

Data Integrator

Event Management Guide

Pervasive Software, Inc.
12365-B Riata Trace Parkway
Austin, Texas 78727 USA

Telephone: 888.296.5969 or 512.231.6000

Fax: 512.231.6010

Email: info@pervasiveintegration.com

Web: <http://www.pervasiveintegration.com>

PERVASIVE[®]

See copyrights.txt in the product installation directory for information on third-party and open source software components.

Event Management Guide
October 2010

About This Manual	v
1 Event Handling	1-1
<i>Executing Actions During Transformations</i>	
Transformation Event Handlers	1-2
Events	1-3
Actions	1-3
Order of Event Action Execution	1-4
Event Precedence	1-5
Source Event Precedence	1-5
Moving From Source to Target or Error Events	1-6
Target Event Precedence	1-6
Error Event Precedence	1-7
Reaching Transformation End	1-8
Source General Event Handlers	1-9
Target General Event Handlers	1-10
Trapping Key Mismatch Errors	1-11
Record Type Event Handlers	1-13
2 Event Actions.	2-1
<i>A Guide to Event Actions in Map Designer</i>	
Event Actions List	2-3
Abort Action	2-6
Assert Action	2-7
ChangeSource Action	2-8
Plus Sign Usage in String Syntax	2-8
Example 2	2-9
ChangeTarget Action	2-10
Plus Sign Usage in String Syntax	2-11
Example 1	2-11
Example 2	2-11
Example 3	2-11
Clear Action	2-13
Clear Tree Action	2-15
ClearInitialize Action	2-17
ClearMap Action	2-18
ClearMapPut Record Action	2-20
Execute Action	2-23
LogMessage Action	2-24
LogTargetRecord Action	2-25
Map Action	2-26
MapPut Record Action	2-28
OnRecordValidationError Event	2-30
Put Record Action	2-31

Put Tree Action	2-33
QueryStatistic Action	2-36
Resume Action	2-38
Terminate Action	2-39
TraceOff Action	2-40
TraceOn Action	2-42
Upsert Record Action	2-44
Validate Record Action	2-46

3 Multimode Event Actions 3-1

Multimode Connectors and Multimode Event Actions

Multimode Event Actions List	3-2
ClearMapInsert Record Action	3-4
Create Index Action	3-6
Create Table Action	3-7
Delete Record Action	3-8
Drop Table Action	3-10
Insert Record Action	3-11
MapInsert Record Action	3-13
SQL File Action	3-15
SQL Statement Action	3-17
Update Record Action	3-19

About This Manual

Event handlers and event actions are where most of the work is accomplished in a transformation. Using event handlers and event actions in Map Designer is important for the following reasons:

- You can receive notifications from the engine that certain things have occurred.
- Do pre-processing or post-processing of records.
- Perform memory management tasks, such as initializing memory and cleaning up memory.
- Create temporary lookup files before the transformation and then destroy them after the transformation runs.
- Let the transformation do the work instead of writing extensive RIFL code.

Some common scenarios using events include the following:

- Do an update in a Source event action. If the update fails, it can trigger an `OnMismatchError` event.
- Use `OnMismatchError` if you are inserting records, and set up rejects for mismatched values.
- Include the `BeforeFirstRecord` and `AfterFirstRecord` events in a transformation to look for header information, and to check if the source file is correct or not.
- Use `OnDataChange` events to monitor parent and child relationships.
- Using an `OnEOF` event, keep a running total of all records processed, and send the value to a file or log. For instance, if you expect 127 records and only receive 126, read the log file, and determine why the record is not written.

This documentation includes the following chapters:

“Event Handling” on page 1-1

“Event Actions” on page 2-1

“Multimode Event Actions” on page 3-1

Event Handling

1

Executing Actions During Transformations

Transformation events are events related to the transformation as a whole. These events occur at certain defined points in a transformation, including before, after, upon aborting, and upon errors. Event handlers are paired with event actions. For instance, you can set a ChangeSource action in the BeforeTransformation event handler.

This chapter includes the following sections:

- “Transformation Event Handlers”
- “Event Precedence”
- “Source General Event Handlers”
- “Target General Event Handlers”
- “Record Type Event Handlers”

Transformation Event Handlers

The event handlers in Map Designer are designed to allow tremendous flexibility in the handling of data, and are triggered at several points in a transformation. The event handlers are as follows:

“BeforeTransformation”
“AfterTransformation”
“OnAbort”
“OnAssertionError”
“OnError”
“OnRecordValidationError”

You can customize the transformation process by setting actions to occur when a particular event is triggered. Some of those actions are the following:

- Log messages
- Execute expressions
- Trace errors
- Manipulate data
- Clear memory
- End or abort the transformation

You have control over when these actions occur, what actions occur, and how many actions occur.

The advantage of event handling is that complex transformations with multiple record types on both source and target can be accomplished. Complex data manipulations, such as record aggregation, unrolling of data, transposing of data, are fairly simple. Event handling allows you much of the flexibility, and customizability that you would get from a custom coded solution, without the hassle of building a custom program yourself every time you want to transform data.

Transformation event handlers include the following:

Name	Description	Possible Uses
BeforeTransformation	Triggered after the transformation is initialized and before anything else is done, this is the first event in every transformation. Note: When running a transformation in Clear File/Table or Replace File/Table mode, the Target table is cleared and recreated prior to execution of the BeforeTransformation event.	Changing source or target connections, setting up dynamic SQL lookups, creating new indexes, creating or dropping tables, inserting or deleting records, and declaring object variables.
AfterTransformation	Triggered after all records have been processed and just before the transformation is terminated. This is the last event in every transformation.	Summing final totals in record aggregation
OnAbort	Triggered when the transformation is aborted.	Logging abort conditions
OnAssertionError	Triggered on the failure of an Assert action. This event handler is specific to either the source or the target. For example, if you set an event on the target side, it does not fire if an Assert action fails on the source side.	Exception error handling
OnError	Fired when an error of any type occurs.	Error handling
OnRecordValidationError	Triggered on the failure of a Validate Record action.	

See Also

“Event Actions List”

“Event Precedence”

Events

Events are opportunities that can be used within the transformation cycle. For example, if you want something to happen as soon as a record is read into the transformation, you choose the AfterEveryRecord event, and then choose the action you want to happen at that time.

Actions

Actions are things that you want to happen. For example, if you want a message to be placed in the error and event log every time a record is discarded, choose the OnDiscard Target event and the LogMessage action. Most actions can be added to any event that you choose.

For a complete list of available actions, see “Event Actions List”.

***Order of Event
Action
Execution***

Event actions are processed in a particular order dependent on the event. If multiple actions are associated with a single event handler, the actions are executed in the order in which they are defined. Events are triggered based on a precedence scheme. For more information on order of execution, see “Event Precedence”.

You can choose to have more than one action occur during a particular event, and you can choose to have the same action occur during more than one event. There are no restrictions on the number or type of actions.

Event Precedence

Every transformation consists of a series of events that run in a particular order. Understanding the flow of event precedence is important so that you can create an effective transformation design with events firing in the correct order.

Source Event Precedence

The following table shows the event precedence for all source events in the order in which they fire.

Event Type	Order of Precedence	Event Name
Source events	1	BeforeTransformation
	2	Source Filter Expressions (for each record)
	3	Source Sort (for each record)
	4	Source Filter Range (for all records)
	5	BeforeFirstRecord
	6	BeforeEveryRecord
Source General events	7	BeforeFirstRecord
	8	BeforeEveryRecord
Source onDataChange events	9	OnAnyDataChange
	10	OnAllDataChange
	11	OnDataChange events 1-5
Source events	12	AfterFirstRecord
	13	AfterEveryRecord
	14	AfterFirstRecord
Source General events	15	AfterEveryRecord
	16	On End of File
Source events	17	AfterTransformation

In the onDataChange events 1-5, if no event 1 is set, event 2 is fired first, event 3 next, and so on. You can rearrange actions by dragging action names within the grid on the Map All tab.



Note The AfterFirstRecord Source event is fired **after** onDataChange events. If you have imported .djs files from previous versions, you may have to change how your events fire so that they are consistent with the current event precedence.

Moving From Source to Target or Error Events

From any point within the source flow of events, you may direct the flow to a sublevel of either target or error event precedence (see below). Actions such as Map, Put Record, ClearMapPut Record, and MapPut Record transfer the flow to the target event precedence. Any error that occurs transfers the flow to the error event precedence unless captured at the expression level using the On Error GoTo trapping.

Target Event Precedence

The target event precedence consists of two separate subroutines, Map and Put. Subroutines are a portion of code within a larger program. If Map Designer encounters an error within the Map subroutine, it disregards the Put subroutine unless told (within the error event precedence) to Put and then Resume. If told to Put without the Resume, Map Designer writes data to your target file and stops. No further data is processed. If Map Designer encounters an error within the Put subroutine, it uses error event precedence, which requires a Resume action to return to transformation event precedence.

The following table shows the event precedence for all target events in the order in which they fire.

Event Type	Order of Precedence	Event Name
Mapping subroutine	1	Target General Before Map event
	2	Target Record Before Map event
	3	Map Action
	4	Target Record After Map event

Event Type	Order of Precedence	Event Name
	5	Target General After Map event
Filter expressions and sampling	6	Target filter expressions
	7	Target filter samples
Put subroutine	8	Target General Before Put Record event
	9	Target Record Before Put Record event
	10	Put action
	11	Target Record After Put Record event
	12	Target General After Put Record event

Error Event Precedence

The following table shows the event precedence for all error events in the order in which they fire.

