Data Flow Task

**SQL Server 2008 R2**

[Other Versions](javascript:;)

The Data Flow task encapsulates the data flow engine that moves data between sources and destinations, and lets the user transform, clean, and modify data as it is moved. Addition of a Data Flow task to a package control flow makes it possible for the package to extract, transform, and load data.

A data flow consists of at least one data flow component, but it is typically a set of connected data flow components: sources that extract data; transformations that modify, route, or summarize data; and destinations that load data. Components are connected in the data flow by paths. Each path specifies the two components that are the start and the end of the path. For more information, see [Data Flow Elements](https://technet.microsoft.com/en-us/library/ms140080(v=sql.105).aspx).

At run time, the Data Flow task builds an execution plan from the data flow, and the data flow engine executes the plan. You can create a Data Flow task that has no data flow, but the task executes only if it includes at least one data flow.

The following diagram shows a Data Flow task with one data flow.

A Data Flow task can include multiple data flows. If a task copies several sets of data, and if the order in which the data is copied is not significant, it can be more convenient to include multiple data flows in the Data Flow task. For example, you might create five data flows, each copying data from a flat file into a different dimension table in a data warehouse star schema.

However, the data flow engine determines order of execution when there are multiple data flows within one data flow task. Therefore, when order is important, the package should use multiple Data Flow tasks, each task containing one data flow. You can then apply precedence constraints to control the execution order of the tasks.

The following diagram shows a Data Flow task that has multiple data flows.

A package can include multiple Data Flow tasks, and complex packages frequently do. For example, if a package requires that data flows be run in a specified sequence, or that other tasks be performed between the data flows, you must use a separate Data Flow task for each data flow.

The Data Flow task also manages error flows. At run time, row-level errors may occur when data flow components convert data, perform a lookup, or evaluate expressions. For example, a data column with a string value cannot be converted to an integer, or an expression tries to divide by zero. Both operations cause errors, and the rows that contain the errors can be processed separately using an error flow. For more information about how to use error flows in package data flow, see [Handling Errors in the Data Flow](https://technet.microsoft.com/en-us/library/ms141679(v=sql.105).aspx).

To bulk insert data from text files into a SQL Server database, you can use the Bulk Insert task instead of a Data Flow task and a data flow. However, the Bulk Insert task cannot transform data. For more information, see [Bulk Insert Task](https://technet.microsoft.com/en-us/library/ms141239(v=sql.105).aspx).

[Using Property Expressions with Data Flow Elements](javascript:void(0))

A number of the data flow components—sources, transformations, and destinations—support the use of property expressions in some of their properties. A property expression is an expression that replaces the value of the property when the package is loaded. At run time, the package uses the updated property values. The expressions are built using the Integration Services expression syntax and can include Integration Services functions, operators, identifiers, and variables. For more information, see [Integration Services Expression Reference](https://technet.microsoft.com/en-us/library/ms141232(v=sql.105).aspx),[Using Expressions in Packages](https://technet.microsoft.com/en-us/library/ms137547(v=sql.105).aspx), and [Using Property Expressions in Packages](https://technet.microsoft.com/en-us/library/ms141214(v=sql.105).aspx).

If you construct a package in Business Intelligence Development Studio, the properties of any data flow components that support property expressions are exposed on the Data Flow task to which they belong. To add, change, and remove the property expressions of data flow components, click the Data Flow task, and then use the Properties window or the editor for the task to add, change, or delete property expressions. Property expressions for the Data Flow task itself are managed in the Properties window.

If the data flow contains any components that use expressions, the expressions are also exposed in the Properties window. To view expressions, select the Data Flow task to which the component belongs. You can view properties by categories, or in alphabetical order. If you use the categorized view in the Properties window, any expressions that are not used in a specific property are listed in the **Misc**category. If you use the alphabetical view, expressions are listed in order of the name of the data flow component.

