ScriptEvent

Field keywords

PosField
FixLenField
ChrSepField
AlwaysCreateField
ScriptField
LabelPrefix
PosLabelPrefix
PosPrefix
ChrSepPrefix
ScriptPrefix
LabelStartVariable
PosLabelStartVariable
ChrStartVariable
LabelFieldCont
PosLabelFieldCont
FieldContString
PosValue
IgnoreBlankFieldValues
KeepFieldSpaces
EndValueDesc

Body text keywords

IncludeTextMode
LabelStartInclude
PosStartInclude
LabelIncludeRow
IncludeContString
LabelEndInclude
PosEndInclude
IncludeField
ScriptIncludeField

Level keywords

LabelStartLevel
PosLabelStartLevel
LevelNotOnSepLine
LabelEndLevel
PosLabelEndLevel
IgnoreLevel

Control keywords

LabelStartHeader|Control|Sort
PosHeader|Control|Sort
ScriptHeader|Control|Sort
StreamIN samples...........................................................................................................47
Creating RecordIN samples .........................................................................................47
Creating FieldIN samples .............................................................................................49
Checking the sample file syntax ..................................................................................50
Creating a StreamIN configuration...............................................................................51
Importing a StreamIN configuration.............................................................................52
Managing blocks and fields .........................................................................................53
Blocks .........................................................................................................................53
Adding blocks ..............................................................................................................53
Configuring blocks .....................................................................................................53
Sorting .........................................................................................................................53
Fields ...........................................................................................................................55
Adding fields ...............................................................................................................55
Configuring fields .......................................................................................................55
Field variables ............................................................................................................56
Numeric and date formats .............................................................................................56
StreamIN tool GUI reference .......................................................................................59
Main window ................................................................................................................59
File menu .....................................................................................................................59
Edit menu ....................................................................................................................60
Insert menu ..................................................................................................................60
Tools menu ...................................................................................................................60
Message view ...............................................................................................................61
Properties view .............................................................................................................62
Message properties ......................................................................................................62
Block properties ..........................................................................................................62
Field properties ..........................................................................................................63
About StreamIN

The StreamServer can receive and process field-based (FieldIN) and record-based (RecordIN) input data as shown in the examples below.

**Example 1** RecordIN input

| INVOICE;1234;JOHN SMITH |
| ARTICLES;010;Ball;30.00 |
| ARTICLES;020;Rope;125.00 |
| AMOUNT;SEK;155.00 |

**Example 2** FieldIN input

BEGIN Invoice
HEADER_Invoice_no 1234
HEADER_Your_ref JOHN SMITH
ARTICLE_Pos_no 010
ARTICLE_Item Ball
ARTICLE_Price 30.00
ARTICLE_Pos_no 020
ARTICLE_Item Rope
ARTICLE_Text 80 Inches
ARTICLE_Price 125.00
AMOUNT_To_pay 155.00
AMOUNT_Currency SEK
FREETEXT_Free_text Merry Christmas!

**Description files**

You must create a description file where you describe all fields that the input data can contain, and which Event(s) the input data is aimed for. See *Description files* on page 9.

**Samples**

You can load one or more StreamIN samples to the StreamIN tool, and use these samples when you create a StreamIN configuration. See *StreamIN samples* on page 47.

**StreamIN configuration**

The description file does not describe how to structure the fields. In the StreamIN tool, you specify how to organize the input data as fields and blocks of fields. See *Creating a StreamIN configuration* on page 51.
Description files

StreamIN input is either field-based (FieldIN) or record-based (RecordIN). In either case, you must create a description file where you describe all fields that the input data can contain, and which StreamIN Event configuration the input data is aimed for. One description file can contain several descriptions, separated by description IDs. You create a description file using a text editor, and import it to a resource set connected to the StreamIN configuration.

Adding the description file to the StreamIN configurations
In the Design Center, you must add the description file to the appropriate StreamIN configurations.

1. In the Message window, right-click the StreamIN Event and select **Settings**. The Event Settings dialog box opens.
2. On the Agent Settings tab, select the appropriate **Input Type** (FieldIN or RecordIN) and edit the settings.

<table>
<thead>
<tr>
<th>Settings</th>
<th>Description resource</th>
<th>Description ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description resource</td>
<td>The description file.</td>
<td></td>
</tr>
<tr>
<td>Description ID</td>
<td>The <code>FIELDIN</code> or <code>STREAMIN</code> keyword in the description file (case sensitive).</td>
<td></td>
</tr>
</tbody>
</table>
RecordIN reference

RecordIN data consists of records with one or more fields. A record can be of fixed or variable length. The fields can be character separated, or located in fixed positions.

Example 3  RecordIN input

INVOICE;1234;JOHN SMITH
ARTICLE;010;Ball;30.00
ARTICLE;020;Rope;125.00
AMOUNT;SEK;155.00

Example 4  RecordIN description file

STREAMIN "STR123"
BEGIN
  TYPEPREFIX;
  RECORD "INVOICE" 1 CHRSEP ";"
    MATCH "INVOICE"
    NEWEVENT "Invoice"
    FIELDS
      "Record_id";
      "Invoice_no";
      "Your_ref";
  END
END
RECORD "ARTICLE" 1 CHRSEP ";"
    MATCH "ARTICLE"
    INEVENT "Invoice"
    FIELDS
      "Record_id";
      "Pos_no";
      "Item";
      "Price";
  END
END
RECORD "AMOUNT" 1 CHRSEP ";"
    MATCH "AMOUNT"
    INEVENT "Invoice"
    FIELDS
      "Record_id";
      "Currency";
      "To_pay";
  END
END
END
RecordIN syntax

STREAMIN <str>
BEGIN
[TYPEPREFIX;
[RECLEN <num>;
RECORD <str> 1 [[IGNORE;][FIXPOS|CHRSEP <str>]]
[FIELDQUOTE <str>]
[ESQUOTE <str>]
[NEWEVENT <str>;
[INEVENT <str1> <str2> <str3> ... <strN>;
[EVENT <str>;
[JOBBEGIN;
[JOBEND;
[MATCH [<str1> <str2> ... <strN>|EMPTY|SCRIPT <{...}>];
[FIELDS
<str1> [<num11> <num12>]
[KEEPSP|KEEPLEADINGSP|KEEPTRAILINGSP|SCRIPT <{...}>]
[VARNAME <str>];
...
<strN> [<numN1> <numN2>]
[KEEPSP|KEEPLEADINGSP|KEEPTRAILINGSP|SCRIPT <{...}>]
[VARNAME <str>];
END]
END
END

- All keywords and arguments are separated with white spaces (space or tab).
- Keywords within “[ ]” are optional.
- Pipe “|” indicates “OR”.

All string and character arguments must be enclosed by quotation marks, for example "String 1" or "A". You can also enter characters as ASCII within angle brackets. For example, enter "<33>" instead of an exclamation mark ("!").

Overall keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STREAMIN</strong></td>
<td>The start of the RecordIN description. This is the Description ID you must specify when you configure the Event settings in the Design Center.</td>
</tr>
<tr>
<td><strong>TYPEPREFIX</strong></td>
<td>This keyword must be included if several records have the same field names.</td>
</tr>
<tr>
<td>Keyword overview</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Keyword</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td><strong>RECLEN</strong></td>
<td>This keyword must be included if the records are of fixed length. It specifies the length in characters.</td>
</tr>
</tbody>
</table>

**STREAMIN**

**Syntax**

STREAMIN *<str>*

**Description**
The start of the RecordIN description. This is the Description ID you must specify when you configure the Event settings in the Design Center.