Event Type	Order of Precedence	Event Name
Expression-level	1	Expression error (for example, On Error Goto statement)
Source or Target record-level	2	Source record error or Target record error
Source or Target General event - level	3	Source General event error or Target General event error
Target record events	4	Target record OnError
Target general events	5	Target General OnError
Transformation-level error events	6	Transformation-level Error



Note If Map Designer encounters an error in the error event precedence, the transformation aborts. You cannot trap errors within an error event.

**Reaching
Transformation
End**

Once Map Designer finishes processing the events within the target sublevel, it automatically returns to the basic transformation event precedence at the point it left off. The transformation continues through that precedence until sent to another sublevel again, or until the transformation ends. However, once Map Designer finishes processing events within the error event precedence, it must have a Resume action to return to the transformation event precedence. If Map Designer does not encounter a Resume action, the transformation stops and no further records are processed.

See Also

“Record Type Event Handlers”

Source General Event Handlers

Source events are moments in the transformation timeline that occur during the reading of source data and before the transforming and writing of target data.

In about 80 percent of transformation designs, source events are used, while target events are used in the remaining 20 percent. This is because data manipulation generally occurs on the source side.

Event Name	Description	Uses
BeforeFirstRecord	Triggered before reading the first source record.	Executing actions once before any records are read, initialize variables, start Dynamic SQL connections.
AfterFirstRecord	Triggered after reading the first source record.	Executing actions after the first record is read. All records are read in this event, including the first record.
AfterEveryRecord	Triggered as soon as a new record is read and before any processing takes place. (If there is an error reading the new record, or if the end of file is reached, this event does not take place.)	Executing actions once per record read.
BeforeEveryRecord	Triggered before a new record is read and after all processing of the previous record is completed.	Initializing variables, starting Dynamic SQL connections.
OnAssertionError	Triggers when an Assert action statement returns false.	Verifying source field values.
OnEOF	Triggered when the end of the source file is reached.	Footers, totals, final record writing, ChangeSource action.
OnError	Triggered when an error occurs reading a source record.	Error handling, ChangeTarget action.
OnRecordValidationError	Triggers when a validation rule returns false.	Verifying data against a schema.
OnTruncateError	Triggered when a data truncation error is detected.	Handling truncation errors.

Target General Event Handlers

Target events are moments in the transformation timeline that occur during the transformation and writing of data to the target.

Event Name	Description	Possible Uses
AfterDeleteRecord	Triggered after a Delete Record action is fired.	
AfterInsertRecord	Triggered after an Insert Record action is fired. This event is not valid without a ClearMapInsert or MapInsert Record action.	
AfterMap	Triggered after a Map action is fired.	Clearing map expression variables.
AfterPutRecord	Triggered after a Put Record action is fired.	Logging successful writing of a record.
AfterPutTree	Triggered after a Put Tree action is fired.	
AfterUpdateRecord	Triggered after an Update Record action is fired.	
AfterUpsertRecord	Triggered after the Upsert Record action is fired. See Also: "Trapping Key Mismatch Errors".	
BeforeDeleteRecord	Triggered before a Delete Record action is fired.	
BeforeInsertRecord	Triggered before an Insert Record action is fired.	
BeforeMap	Triggered before a Map action is fired.	Initializing map expression variables needed for Map expressions
BeforePutRecord	Triggered before a Put Record action is fired.	Doing final validation before putting data in a target file or table.
BeforePutTree	Triggered before a Put Tree action is fired.	
BeforeUpdateRecord	Triggered before an Update Record action is fired.	Update record, and conditionally Insert based upon matching records

Event Name	Description	Possible Uses
BeforeUpsertRecord	Triggered before an Upsert Record action is fired. See Also: "Trapping Key Mismatch Errors".	Update record, and conditionally Insert based upon matching records
OnAbort	Triggered when a transformation is aborted.	Logging abort conditions.
OnAssertionError	Triggered when an Assert action fails. This event handler is specific to the target (for example, if you set an event on the target side, it does not fire if an Assert action fails on the source side).	Exception error handling.
OnDiscard	Triggered when a source record is deleted from the transformation and not written to any target file.	Logging discards, aborting or terminating when a discard is done.
OnError	Triggered when any type of error in the target occurs.	Error handling.
OnRecordValidationError	Triggered when the content of a record fails validation.	
OnReject	Triggered when a record is written to a specified reject file.	Logging record rejects, aborting or terminating when a record is rejected.

Trapping Key Mismatch Errors

The BeforeUpsertRecord and AfterUpsertRecord actions are actually combination actions. They are equivalent to doing an Update Record, followed by a conditional Insert Record. It is conditional because the Insert is done only when the Update does not match any records.

Sometimes using UpsertRecord actions may be more efficient than using the OnMismatch Error to trap key mismatch errors from the Update Record, and then calling Insert Record from the error handler.

Some actions are specific to certain connectors. See the following table for more information on these connectors.

Name	Description	Supported Connectors
OnConstraintError	Triggered when a constraint error occurs, or an integrity rule is violated.	<p>ODBC and ODBC Multimode</p> <p>ODBC does not distinguish between key and null value violations. The OnConstraintError event should catch both types of errors as well as other integrity violations.</p>
OnDuplicateKeyError	Triggered when a duplicate key error occurs. (These errors occur when you try to insert a duplicate key in an indexed target where indexes must be unique.)	Access, Oracle, Oracle Multimode
OnMismatch	Triggered when a mismatch occurs because of an UpdateRecord or DeleteRecord error event. UpdateRecord or DeleteRecord fails when there is no record in the target that matches the key of the record being updated or deleted.	All multimode connectors
OnNullValueError	Triggered when a null value is detected. Null errors occur when a null value is placed into a target field that does not allow null values.	Access, Oracle, Centura SQL Base, SQL Server, Sybase SQL Server, Sybase SQL Anywhere
OnOverflowError	Triggered when a data overflow error occurs. Overflow errors occur when the source data is numeric and contains more digits than the target field's precision.	All multimode connectors
OnTruncateError	Triggered when a data truncation error occurs. Truncation errors occur when the source data is text and contains more characters than the target field's size.	Access, ASCII (Delimited), ASCII (Fixed), BCP, Oracle, Oracle Direct Path, ODBC, SQL Loader

Record Type Event Handlers

Source or target events that relate to a specific record type are called "record type" event handlers. This helps distinguish them from other event handlers that have a scope covering the entire transformation. Recognition rules for each record type work together with record type event handlers and their event actions. Whenever a record is read from source or written to target, the event handlers and actions fires for that particular record type.

Many of the record specific events are the same as for the general source or target events. The difference is that if your source has three record types (for example, R1, R2 and R3) and you want an action to happen after any one of the three type records is read, add the action in the General Source AfterNextRecord event.

If you want something different to happen when each of the three types is read, add an action to each one of the record specific events:

- R1 AfterNextRecord event
- R2 AfterNextRecord event
- R3 AfterNextRecord event

If you want only one thing different for R2 followed by the same action for all three record types, add a specific action to R2 AfterNextRecord event. Then, add another action under the general Source AfterNextRecord event. Remember the event precedence order discussed in "Event Precedence".

Name	Description	Possible Uses
AfterFirstRecord	An event triggered immediately after the first record of this type is read	To trigger a ChangeSource or ChangeTarget, Validation
AfterEveryRecord	As soon as a record of this type is read before it is processed	Needed to handle multiple record types. This is where actions associated with any particular record type are placed to answer the question, "What should be done with a record of this type?"
BeforeFirstRecord	An event triggered before the first record of this type is read	To trigger a ChangeSource or ChangeTarget, Validation

Event Handling

Name	Description	Possible Uses
BeforeEveryRecord	An event triggered before the next record of this type is read and after all processing of the previous record of this type is completed	Needed to handle multiple record types. This is where actions associated with any particular record type are placed to answer the question, "What should be done with a record of this type?"
OnDataChange1 OnDataChange2 OnDataChange3 OnDataChange4 OnDataChange5	To set the action taken when the monitored data changes. onDataChange1 is the first event to be triggered, onDataChange2 is the second event, and so on.	Allows you to monitor changes in your data. For instance, you could monitor a list of sales accounts by state to determine which states were buying the most of your product in a given period.

Event Actions

A Guide to Event Actions in Map Designer

Event actions are actions that you specify to take place at a specific moment in the transformation. Which event they are associated with determines when they are executed. For instance, if you place a `TraceOn` Action in the `BeforeTransformation` event handler, the `TraceOn` action fires before the transformation is run.

For details on which events fire in what order, see “Event Precedence”.

This chapter includes the following sections:

- “Event Actions List”
- “Abort Action”
- “Assert Action”
- “ChangeSource Action”
- “ChangeTarget Action”
- “Clear Action”
- “Clear Tree Action”
- “ClearInitialize Action”
- “ClearMap Action”
- “ClearMapPut Record Action”
- “Execute Action”
- “LogMessage Action”
- “LogTargetRecord Action”
- “Map Action”
- “MapPut Record Action”
- “OnRecordValidationError Event”
- “Put Record Action”
- “Put Tree Action”

Event Actions

- “QueryStatistic Action”
- “Resume Action”
- “Terminate Action”
- “TraceOff Action”
- “TraceOn Action”
- “Upsert Record Action”
- “Validate Record Action”

Event Actions List

The following event action list is for single-mode connectors. For multimode connector actions, see “Multimode Event Actions List”.

