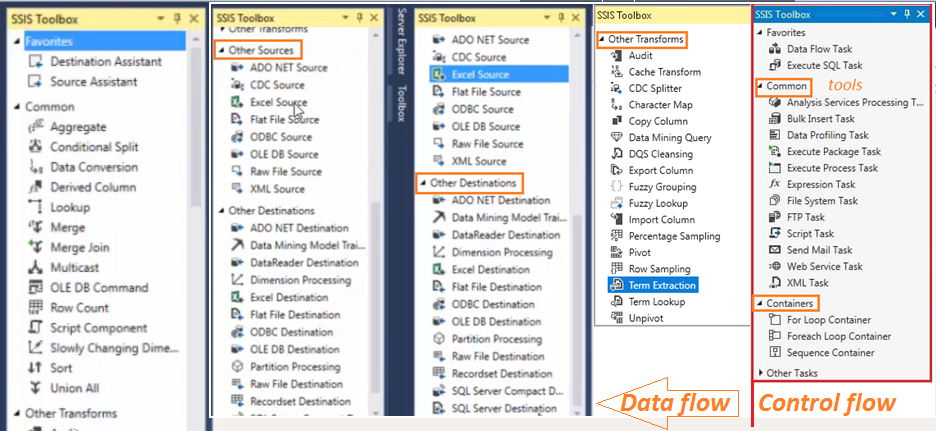
****

Depending on the functionality transformations are divided into the following five groups:

1. Business Intelligence Transformations
2. Split And Join Transformations
3. Row Transformations
4. Row-Set Transformations
5. Other Transformations.

**Business Intelligence Transformations**

1. **Fuzzy Group Transformation**: The Fuzzy Group Transformation is used to do data cleansing by finding rows that are likely duplicates and *reduce the number of duplicates within a dataset based on the Matching Decision.* This Transformation will accept only string data types while reducing the amount of duplicate data.
2. **Fuzzy Lookup Transformation**: The Fuzzy Lookup Transformation is used to return *close matches of reference* data for the incoming data stream. It can be matched and standardizes the data based on fuzzy logic.
3. **Term Extraction Transformation**: *Extracts terms* (nouns and noun phrases) from the input text into the transformation output column.
4. **Term Lookup Transformation**: Extracts terms from the *input column with TEXT data type and match them with the 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 then the transformation returns the value as well as the number of times it occurs in the row. You can configure this transformation *to do a case-sensitive search*.
5. **Data Mining Query Transformation**: Queries a data mining model. Includes a query builder to *assist you with the development of Data Mining eXpressions (DMX) prediction queries*.
6. **Data Cleansing Transformation**: This transformation is used to *do automating data cleansing and monitoring* the overall status of the data cleansing process.

**Split and Join Transformations**

1. **Cache Transformation**: This transformation is used to *store data as a file or in memory* for use in a lookup transformation
2. **Conditional Split**: This transformation is used to accepts input and determine which destination to pipe the data into based on the result of an expression. It *redirects rows of data that meet specific conditions to different outputs*
3. **Look-Up Transformation**: This Look Up Transformation is used to *join 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.
4. **Merge Transformation**: This transformation is used to *merge 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
5. **Merge-Join Transformation**: This transformation is used to merge two datasets into a single dataset using a *JOIN* function.
6. **Multicast Transformation**: This transformation is used to *send a copy of the data to an additional path* in the workflow. It duplicates the data in the dataflow that again we can send the data in parallel, or when we want to send the data to multiple destinations simultaneously.
7. **Union-All Transformation**: *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.

**Row Transformations**

1. **Character Map Transformation**: The Character Map transformation allows you to do *character operations on string* columns. It makes common string data changes for you.
2. **Copy Column Transformation**: This transformation is used to add a *copy of column to the transformation output*. You can later transform the copy. Makes a copy of a single or multiple columns that will be further transformed by subsequent tasks in the package
3. **Data Conversion Transformation**: This transformation is used *to convert a column data type* to a new (another) column data type.
4. **Derived Column Transformation**: This transformation is used to *apply expression to a data column and create a new derived column calculated* from an expression.
5. **OLEDB Command Transformation**: Runs a SQL *command for each input data row.* Normally your SQL statement will include a parameter (denoted by the question mark)
6. **Script Component Transformation**: This transformation is used to do a custom transformation. It uses a *script to transform the data and you can apply specialized business logic* to your data flow.