[Log Entries](javascript:void(0))

Integration Services provides a set of log events that are available to all tasks. Integration Services also provides custom log entries to many tasks. For more information, see [Implementing Logging in Packages](https://technet.microsoft.com/en-us/library/ms138020(v=sql.105).aspx) and [Custom Messages for Logging](https://technet.microsoft.com/en-us/library/ms345174(v=sql.105).aspx). The Data Flow task includes the following custom log entries:

|  |  |
| --- | --- |
| **Log entry** | **Description** |
| **BufferSizeTuning** | Indicates that the Data Flow task changed the size of the buffer. The log entry describes the reasons for the size change and lists the temporary new buffer size. |
| **OnPipelinePostEndOfRowset** | Denotes that a component has been given its end-of-rowset signal, which is set by the last call of the**ProcessInput** method. An entry is written for each component in the data flow that processes input. The entry includes the name of the component. |
| **OnPipelinePostPrimeOutput** | Indicates that the component has completed its last call to the **PrimeOutput** method. Depending on the data flow, multiple log entries may be written. If the component is a source, this log entry means that the component has finished processing rows. |
| **OnPipelinePreEndOfRowset** | Indicates that a component is about to receive its end-of-rowset signal, which is set by the last call of the **ProcessInput** method. An entry is written for each component in the data flow that processes input. The entry includes the name of the component. |
| **OnPipelinePrePrimeOutput** | Indicates that the component is about to receive its call from the **PrimeOutput** method. Depending on the data flow, multiple log entries may be written. |
| **OnPipelineRowsSent** | Reports the number of rows provided to a component input by a call to the **ProcessInput** method. The log entry includes the component name. |
| **PipelineBufferLeak** | Provides information about any component that kept buffers alive after the buffer manager goes away. If a buffer is still alive, buffers resources were not released and may cause memory leaks. The log entry provides the name of the component and the ID of the buffer. |
| **PipelineComponentTime** | Reports the amount of time (in milliseconds) that the component spent in each of its five major processing steps—Validate, PreExecute, PostExecute, ProcessInput, and ProcessOutput. |
| **PipelineExecutionPlan** | Reports the execution plan of the data flow. The execution plan provides information about how buffers will be sent to components. This information, in combination with the PipelineExecutionTrees log entry, describes what is happening within the Data Flow task. |
| **PipelineExecutionTrees** | Reports the execution trees of the layout in the data flow. The scheduler of the data flow engine uses the trees to build the execution plan of the data flow. |
| **PipelineInitialization** | Provides initialization information about the task. This information includes the directories to use for temporary storage of BLOB data, the default buffer size, and the number of rows in a buffer. Depending on the configuration of the Data Flow task, multiple log entries may be written. |

These log entries provide a wealth of information about the execution of the Data Flow task each time you run a package. As you run the packages repeatedly, you can capture information that over time provides important historical information about the processing that the task performs, issues that might affect performance, and the data volume that task handles.

For more information about how to use these log entries to monitor and improve the performance of the data flow, see one of the following topics:

* [Monitoring the Performance of the Data Flow Engine](https://technet.microsoft.com/en-us/library/ms137622(v=sql.105).aspx)
* [Improving the Performance of the Data Flow](https://technet.microsoft.com/en-us/library/ms141031(v=sql.105).aspx)

Sample Messages From a Data Flow Task

The following table lists sample messages for log entries for a very simple package. The package uses an OLE DB source to extract data from a table, a Sort transformation to sort the data, and an OLE DB destination to writes the data to a different table.