**Example**

STREAMIN "Invoice"

**TYPEPREFIX**

**Syntax**

TYPEPREFIX

**Description**
This keyword must be included if several records have the same field names.

**Example**

TYPEPREFIX

**RECLEN**

**Syntax**

RECLEN *<num>*

**Description**
This keyword must be included if the records are of fixed length. It specifies the length in characters.

**Example**

RECLEN 15
## Record keywords

### Keyword overview

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RECORD</strong></td>
<td>The record ID.</td>
</tr>
<tr>
<td><strong>IGNORE</strong></td>
<td>Ignore the fields in the record.</td>
</tr>
<tr>
<td><strong>FIXPOS</strong></td>
<td>Use this keyword if the record fields are in fixed positions.</td>
</tr>
<tr>
<td><strong>CHRSEP</strong></td>
<td>The field separator for character separated record fields.</td>
</tr>
<tr>
<td><strong>FIELDQUOTE</strong></td>
<td>Quotation marks in the input data indicate that two or more fields belong together. The <code>FIELDQUOTE</code> keyword must be the same character as the quotation mark used in the input data.</td>
</tr>
<tr>
<td><strong>ESCQUOTE</strong></td>
<td>To be able to use the quotation mark as a character in a field, it must be escaped by an escape character. The <code>ESCQUOTE</code> keyword must be the same character as the escape character used in the input data.</td>
</tr>
<tr>
<td><strong>NEWEVENT</strong></td>
<td>Associates the record with a StreamIN configuration. This type of record will trigger a new Message.</td>
</tr>
<tr>
<td><strong>INEVENT</strong></td>
<td>Associates the record with one or more StreamIN configurations. This type of record does not trigger any new Message. The record must be associated with one or more StreamIN configurations that are already active, otherwise the data will be lost.</td>
</tr>
<tr>
<td><strong>EVENT</strong></td>
<td>Associates the record with a StreamIN configuration. If the corresponding StreamIN configuration is not active, a new Message will be triggered by the record. If it is active, the record data will be added to the current Message.</td>
</tr>
<tr>
<td><strong>JOBBEGIN</strong></td>
<td>Sets a JobBegin command.</td>
</tr>
<tr>
<td><strong>JOBEND</strong></td>
<td>Sets a JobEnd command.</td>
</tr>
<tr>
<td><strong>MATCH</strong></td>
<td>Match criteria for a record. Start position of the match is position 1 by default. You can omit the <code>MATCH</code> keyword if the match criterion and record ID are the same.</td>
</tr>
<tr>
<td><strong>EMPTY</strong></td>
<td>Match empty records, i.e. a single carriage return/line feed or a single line feed.</td>
</tr>
</tbody>
</table>
**RecordIN reference**

**Description files**

### RECORD

**Syntax**

 RECORD <str>

**Description**

The record ID.

**Example**

 RECORD "INVOICE"

### IGNORE

**Syntax**

 IGNORE

**Description**

Ignore the fields in the record.

**Comment**

If you use the IGNORE keyword, you cannot use the MATCH keyword.

**Example**

 RECORD "INV_TEST" 1 IGNORE;

### FIXPOS

**Syntax**

 FIXPOS

**Description**

Use this keyword if the record fields are in fixed positions.

**Example**

 RECORD "INV_ARTICLE" 1 FIXPOS

### CHRSEP

**Syntax**

 CHRSEP <chr>

**Description**

The field separator for character separated record fields.

**Example**

 RECORD "INV_ARTICLE" 1 CHRSEP ";"
### FIELDQUOTE

**Syntax**

FIELDQUOTE <chr>

**Description**

Quotation marks in the input data indicate that two or more fields belong together. The FIELDQUOTE keyword must be the same character as the quotation mark used in the input data.

**Comment**

You cannot write FIELDQUOTE "". If you want to use " as field quote, you must use the hexadecimal ASCII representation <22> (FIELDQUOTE "<22>").

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
</tr>
<tr>
<td>INVOICE 1234 &quot;William Smith&quot;</td>
</tr>
</tbody>
</table>

### ESCQUOTE

**Syntax**

ESCQUOTE <chr>

**Description**

To be able to use the quotation mark as a character in a field, it must be escaped by an escape character. The ESCQUOTE keyword must be the same character as the escape character used in the input data.

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
</tr>
<tr>
<td>INVOICE 1234 &quot;William &quot;\Bill&quot;\ Smith&quot;</td>
</tr>
</tbody>
</table>
NEWEVENT

Syntax  
NEWEVENT <str>;

Description  
Associates the record with a StreamIN configuration. This type of record will trigger a new Message.  
The case sensitive string argument must be the same as the name given to the StreamIN Event configuration in the Design Center.

Example  
RECORD "INV_ABC" 1 ChrSep " "  
       NEWEVENT "invoice_1";

INEVENT

Syntax  
INEVENT <str1>...<strN>

Description  
Associates the record with one or more StreamIN configurations. This type of record does not trigger any new Message. The record must be associated with one or more StreamIN configurations that are already active, otherwise the data will be lost.  
The case sensitive string arguments must be the same as the names given to the StreamIN Event configurations in the Design Center.

Example  
RECORD "INV_ABC" 1 ChrSep " "  
       INEVENT "invoice_1" "invoice_3" "invoice_5";

EVENT

Syntax  
EVENT <str>

Description  
Associates the record with a StreamIN configuration. If the corresponding StreamIN configuration is not active, a new Message will be triggered by the record. If it is active, the record data will be added to the current Message.  
The case sensitive string argument must be the same as the name given to the Event in the Design Center.

Example  
RECORD "INV_ABC" 1 ChrSep " "  
       EVENT "invoice_1";
JOBBEGIN
Syntax  JOBBEGIN
Description Sets a JobBegin command.
Example  RECORD "INV_ABC" 1 ChrSep " 
EVENT "invoice_1";
JOBBEGIN;

JOBEND
Syntax  JOBEND
Description Sets a JobEnd command.
Example  RECORD "INV_DEF" 1 ChrSep " 
EVENT "invoice_1";
JOBEND;

MATCH
Syntax  MATCH <str1>... <strN>|EMPTY|SCRIPT <{...}>;
Description Match criteria for a record. Start position of the match is position 1 by default.
You can omit the MATCH keyword if the match criterion and record ID are the same.
Comment Not applicable to IGNORE records.
You can use ? as wildcard.
Example MATCH "CLASSIC" "ROCK";
The match criterion is CLASSIC or ROCK in this example.

EMPTY
Syntax  EMPTY
Description Match empty records, i.e. a single carriage return/line feed or a single line feed.
Example  MATCH EMPTY;
Field keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FIELDS</strong></td>
<td>A field definition. Includes the field names and, in case of fixed position records, the start and end positions.</td>
</tr>
<tr>
<td><strong>KEEPLEADINGSP</strong></td>
<td>Use this keyword to keep leading spaces in the field.</td>
</tr>
<tr>
<td><strong>KEEPSP</strong></td>
<td>Use this keyword to keep both leading and trailing spaces in the field.</td>
</tr>
<tr>
<td><strong>KEEPTRAILINGSP</strong></td>
<td>Use this keyword to keep trailing spaces in the field.</td>
</tr>
<tr>
<td><strong>VARNAME</strong></td>
<td>Specifies a field variable.</td>
</tr>
</tbody>
</table>

**FIELDS**

**Syntax**

FIELDS <str1> [<num11> <num12>] [KEEPSP | KEEPLEADINGSP | KEEPTRAILINGSP | SCRIPT < {...}>] [VARNAME <str>];

**Description**

A field definition. Includes the field names and, in case of fixed position records, the start and end positions.

**Example**

FIELDS

"Record_id" 1 10;
"Invoice_no" 17 22;
"Invoice_date" 25 34;
"Your_ref" 52 71;
"Our_ref" 76 95;
"Email_address" 97 126;
"Fax_no" 128 145;
"Country_code" 149 151;

END
### KEEPLEADINGSP

**Syntax**  
`KEEPLEADINGSP`

**Description**  
Use this keyword to keep leading spaces in the field.

**Example**  
```plaintext
FIELDS
  "Record_id" 1 10 KEEPLEADINGSP;
```

### KEEPSP

**Syntax**  
`KEEPSP`

**Description**  
Use this keyword to keep both leading and trailing spaces in the field.

**Example**  
```plaintext
FIELDS
  "Record_id" 1 10 KEEPSP;
```

### KEEPTRAILINGSP

**Syntax**  
`KEEPTRAILINGSP`

**Description**  
Use this keyword to keep trailing spaces in the field.

**Example**  
```plaintext
FIELDS
  "Record_id" 1 10 KEEPTRAILINGSP;
```

### VARNAME

**Syntax**  
`VARNAME <str>`

**Description**  
Specifies a field variable.

**Comment**  
Do not enter the `$` prefix and do not use white spaces.

**Example**  
```plaintext
FIELDS
  "cust_no" 1 10 VARNAME "cust_no";
```
FieldIN reference

FieldIN input consists of field IDs and field values. The ID and the value can be character separated, or located in fixed positions.