Use CTRL+F to find information on a desired event action.

Name	Description
“Abort Action”	Causes the transformation to abort. If the transformation is transactional, the transaction is rolled back. If there is an Action associated with the AfterTransformation event, it is not executed.
“Assert Action”	Executes an expression that you specify. Similar to the “Execute Action”, with one exception – if the result of an expression is a False Boolean value, an OnAssertionError event is fired.
“ChangeSource Action”	Changes the source connections in transformation processing. Several source connections can be made in a single transformation using this action.
“ChangeTarget Action”	Changes the target connections in transformation processing. Several target connections can be made in a single transformation using this action.
“Clear Action”	Clears the contents of a target record memory buffer.
“ClearInitialize Action”	Clears the contents of a target record buffer and initializes non-null fields with "0" values.
“ClearMap Action”	When a new record is read, the data stored in the target memory buffer from the previous record is cleared, and new field data is manipulated and assigned to target fields.
“ClearMapPut Record Action”	<p>Combines the functionality of Clear, Map, and Put into one Action. (The target record buffer is cleared, target field expressions are executed, and the resulting data is written out to the target file.)</p> <p>Note: Before you run the transformation, you will not see data in the Results column of the Target Record Layout grid unless you have defined a Put action in source events. The automatic ClearMapPut does not take effect until the transformation is run. To see results for multiple source/target records, you must have rules defined to see results for only the records that are identified, mapped, and written to the target record.</p>

Name	Description
"Clear Tree Action"	Similar to the "Clear Action", with one exception. Clear Tree clears the contents of the specified record, walks the tree of child records, and finally clears the record buffer.
"Execute Action"	Executes an expression that you specify. Can also be used to call external .dll files and ActiveX components.
"LogMessage Action"	Writes a message to the error and event log file.
"LogTargetRecord Action"	Writes current target record field values into the error and event log file. (This is very useful for diagnostic purposes.)
"Map Action"	Executes target field expressions and maps the results of the expressions to the target buffer. The target field expressions usually include data from source fields.
"MapPut Record Action"	<p>Executes target field expressions, assigns the resulting data to the target buffer and writes the data, but does not clear the buffer. If data from previous records is assigned to fields that do not exist in the new record, all of that data is written along with the data from the new record.</p> <p>Before you run the transformation, you will not see data in the Results column of the Target Record Layout grid unless you have defined a Put action within your source events. The automatic ClearMapPut does not take effect until the transformation is run. To see results for multiple source/target records, you must have rules defined, and you will see results for only the records that are identified, mapped, and written to the target record.</p>
"Put Record Action"	<p>Takes data from target buffer and writes to the file or table.</p> <p>Before you run the transformation, you will not see data in the Results column of the Target Record Layout grid unless you have defined a Put action within your source events. The automatic ClearMapPut does not take effect until the transformation is run. To see results for multiple source/target records, you must have rules defined, and you will see results for only the records that are identified, mapped, and written to the target record.</p>
"Put Tree Action"	A recursive tree-walking variant of the Put Record action. In Put Tree, Boolean expressions (Put rules) are used to control whether or not a particular record is included as part of the general tree operation. This action allows you to collect data in memory in the target buffer. Then you can decide to write the data to the target or to clear the data. See Also: "Put Record Action"

Name	Description
"QueryStatistic Action"	Returns a user-specified run time statistic to a variable. Once populated by this action, the related variable may then be used to populate a message box, message object, or a report file.
"Resume Action"	Causes the transformation to continue normally after an error is detected.
"Terminate Action"	Ends the transformation without aborting it. If the transformation is transactional, it does not roll back, but ends normally. If there is an action associated with the AfterTransformation event, it is executed.
"TraceOff Action"	Turns off error tracing and returns the logging to its normal state.
"TraceOn Action"	Turns on error tracing so that details about each record transformation are written in the error and event log file.
"Upsert Record Action"	Combines two actions into one action. It is equivalent to doing an Update Record, followed by a conditional Insert Record, inserted only when Update does not match any records.
"Validate Record Action"	This action executes the validation rule for the specified source or target record. Validate Record raises an OnRecordValidationError event if the validation rule returns False. See Also: "OnRecordValidationError Event"

Abort Action

The Abort event action causes the transformation to abort. It can also write a message in the log if you wish.

If the transformation is transactional, the transaction is rolled back. If there is an action associated with the AfterTransformation event, it is not executed.

Parameters	Description
Message	<p>What is written to the error and event log when the transformation is aborted.</p> <p>Not required.</p> <p>Options: Any written message up to 255 characters.</p> <p>The message you type is passed to the Event Actions Parameters Values column, and appears in brackets in the Parameters column.</p>
Example	
Situation	You would like the transformation to abort if any source data fields are truncated.
Action	<p>Choose the target event OnTruncationError.</p> <p>Choose the Abort action.</p> <p>Set the Message parameter to "Data was truncated." by typing that in the cell.</p>
Result	If a text field is read that has too many characters for the target field, a message is written to the error and event log that says "Data was truncated." and the transformation is aborted.

Assert Action

The Assert action is similar to the “Execute Action”, with one exception. If the result of an expression is a False value, an OnAssertionError event is fired.

Assertions allow more control over the handling of exceptions. Using the Reject function gives you control over writing records, but does not allow control of events.

Parameters	Description
Expression	True is the default option. Choose False to fire an OnAssertionError event.

ChangeSource Action

The ChangeSource event action changes your source connection. It allows you to change source files (or tables) during a transformation.



Caution Actions set in an event handler after the ChangeSource action are ignored. If you have actions that must be fired by the same event handler, you must set these first.

Parameters	Description
Source Name	Select source from the list. Required. Options: Source – the normal source file.
Connection String	The connection string to the source file. Required. Example: The following connection string specifies a connection to the database "test.mdb" and to table "tutor1". Note: In the example below, the lines of code are shown with breaks at the semicolon (;). Actual connection strings must be one continuous line with no line breaks. Database=test.mdb;UserID=Admin; Table=tutor1;WhereStmt=SystemTables=False; Views=True;CodePage=ANSI;Password=; SystemDB=system.mdw
Example 1	Choose an event handler, such as BeforeTransformation, and select the ChangeSource Action. Enter the following connection string: "+File=C:\MyData\testfile.asc" If the path uses spaces, three options are recommended, in descending order of best practice: <ul style="list-style-type: none"> • Use a variable based on a macro for the path string. • Map to a network share name without spaces. • Enclose the path in single quotation marks: "+File='C:\My Data\testfile.asc'"

Plus Sign Usage in String Syntax

If the string starts with a plus sign (+), the contents of the connect string are used to update the current connection settings instead of doing a complete replacement. If the previous connection string was used to define a connection to table "tutor1", you could switch to "tutor2" with the connect string: +Table=tutor2

Example 2

To create a different source file for each run of a given key in the source file, add the following to an *OnDataChange* event handler:

ChangeSource Connection String:

```
" +File=" & MyVariable & ".asc"
```

This would use the current contents of MyVariable concatenated with ".asc" to build a new file name. Since the connection string was prefixed with a plus (+) sign, all of the other connection settings come from the previous connection (if there was one), or it has the Connector defaults.

To make this action conditional, simply pass an empty connection string and it does not fire.



Note If you run a ChangeSource in OnEOF, it jumps into an infinite loop. To avoid this issue, place a flow control in RIFL Script Editor.

ChangeTarget Action

The ChangeTarget event action changes the target connection information in transformation processing. It allows you to change target files (or tables) during a transformation. ChangeTarget is well suited for splitting output based on logical data control breaks using the OnDataChange event handler.



Caution Actions specified in an event handler after the ChangeSource action are ignored. If you have actions that must be fired by the same event handler, specify these first.

Parameters	Description
Target Name	The name target is inserted as a default. Required. Options: Target (default) or Reject.
Connection String	A script for building the connection string. Required. Example: The following connection string sets a connection to the database test.mdb and to table tutor1. <code>Database=test.mdb;UserID=Admin;Table=tutor1;Where Stmt=SystemTables=False;Views=True;CodePage=ANSI; Password=;SystemDB=system.mdw</code>

Parameters	Description
Append	<p>Append output to an existing target file (or table) with the specified name. The default is False.</p> <p>An expression can be used. Alternatively, you can build a script using RIFL Script Editor if you want to change the mode dynamically. The script must evaluate as false (or 0) if the file is to be replaced or true (any nonzero value) if you want to append to it.</p> <p>Example: You have a transformation that writes to two target files, one that contains records specific to this run and a second that accumulates information for all transformations of this type. You create a script that tells Map Designer to write or replace the first target file and append records to the second.</p>
Clear Records	<p>Clears record values from the target record buffer. The default is True. This parameter is ignored if the target connector of the action is not the target connector (for example the Reject connector). Performing a ChangeTarget action on the Reject connector does not clear the record buffer.</p>

Plus Sign Usage in String Syntax

If the string starts with a plus (+) sign, the contents of the connect string are used to update the current connection settings instead of doing a complete replacement. If the previous connection string was used to define a connection to table "tutor1", you could switch to "tutor2" with the connect string "+Table=tutor2".