**Row Set Transformations**:

1. **Aggregate Transformation**: This transformation is used to aggregates the data from transformation or source and it *aggregates the values by group.*
2. **Row Sampling Transformation**: This transformation is used to capture a *sampling of the data* from the data flow by using a *row count* of the data flow's total rows. It Loads only a subset of your data, defined as the number of rows and it *randomly selects the data* and is delivered to somewhere.
3. **Percentage Sampling Transformation**: Loads only a *subset of your data*, defined as the percentage of all rows in the data source. It randomly selects percentage of rows.
4. **Sort Transformation**: This transformation is used to *sort the data* in the data flow by a given column and discard with duplicate values (optionally eliminating duplicates).
5. **Pivot Transformation**: This transformation is used to pivot *the data on a column into a more non-relational form*. It converts rows into columns.
6. **UnPivot Transformation**: This transformation is used to *unpivot the data from a non-normalized format* to a relational format.

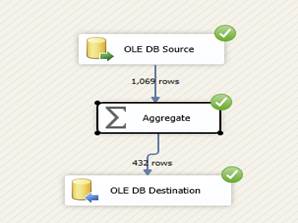
**Other SSIS Transformations**:

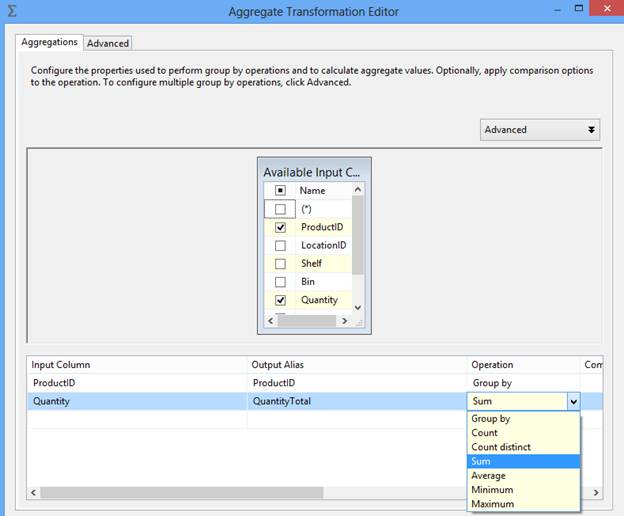
1. **Audit Transformation:** This transformation is used to *expose auditing information from the package* to the data pipe, such as package execution and Execution Time.
2. **Row Count Transformation:** This transformation is used to *count the rows* in the data flow and stores them as a variable.
3. **SCD Transformation:** This transformation *maintains the historical values of the dimension members* when new members are introduced. It automatically generates transformations for TYPE1 and TYPE2 SCD's.
4. **Export Transformation:** This transformation is used to *export the column from the data flow to the system*.
5. **Import Transformation:** This transformation is used to *read data from files and appends it to the data flow*.

**Dataflow - Common Transformation**

**Aggregate**

An Asynchronous full blocking transformation, Aggregate transformation allows to aggregate data from Data Flow to apply certain T-SQL functions that are done in a GROUP BY statement.