Example 5  FieldIN input

BEGIN Invoice
HEADER_Invoice_no 1234
HEADER_Your_ref JOHN SMITH
ARTICLE_Pos_no 010
ARTICLE_Item Ball
ARTICLE_Price 30.00
ARTICLE_Pos_no 020
ARTICLE_Item Rope
ARTICLE_Text 80 Inches
ARTICLE_Price 125.00
AMOUNT_To_pay 155.00
AMOUNT_Currency SEK
FREETEXT_Free_text Merry Christmas!

Example 6  FieldIN description file

FIELDIN "STR456"
  PosLabelStartEvent 1
  LabelStartEvent "BEGIN"
  PosEvent 25
  IgnoreLevel
  PosField 1
  FixLenField 24
  PosValue 25
END

FieldIN syntax

FIELDIN <str>
  [Comment <str>]
  [DisableLookback]
  Event keywords [<num>|<str>|{<...>}]  
  Field keywords [<num>|<str>|{<...>}]  
  [Body text keywords [<num>|<str>|{<...>}]  
  [Level keywords [<num>|<str>] ]  
  [Control keywords [<num>|<str>|{<...>}]]
END

- Entries within “{...}” are optional.
- Pipe “|” indicates “OR”.
- Scripts are indicated as “<...>”.

StreamServe Persuasion SP4 StreamIN User Guide Rev A
All string and character arguments must be enclosed by quotation marks, for example "String 1" or "A". You can also enter characters as ASCII within angle brackets. For example, enter "<33>" instead of an exclamation mark ("!").

Overall keywords

<table>
<thead>
<tr>
<th>Keyword overview</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyword</strong></td>
</tr>
<tr>
<td>FieldIn</td>
</tr>
<tr>
<td>Comment</td>
</tr>
<tr>
<td>DisableLookBack</td>
</tr>
</tbody>
</table>
FieldIN

**Syntax**
FieldIN <str>

**Description**
The start of the FieldIN description. This is the Description ID you must specify when you configure the Event settings in the Design Center.

**Example**
FieldIN "STR456"

Comment

**Syntax**
Comment <str>

**Description**
A string that determines when to treat a line of text as a comment. All lines in the input starting with this string will be ignored.

**Example**
Comment "//*"

DisableLookBack

**Syntax**
DisableLookBack

**Description**
Turn off the LookBack function.

The LookBack function looks in the previous instance of a block (i.e. block defined in the StreamIN Event) to see if there are any missing fields. If there are, the incoming field value will be added to previous instance, and not to the current instance.

Event keywords

<table>
<thead>
<tr>
<th>Keyword overview</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyword</strong></td>
</tr>
<tr>
<td>LabelStartEvent</td>
</tr>
<tr>
<td>PosLabelStartEvent</td>
</tr>
<tr>
<td>EndEventDesc</td>
</tr>
</tbody>
</table>
### Keyword overview

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LabelEndEvent</strong></td>
<td>Text string in input that specifies the end of an Event. Any input data between a <code>LabelEndEvent</code> and a new <code>LabelStartEvent</code> will be ignored.</td>
</tr>
<tr>
<td><strong>PosLabelEndEvent</strong></td>
<td>The start position of <code>LabelEndEvent</code>.</td>
</tr>
<tr>
<td><strong>LabelPageBreak</strong></td>
<td>Triggers a page break in the corresponding Process output.</td>
</tr>
<tr>
<td><strong>PosPageBreak</strong></td>
<td>The start position of <code>LabelPageBreak</code>.</td>
</tr>
<tr>
<td><strong>PosEvent</strong></td>
<td>The start position of the Event trigger. The name of the Event in the Design Center must be the same as the Event trigger text string.</td>
</tr>
<tr>
<td><strong>ChrSepEvent</strong></td>
<td>The character used in the input to separate <code>LabelStartEvent</code> and the Event trigger.</td>
</tr>
<tr>
<td><strong>FixLenEvent</strong></td>
<td>Specifies a maximum length, in characters, for the Event trigger.</td>
</tr>
<tr>
<td><strong>BestMatchEvent</strong></td>
<td>Takes the Event trigger specified in the input and compares it with all StreamIN Event names defined in the Project. If there is no matching Event name, the last character is removed from the Event name specified in the input, and a new round starts. This procedure is repeated until a matching Event name is found.</td>
</tr>
<tr>
<td><strong>AliasEventPath</strong></td>
<td>Use an alias table to determine which Event to trigger.</td>
</tr>
<tr>
<td><strong>ScriptEvent</strong></td>
<td>Use a script to determine which Event to trigger.</td>
</tr>
</tbody>
</table>

### LabelStartEvent

**Syntax**  
`LabelStartEvent <str>`

**Description**  
The text string in the input that indicates the start of an Event.

**Example**  
`LabelStartEvent "BEGIN"`
PosLabelStartEvent

Syntax: `PosLabelStartEvent <num>`
Description: The start position of `LabelStartEvent`.
Example: `PosLabelStartEvent 1`

EndEventDesc

Syntax: `EndEventDesc <chr>`
Description: The character that specifies the end of the Event description section in the input data. Default is line feed.
Example:

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN;Invoice*ARTICLE_1;Gambozola...</td>
<td>EndEventDesc &quot;*&quot;</td>
</tr>
</tbody>
</table>

LabelEndEvent

Syntax: `LabelEndEvent <str>`
Description: Text string in input that specifies the end of an Event. Any input data between a `LabelEndEvent` and a new `LabelStartEvent` will be ignored.

Example:

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td>CLASSIC</td>
</tr>
<tr>
<td>ARTIST</td>
<td>VIVALDI</td>
</tr>
<tr>
<td>ARTIST</td>
<td>GRIEG</td>
</tr>
<tr>
<td>END_CL</td>
<td></td>
</tr>
<tr>
<td>ARTIST</td>
<td>STING</td>
</tr>
<tr>
<td>ARTIST</td>
<td>EZRA</td>
</tr>
</tbody>
</table>

`ARTIST STING` and `ARTIST EZRA` will be ignored.
PosLabelEndEvent

Syntax: \texttt{PosLabelEndEvent <num>}

Description: The start position of \texttt{LabelEndEvent}.

Example: \texttt{PosLabelEndEvent 1}

LabelPageBreak

Syntax: \texttt{LabelPageBreak <str>}

Description: Triggers a page break in the corresponding Process output.

Example: \texttt{LabelPageBreak "BREAK"}

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN ARTISTS</td>
<td></td>
</tr>
<tr>
<td>ARTIST VIVALDI</td>
<td></td>
</tr>
<tr>
<td>ARTIST GRIEG</td>
<td></td>
</tr>
<tr>
<td>BREAK</td>
<td></td>
</tr>
<tr>
<td>ARTIST STING</td>
<td></td>
</tr>
<tr>
<td>ARTIST EZRA</td>
<td></td>
</tr>
</tbody>
</table>

PosPageBreak

Syntax: \texttt{PosPageBreak <num>}

Description: The start position of \texttt{LabelPageBreak}.

Example: \texttt{PosPageBreak 1}
PosEvent

**Syntax**: `PosEvent <num>`

**Description**: The start position of the Event trigger. The name of the Event in the Design Center must be the same as the Event trigger text string.

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN ARTISTS</td>
<td>PosLabelStartEvent 1</td>
</tr>
<tr>
<td>ARTIST VIVALDI</td>
<td>LabelStartEvent &quot;BEGIN&quot;</td>
</tr>
<tr>
<td>ARTIST GRIEG</td>
<td>PosEvent 14</td>
</tr>
</tbody>
</table>

ChrSepEvent

**Syntax**: `ChrSepEvent <chr>`

**Description**: The character used in the input to separate LabelStartEvent and the Event trigger (BEGIN and ARTISTS in the example below).

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN;ARTISTS</td>
<td>PosLabelStartEvent 1</td>
</tr>
<tr>
<td>ARTIST VIVALDI</td>
<td>LabelStartEvent &quot;BEGIN&quot;</td>
</tr>
<tr>
<td>ARTIST GRIEG</td>
<td>ChrSepEvent &quot;;&quot;</td>
</tr>
</tbody>
</table>

FixLenEvent

**Syntax**: `FixLenEvent <num>`

**Description**: Specifies a maximum length, in characters, for the Event trigger.

**Example**: `FixLenEvent 16`
BestMatchEvent

Syntax
BestMatchEvent <num>

Description
Takes the Event trigger specified in the input, and compares it with all StreamIN Event names defined in the Project. If there is a matching Event name, the corresponding Event is triggered.

