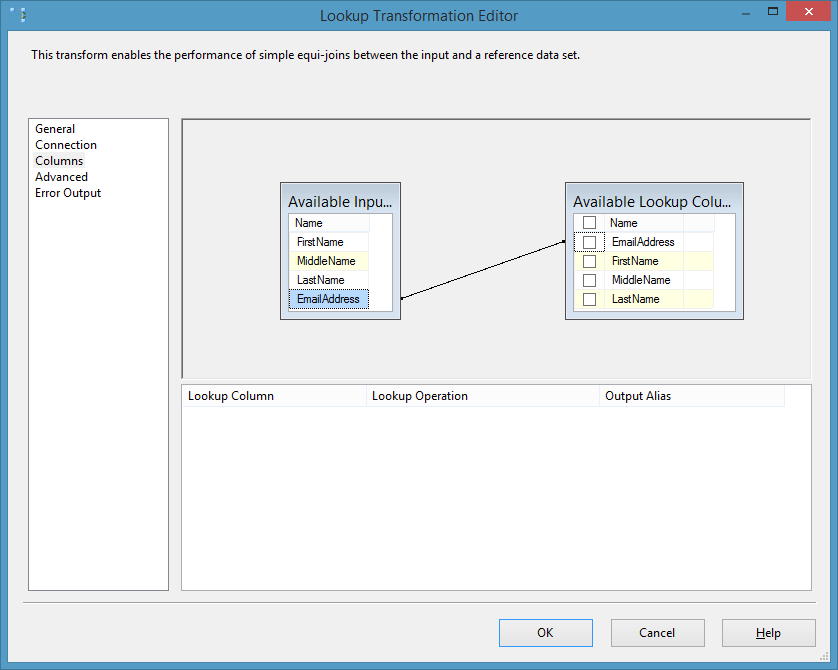
In No Cache mode, the lookup operation occurs as each row flows through the transformation. Whenever a row passes through the Lookup, the Transform executes a query against the Lookup table, view, or query; and adds any returned values to the rows as they flow through the Transformation.

In Full Cache mode, the lookup operation attempts to load all the rows from the Lookup table, view, or query into the Lookup cache in RAM before the Data Flow Task executes. Did you catch the word "attempts" in that last sentence? If the Lookup table, view, or query returns a large data set - or if the server is RAM-constrained (either running low or doesn't have enough RAM installed) - the Lookup will fail. The Lookup cache holds the values from the configured table, view, or query. Matches found in this cache are applied / added to the rows as they flow through the Transformation.

In Partial Cache mode, the Transformation first checks the Lookup cache as each row flows through - seeking a match. If there's no match in the cache, a lookup operation occurs. Matching data is added to the row and to the Lookup cache. If another row seeking the same matching columns flows through the Transformation, matching data is supplied from the Lookup cache.

change the dropdown labeled "Specify how to handle rows with no matching entries" to "Redirect rows to no match output":

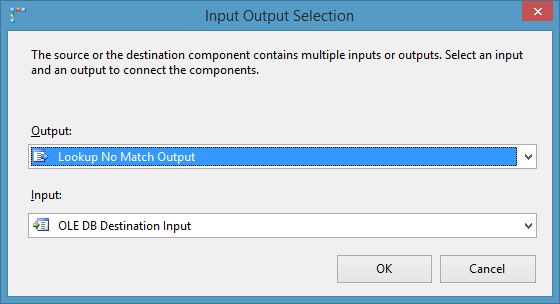




Available Input Columns and the Available Lookup Columns Email column - is analogous to the ON clause of the join. It defines the matching criterion that drives the Lookup function.

The Available Lookup Columns have checkboxes next to them and a "check all" checkbox in the grid header. If the Lookup Transformation is similar to a Join, the checkboxes are a mechanism for adding columns from the joined table to the SELECT clause

We have configured a Lookup Transformation to open the destination table and match records that exist in the Data Flow pipeline with records in the destination table. The records that exist in the Data Flow pipeline come from the OLE DB Source adapter, and are loaded into the Data Flow from the Person.Contact table. The destination table is dbo.Contact and we accessed it using a T-SQL query on the Lookup Transformation's Connections page (see Figure 18). We configured the Lookup Transformation to look for matches by comparing the Email column values in the Destination table with the Email column values from the Source table (via the OLE DB Source adapter). We configured the Lookup Transformation to send rows that do not match to the Lookup Transformation's No Match Output. If the rows find a match between the Email column values in the Destination table and the Email column values in the Source table, the Lookup Transformation will send those rows to the Match Output.



Why the Lookup No Match Output? In SSIS 2008 and SSIS 2008 R2, the Lookup Transformation provides this built-in output to catch records in the Lookup table (the dbo.Contact destination table, in this case) that do not exist in the source (the Person.Contact table) - it's the Lookup No Match Output.

Why would there be no match? Because the value in the Email column doesn't exist in the destination table! If it's in the source table and not in the destination table, it's a new row - one that's been added to the source table since the last load. This is a row we want to load - it's new.

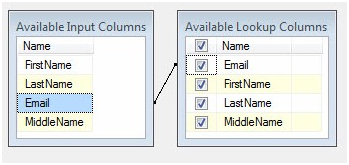
we've built a loader that only adds new rows from the source table to the destination table

Incremental Loads sometimes only load new rows. Consider a table that contains historical daily currency-conversion rates; this data isn't going to change over time. It's fixed at the end of each day. Another use case is a table that holds the high temperature for each day. Again, this data is never updated; new data is appended. The loader you just built will serve well in these cases.

Another consideration is the source data may change more or less rapidly. This incremental load pattern gives you the flexibility to load data once per year, once per minute, or at any interval - regular or irregular - in between. Only new rows are loaded into the destination. How flexible is that?

## ****Reviewing Incremental Loads****

Remember, incremental loads in SSIS have three use cases:

1. New rows – add rows to the destination that have been added to the source since the previous load.
2. Updated rows – update rows in the destination that have been updated in the source since the previous load.
3. Deleted rows – remove rows from the destination that have been deleted from the source.
4. If the Lookup Transformation is similar to a Join, these checkboxes are a mechanism for adding columns from the joined table to the SELECT clause. Click the unlabeled “Check All” checkbox to the left of the column header “Name” in the Available Lookup Columns grid. This will check all the columns as shown in Figure 2:
5. 

I often use “LkUp\_” or “Dest\_” to alias them. It helps me separate the columns that came in from the OLE DB Source and the columns that were returned from the Lookup Transformation. Plus, if the columns are named the same, SSIS will stick a “(1)” on the end of the column name.