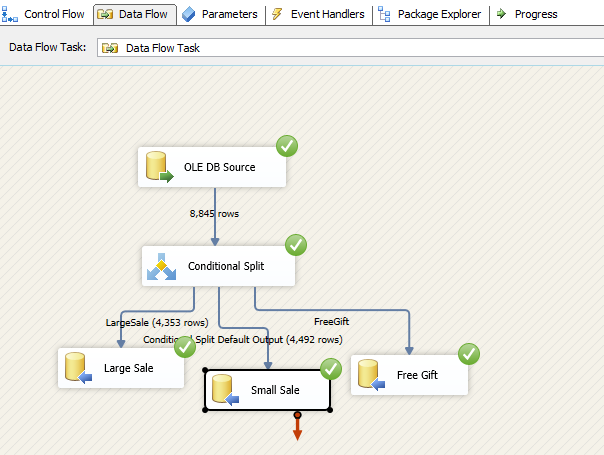


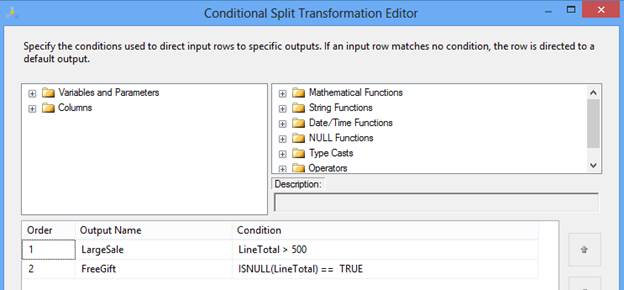


In the above example we have applied SUM aggregation but Aggregation transformation provides other options to aggregate data like Count, Count distinct, Average, Minimum and Maximum.

**Conditional Split**

Synchronous transformation, allows you to send the data from a single data path to various outputs or paths based on conditions that use the SSIS expressions.





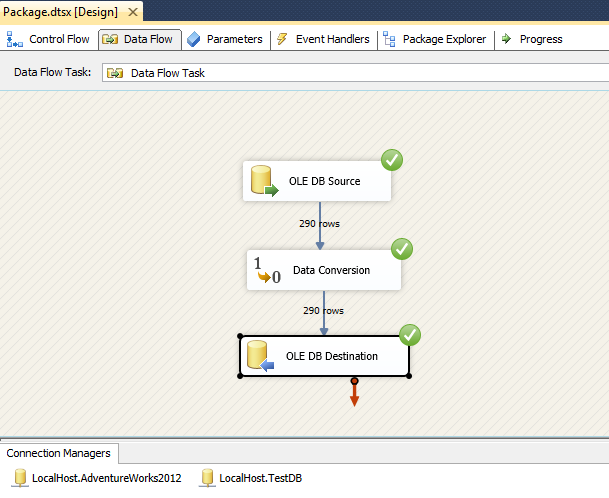
In the above example, we are splitting input records based on total order cost. If cost is more than 500, the record will be considered as part of a large sale. If LineTotal is NULL, we are assuming it's a free gift and no cost is associated with it. The rest we can consider part of small sale, in current implementation it is the default output of Conditional Split transformation.

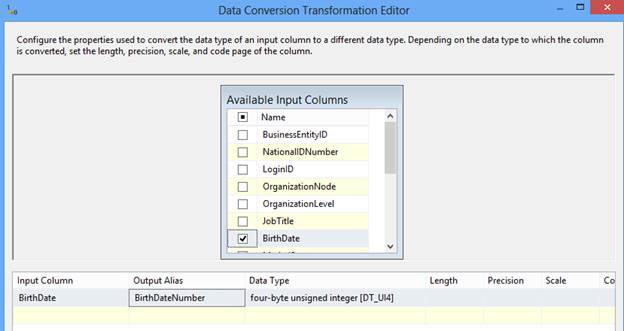
After execution of DFT the data will move in three different destinations as per ETL design.

## Data Conversion

Synchronous transformation is used for data conversion. It is a similar function to the Convert or Cast functions in T-SQL. It is a very useful transformation if we are pulling same data from multiple sources.