If there is no matching Event name, the last character is removed from the Event name specified in the input, and a new round starts. This procedure is repeated up to <num> number of times until a matching Event name is found.

Comment
The Event name in the input data cannot have fewer characters than the Event names defined in the Project.

Example

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td>STRIN001</td>
</tr>
<tr>
<td>ARTIST</td>
<td>VIVALDI</td>
</tr>
<tr>
<td>ARTIST</td>
<td>GRIEG</td>
</tr>
</tbody>
</table>

There are two StreamIN Events defined in the Project: STR_US and STRIN. The match procedure will be run four times, and the Event STRIN will be triggered.

AliasEventPath

Syntax
AliasEventPath <str>

Description
Use an alias table to determine which Event to trigger.

Comment
You must specify the file relative to the Project’s export directory.

Example

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td>STRIN002</td>
</tr>
<tr>
<td>ARTIST</td>
<td>VIVALDI</td>
</tr>
<tr>
<td>ARTIST</td>
<td>GRIEG</td>
</tr>
</tbody>
</table>

Alias table STRALIAS.txt

<table>
<thead>
<tr>
<th>STRIN001</th>
<th>CGW32</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRIN002</td>
<td>AFW104</td>
</tr>
<tr>
<td>STRIN003</td>
<td>RTEA36</td>
</tr>
<tr>
<td>STRIN004</td>
<td>HGH326</td>
</tr>
</tbody>
</table>

The Event AFW104 will be triggered.
ScriptEvent

Syntax

ScriptEvent $\langle\ldots\rangle$

Description

Use a script to determine which Event to trigger.

Example

```script
{ $time=gettime();
  if ($time <= "120000")
    return "AFW104";
  else
    return "HGH326";
};
```

Field keywords

<table>
<thead>
<tr>
<th>Keyword overview</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyword</strong></td>
</tr>
<tr>
<td>PosField</td>
</tr>
<tr>
<td>FixLenField</td>
</tr>
<tr>
<td>ChrSepField</td>
</tr>
<tr>
<td>AlwaysCreateField</td>
</tr>
<tr>
<td>ScriptField</td>
</tr>
<tr>
<td>LabelPrefix</td>
</tr>
<tr>
<td>PosLabelPrefix</td>
</tr>
<tr>
<td>PosPrefix</td>
</tr>
<tr>
<td>ChrSepPrefix</td>
</tr>
</tbody>
</table>
### Keyword overview

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ScriptPrefix</strong></td>
<td>Use a script to define a prefix.</td>
</tr>
<tr>
<td><strong>LabelStartVariable</strong></td>
<td>A key that triggers the creation of a variable. The key should match one or more characters in a field name. When this key is found in the input data, a variable $&lt;field name&gt; is created.</td>
</tr>
<tr>
<td><strong>PosLabelStartVariable</strong></td>
<td>The start position of <strong>LabelStartVariable</strong>.</td>
</tr>
<tr>
<td><strong>ChrStartVariable</strong></td>
<td>A single-character key that triggers the creation of a variable. The key should match the first character in a field name. When this key is found in the input data, a variable $&lt;field name&gt; is created. Note that the key character is removed from the variable name.</td>
</tr>
<tr>
<td><strong>LabelFieldCont</strong></td>
<td>Several fields in the input data can be concatenated to one single field in the output. When the key specified by <strong>LABELFIELDCONT</strong> is found in the input, the current field and the next field will be concatenated.</td>
</tr>
<tr>
<td><strong>PosLabelFieldCont</strong></td>
<td>The start position of <strong>LabelFieldCont</strong>.</td>
</tr>
<tr>
<td><strong>FieldContString</strong></td>
<td>Determines which character to insert between the concatenated field values.</td>
</tr>
<tr>
<td><strong>PosValue</strong></td>
<td>The start position of the field value.</td>
</tr>
<tr>
<td><strong>IgnoreBlankFieldValues</strong></td>
<td>Ignore empty fields, i.e. field values that contain only white spaces.</td>
</tr>
<tr>
<td><strong>KeepFieldSpaces</strong></td>
<td>Keep leading spaces in field values.</td>
</tr>
<tr>
<td><strong>EndValueDesc</strong></td>
<td>The character that specifies the end of the field value in the input data. Default is line feed.</td>
</tr>
</tbody>
</table>

### PosField

**Syntax**  
PosField <num>

**Description**  
The start position of the field names.

**Example**  
PosField 1
FixLenField

**Syntax**

FixLenField <num>

**Description**

Specifies a fixed length, in characters, for the field names. If the field names and field values in the input data are column separated, you must specify a fixed length.

**Example**

FixLenField 24

ChrSepField

**Syntax**

ChrSepField <chr>

**Description**

The character used in the input to separate the field name and field value (ARTICLE_1 and Gambozola in the example below).

**Example**

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN STRIN002</td>
<td>ChrSepField &quot;;&quot;</td>
</tr>
<tr>
<td>ARTICLE_1;Gambozola</td>
<td></td>
</tr>
</tbody>
</table>

AlwaysCreateField

**Syntax**

AlwaysCreateField

**Description**

Always create a field. If the field has no value in the input, a field will be created with an empty value ("").

ScriptField

**Syntax**

ScriptField <...>

**Description**

Use a script to define the start position, length, etc. of the fields.
LabelPrefix

**Syntax**

LabelPrefix <str>

**Description**
The prefix label.

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td>LABELPREFIX &quot;PREFIX&quot;</td>
</tr>
<tr>
<td>PREFIX</td>
<td>POSLABELPREFIX 1</td>
</tr>
<tr>
<td>ARTIST</td>
<td>POSPREFIX 14</td>
</tr>
<tr>
<td>ARTIST</td>
<td>POSFIELD 1</td>
</tr>
<tr>
<td>PREFIX</td>
<td>POSVALUE 14</td>
</tr>
<tr>
<td>ARTIST</td>
<td>...</td>
</tr>
</tbody>
</table>

The corresponding field names in the StreamIN configuration are CLASSIC_ARTIST and ROCKPOP_ARTIST respectively.

PosLabelPrefix

**Syntax**

PosLabelPrefix <num>

**Description**
The start position of the prefix label.

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td>LABELPREFIX &quot;PREFIX&quot;</td>
</tr>
<tr>
<td>PREFIX</td>
<td>POSLABELPREFIX 1</td>
</tr>
<tr>
<td>ARTIST</td>
<td>POSPREFIX 14</td>
</tr>
<tr>
<td>ARTIST</td>
<td>POSFIELD 1</td>
</tr>
<tr>
<td>PREFIX</td>
<td>POSVALUE 14</td>
</tr>
<tr>
<td>ARTIST</td>
<td>...</td>
</tr>
</tbody>
</table>

The corresponding field names in the StreamIN configuration are CLASSIC_ARTIST and ROCKPOP_ARTIST respectively.
FieldIN reference

Description files

**PosPrefix**

**Syntax**

```
PosPrefix <num>
```

**Description**

The position of the prefix.

**Example**

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN STRIN002</td>
<td>LABELPREFIX &quot;PREFIX&quot;</td>
</tr>
<tr>
<td>PREFIX CLASSIC VIVALDI</td>
<td>POSLABELPREFIX 1</td>
</tr>
<tr>
<td>ARTIST GRIEG</td>
<td>POSFIELD 1</td>
</tr>
<tr>
<td>PREFIX ROCKPOP SPRINSTEEN</td>
<td>POSVALUE 14</td>
</tr>
<tr>
<td>ARTIST DYLAN</td>
<td></td>
</tr>
</tbody>
</table>

The corresponding field names in the StreamIN configuration are `CLASSIC_ARTIST` and `ROCKPOP_ARTIST` respectively.

**ChrSepPrefix**

**Syntax**

```
ChrSepPrefix <chr>
```

**Description**

The character used in the input to separate the prefix label and prefix.