Example 1

ChangeTarget Connection String:

```
" +File=C:\MyData\testfile.asc"
```

If the path uses spaces, three options are recommended, in descending order of best practice:

- Use a variable based on a macro for the path string.
- Map to a network share name without spaces.
- Enclose the path in single quotation marks: "+File='C:\MyData\testfile.asc'"

Example 2

To create a different target file for each run of a given key in the target file, add the following to an **OnDataChange** event handler:

ChangeTarget Connection String:

```
" +File=" & MyVariable & ".asc"
```

This uses the current contents of MyVariable concatenated with ".asc" to build a new file name. Since the connection string is prefixed

with a plus (+) sign, all of the other connection settings come from a previous connection, or have the Connector defaults.

Example 3

This example examines representing hexadecimal values in a ChangeTarget connection string. You might want to reference Connector properties in a ChangeTarget connection string, and sometimes hexadecimal representations are a requirement of the Connector property. For instance, a good example is the EDI SegmentTerminator property. Hexadecimal values for properties must be quoted with double quotation marks. In addition, since the string passed to ChangeTarget is an expression, the quoted part needs to be double-double quoted, as in the following example:

```
"File=xyzyzy.out;RecordSeparator=""\x0d\x0a"""
```



Note You must use the \xdd escape sequence for each hexadecimal character. In addition to hexadecimal strings, C-language style escape sequences for some common control characters are also supported. These include carriage return (\r), tab (\t), and new line or line feed (\n).

The following is equivalent to the previous example:

```
"File=xyzyzy.out;RecordSeparator=""\r\n"""
```



Tip To make this action conditional, pass an empty connection string and it does not fire.

Clear Action

The Clear event action clears all data from the specified source or target record that is held in memory. The data is held from all fields in previous records in a memory buffer, until it is either replaced by new data from a new input record or it is cleared by this action.

Source records (not just target records) can now be cleared. This can be useful when there are optional elements of a parent and child relationship, and you want to discard the contents of child records from the previous parent.

Parameters	Description
Source/Target Name	Name of the source or target file or table whose buffer you want cleared. Required. Options: Source – the normal source file or table. Target – the normal target file or table.
Record Layout	Name of the source or target layout whose buffer you want cleared. Required. Options: In most cases, the only choice is Record1 (R1). If you have specified multiple target record layouts and recognition rules, each layout is listed as a choice in the list.
Clear Fields	Boolean expression that indicates whether or not record layout field values are cleared. The default is True. Select False if you do not want to clear field values.
Clear Buffer	Boolean expression that indicates whether or not buffered output records are cleared. The default is True. Select False if you do not want to clear the record memory buffer.
Example	
Situation	You are storing accumulated data in a running total field. When you get a new header record, it indicates that the data following pertains to a different department. That running total needs to begin again from nothing.

Parameters	Description
Action	<ol style="list-style-type: none">1. Define a record layout with recognition rules that specify the header record, and name the layout "Department".2. Choose the AfterEveryRecord event in the Department Event Handlers list in the tree structure.3. Choose the Clear action.4. Set the Target Name parameter to Target.5. Set the Target Layout parameter to Record1 (R1).6. Leave the Clear Fields and Clear Buffer parameters set to True.
Result	<p>When the Department record is read, the Clear Action clears the Dependent Info fields and the target buffer.</p> <p>You may decide to clear only the field values and not the target buffer. Then, you can write data in the target buffer to a different field, such as Department Info History. In this case, you would set Clear Fields to True and Clear Buffer to False.</p>
Remarks	<p>Use this action for single record type files. For hierarchical files, see "Clear Tree Action".</p>

Clear Tree Action

The Clear Tree event action is similar to the “Clear Action”, with one exception. This action clears the contents of the specified record, and then walk and clear the tree of child records as well. Clear Tree is used for hierarchical records. For flat files, use the Clear Action.

Parameters	Description
Source/Target Name	Clears the contents of a target record buffer and initializes non-null fields with "0" values. Required.
Record Layout	Name of the source/target layout whose buffer you want cleared. Required.
Clear Fields	Boolean expression that indicates whether or not record layout field values are cleared. The default is True. Select False if you do not want to clear field values.
Clear Buffer	Boolean expression that indicates whether or not buffered output records are cleared. The default is True. Select False if you do not want to clear the record memory buffer.
Remarks	This action should be used on hierarchical data, not on flat files. Use the Clear action when you are working with single record type files.
Example	The following example is taken from the transformation sample “Using Buffered Put Tree to Create Hierarchical Records”. You can download the sample on the sample home page .
Situation	Before the sample map runs, a target XML file contains fields that are placeholders to store multiple records and the source file structure. When the transformation is successful, records are written to the appropriate fields and grouped in a hierarchical format.
Action	<ol style="list-style-type: none"> 1. Set up a Source onDataChange1 event to define the actions to take place when the Data Change Monitor value changes. Choose onDataChange1 in the R1 Event Handlers list. 2. Choose the Clear Tree action. 3. Set the Target Name parameter to Target. 4. Set the Target Layout parameter to Dependent Info. 5. Leave the Clear Fields and Clear Buffer parameters set to True.

Event Actions

Parameters	Description
Result	<p>When the Dependent Info record is read, the ClearTree Action clears the current tree, clearing the Dependent Info fields and the target buffer.</p> <p>You may decide to clear only the field values and not the target buffer. Then, you can write data in the target buffer to a different field, such as Dependent Info History. In this case, you would set Clear Fields to True and Clear Buffer to False.</p>
Remarks	<p>Use this action for multiple record type files. For single record type files, see "Clear Action".</p>

ClearInitialize Action

This action clears the buffer of target records and initializes the non-null field values. The ClearInitialize action is similar to the Clear Action, however Clear initializes all fields in the specified record layout to null values, while ClearInitialize initializes all values to zero. By default, the action initializes numeric and text fields with a zero value.

Parameters	Description
Target Name	<p>Clears the contents of a target record buffer and initializes nonnull fields with zero values.</p> <p>Required.</p> <p>Options: Target – the normal target file. This is the only choice in most cases.</p>
Record Layout	<p>Name of the target layout whose buffer you want cleared.</p> <p>Required.</p> <p>Options: In most cases, the only choice is Record1 (R1). If you have specified multiple target record layouts and recognition rules, then each layout is listed as a choice in the list.</p>
Default Value	<p>Specify a simple expression to define the initial value for non-null fields. Useful in situations where 0 is not a good initial value.</p> <p>Not required.</p> <p>Options: A simple expression can be any expression that does not reference Fields, for example: "1.25", "Date()", "Serial()"</p>
Example	
Situation	You do not want the target field to be initialized with a Null value, because the results returned to the target are Null.
Action	<p>This example shows a mapping expression for a target field "TrgField".</p> <pre>TrgField = Targets(0).Fields("SrcField") + Sources(0).Fields("SrcField")</pre>
Result	When a field is initialized with the ClearInitialize Action, the result is the value of "SrcField" added to the value of "TrgField".

ClearMap Action

In the ClearMap event action, when a new record is read, the data stored in the target memory buffer from the previous record is cleared, executes new data from target field expressions and maps the results of the expressions to the target buffer. This combines the functionality of the Clear action and the Map action. A Put Record action or Insert Record action can be used separately after this action.

Parameters	Description
Target Name	Name of the target file or table. Required. Options: Target – the normal target file. This is the only choice in most cases.
Record Layout	Name of the target layout to clear, and to have data written to it. Required. Options: In most cases, the only choice is R1 (for single, not multiple record layouts). If you have specified multiple target record layouts and recognition rules, then each layout is listed as a choice in the list.

Parameters	Description
Count	<p>Expression to evaluate the repetition count (enter a numeric value). Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <p>Leave the parameter blank (defaults to 1, writing data to the Target).</p> <p>Use a Private or Public variable (not Local variables, which are not recognized here).</p> <p>Use literal numerics (for example, enter a 3 to write the data 3 times).</p> <p>Use any expression that includes a numeric return value.</p> <p>Not required</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>
Counter Variable	<p>Name of the variable used to store the current repetition count.</p> <p>Ensure that the variable has been declared Private or Public (see above note in Count).</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>

ClearMapPut Record Action

The ClearMapPutRecord action is the default operation done automatically by Map Designer when a transformation is run without any events being specified.

When a new record is read, the following things happen:

- **Clear:** The data stored in the target memory buffer from the previous record is cleared.
- **Map:** Executes target field expressions and maps the results of the expressions to the target buffer.
- **Put:** Takes data from the target buffer and writes to the file or table.

The above actions combine the functionality of the Clear, Map, and Put Actions into this one action, so that you do not have to set each one separately. Of course, it is also possible to break the process down into separate or double actions. See “Clear Action”, “Map Action”, “Put Record Action”, “ClearMap Action”, and “MapPut Record Action”.

For a complete list of actions, see “Event Actions”.

Parameters	Description
Target Name	Name of the target file or table. This is the only choice in most cases. Required. Options: Target – The normal target file. This is the only choice in most cases. Reject – The reject records file as specified in the Reject Records tab of Transformation and Map Properties.
Record Layout	Name of the target layout to clear, and to write data to it. Required. Options: In most cases, the only choice is R1. If you have specified multiple target record layouts and recognition rules, then each layout is listed as a choice in the list.