Data flow task design for Data conversion:

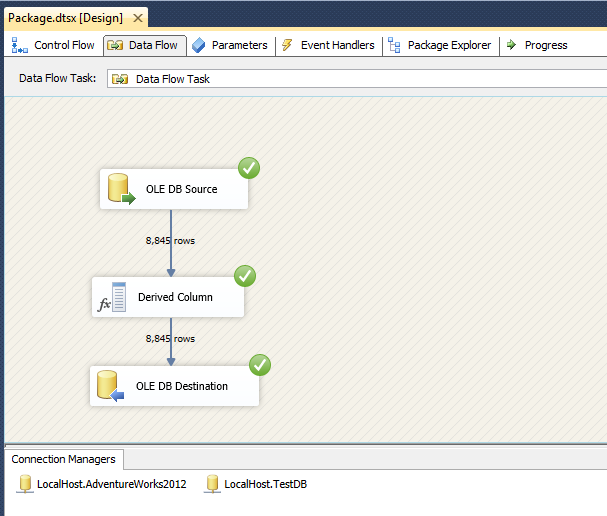


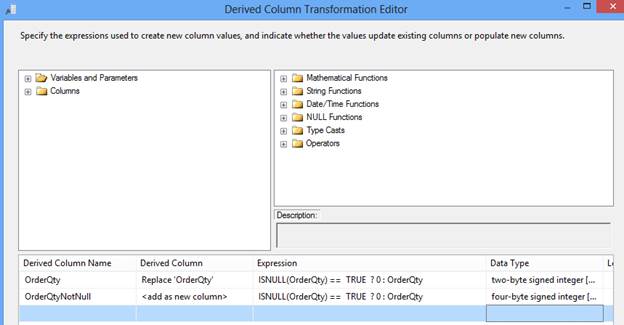


In this example we converted the BirthDate column of the datetime data type in another column BirthDateNumber of Integer data type.

## Derived Column

Synchronous transformation, this transformation creates a new column that is derived from the output of another column. This transformation provides you two options; either you can create a new column as a derived column or replace the existing column with a new derived column.

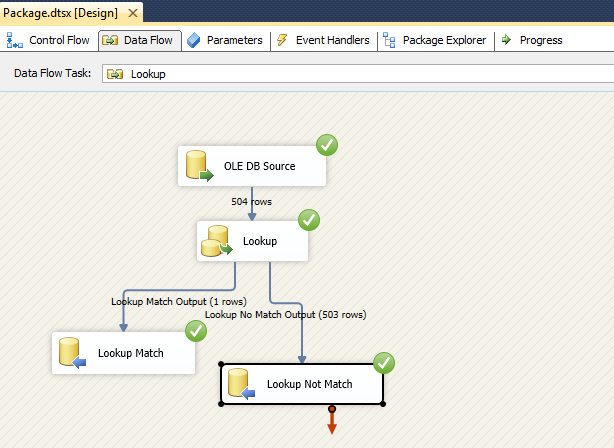




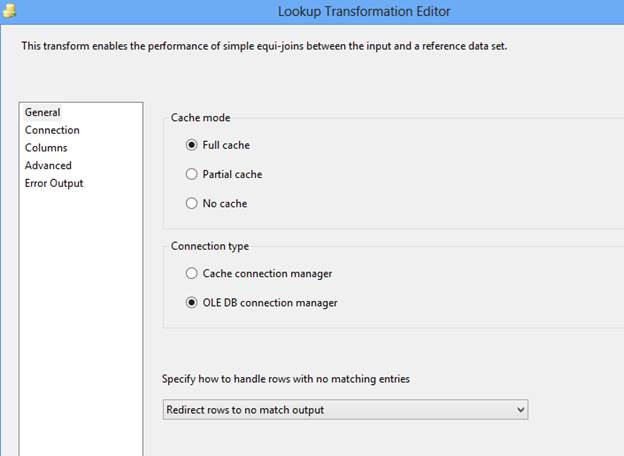
In this example, in the first row, check the if OrderQty value is NULL then update with 0 and in the second row apply the same operation as in the first row; the only difference is it will create one new column OrderQtyNotNull in the output. So, with the help of Derived Column transformation you can either update an existing column value or introduce a new column in the output.