**Example**

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN STRIN002</td>
<td>LABELPREFIX &quot;PREFIX&quot;</td>
</tr>
<tr>
<td>PREFIX;CLASSIC VIVALDI</td>
<td>POSLABELPREFIX 1</td>
</tr>
<tr>
<td>ARTIST GRIEG</td>
<td>CHRSEPREFIX &quot;;&quot;</td>
</tr>
<tr>
<td>PREFIX;ROCKPOP SPRINSTEEN</td>
<td>POSFIELD 1</td>
</tr>
<tr>
<td>ARTIST DYLAN</td>
<td>POSVALUE 14</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ScriptPrefix

**Syntax**

```plaintext
ScriptPrefix < {... }>
```

**Description**

Use a script to define the prefix.

---

LabelStartVariable

**Syntax**

```plaintext
LabelStartVariable < str>
```

**Description**

A key that triggers the creation of a variable. The key should match one or more characters in a field name. When this key is found in the input data, a variable `$<field name>` is created.

**Comment**

You can also use the `ChrStartVariable` keyword instead. See `ChrStartVariable`. You can use either `LabelStartVariable` or `ChrStartVariable` – not both.

---

### Example

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN STRIN002</td>
<td>POSLABELSTARTVARIABLE 2</td>
</tr>
<tr>
<td>...</td>
<td>LABELSTARTVARIABLE &quot;RE&quot;</td>
</tr>
<tr>
<td>URES 123</td>
<td></td>
</tr>
<tr>
<td>ASER Gold</td>
<td></td>
</tr>
<tr>
<td>BRET Agir</td>
<td></td>
</tr>
<tr>
<td>GBD 080</td>
<td></td>
</tr>
</tbody>
</table>

The variables `$URES` and `$BRET` are created.

---

PosLabelStartVariable

**Syntax**

```plaintext
PosLabelStartVariable < num>
```

**Description**

The start position of `LabelStartVariable`.

**Example**

```plaintext
PosLabelStartVariable 1
```
FieldIN Reference

Description files

**ChrStartVariable**

**Syntax**

ChrStartVariable <chr>

**Description**

A single-character key that triggers the creation of a variable. The key should match the first character in a field name. When this key is found in the input data, a variable $<field name>$ is created. Note that the key character is removed from the variable name.

**Comments**

You can also use the ChrStartVariable keyword instead. See LabelStartVariable. You can use either LabelStartVariable or ChrStartVariable — not both.

### Example

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN STRIN002</td>
<td>CHRSTARTVARIABLE &quot;U&quot;</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>URES 123</td>
<td></td>
</tr>
<tr>
<td>ASER Gold</td>
<td></td>
</tr>
<tr>
<td>BRET Agir</td>
<td></td>
</tr>
<tr>
<td>UBGD 080</td>
<td></td>
</tr>
</tbody>
</table>

The variables $RES$ and $BGD$ are created.

**LabelFieldCont**

**Syntax**

LabelFieldCont <str>

**Description**

Several fields in the input data can be concatenated to one single field in the output. When the key specified by LABELFIELDCONT is found in the input, the current field and the next field will be concatenated.

### Example

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN STRIN002</td>
<td></td>
</tr>
<tr>
<td>ARTIST @Antonio</td>
<td>PosLabelFieldCont 10</td>
</tr>
<tr>
<td>ARTIST Vivaldi</td>
<td>LabelFieldCont &quot;@&quot;</td>
</tr>
<tr>
<td>ALBUM 4 seasons</td>
<td>FieldContString &quot; &quot;</td>
</tr>
<tr>
<td>PRICE 32</td>
<td></td>
</tr>
</tbody>
</table>

The two ARTIST fields with values Antonio and Vivaldi are concatenated to one ARTIST field with the value Antonio Vivaldi.
FieldCont

**PosLabelFieldCont**

**Syntax**

```
PosLabelFieldCont <num>
```

**Description**

The start position of LabelFieldCont.

**Example**

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN STRIN002</td>
<td>PosLabelFieldCont 10</td>
</tr>
<tr>
<td>ARTIST @Antonio</td>
<td>LabelFieldCont &quot;@&quot;</td>
</tr>
<tr>
<td>ARTIST Vivaldi</td>
<td>FieldContString &quot; &quot;</td>
</tr>
<tr>
<td>ALBUM 4 seasons</td>
<td></td>
</tr>
<tr>
<td>PRICE 32</td>
<td></td>
</tr>
</tbody>
</table>

The two ARTIST fields with values Antonio and Vivaldi are concatenated to one ARTIST field with the value Antonio Vivaldi.

**FieldContString**

**Syntax**

```
FieldContString <chr>
```

**Description**

Determines which character to insert between the concatenated field values.

**Example**

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN STRIN002</td>
<td>PosLabelFieldCont 10</td>
</tr>
<tr>
<td>ARTIST @Antonio</td>
<td>LabelFieldCont &quot;@&quot;</td>
</tr>
<tr>
<td>ARTIST Vivaldi</td>
<td>FieldContString &quot; &quot;</td>
</tr>
<tr>
<td>ALBUM 4 seasons</td>
<td></td>
</tr>
<tr>
<td>PRICE 32</td>
<td></td>
</tr>
</tbody>
</table>

The two ARTIST fields with values Antonio and Vivaldi are concatenated to one ARTIST field with the value Antonio Vivaldi.

**PosValue**

**Syntax**

```
PosValue <num>
```

**Description**

The start position of the field value.

**Example**

PosValue 25
IgnoreBlankFieldValues

Syntax: IgnoreBlankFieldValues
Description: Ignore empty fields, i.e. field values that contain only white spaces.

KeepFieldSpaces

Syntax: KeepFieldSpaces
Description: Keep leading spaces in field values.

EndValueDesc

Syntax: EndValueDesc <chr>
Description: The character that specifies the end of the field value in the input data. Default is line feed.

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
</tr>
<tr>
<td>BEGIN;Invoice</td>
</tr>
<tr>
<td>ARTICLE;Gambozola+ARTICLE;Gouda+...</td>
</tr>
</tbody>
</table>

Body text keywords

Body text keywords identify and extract body texts. Body text can cover one or several lines.

<table>
<thead>
<tr>
<th>Keyword overview</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyword</strong></td>
</tr>
<tr>
<td>IncludeTextMode</td>
</tr>
<tr>
<td>LabelStartInclude</td>
</tr>
<tr>
<td>PosStartInclude</td>
</tr>
<tr>
<td>LabelIncludeRow</td>
</tr>
</tbody>
</table>
### Keyword overview

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IncludeContString</td>
<td>Specifies which characters to use when concatenating the body text lines (specified by LabelIncludeRow) in the output.</td>
</tr>
<tr>
<td>LabelEndInclude</td>
<td>The label that indicates the end of a body text.</td>
</tr>
<tr>
<td>PosEndInclude</td>
<td>The start position of LabelEndInclude.</td>
</tr>
<tr>
<td>IncludeField</td>
<td>Enables the use of ScriptIncludeField.</td>
</tr>
<tr>
<td>ScriptIncludeField</td>
<td>A script that returns the field name.</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN STRIN002</td>
<td>IncludeTextMode</td>
</tr>
<tr>
<td>...</td>
<td>PosStartInclude 1</td>
</tr>
<tr>
<td>&lt;--</td>
<td>LabelStartInclude &quot;&lt;--&quot;</td>
</tr>
<tr>
<td>::This is not the official version of the XCH-04. See D-12 for more information.</td>
<td>LabelIncludeRow &quot;::&quot;</td>
</tr>
<tr>
<td>--&gt;</td>
<td>IncludeContString &quot;&lt;0d&gt;&lt;0A&gt;&quot;</td>
</tr>
<tr>
<td></td>
<td>PosEndInclude 1</td>
</tr>
<tr>
<td></td>
<td>LabelEndInclude &quot;--&gt;&quot;</td>
</tr>
<tr>
<td></td>
<td>IncludeField</td>
</tr>
<tr>
<td></td>
<td>ScriptIncludeField</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>return &quot;TEXT&quot;;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

The text between the labels <-- and --> will be extracted from the input data, and included in the field TEXT.
IncludeTextMode

**Syntax**
IncludeTextMode

**Description**
Enable extraction of body texts.

**Comment**
If this keyword is not specified in the description file, only the first line in a body text will be extracted from the input.