Parameters	Description
Count	<p>Expression to evaluate the repetition count (enter a numeric value). Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <p>Leave the parameter blank (defaults to 1, writing data to the target).</p> <p>Use a Private or Public variable (not Local variables, which are not recognized here).</p> <p>Use literal numerics (for example, enter a 3 to write the data 3 times).</p> <p>Use any expression that includes a numeric return value.</p> <p>Not required.</p> <p>Options: Any positive integer, or expression that evaluates to a positive integer.</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>
Counter Variable	<p>Name of the variable used to store the current repetition count.</p> <p>Ensure that the variable has been declared Private or Public (see Note in Count above).</p> <p>Not required.</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>
Buffered	<p>The default option is False. If you select True, the output record is appended to the memory target buffer (instead of written to the physical target file).</p>
Situation	<p>You have a source file with two record types that you want combined into one record in the target.</p>

Parameters	Description
<p>Action</p>	<p>Create record layouts for both record types with recognition rules to distinguish them.</p> <p>Map all the fields from both source record layouts into a single target record layout.</p> <p>Select the AfterEveryRecord event from the Record Layout Events of the second source record layout.</p> <p>Select the ClearMapPut Record action.</p> <p>Set the Target Name parameter to target.</p> <p>Set the Target Layout parameter to R1.</p>
<p>Result</p>	<p>When the transformation is run, data from the first record type is held in the memory buffer. When a record of the second record type is read, it joins the data from the other record in the buffer, is assigned to appropriate fields, and all of the data is written as one record.</p>



Note When combining the fields of two record types, on AfterEveryRecord of the first record type, the buffered property has to be set to True in the ClearMapPut action.

This action can be used after the “Drop Table Action” and “Create Table Action” when using multimode connectors.

Execute Action

The Execute event action executes an expression that you specify. Execute can also be used to call external .dll files and ActiveX components.

Parameters	Description
Expression	The expression that you want to execute. Required. Options: This can be virtually any expression you can write in the Integration expression language. Stored procedures and ActiveX components should be called normally using standard expression syntax.

See Also

Search for the keywords “Using ActiveX” in the online help.

LogMessage Action

The LogMessage event action writes a message to the error and event log file.

Parameters	Description
Message	The message text that you want returned to the log file. Required. Options: Type any text that you want returned to the error and event log file.
Example	
Situation	You have source data that may contain more data in a text field than fits into the target field, therefore truncation may occur. You do not want the transformation to stop, but you would like to know when truncation occurs.
Action	Choose the OnTruncationError target event. Select the LogMessage Action. Type a message such as "Data has been truncated" in the Message parameter.
Result	Every time data is truncated due to a lack of space in the target field, the message "Data has been truncated" is written to the error and event log.

LogTargetRecord Action

The LogTargetRecord action writes current target record field values into the error and event log file. It is useful for diagnostic purposes.



Tip Before following these instructions, you must turn on debugging. In Transformation and Map Properties under Error Logging, select the Debug Messages check box so that this action writes messages to your log file.

Parameters	Description
Target Name	Name of the target file or table. Required. Options: Target – the normal target file. This is the only choice in most cases. Reject – the reject records file as specified in the Reject Records tab of Transformation and Map Properties.
Record Layout	Name of the target layout to log. Required.
Field Count	Number of target record fields to write to the log file. The default setting is to print all fields. Optional.

Options: In most cases the only choice is Record1 (R1). If you have specified multiple target record layouts and recognition rules, then each layout is listed as a choice.

Map Action

The Map event action executes target field expressions and maps the resulting data from the expressions to the target buffer. Here you can initialize variables needed for map expressions.

Parameters	Description
Target Name	Name of the target file or table. Required. Target – The normal target file. This is the only choice in most cases. Reject – The reject records file as specified in the Reject Records tab of Transformation and Map Properties.
Record Layout	Name of the target layout to map. Required. Options: In most cases the only choice is R1 (for single record layouts). If you have specified multiple target record layouts and recognition rules, each layout is listed as a choice in the list.

Parameters	Description
Count	<p>Expression to evaluate the repetition count (enter a numeric value). Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <p>Leave the parameter blank (defaults to 1, writing data to the target).</p> <p>Use a Private or Public variable (not Local variables, which are not recognized here).</p> <p>Use literal numerics (for example, enter a 3 to write the data 3 times).</p> <p>Use any expression that includes a numeric return value.</p> <p>Not required.</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>
Counter Variable	<p>Name of the variable used to store the current repetition count. Ensure that the variable has been declared Private or Public (see Note in Count above).</p> <p>Not required.</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>

MapPut Record Action

The MapPut Record event action executes target field expressions, assigns the resulting data to the target buffer, takes data from the target buffer, and writes to the file or table.

Parameters	Description
Target Name	<p>Name of the target file or table.</p> <p>Required.</p> <p>Options:</p> <p>Target – The normal target file. This is the only choice in most cases.</p> <p>Reject – The reject records file as specified in the Reject Records tab of Transformation and Map Properties.</p>
Record Layout	<p>Name of target layout to map and write.</p> <p>Required.</p> <p>Options: In most cases, the only choice is R1. If you have specified multiple target record layouts and recognition rules, then each layout is listed as a choice in the list.</p>
Count	<p>Repetition count the number of records to be written out for a single record read.</p> <p>Note: The Count parameter is a variable (the variable is already declared). You can specify 0 to suppress the Put, or a 1 to write the Put. Writing simple SQL expressions here (and in Counter Variable) give you precise control over what data is written. Useful when updating records in Update Mode.</p> <p>Not required.</p> <p>Options: Any positive integer, or expression that evaluates to a positive integer.</p> <p>See Also: Search for the keywords “Counter Variable” in the online help.</p>

Parameters	Description
Counter Variable	Name of the variable used to store the current repetition count. Ensure that the variable has been declared Private or Public (see Note in Count above). Not required. See Also: Search for the keywords "Counter Variable" in the online help.
Buffered	The default option is No. If you select Yes, the output record is appended to the memory buffer (instead of written to the physical target file).

OnRecordValidationError Event

The OnRecordValidationError event handler is raised if the validation rule for a record evaluates to False.

Parameters	Description
Expression	The expression that you want to execute. Required.

Put Record Action

The Put Record event action takes data from the target buffer and writes the data to the file or table.

Parameters	Description
Target Name	<p>Name of the target file or table.</p> <p>Required.</p> <p>Target – The normal target file. This is the only choice in most cases.</p> <p>Reject – The reject records file as specified in the Reject Records tab of Transformation and Map Properties.</p>
Record Layout	<p>Name of the target layout to be written.</p> <p>Required.</p> <p>Options: In most cases, the only choice is R1. If you have specified multiple target record layouts and recognition rules, then each layout is listed as a choice in the list.</p>
Count	<p>Expression to evaluate the repetition count (enter a numeric value). Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <ul style="list-style-type: none"> • Leave the parameter blank (defaults to 1, writing data to the target). • Use a Private or Public variable (not Local variables, which are not recognized here). • Use literal numerics (for example, enter a 3 to write the data 3 times). • Use any expression that includes a numeric return value. Not required. <p>See Also: Search for the keywords “Counter Variable” in the online help.</p>

Parameters	Description
Counter Variable	<p>Name of the variable used to store the current repetition count.</p> <p>Ensure that the variable has been declared Private or Public (see Note in Count above).</p> <p>Not required.</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>
Buffered	<p>The default option is No. If you select Yes, the output record is appended to the memory buffer (instead of written to the physical target file).</p>
Example	<p>To manually set the target buffer, add a Put record action with an Execute expression:</p> <pre>Targets (0) Records (R1) Fields ("Name") = "PM"</pre> <p>To reference the target buffer, add Targets(0) at the front of a default expression.</p>

Put Tree Action

The Put Tree event action is a recursive tree-walking variant of the “Put Record Action”, writing the buffered records to target. In Put Tree, Boolean expressions (Put conditions) are used to control whether or not a particular record is included as part of the general tree operation. See “Put Conditions”.

Parameters	Description
Target Name	Name of target file or table. Required. Options: Target – The normal target file. This is the only choice in most cases.
Record Layout	Name of the target layout to be written.

When using a Put Tree action for a hierarchical record layout (such as XML), you must specify the parent in the hierarchy, and then children are written in their appropriate order. Note that the AfterPutTree event handler is fired for the parent record layout after the entire tree is written.

When you use the Put Tree Action for a multirecord file with an implied hierarchical relationship, such as ASCII (Fixed), you must plan your map carefully. Map Designer does not define the "parent/child" relationship, so it cannot interpret the relationships between the parent and child tree buffers to write them out in a specific order. This logic must be handled manually.

A Put Tree Action should be followed by a Clear Tree Action to avoid writing duplicate entries. Enter the ClearTree Action into an AfterPutTree event handler.

When using a Put Tree Action for a reject record, follow the Put Tree Action with a Clear Tree Action for the same reject record in the same event.

When working with a hierarchical unit of work (UOW) using the PutTree Action, it is sometimes necessary to discard a branch of the UOW or the complete UOW based on the successful transformation

of the data within that branch or UOW. Use the put condition in the Target Schema Rules node to control this logic. The put condition should be specified at the upper level of the hierarchy for which the exclusion applies. For example, you have an XML file with the following hierarchy:

```
<Root>
  <Owner>
    <Name/>
    <Address/>
    <City/>
    <State/>
    <Zip/>
    <Vehicles>
      <Vehicle>
        <Year/>
        <Make/>
        <Model/>
        <Color/>
        <Style/>
      </Vehicle>
    </Vehicles>
  </Owner>
</Root>
```

To omit vehicles built before 1995, set up your put condition on the Vehicle record using the following expression:

```
Option EvaluateManual
Targets(0).Records("Vehicle").Fields("Year") >= "1995"
```

To omit all vehicles if any of the vehicles belonging to an owner were built before 1995, enter this expression in the put condition for the Vehicles record. Finally, to omit any owners with vehicles built before 1995, you would enter the expression in the put condition for the Owner record.