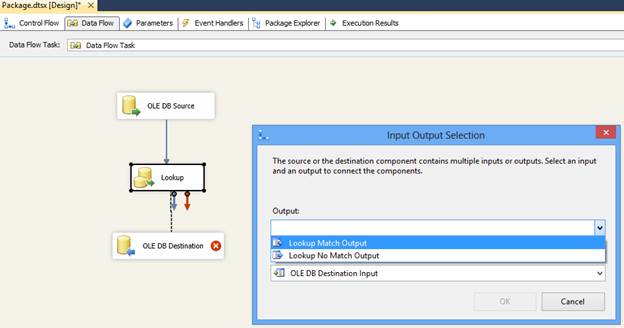
## Lookup

Synchronous transformation, allows you to perform an equi-join between values in the transformation input and values in the reference dataset similar to T-SQL. This transformation is used to join two datasets at a time.  To join more than two datasets we need to put multiple Lookup transformations, similar to a T-SQL join condition.

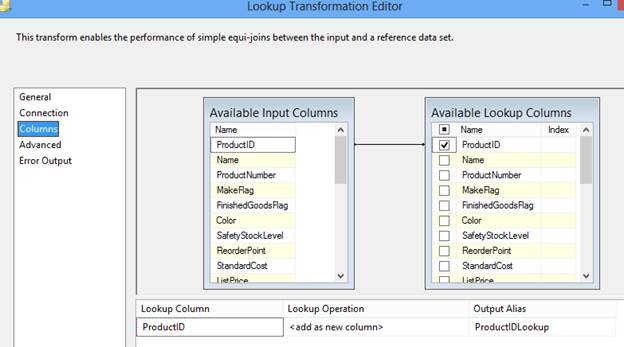


If there is no matching entry in the reference dataset, no join occurs. By default, the Lookup transformation treats rows without matching entries as errors. However, it can configure the Lookup transformation to redirect such rows to a no match output as shown in the images below:





The join can be a composite join, which means that multiple columns can be used in the join in the transformation input to columns in the reference dataset; for simplification we used only one column. Refer to the below image:

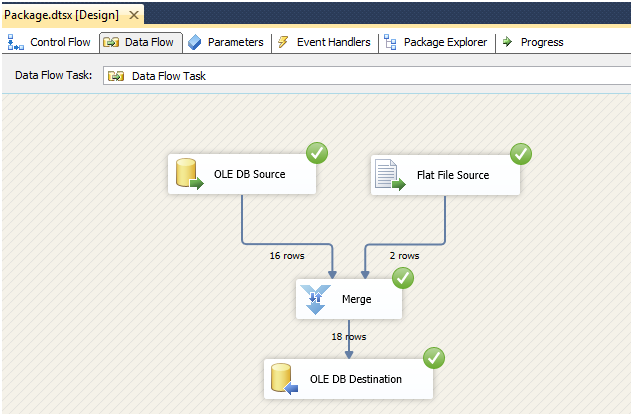


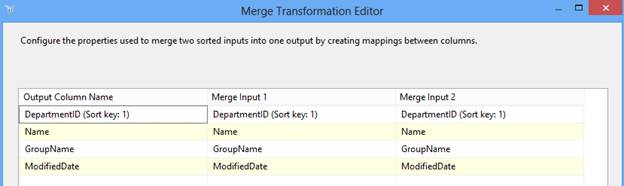
In above image, you can observe in the Lookup Operation that we specified “<add as new column>”; its mean values from the reference dataset are added as a new column to the transformation output. For example, the Lookup transformation can extract the ProductID details from a table using a value from an input column, and then add the ProductIDLookup to the transformation output. The values from the reference table can replace column values or can be added to new columns.

Lookup transformations provides several modes of operations, Full cache, Partial cache or No cache, that allows a trade-off between performance and resource usage.

## Merge

An Asynchronous partial blocking transformation merges two sorted data sets into a single dataset. This transformation is very useful when during ETL its needs to merge data from two different data sources. Merge transformation can’t merge a column that has a numeric data type with a column that has a character data type.