**Example**
```
IncludeTextMode
PosStartInclude 1
LabelStartInclude "<--"
LabelIncludeRow "::"
IncludeContString "<0d><0A>"
PosEndInclude 1
LabelEndInclude "-->"
IncludeField
ScriptIncludeField
{
    return "TEXT";
}
```

LabelStartInclude

**Syntax**
LabelStartInclude <str>

**Description**
The label that indicates the beginning of a body text.

**Example**
```
IncludeTextMode
PosStartInclude 1
LabelStartInclude "<--"
LabelIncludeRow "::"
IncludeContString "<0d><0A>"
PosEndInclude 1
LabelEndInclude "-->"
IncludeField
ScriptIncludeField
{
    return "TEXT";
}
```
PosStartInclude

**Syntax**

PosStartInclude <num>

**Description**
The start position of LabelStartInclude.

**Example**

IncludeTextMode

PosStartInclude 1

LabelStartInclude "<--"

LabelIncludeRow "::"

IncludeContString "<0d><0A>"

PosEndInclude 1

LabelEndInclude "-->"

IncludeField

ScriptIncludeField

{ return "TEXT";
  }

LabelIncludeRow

**Syntax**

LabelIncludeRow <str>

**Description**
A label that specifies whether or not to include a body text line in the output.

**Example**

IncludeTextMode

PosStartInclude 1

LabelStartInclude "<--"

LabelIncludeRow "::"

IncludeContString "<0d><0A>"

PosEndInclude 1

LabelEndInclude "-->"

IncludeField

ScriptIncludeField

{ return "TEXT";
  }
IncludeContString

Syntax
IncludeContString <str>

Description
Specifies which characters to use when concatenating the body text lines (specified by LabelIncludeRow) in the output.

Example
IncludeTextMode
PosStartInclude 1
LabelStartInclude "<--"
LabelIncludeRow "::"
IncludeContString "<0d><0A>"
PosEndInclude 1
LabelEndInclude "-->"
IncludeField
ScriptIncludeField
{
    return "TEXT";
};

LabelEndInclude

Syntax
LabelEndInclude <str>

Description
The label that indicates the end of a body text.

Example
IncludeTextMode
PosStartInclude 1
LabelStartInclude "<--"
LabelIncludeRow "::"
IncludeContString "<0d><0A>"
PosEndInclude 1
LabelEndInclude "-->"
IncludeField
ScriptIncludeField
{
    return "TEXT";
};
**PosEndInclude**

**Syntax**

PosEndInclude <num>

**Description**

The start position of LabelEndInclude.

**Example**

IncludeTextMode
PosStartInclude 1
LabelStartInclude "<--"
LabelIncludeRow "::"
IncludeContString "<0d><0A>"
PosEndInclude 1
LabelEndInclude "-->"
IncludeField
ScriptIncludeField
{
    return "TEXT";
};

**IncludeField**

**Syntax**

IncludeField

**Description**

Enables the use of ScriptIncludeField.

**Example**

IncludeTextMode
PosStartInclude 1
LabelStartInclude "<--"
LabelIncludeRow "::"
IncludeContString "<0d><0A>"
PosEndInclude 1
LabelEndInclude "-->"
IncludeField
ScriptIncludeField
{
    return "TEXT";
};
ScriptIncludeField

Syntax

ScriptIncludeField </.../>

Description
A script that returns the field name.

Example
IncludeTextMode
PosStartInclude 1
LabelStartInclude "<--"
LabelIncludeRow "::"
IncludeContString "<0d><0A>"
PosEndInclude 1
LabelEndInclude "-->"
IncludeField
ScriptIncludeField
{
    return "TEXT";
}

Level keywords

You can use level keywords to categorize the fields in the input data as blocks. If you do not want to categorize any fields, you must include the IGNORELEVEL keyword in the description file.

The LOOKBACK function is automatically disabled for all fields categorized by level keywords.

<table>
<thead>
<tr>
<th>Keyword overview</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyword</strong></td>
</tr>
<tr>
<td>LabelStartLevel</td>
</tr>
<tr>
<td>PosLabelStartLevel</td>
</tr>
<tr>
<td>LevelNotOnSepLine</td>
</tr>
<tr>
<td>LabelEndLevel</td>
</tr>
<tr>
<td>PosLabelEndLevel</td>
</tr>
<tr>
<td>IgnoreLevel</td>
</tr>
</tbody>
</table>
**LabelStartLevel**

**Syntax**

```
LabelStartLevel <str>
```

**Description**
The label that indicates the beginning of a block.

**Example**

```
LabelStartLevel "BLOCK_START"
PosLabelStartLevel 1
LabelEndLevel "BLOCK_END"
PosLabelEndLevel 1
```

**PosLabelStartLevel**

**Syntax**

```
PosLabelStartLevel <num>
```

**Description**
The start position of `LabelStartLevel`.

**Example**

```
LabelStartLevel "BLOCK_START"
PosLabelStartLevel 1
LabelEndLevel "BLOCK_END"
PosLabelEndLevel 1
```

**LevelNotOnSepLine**

**Syntax**

```
LevelNotOnSepLine
```

**Description**
Specifies that `LabelStartLevel` is not a separate line in the input data. The name of the first field, or a part of this name, is used as label instead.

**Example**

```
LabelStartLevel "ARTICLE_pos"
PosLabelStartLevel 1
LabelEndLevel "ARTICLE_total"
PosLabelEndLevel 1
LevelNotOnSepLine
```
LabelEndLevel

**Syntax**

```
LabelEndLevel <str>
```  

**Description**
The label that indicates the end of a block.

**Example**
```
LabelStartLevel "BLOCK_START"
PosLabelStartLevel 1
LabelEndLevel "BLOCK_END"
PosLabelEndLevel 1
```  

PosLabelEndLevel

**Syntax**

```
PosLabelEndLevel <num>
```  

**Description**
The start position of `LabelEndLevel`.

**Example**
```
LabelStartLevel "BLOCK_START"
PosLabelStartLevel 1
LabelEndLevel "BLOCK_END"
PosLabelEndLevel 1
```  

IgnoreLevel

**Syntax**

```
IgnoreLevel
```  

**Description**
Disables any other block keyword defined in the description file.

Control keywords

You can use control keywords to specify scripts that will be triggered by input data. The scripts usually assign values to variables that can be used in other keyword scripts.
You can specify up to three control scripts. To make this possible, i.e. to prevent the script from being overwritten, each control script has a unique name: `ScriptHeader`, `ScriptControl`, and `ScriptSort`.

### Keyword overview

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>`LabelStartHeader</td>
<td>Control</td>
</tr>
<tr>
<td>`PosHeader</td>
<td>Control</td>
</tr>
<tr>
<td>`ScriptHeader</td>
<td>Control</td>
</tr>
</tbody>
</table>

### Example

The script `ScriptHeader` is run when the text "JOHN SMITH" is found in position 25.

```
<table>
<thead>
<tr>
<th>Input</th>
<th>Description file</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LabelStartHeader  &quot;JOHN SMITH&quot;</td>
</tr>
<tr>
<td></td>
<td>PosHeader 25</td>
</tr>
<tr>
<td></td>
<td>ScriptHeader {</td>
</tr>
<tr>
<td></td>
<td>$Type=&quot;GOLD&quot;;</td>
</tr>
<tr>
<td></td>
<td>};</td>
</tr>
</tbody>
</table>
```

### LabelStartHeader|Control|Sort

**Syntax**

- `LabelStartHeader <str>`
- `LabelStartControl <str>`
- `LabelStartSort <str>`

**Description**

The input string that will trigger the script.

**Example**

```
LabelStartHeader  "JOHN SMITH"
PosHeader 25
ScriptHeader {
   $Type="GOLD";
};
```
PosHeader|Control|Sort

**Syntax**

PosHeader <num>
PosControl <num>
PosSort <num>

**Description**
The start position of LabelStartHeader, LabelStartControl, or LabelStartSort.

**Example**

LabelStartHeader  "JOHN SMITH"
PosHeader  25

ScriptHeader {
    $Type="GOLD";
    }

---

ScriptHeader|Control|Sort

**Syntax**

ScriptHeader {...}
ScriptControl {...}
ScriptSort {...}

**Description**
The control script.

**Example**

LabelStartHeader  "JOHN SMITH"
PosHeader  25

ScriptHeader {
    $Type="GOLD";
    }
StreamIN samples

You can load one or more StreamIN samples in the StreamIN tool, and use these samples when you create a StreamIN configuration. A StreamIN sample is loaded as a structure of blocks and fields that you can drag to the appropriate position in the StreamIN tree structure.