Put Conditions

Put conditions are conditional expressions used to control the behavior of the Put Tree action. Using the buffered put capability of the engine, a map processes a record and stores the result in a temporary tree of data with other related records. As the Put Tree

action scans the buffered data, it writes data to the target or skips it based on the put conditions.

Put conditions are useful when you are working with a group of similar records or with parent and child records, and you do not want to write the data until you have a complete set or until you have found out more about the data.

For example, you can set a put condition on a record type in the target schema so that those records are written only if a particular other source record type is found.

QueryStatistic Action

The Query Statistic event action returns a user-specified run time statistic to a variable. Once populated by QueryStatistic, the related variable may then be used, for example, to populate a message box, message object, or a report file.

QueryStatistic is a transformation action selected and configured in the Event Handler dialog box. QueryStatistic returns a value to a variable. The variable is also declared in the Event Handler dialog box. The value returned in each QueryStatistic action is one of the same run-time statistics recorded in the log file. One advantage of this action is that values are accessible within the transformation at run time. Using QueryStatistic, these values may be obtained without parsing the log file.

Statistic Name	Execution Statistic	Data Source
Reads	Total records read	source records read
Buffered Writes	Total records buffered	map events (buffered puts)
Writes	Total records written	map events (sent to target)
Inserts	Total records inserted	target
Updates	Total records updated	target
Deletes	Total records deleted	target
Discards	Total records discarded	map events
Rejects	Total records rejected	map events
Errors	Total error count	target (This is associated with the Writes event action and includes unhandled errors only.)
RowCounts	Total reject records written	map events (sent to reject file/table)

To use the QueryStatistic Action, first select an event handler. In the Event Handler dialog box, select the Event Name and the QueryStatistic Action. Configure the event action parameters in the next dialog display by choosing the value you want to return and declaring the variable name that contains this value.

Parameters	Description
Statistic Name	Name of the statistic query that you want to run. Required.
Variable Name	Name of the variable that contains the value you want to return. Required. Tip: Public process-level variables are not supported. If you need to pass the return value to a public process-level variable, in this parameter first specify a global variable that is not public and then use a RIFL script to assign the desired value of this variable.
Example	
Situation	At run time, you want to know when the first 100 records have been written to the target.
Action	This example shows a mapping expression for a target field "TrgField". Set the QueryStatistic action statistic name to "Writes". Set the variable name to "var". Set the target field expression contents for "TrgField" to: If var = 100 Then "Record100" End If
Result	After the 100th record writes to the target, "Record100" writes to "TrgField" in the 101st target record.
Returns	Buffered Writes Deletes Discards Errors Inserts Reads Rejects RowCount Updates Writes

Use the QueryStatistic Event Action for the following:

- Return transformation statistics at run time without parsing the log file
- Report run-time transformation milestones
- Trigger an action based on a transformation milestone

Resume Action

The Resume event action causes the transformation to continue normally after an error is detected.

Parameters	Description
None	
Example	
Situation	You do not want overflow errors to interrupt your transformation process.
Action	Select the OnOverflowError event. Select the Resume action.
Result	When a numeric field is encountered in the source data that has too many digits to fit in the target field, the transformation ignores the problem and continues converting the rest of the records.

Terminate Action

The Terminate event action ends the transformation. If the transformation is transactional, it does not roll back, but ends normally. If there is an action associated with the AfterTransformation event, it is executed.

Parameters	Description
<p>Message</p>	<p>Text written into the error and event log when the transformation is terminated.</p> <p>Not required.</p> <p>Options: Any written message up to 255 characters.</p> <p>The message you type is passed to the Event Actions Parameters Values column, and appears in brackets in the Parameters column.</p>
<p>Example</p>	
<p>Situation</p>	<p>You would like the transformation to end if it encounters a record that cannot be written to the target table.</p>
<p>Action</p>	<p>Choose the OnDiscard event.</p> <p>Choose the Terminate action.</p> <p>Type the message "A record was discarded." in the parameter value column.</p>
<p>Result</p>	<p>When the transformation is run, records are written normally to the target file until a record is discarded for any reason. At that time, the message "A record was discarded." is written in the error and event log, and the transformation ends. Any AfterTransformation event actions are executed and normal transformation termination routines will happen.</p>

TraceOff Action

The TraceOff event action turns off error tracing and returns the logging to its normal state. For single-mode connectors only.



Note Before following these instructions, you must turn on debugging. To do this, go to **Transformation and Map Properties > Error Logging** and select the **Debug Messages** check box. Now this action writes messages to your log file.

Parameters	Description
None	
Example	
Situation	You would like some more detailed information about what happens to the records right after the data changes in an account number field.
Action	TraceOff is used after the TraceOn event. See “TraceOn Action”. Choose the TraceOff Action after TraceOn so that trace logging is turned off just after the monitored record is read.
Result	When the data in the account number changes, detailed messages are written to the error and event log about what happened to the data. Once the TraceOff Action is applied, logging returns to normal (just before a new record is read).

➤ To debug using TraceOff and TraceOn actions

If you need to troubleshoot a transformation that is failing and not producing detailed log messages, follow these steps to set up debugging.

- 1 Clear your log file if necessary.
- 2 Open the transformation and select **View > Transformation and Map Properties**.
- 3 Select **Debug Messages** and **Flush log to disk after each message**.
This sets error logging to check debug messages and to return all messages to the log.

- 4** In a Before Transformation event, add a TraceOn action as the first action.
- 5** In an After Transformation event, add the TraceOn action as the last action.

The TraceOn and TraceOff actions send more details about each record transformation to the log.

- 6** Run the transformation and watch the memory usage.
- 7** Review the error log messages to determine why the transformation failed.

See Also

“TraceOn Action”

Search for the keywords “Viewing the Error and Event Log File” in the online help.

TraceOn Action

The TraceOn event action turns on error tracing so that details about each record transformation are written in the error and event log file. For single-mode connectors only.



Tip Before following these instructions, you must turn on debugging. To do this, select **Transformation and Map Properties > Error Logging** and then select the **Debug Messages** check box. Now this action writes messages to your log file.

Parameters	Description
None	
Example	
Situation	You would like some more detailed information about what happens to records right after the data changes in an account number field.
Action	<p>In the source event handlers (Map All tab, upper left quadrant), select the plus sign (+) next to Data Change Events.</p> <p>At Data Change Monitor, select the record you want to trace from the list, such as the Source "Account No" record name in the tutor1.asc tutorial file.</p> <p>Then select the Data Change Event Management Option that you need. The default is "Suppress First ODC event/Fire extra ODC event at EOF." For an explanation of all the options, see "Record Type Event Handlers".</p> <p>Under Action, click in the cell and select the ellipsis. Choose the TraceOn action from the list. Ignore the Value text box (nothing needs to be entered here). Select OK.</p> <p>Again, under Action, select the ellipsis under Value and choose the TraceOff Action to turn the trace logging off. Select OK.</p>
Result	When the data in the Account No changes, detailed messages are written to the error and event log about what happened to the data. The logging returns to normal just before a new record is read.

See Also

For an example on how to set up debugging in a transformation, see “To debug using TraceOff and TraceOn actions”.

Upsert Record Action

The Upsert Record event action combines two actions into one action. It is equivalent to doing an Update Record, followed by a conditional Insert Record. It is conditional because the Insert is done only when the Update does not match any records.

There are cases where using Upsert Record actions may be more efficient than using the OnMismatch Error to trap key mismatch errors from the Update Record and then calling Insert Record from the error handler.

The following table gives the action parameters and descriptions.

Parameters	Description
Target Name	Name of the target file. Required. Options: Target – The normal target file.
Record Layout	Names of the target layouts to write. Multiple record layouts may be selected. Required.
Table Name	Names of tables to write. SQL Script as a target – Refer to your SQL application to get table names, since no list is available for SQL Script. Required.

Parameters	Description
Count	<p>Expression to evaluate the repetition count (enter a numeric value). Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <p>Leave parameter blank (defaults to 1, writing data to the target).</p> <p>Use a Private or Public variable (not Local variables, which are not recognized here).</p> <p>Use literal numerics (for example, enter a 3 to write the data 3 times).</p> <p>Use any expression that includes a numeric return value.</p> <p>Not required.</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>
Counter Variable	<p>Name of the variable used to store the current repetition count.</p> <p>Ensure that the variable has been declared Private or Public (see Note in Count above).</p> <p>Not required.</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>

Validate Record Action

The Validate Record event action executes the validation rule for the specified source or target record. This action raises an `OnRecordValidationError` even if the validation rule returns `False`. See “`OnRecordValidationError` Event”.

Validation rules are Boolean expressions that can be associated with any source or Target Record Layout.

The Validate Record action can be used to build filters. To do this, write an expression that tests the record on your criteria, and returns a Boolean value. `True` passes validation, while `False` fails validation and triggers the `OnRecordValidationError`.



Tip Use the `OnRecordValidationError` action to set up error handling.