In the above example, we are merging data from two sources; OLEDB and Flat File. The Merge transformation automatically maps columns that have the same metadata. You can then manually map other columns that have compatible data types.

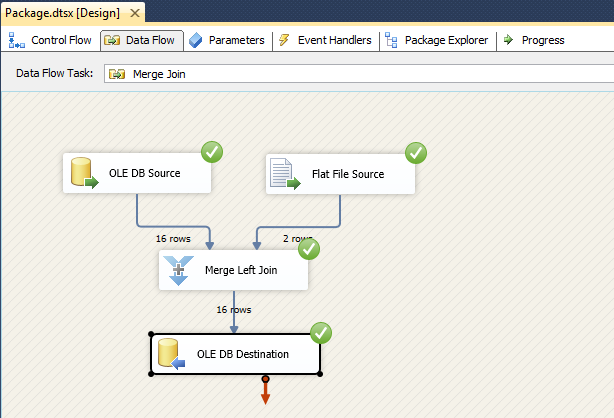
This transformation has two inputs and one output. It does not support an error output.

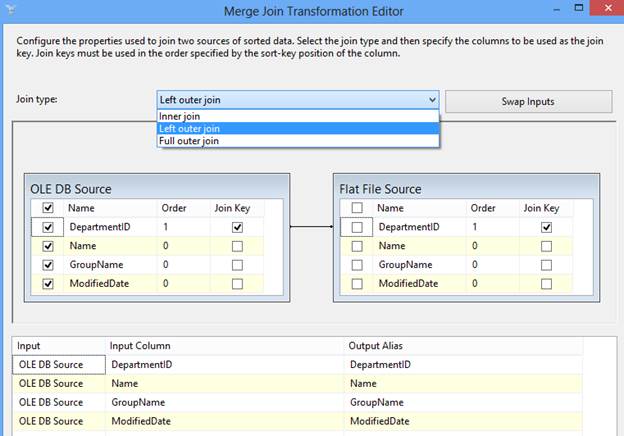
## Merge Join

An Asynchronous partial blocking transformation, allows joining data from two sorted datasets using a FULL, LEFT, or INNER join.

It also has two inputs and one output and like Merge transformation, does not support an error output.

Data Flow task design for Merge Join:



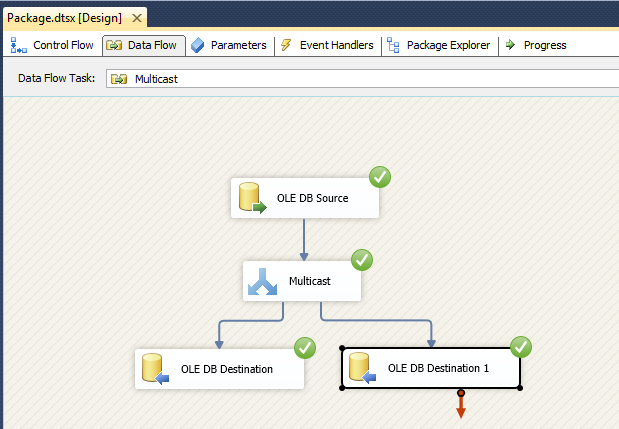


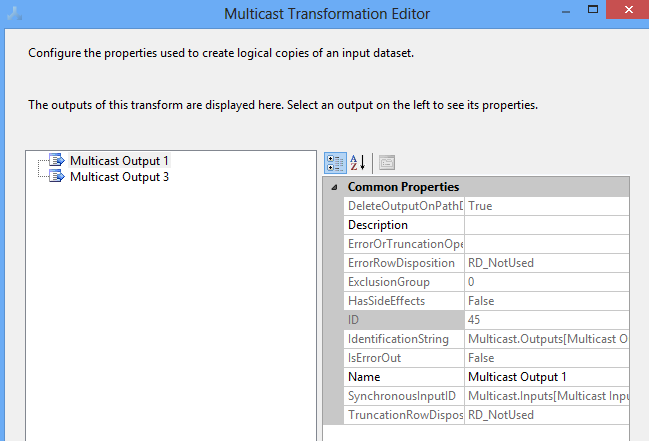
In above example, we merged data from two different sources; OLEDB and Flat File, applying a Left outer join on DepartmentID.