**To load a sample**

1. In the Integration Tool view, select **Document > New Connection**. The Create new Connection dialog box opens.
2. Select **SXD Parser** and click **OK**. The Select Resource dialog box opens.
3. Browse to, and select, the appropriate sample.

**Creating RecordIN samples**

For RecordIN data, you can generate a sample file automatically from the description file. If the description file is only available as a resource, you must first extract it to file.

*Example 7 Sample file for RecordIN*

```xml
<?xml version="1.0" ?>
<strsdictionary version="2.0" name="RecordIN">
    <field id="INVOICE_Record_id"/>
    <field id="INVOICE_Invoice_no"/>
    <field id="INVOICE_Your_ref"/>
    <block id="Invoice">
        <field id="ARTICLE_Record_id"/>
        <field id="ARTICLE_Pos_no"/>
        <field id="ARTICLE_Item"/>
        <field id="ARTICLE_Price"/>
        <block id="Text">
            <field id="ARTICLE_TEXT_Record_id"/>
            <field id="ARTICLE_TEXT_Description"/>
        </block>
    </block>
    <field id="AMOUNT_Record_id"/>
    <field id="AMOUNT_Currency"/>
    <field id="AMOUNT_To_pay"/>
    <field id="FREETEXT_Record_id"/>
    <field id="FREETEXT_Free_text"/>
</strsdictionary>
```

The description file does not indicate whether a record is recurring or not. This means that you may have to edit the sample to make sure that the fields and blocks complies to the input data structure.
To create a sample

1. Create an empty SXD resource in a resource set connected to the StreamIN Event.

2. In the StreamIN tool, select **Tools > SXD Converter**. The SXD converter dialog box opens.

3. In **Dictionary source**, browse to and select the description file.

4. In **SXD target**, browse to and select the SXD resource you created.

5. Click **OK**. The sample is created in the SXD resource.

You can now load the SXD resource as a sample in the StreamIN tool.
Creating FieldIN samples

For FieldIN data, you must create the sample file manually.

Example 8  Sample file for FieldIN

```xml
<?xml version="1.0"?>
<strsdictionary version="2.0" name="Invoice">
    <field id="HEADER_Invoice_no"/>
    <field id="HEADER_Your_ref"/>
    <block id="Article_Block">
        <field id="ARTICLE_Pos_no"/>
        <field id="ARTICLE_Item"/>
        <field id="ARTICLE_Price"/>
        <block id="Text">
            <field id="ARTICLE_Text"/>
        </block>
    </block>
    <field id="AMOUNT_To_pay"/>
    <field id="AMOUNT_Currency"/>
    <field id="FREETEXT_Free_Text"/>
</strsdictionary>
```

Creating a sample file using a text editor

You can create the sample file using a text editor. The sample file must conform to the following DTD:

`www.streamserve.com/strs-xml/strsdictionary.dtd`

The following characters must be escaped:

<table>
<thead>
<tr>
<th>Character</th>
<th>Escape sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backslash: &quot;&quot;</td>
<td>&quot;\&quot;</td>
</tr>
<tr>
<td>Space: &quot; &quot;</td>
<td>&quot;\w&quot; (only required for leading and trailing spaces).</td>
</tr>
<tr>
<td>Tab: &quot; &quot;</td>
<td>&quot;\t&quot; (always required to differentiate between space and tab).</td>
</tr>
</tbody>
</table>

Creating a sample file using the StreamIN tool

You can create a StreamIN structure manually in the StreamIN tool by adding blocks and fields to the Message tree. You can then save this structure as a sample file resource in the appropriate resource set.

1 Add the blocks and fields to the Message tree.
2 Select File > Save Message Definition. The Select Resource dialog box opens.
3 Browse to, and save the resource, in the appropriate resource set.
Checking the sample file syntax

Change the file extension from `.sxd` to `.xml` and open the XML file in Internet Explorer (version 5.0 or higher). If the XML file is correctly displayed in the window, it is well-formed.
Creating a StreamIN configuration

In the StreamIN tool, you specify how to organize the input data as fields and blocks of fields.

You create the instructions based on analysis of the input data structure. You can create the structure by adding fields and blocks manually, or you can use dictionaries to drag-and-drop fields and blocks. See *StreamIN samples* on page 47.

**Example 9**  
*Input data structure analysis*

Each field belongs to a group, and each group can appear once, several ($N$) times, or not at all.

```
INVOICE
    1
      N
      0
      N
      1
      1

ARTICLE
    1
      Position number
      Item
      Price

TEXT
    Description

AMOUNT
    Currency
    Amount to pay

FREETEXT
    Free Text
```
Importing a StreamIN configuration

You can import an existing StreamIN configuration and add it to your current StreamIN configuration. Your current configuration can be empty, or it can already contain blocks and fields.

The configuration you import must be available as a sample file. See StreamIN samples on page 47.

To import a StreamIN configuration

1. Select File > Load Message Definition. The Load Message Definition dialog box opens.
2. Browse to, and select, the configuration you want to import.
3. Click OK.
Managing blocks and fields

Blocks

A block is a set of recurring fields. You use separate blocks for every set of recurring fields. A block can contain any number of fields and sub-blocks.

Adding blocks

If you have loaded a sample, see StreamIN samples on page 47, you can drag-and-drop blocks from the Integration Tool view to the Message view. You can only add a block from the sample once. If you try to add a block that already exists in the Message view, the StreamIN tool cancels the action.

To add a block using a sample

Drag the block, including all fields, from the Integration tool, and drop it at the appropriate position in the Message tree.

To add a block manually

1 Right-click the node (Field folder or block) below which you want to insert the block.
2 Select New > Block. The new block is added to the Message tree.
3 Rename the block.

Configuring blocks

1 Select the block you want to configure. The block properties are displayed in the Properties view.
2 Edit the properties. See Block properties on page 62.

Sorting

You can use sort criteria to specify the order in which data will be delivered to the subsequent Processes. If no sort criteria is used, data will be delivered in the same order as it arrives. Performance will be less affected if data is sorted at Event level, compared to sorting at Process level.

You specify the sort criteria by assigning priorities to blocks. Data associated with blocks with the highest priority will be delivered first, and so on.

To specify sort criteria for Parent-level blocks

1 Select the Message node in the Message tree. The Message properties are displayed in the Properties view.
2 Set Use block sort priority to Yes.
3 Select the first Parent-level block. The Block properties are displayed in the Properties view.

4 Set the **Block sort priority** level. The lower the number, the higher the priority.

5 Repeat steps 3 and 4 for all Parent-level blocks.

**To specify sort criteria for sub-blocks within a block**

1 Select the block. The block properties are displayed in the Properties view.

2 Set **Use block sort priority** to **Yes**.

3 Select the first sub-block. The Block properties are displayed in the Properties view.

4 Set the **Block sort priority** level. The lower the number, the higher the priority.

5 Repeat steps 3 and 4 for all sub-blocks.

**Sorting examples**

The following examples illustrate how the order of the output from an Event changes when sort criteria is specified for the main blocks in the Event.

---

**Example 10** *Event output without sorting*

All blocks have sort priority set to 0. Output is delivered in the same order as it arrived.

<table>
<thead>
<tr>
<th>Input</th>
<th>Configuration</th>
<th>Unsorted data</th>
</tr>
</thead>
<tbody>
<tr>
<td>E Dylan</td>
<td></td>
<td>E Dylan</td>
</tr>
<tr>
<td>L Armstrong</td>
<td></td>
<td>L Armstrong</td>
</tr>
<tr>
<td>D Gillespie</td>
<td></td>
<td>D Gillespie</td>
</tr>
<tr>
<td>J S Bach</td>
<td></td>
<td>J S Bach</td>
</tr>
<tr>
<td></td>
<td><strong>Classic</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Priority = 0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Jazz and Blues</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Priority = 0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Rock and Pop</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Priority = 0)</td>
<td></td>
</tr>
</tbody>
</table>
Example 11  Event output with sorting

The blocks have sort priority set to 1, 2, and 3. Output is delivered in this order.