Parameters	Description
Source/Target Name	Name of the source or target record you want to validate. Required. Options: Source – the normal Source file. Target – the normal target file.
Record Layout	Name of the target layout you want to validate. Required.
Options	In most cases, the only choice is Record1 (R1). If you have specified multiple target record layouts and recognition rules, then each layout are listed as a choice in the list.
Example	Your database consists of records of clients throughout the nation, and you want to create a mailing list to target the central Austin, Texas market (ZIP Code 78701). Set <code>ValidateRecord</code> to the expression: <code>ZIP Code = 78701</code> To write only these records to target.

Multimode Event Actions

Multimode Connectors and Multimode Event Actions

When you use a multimode connector, a single data stream can be sent from multiple tables to a target database. You can perform multiple operations in one transformation, such as table drops, inserts, and updates directly on your target database.

Multiple output is the only output option for multimode connectors. This means that if you need to update, append, or delete records with a multimode connector, you need to use the multimode event actions outlined in this chapter.

- “Multimode Event Actions List”
- “ClearMapInsert Record Action”
- “Create Index Action”
- “Create Table Action”
- “Delete Record Action”
- “Drop Table Action”
- “Insert Record Action”
- “MapInsert Record Action”
- “SQL File Action”
- “SQL Statement Action”
- “Update Record Action”

Multimode Event Actions List

Event actions are actions that you specify to take place at a specific moment in the transformation. Which event they are associated with determines when they are executed. Multimode event actions can be used with multimode connectors only.

Use CTRL+F to find information on a desired event action.

Name	Description
"ClearMapInsert Record Action"	Clears the target record buffer, executes any map expressions, and inserts records into the target table.
"Create Index Action"	Creates new indexes with attributes specified in action parameters. Often used in a BeforeTransformation event.
"Create Table Action"	Creates new tables with attributes specified in action parameters. Often used in a BeforeTransformation event. It can be placed after a Drop Action event in a transformation designed to be run numerous times. In this scenario, the existing target tables with old data are dropped, then tables are created for new data.
"Delete Record Action"	Deletes records from multiple tables specified in the action parameters. Write to multiple tables by using this action in a BeforeTransformation event.
"Drop Table Action"	Drops tables specified in the action parameters. Often used in a BeforeTransformation event. It can be placed before a CreateTable Action event in a transformation designed to be run numerous times. In this scenario, the existing target tables with old data are dropped, then tables are created for new data.
"Insert Record Action"	Acts as a query statement so that records can be inserted into specified tables. Write to multiple tables by using this action in a BeforeTransformation event.
"MapInsert Record Action"	Executes target field expressions, assigns the resulting data to the target buffer, and inserts records into the target tables.
"SQL File Action"	Calls specified SQL files to be used in a transformation.

Name	Description
"SQL Statement Action"	Includes code that contains one or many SQL statements. These statements are written to fire off various events within your transformation, such as the Insert Action or Delete Action events. SQL commands, such as Truncate or Create Procedure, can be called in SQL Statement Action code.
"Update Record Action"	Updates records in multiple tables specified in the action parameters. Write to multiple tables by using this action in a BeforeTransformation event.

See Also

"Transformation Event Handlers"

"Event Precedence"

ClearMapInsert Record Action

This action can be used by multimode connectors only. The ClearMapInsert Record event action clears the target record buffer, executes any map expressions, and inserts records into the target table.

Parameters	Description
Target Name	The data target or reject target. Required. Options: Target – the data target or reject target.
Record Layout	Name of the target layout. To insert to more than one target layout, add additional event actions to your transformation. Required.
Table Name	Name of table. To insert to more than one target table, add additional event actions to your transformation. SQL script as a target: Refer to your SQL application to get table names, since no list is available for SQL script. Required.

Parameters	Description
Count	<p>Expression to evaluate the repetition count (enter a numeric value). Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <ul style="list-style-type: none"> • Leave the parameter blank (defaults to 1, writing data to the target). • Use a Private or Public variable (not Local variables, which are not recognized here). • Use literal numerics (for example, enter a 3 to write the data 3 times). • Use any expression that includes a numeric return value. <p>Not required.</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>
Counter Variable	<p>Name of the variable used to store the current repetition count. Ensure that the variable is declared Private or Public (see Note in Count above).</p> <p>Not required (unless you write an expression in Count).</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>

Create Index Action

The Create Index action can be used by multimode connectors only. It creates new indexes with attributes specified in Action parameters. Create Index is often used as a BeforeTransformation event.



Note If single-mode connectors are used in a transformation and this action is selected, it is ignored in the transformation process. This action is for multimode connectors only.

Parameters	Description
Target Name	The data target or reject target. Required. Options: Target – the data target or reject target.
Record Layout	Name of the target layout. To use more than one target layout, add additional event actions to your transformation. Required.
Table Name	Name of table to create an index. To create an index in more than one target table, add additional event actions to your transformation. Note: SQL Script as a target: Refer to your SQL application to get table names, since no list is available for SQL Script. Required.
Index Name	Name of the index to create. Select the ellipsis to open the RIFL Script Editor. Type in a SQL statement to name the index. Required.
Unique	Unique Key. Select True or False from the list. Not required.

Create Table Action

This action can be used by multimode connectors only. It creates new tables with attributes specified in Action parameters. The Create Table event action is often used in BeforeTransformation events. It can be placed after a Drop Table action event in a transformation designed to be run numerous times. In this scenario, the existing target tables with old data are dropped, then tables are created for new data.



Note If single-mode connectors are used in a transformation and this action is selected, it is ignored in the transformation process. This action is for multimode connectors only.

Parameters	Descriptions
Target Name	The data target or reject target. Required. Options: Target – the data target or reject target.
Record Layout	Name of the target layout to add. To add more than one target layout, add additional event actions to your transformation. Required.
Table Name	Name of table to create. To create more than one target table, add additional event actions to your transformation. Note: SQL Script as a target: Refer to your SQL application to get table names, since no list is available for SQL Script.

Delete Record Action

The Delete Record event action may only be used by multimode connectors. It deletes records from multiple tables specified in the action parameters. You may write to multiple tables by using this action in a BeforeTransformation event.

Parameters	Description
Target Name	The data target or reject target. Required. Options: Target – the data target or reject target.
Record Layout	Name of the target layout to delete. To delete more than one target layout, add additional event actions to your transformation. Required.
Table Name	The name of the table from which records are deleted. To select more than one target table, add additional event actions to your transformation. For SQL Script as a target, refer to your SQL application to get table names, since no list is available for SQL Script. Required.

Parameters	Description
Count	<p>Expression to evaluate the repetition count (enter a numeric value). Select the ellipsis in the Value column of the Event Action Parameter dialog box to display the RIFL Script Editor to write your expression. Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <p>Leave the parameter blank (defaults to 1, writing data to the target).</p> <p>Use a Private or Public variable (not Local variables, which are not recognized here).</p> <p>Use literal numerics (for example, enter a 3 to write the data 3 times)</p> <p>Use any expression that includes a numeric return value.</p> <p>Not required.</p>
Counter Variable	<p>Name of the variable used to store the current repetition count. Select the ellipsis in the Value column of the Event Action Parameter dialog box to display a text box to enter the Counter variable text value.</p> <p>Ensure that the variable has been declared Private or Public (see Note in Count above).</p> <p>Not required.</p> <p>See Also: Search for the keywords "Counter Variable" in the online help.</p>

Drop Table Action

The Drop Table event action can be used by multimode connectors only. It drops tables specified in the action parameters. This action is often used as a BeforeTransformation event. It can be placed before a CreateTable Action event in a transformation designed to be run numerous times. In this scenario, existing target tables with old data are dropped, then tables are created for new data.



Note If single-mode connectors are used in a transformation and this action is selected, it is ignored in the transformation Execute Action

This action executes an expression that you specify. Execute can also be used to call external .dll files and ActiveX components.

Parameters	Description
Target Name	The data target or reject target. Required. Options: Target – the data target or reject target.
Table Name	Name of table to drop. To drop more than one target table, add additional event actions to your transformation. SQL Script as a target: Refer to your SQL application to get table names, since no list is available for SQL Script.

Insert Record Action

The Insert Record event action may only be used by multimode connectors. It acts as a query statement so that records can be inserted into specified tables. You may write to multiple tables by using this action as a BeforeTransformation event.

Parameters	Description
Target Name	The data target or reject target. Required. Options: Target – the data target or reject target.
Record Layout	Name of the target layout. To use more than one target layout, add additional event actions to your transformation. Required.
Table Name	Name of table in which to insert records. To insert to more than one target table, add additional event actions to your transformation. For SQL Script as a target, refer to your SQL application to get table names, since no list is available for SQL Script. Required.

Parameters	Description
<p>Count</p>	<p>Expression to evaluate the repetition count (enter a numeric value). Select the ellipsis in the Value column of the Event Action Parameter dialog box to display the RIFL Script Editor to write your expression. Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <ul style="list-style-type: none"> • Leave the parameter blank (defaults to 1, writing data to the Target). • Use a Private or Public variable (not Local variables, which are not recognized here). • Use literal numerics (for example, enter a 3 to write the data 3 times). • Use any expression that includes a numeric return value. <p>Not required.</p>
<p>Counter Variable</p>	<p>Name of the variable used to store the current repetition count. Select the ellipsis in the Value column of the Event Action Parameter dialog box to display a text box to enter the Counter variable text value.</p> <p>Ensure that the variable has been declared Private or Public (see Note in Count above).</p> <p>Not required.</p> <p>See Also: Search for the keywords “Counter Variable” in the online help.</p>

MapInsert Record Action

The MapInsert event action may only be used by multimode connectors. MapInsert Record executes target field expressions, assigns the resulting data to the target buffer, and inserts records into the target tables.