|  |  |
| --- | --- |
| **Log entry** | **Messages** |
| **BufferSizeTuning** | Rows in buffer type 0 would cause a buffer size greater than the configured maximum. There will be only 9637 rows in buffers of this type.  Rows in buffer type 2 would cause a buffer size greater than the configured maximum. There will be only 9497 rows in buffers of this type.  Rows in buffer type 3 would cause a buffer size greater than the configured maximum. There will be only 9497 rows in buffers of this type. |
| **OnPipelinePostEndOfRowset** | A component will be given the end of rowset signal. : 1180 : Sort : 1181 : Sort Input  A component will be given the end of rowset signal. : 1291 : OLE DB Destination : 1304 : OLE DB Destination Input |
| **OnPipelinePostPrimeOutput** | A component has returned from its PrimeOutput call. : 1180 : Sort  A component has returned from its PrimeOutput call. : 1 : OLE DB Source |
| **OnPipelinePreEndOfRowset** | A component has finished processing all of its rows. : 1180 : Sort : 1181 : Sort Input  A component has finished processing all of its rows. : 1291 : OLE DB Destination : 1304 : OLE DB Destination Input |
| **OnPipelinePrePrimeOutput** | PrimeOutput will be called on a component. : 1180 : Sort  PrimeOutput will be called on a component. : 1 : OLE DB Source |
| **OnPipelineRowsSent** | Rows were provided to a data flow component as input. : : 1185 : OLE DB Source Output : 1180 : Sort : 1181 : Sort Input : 76  Rows were provided to a data flow component as input. : : 1308 : Sort Output : 1291 : OLE DB Destination : 1304 : OLE DB Destination Input : 76 |
| **PipelineComponentTime** | The component "Calculate LineItemTotalCost" (3522) spent 356 milliseconds in ProcessInput.  The component "Sum Quantity and LineItemTotalCost" (3619) spent 79 milliseconds in ProcessInput.  The component "Calculate Average Cost" (3662) spent 16 milliseconds in ProcessInput.  The component "Sort by ProductID" (3717) spent 125 milliseconds in ProcessInput.  The component "Load Data" (3773) spent 0 milliseconds in ProcessInput.  The component "Extract Data" (3869) spent 688 milliseconds in PrimeOutput filling buffers on output "OLE DB Source Output" (3879).  The component "Sum Quantity and LineItemTotalCost" (3619) spent 141 milliseconds in PrimeOutput filling buffers on output "Aggregate Output 1" (3621).  The component "Sort by ProductID" (3717) spent 16 milliseconds in PrimeOutput filling buffers on output "Sort Output" (3719). |
| **PipelineExecutionPlan** | SourceThread0  Drives: 1  Influences: 1180 1291  Output Work List  CreatePrimeBuffer of type 1 for output ID 11.  SetBufferListener: "WorkThread0" for input ID 1181  CreatePrimeBuffer of type 3 for output ID 12.  CallPrimeOutput on component "OLE DB Source" (1)  End Output Work List  End SourceThread0  WorkThread0  Drives: 1180  Influences: 1180 1291  Input Work list, input ID 1181 (1 EORs Expected)  CallProcessInput on input ID 1181 on component "Sort" (1180) for view type 2  End Input Work list for input 1181  Output Work List  CreatePrimeBuffer of type 4 for output ID 1182.  SetBufferListener: "WorkThread1" for input ID 1304  CallPrimeOutput on component "Sort" (1180)  End Output Work List  End WorkThread0  WorkThread1  Drives: 1291  Influences: 1291  Input Work list, input ID 1304 (1 EORs Expected)  CallProcessInput on input ID 1304 on component "OLE DB Destination" (1291) for view type 5  End Input Work list for input 1304  Output Work List  End Output Work List  End WorkThread1 |
| **PipelineExecutionTrees** | begin execution tree 0  output "OLE DB Source Output" (11)  input "Sort Input" (1181)  end execution tree 0  begin execution tree 1  output "OLE DB Source Error Output" (12)  end execution tree 1  begin execution tree 2  output "Sort Output" (1182)  input "OLE DB Destination Input" (1304)  output "OLE DB Destination Error Output" (1305)  end execution tree 2 |
| **PipelineInitialization** | No temporary BLOB data storage locations were provided. The buffer manager will consider the directories in the TEMP and TMP environment variables.  The default buffer size is 10485760 bytes.  Buffers will have 10000 rows by default  The data flow will not remove unused components because its RunInOptimizedMode property is set to false. |

Many log events write multiple entries, and the messages for a number of log entries contain complex data. To make it easier to understand and to communicate the content of complex messages you can parse the message text. Depending on the location of the logs, you can use Transact-SQL statements or a Script component to separate the complex text into columns or other formats that you find more useful.