## Multicast

Synchronous transformation allows you to distribute its input to one or more outputs. This transformation is similar to the Conditional Split transformation. Both transformations direct an input to multiple outputs. The difference between the two is that the Multicast transformation directs every row to every output, and the Conditional Split directs a row to a single output.

In nutshell, a Multicast transformation is used to create/distribute exact copies of the source dataset to one or more destination datasets.

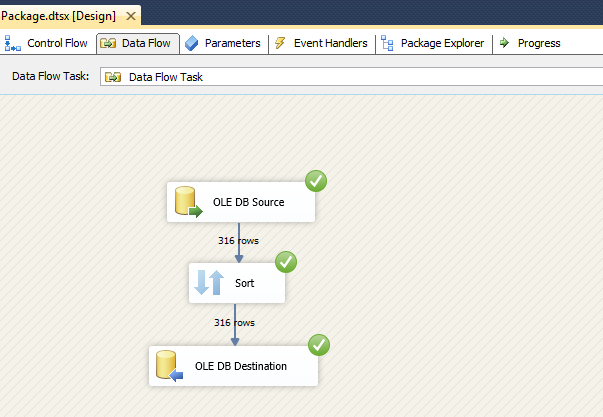


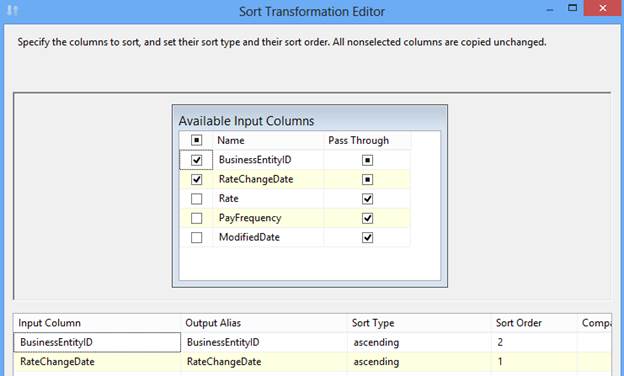


In the above example, we are distributing log data to two different destinations

## Sort

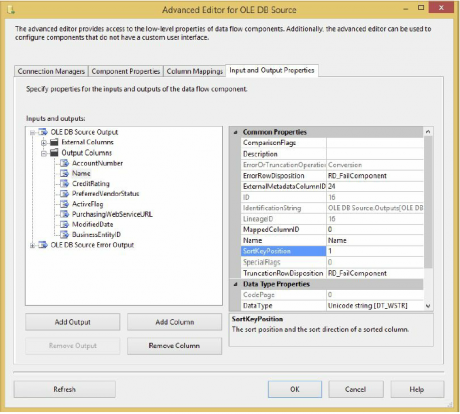
An Asynchronous full blocking transformation allows sort or arrange input data in ascending or descending order and copies the sorted data to the transformation output. You can apply multiple sorts to an input; the column with the lowest number is sorted first, the sort column with the second lowest number is sorted next.





In above example, we arranged input data in ascending order of RateChangeDate first and BusinessEntityID column second.

Sort transformation has one input and one output. It does not support error outputs.