Parameters	Description
Target Name	The data target or reject target. Required. Options: Target – the data target or reject target.
Record Layout	Name of the target layout. To insert to more than one target layout, add additional event actions to your transformation. Required.
Table Name	Name of the table in which to insert records. To insert to more than one table, add additional event actions to your transformation. SQL Script as a target: Refer to your SQL application to get table names, since no list is available for SQL Script. Required.

Parameters	Description
<p>Count</p>	<p>Expression to evaluate the repetition count (enter a numeric value). Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <ul style="list-style-type: none"> • Leave the parameter blank (defaults to 1, writing data to the target). • Use a Private or Public variable (not Local variables, which are not recognized here). • Use literal numerics (for example, enter a 3 to write the data 3 times). • Use any expression that includes a numeric return value. <p>Not required.</p> <p>See Also: Search for the keywords “Counter Variable” in the online help.</p>
<p>Counter Variable</p>	<p>Name of the variable used to store the current repetition count. Ensure that the variable has been declared Private or Public (see Note in Count above).</p> <p>Not required (unless you write an expression in Count).</p> <p>See Also: Search for the keywords “Counter Variable” in the online help.</p>

SQL File Action

The SQL File event action may only be used by multimode connectors. It calls specified SQL files to be used in a transformation.

Parameters	Description
Target Name	The data target or reject target. Required.
File Name	Name of the SQL files to be called. Required.

Parameters	Description
<p>Count</p>	<p>Expression to evaluate the repetition count (enter a numeric value). Select the ellipsis in the Value column of the Event Action Parameter dialog box to display the RIFL Script Editor to write your expression. Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <p>Leave the parameter blank (defaults to 1, writing data to the target).</p> <p>Use a Private or Public variable (not Local variables, which are not recognized here).</p> <p>Use literal numerics (for example, enter a 3 to write the data three times).</p> <p>Use any expression that includes a numeric return value.</p> <p>Not required.</p>
<p>Counter Variable</p>	<p>Name of the variable used to store the current repetition count. Select the ellipsis in the Value column of the Event Action Parameter dialog box to display a text box to enter the Counter variable text value.</p> <p>Ensure that the variable has been declared Private or Public (see Note in Count above).</p> <p>Not required.</p> <p>Search for the keywords “Counter Variable” in the online help.</p>

SQL Statement Action

The SQL Statement event action may only be used by multimode connectors. It includes code that contains one or many SQL statements. These statements are written to fire events within your transformation, such as “Insert Record Action” or “Delete Record Action”. SQL commands, such as Truncate or Create Procedure, can be called in SQL Statement action code. This action does not return a value.

You may also embed Rapid Integration Flow Language (RIFL) expressions within your SQL statements using DJX. The syntax for one-line statements is:

DJX(statement)

For multiple lines or block statements, the syntax is:

```
DJXBegin
[Insert your multiline expression here]
DJXEnd
```

For more information, search for the keywords “Using DJX to Create SQL Statements” in the online help.

Parameters	Description
Target Name	The data target or reject target. Required.
SQL Statement	The SQL statement to be executed. Required.

Parameters	Description
<p>Count</p>	<p>Expression to evaluate the repetition count (enter a numeric value). Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <p>Leave the parameter blank (defaults to 1, writing data to the target).</p> <p>Use a Private or Public variable (not Local variables, which are not recognized here).</p> <p>Use literal numerics (for example, enter a 3 to write the data 3 times).</p> <p>Use any expression that includes a numeric return value.</p> <p>Not required.</p> <p>See Also: Search for the keywords “Counter Variable” in the online help.</p>
<p>Counter Variable</p>	<p>Name of the variable used to store the current repetition count.</p> <p>Ensure that the variable has been declared Private or Public (see Note in Count above).</p> <p>Not required.</p> <p>See Also: Search for the keywords “Counter Variable” in the online help.</p>

See Also

“SQL File Action”

Update Record Action

The Update Record event action can be used by multimode connectors only. It updates records in multiple tables specified in the action parameters. You may write to multiple tables by using this action as a BeforeTransformation event.



Note If single-mode connectors are used in a transformation and this action is selected, it is ignored in the transformation process. This action is for multimode connectors only, such as SQL Script, or Oracle 8.x Multimode.

If you set the action key on a primary field, note that updates are made to target records that have the following:

- Action Key fields set to Yes
- A match in the source

Parameters	Description
Target Name	The data target or reject target. Required. Options: Target – the data target or reject target.
Record Layout	Name of the target layout. To update to more than one target layout, add additional event actions to your transformation. Required.
Table Name	Name of table to update. To update to more than one target table, add additional event actions to your transformation. SQL Script as a target – Refer to your SQL application to get table names, since no list is available for SQL Script. Required.

Parameters	Description
<p>Count</p>	<p>Expression to evaluate the repetition count (enter a numeric value). Specify 0 to suppress the Insert, or a 1 to write the Insert.</p> <p>Note: If you mention variable names in this parameter, you must have declared them as Private or Public. Local variables are not recognized in Count. You may enter the variable names in Global Variables, or declare the variable name with the Private or Public keyword in your statement.</p> <p>Writing simple SQL expressions here (and in Counter Variable) gives you precise control over what data is written. This is useful, for instance, when writing records in Update Mode.</p> <p>Ways to Use the Count Parameter</p> <p>Leave the parameter blank (defaults to 1, writing data to the target).</p> <p>Use a Private or Public variable (not Local variables, which are not recognized here).</p> <p>Use literal numerics (for example, enter a 3 to write the data 3 times).</p> <p>Use any expression that includes a numeric return value.</p> <p>Not required.</p>
<p>Counter Variable</p>	<p>Name of the variable used to store the current repetition count.</p> <p>Ensure that the variable has been declared Private or Public (see Note in Count above).</p> <p>Not required.</p>

Index

E

error

tracing 2-40, 2-42

event action execution, order of 1-4

event actions

Abort 2-6

Assert 2-7

ChangeSource 2-8

ChangeTarget 2-10

Clear 2-13

Clear Tree 2-15

ClearInitialize 2-17

ClearMap 2-18

ClearMapInsert Record 3-4

ClearMapPut Record 2-20

Create Index 3-6

Create Table 3-7

Delete Record 3-8

Drop Table 3-10

Execute 3-10

Insert Record 3-11

LogMessage 2-24

LogTargetRecord 2-25

Map 2-26

MapInsert Record 3-13

MapPut Record 2-28

Put Record 2-31

Put Tree 2-33

QueryStatistic 2-36

Resume 2-38

sql file action 3-15

SQL Statement 3-17

Terminate 2-39

TraceOff 2-40

TraceOn 2-42

Update Record 3-19

Upsert Record 2-44

Validate Record 2-46

event actions list 2-3

event handler

OnRecordValidationError 2-30

event handlers

AfterTransformation 1-3

BeforeTransformation 1-3

OnAbort 1-3

OnAssertionError 1-3

OnError 1-3

OnRecordValidationError 1-3

source 1-9

target 1-10

event precedence 1-5

error 1-7

target 1-6

exception handling for event control 2-7

G

general event handlers 1-9

source 1-9

target 1-10

H

handling events 1-9, 1-10

M

memory buffer

appending 2-20, 2-28

clearing 2-13, 2-15

multimode event actions list 3-2

O

order of event action execution 1-4

P

precedence

source event 1-5

put conditions 2-33

R

record type event handlers 1-13

AfterEveryRecord 1-13

AfterFirstRecord 1-13

- BeforeEveryRecord 1-14
- BeforeFirstRecord 1-13
- OnChange 1-14
- record validation 2-46

S

source event handlers

- AfterEveryRecord 1-9
- AfterFirstRecord 1-9
- BeforeEveryRecord 1-9
- BeforeFirstRecord 1-9
- OnAssertionError 1-9
- OnEOF 1-9
- OnError 1-9
- OnRecordValidationError 1-9
- OnTruncateError 1-9

sources

- event handlers 1-9

SQL

- statement action 3-17

T

target event handlers

- AfterDeleteRecord 1-10
- AfterInsertRecord 1-10
- AfterMap 1-10
- AfterPutRecord 1-10
- AfterPutTree 1-10
- AfterUpdateRecord 1-10
- AfterUpsertRecord 1-10
- BeforeDeleteRecord 1-10
- BeforeInsertRecord 1-10
- BeforeMap 1-10
- BeforePutRecord 1-10
- BeforePutTree 1-10
- BeforeUpdateRecord 1-10
- BeforeUpsertRecord 1-11
- OnAbort 1-11
- OnAssertionError 1-11
- OnConstraintError 1-12
- OnDiscard 1-11
- OnDuplicateKeyError 1-12
- OnError 1-11
- OnMismatch 1-12
- OnNullValueError 1-12
- OnOverflowError 1-12

- OnRecordValidationError 1-11

- OnReject 1-11

- OnTruncateError 1-12

targets

- general event handlers 1-10

transformations

- event handlers 1-2

- troubleshoot transformations using event actions 2-40

V

variables

- process level

- public

- workaround 2-37