For example, the following table contains the message "Rows were provided to a data flow component as input. : : 1185 : OLE DB Source Output : 1180 : Sort : 1181 : Sort Input : 76", parsed into columns. The message was written by the **OnPipelineRowsSent** event when rows were sent from the OLE DB source to the Sort transformation.

|  |  |  |
| --- | --- | --- |
| **Column** | **Description** | **Value** |
| **PathID** | The value from the **ID** property of the path between the OLE DB source and the Sort transformation. | 1185 |
| **PathName** | The value from the **Name** property of the path. | OLE DB Source Output |
| **ComponentID** | The value of the **ID** property of the Sort transformation. | 1180 |
| **ComponentName** | The value from the **Name** property of the Sort transformation. | Sort |
| **InputID** | The value from the **ID** property of the input to the Sort transformation. | 1181 |
| **InputName** | The value from the **Name** property of the input to the Sort transformation. | Sort Input |
| **RowsSent** | The number of rows sent to the input of the Sort transformation. | 76 |

Sources

The following sources have properties that can be updated by property expressions:

* [ADO NET Source](https://technet.microsoft.com/en-us/library/ms137897(v=sql.105).aspx)
* [XML Source](https://technet.microsoft.com/en-us/library/ms140277(v=sql.105).aspx)

For more information, see [Source Custom Properties](https://technet.microsoft.com/en-us/library/ms135923(v=sql.105).aspx).

Transformations

The following transformations have properties that can be updated by property expressions:

* [Conditional Split Transformation](https://technet.microsoft.com/en-us/library/ms137886(v=sql.105).aspx)
* [Derived Column Transformation](https://technet.microsoft.com/en-us/library/ms141069(v=sql.105).aspx)
* [Fuzzy Grouping Transformation](https://technet.microsoft.com/en-us/library/ms141764(v=sql.105).aspx)
* [Fuzzy Lookup Transformation](https://technet.microsoft.com/en-us/library/ms137786(v=sql.105).aspx)
* [OLE DB Command Transformation](https://technet.microsoft.com/en-us/library/ms141138(v=sql.105).aspx)
* [Percentage Sampling Transformation](https://technet.microsoft.com/en-us/library/ms139864(v=sql.105).aspx)
* [Pivot Transformation](https://technet.microsoft.com/en-us/library/ms140308(v=sql.105).aspx)
* [Row Sampling Transformation](https://technet.microsoft.com/en-us/library/ms141200(v=sql.105).aspx)
* [Sort Transformation](https://technet.microsoft.com/en-us/library/ms140182(v=sql.105).aspx)
* [Unpivot Transformation](https://technet.microsoft.com/en-us/library/ms141723(v=sql.105).aspx)

For more information, see [Transformation Custom Properties](https://technet.microsoft.com/en-us/library/ms136014(v=sql.105).aspx).

Destinations

The following destinations have properties that can be updated by property expressions:

* [Flat File Destination](https://technet.microsoft.com/en-us/library/ms141668(v=sql.105).aspx)
* [SQL Server Compact Edition Destination](https://technet.microsoft.com/en-us/library/ms140269(v=sql.105).aspx)

For more information, see [Destination Custom Properties](https://technet.microsoft.com/en-us/library/ms135971(v=sql.105).aspx).

[Configuring the Data Flow Task](javascript:void(0))

You can set properties in the **Properties** window or programmatically.

For more information about how to set these properties in the **Properties** window, click the following topic:

* [How to: Set the Properties of a Task or Container](https://technet.microsoft.com/en-us/library/ms139733(v=sql.105).aspx)

Configuring the Data Flow Task Programmatically

For more information about programmatically adding a data flow task to a package and setting data flow properties, click the following topic:

* [Adding the Data Flow Task Programmatically](https://technet.microsoft.com/en-us/library/ms135997(v=sql.105).aspx)