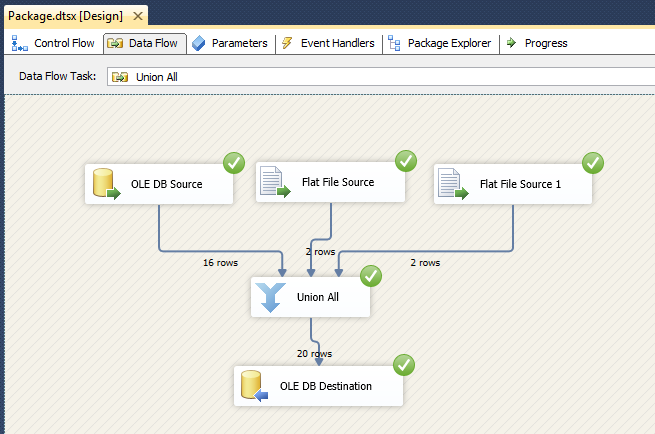
If you have an ORDER BY clause in your T-SQL statement in the OLE DB Source or the ADO.NET Source, you can notify SSIS that the data is already sorted, obviating the need for the Sort Transformation in the Advanced Editor. After ordering the data in your SQL statement, right-click the source and select Advanced Editor. From the Input and Output Properties tab, select OLE DB Source Output. In the Properties pane, change the IsSorted property to True.

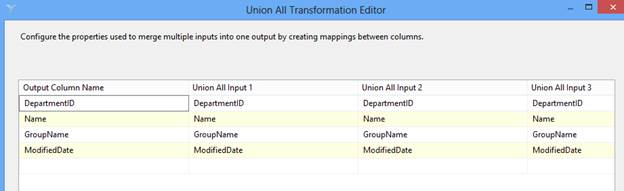
Then, under Output Columns, select the column you are ordering on in your SQL statement, and change the SortKeyPosition to 1 if you’re sorting only by a single column ascending, as shown in Figure. If you have multiple columns, you could change this SortKeyPosition value to the column position in the ORDER BY statement starting at 1. A value of -1 will sort the data in descending order.

## Union All

An Asynchronous partial blocking transformation, allows you to combine multiple (more than two) input and produce one output. Its add inputs to transformation output one after the other and doesn’t sort the data.

Data Flow task design of Union All:

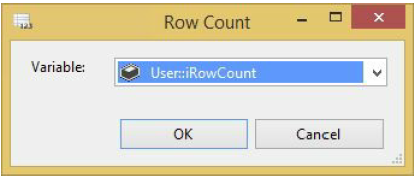




In above example, we used three sources as input and combine all using the Union All transformation before inserting into  the destination. Here, we took two different type of sources; OLEDB and Flat File.

### Row Count

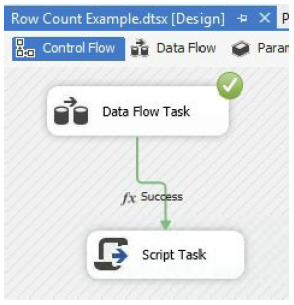
The Row Count Transformation provides the capability to count rows in a stream that is directed to its input source. This transformation must place that count into a variable that could be used in the Control Flow — for insertion into an audit table, for example.  If you need to know how many rows are split during the Conditional Split Transformation, direct the output of each side of the split to a separate Row Count Transformation. You can then use this variable to log information into storage, to build e-mail messages, or to conditionally run steps in your packages.



In the Control Flow tab, add a variable named **iRowCount**. Ensure that the variable is package scoped and of type Int32

### Script Component

The Script Component enables you to write custom .NET scripts as transformations, sources, or destinations. Once you drag the component over, it will ask you if you want it to be a source, transformation, or destination.



Some of the things you can do with this transformation include the following:

* Create a custom transformation that would use a .NET assembly to validate credit card numbers or mailing addresses.
* Validate data and skip records that don’t seem reasonable. For example, you can use it in a human resource recruitment system to pull out candidates that don’t match the salary requirement at a job code level.
* GIACT - validates routing number and bank account no
* Check image OCR
* vendor insurance information

Scripts used as sources can support multiple outputs, and you have the option of precompiling the scripts for runtime efficiency.

### Slowly Changing Dimension

A dimension table contains a set of discrete values with a description and often other measurable attributes such as price, weight, or sales territory. The classic problem is what to do in your dimension data when an attribute in a row changes — particularly when you are loading data automatically through an ETL process. This transformation can shave days off of your development time in relation to creating the load manually through T-SQL, but it can add time because of how it queries your destination and how it updates with the OLE DB Command Transform (row by row).