<table>
<thead>
<tr>
<th>Input</th>
<th>Configuration</th>
<th>Sorted data</th>
</tr>
</thead>
<tbody>
<tr>
<td>D. Dylan</td>
<td>[Sort by: 1]</td>
<td>1 Dylan</td>
</tr>
<tr>
<td>L. Armstrong</td>
<td>[Sort by: 2]</td>
<td>2 L. Armstrong</td>
</tr>
<tr>
<td>D. Gillespie</td>
<td>[Sort by: 3]</td>
<td>3 D. Gillespie</td>
</tr>
<tr>
<td>J. Brahms</td>
<td></td>
<td>4 J. Brahms</td>
</tr>
<tr>
<td>M.S.</td>
<td></td>
<td>5 M.S.</td>
</tr>
<tr>
<td>J. S. Bach</td>
<td></td>
<td>6 J. S. Bach</td>
</tr>
</tbody>
</table>

Fields

A field in the StreamIN tool corresponds to a field in a description file. In the StreamIN tool, you specify how to organize and configure the fields. You can organize the fields by adding them manually, or you can use dictionaries to drag-and-drop fields. See StreamIN samples on page 47.

Adding fields

If you have loaded a sample, see StreamIN samples on page 47, you can drag-and-drop fields from the Integration Tool view to the Message view. You can only add a field from the sample once. If you try to add a field that already exists in the Message view, the StreamIN tool cancels the action.

To add a field using a sample

Drag the field from the Integration tool, and drop it at the appropriate position in the Message tree.

To add a field manually

1. Right-click the Field folder below which you want to insert the field.
2. Select New > Field. The new field is added to the Message tree.
3. Rename the field.

Configuring fields

1. Select the field you want to configure. The field properties are displayed in the Properties view.
2. Edit the properties. See Field properties on page 63.
Field variables

You can define field variables, and later on refer to the variable instead of static values. Variables affect performance, so only use them when necessary.

To create a variable

1. Right-click the field and select Edit > Make Variable. The variable is added to the field properties (Properties view).
2. The variable name will be the same as the field name. If needed, rename the variable.

You can also multi-select fields (SHIFT + select or CONTROL + select) and define variables for all fields in one action.

Numeric and date formats

There are three input format categories for the fields:

- **General.** Data will be treated as a regular string of characters. This is the default format.
- **Numeric.** Enables the StreamServer to handle input data as numeric data.
- **Date.** Enables the StreamServer to handle input data as date formatted data.

Format tables

Numeric and date formats are made available through format tables. Before you specify numeric or date formats for the fields, you must add a format table to a resource set connected to the Message. You can import Formats.txt from `<StreamServe installation>\Applications\StreamServer\<version>\Tools\Samples` to the resource set.

The first time you specify a numeric or date format for a field, a resource selection dialog box opens. In this dialog box you must browse to, and select, the format table you want to use. This table will be selected by default the next time you specify a format for any of the fields in the Event configuration.

To select a numeric | date format for a field

1. Select the field. The field properties are displayed in the Properties view.
2. At the Input format field, click the browse button. The Formats dialog box opens.
3. Select the Numeric | Date category.
4. Double-click the Format that corresponds to the input format.

**Example 12**  

**Numeric formats**

Input `1000000,25` corresponds to `k=d=`,
Input 1000,000.25 corresponds to $k=, d=.$

---

**Example 13** Date formats

Input 31/10/03 corresponds to dd/mm/yy
Input 2003-10-03 corresponds to yyyy-mm-dd

---

**To add a new format**

Enter the new format in the **Format** field and click **Add**.

You can also add new formats directly to the format table resource.
StreamIN tool GUI reference

Main window

The Main window contains three views:

- **Message view**
  This is where you create and configure the structure of the StreamIN Event.

- **Properties view**
  Select a field or block in the Message view, and configure the corresponding properties in the Properties view.

- **Integration Tool view**
  Load dictionaries to the Integration Tool view, and drag-and-drop fields and blocks to the Message view.

File menu

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Create a new StreamIN configuration.</td>
</tr>
<tr>
<td>Open</td>
<td>Open an existing (stand-alone) StreamIN configuration file.</td>
</tr>
<tr>
<td>Save</td>
<td>Save the StreamIN configuration as data embedded in the corresponding Message file in the Design Center Project.</td>
</tr>
<tr>
<td>Save As</td>
<td>Save the StreamIN configuration as a separate file.</td>
</tr>
<tr>
<td>Load Message Definition</td>
<td>Import and add a StreamIN configuration to the current StreamIN configuration. The current configuration can be empty, or it can already contain blocks and fields.</td>
</tr>
<tr>
<td>Save Message Definition</td>
<td>Save the current StreamIN structure as a sample resource in the appropriate resource set.</td>
</tr>
<tr>
<td>Event Information</td>
<td>View and edit information – author, company, etc.</td>
</tr>
<tr>
<td>Exit</td>
<td>Exit the StreamIN tool.</td>
</tr>
</tbody>
</table>
Edit menu

Sort  
Applicable to the Message node and on Block nodes.
Sort the items beneath the selected node.

Make variable  
Define a variable for the selected field.
You can multi-select fields (SHIFT + select or CONTROL +select) and define variables for all fields in one action.

Insert menu

New Field  
Insert a new field.

New Block  
Insert a new block below the node selected in the Message tree.

New Parent-level block  
Insert a new block below the Message node in the Message tree.

Tools menu

Customize  
Customize the look-and-feel of the StreamIN tool (tool bars, tool tip, etc.).

SXD Converter  
Convert *.dic or *.dsc files to *.sxd files.
Message view

Used for: Creating and configuring the StreamIN Event structure.

<table>
<thead>
<tr>
<th>Shortcut menu options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expand subnodes</td>
</tr>
<tr>
<td>Collapse subnodes</td>
</tr>
<tr>
<td>New &gt; Field</td>
</tr>
<tr>
<td>New &gt; Block</td>
</tr>
<tr>
<td>New &gt; Parent-level block</td>
</tr>
<tr>
<td>Load Message Definition</td>
</tr>
<tr>
<td>Save Message Definition</td>
</tr>
<tr>
<td>Edit &gt; Make Variable</td>
</tr>
<tr>
<td>Edit &gt; Sort</td>
</tr>
</tbody>
</table>
Properties view

**Used for:** Viewing and editing field and block properties. Select field/block in the Message view, and edit the properties in this view.

Message properties

**Used for:** Specifying whether or not to use sort criteria for the Parent-level blocks.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use block sort priority</td>
<td>Select to enable sorting of data.</td>
</tr>
</tbody>
</table>

Block properties

**Used for:** Editing block properties. Select the block in the Message view, and edit the properties in this view.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label (name)</td>
<td>Block name. Will be displayed in the Process tool.</td>
</tr>
<tr>
<td>Language</td>
<td>Language for the description below.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the block. You can enter descriptions in several languages.</td>
</tr>
<tr>
<td>Comment</td>
<td>Additional description. Language selection does not apply to this property.</td>
</tr>
<tr>
<td>Block sort priority</td>
<td>Set the sort criteria for this block. See <em>Sorting</em> on page 53.</td>
</tr>
<tr>
<td>Use block sort priority</td>
<td>Select to enable sorting of data in the sub-blocks.</td>
</tr>
<tr>
<td>Array type</td>
<td>Select whether or not to create an array of field instances within the block. Only applicable to fields defined as variables.</td>
</tr>
</tbody>
</table>

**Example**

A variable $A$ is specified for Field_1 in Block_1. At the first occurrence of Block_1, the data in Field_1 will be placed in element $A[0]$ of the array. At the second occurrence of Block_1, the data in Field_1 will be placed in element $A[1]$, etc. $A$ is the common instance and $[n]$ is the index.
# Field properties

**Used for:** Editing field properties. Select the field in the Message view, and edit the properties in this view.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Label (name)</strong></td>
<td>Field name. Will be displayed in the Process tool. <strong>RecordIN</strong> The same name as in the description file. If you use the same field names in different records, you must add the record name to the field name: <code>&lt;record name&gt;_&lt;field name&gt;</code> <strong>FieldIN</strong> Must have the same name as the field ID in the input data.</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>Language for the description and sample data below.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Description of the field. You can enter descriptions in several languages.</td>
</tr>
<tr>
<td><strong>Sample data</strong></td>
<td>An example of field content. You can enter sample data in several languages.</td>
</tr>
<tr>
<td><strong>Comment</strong></td>
<td>Additional description. Language selection does not apply to this property.</td>
</tr>
<tr>
<td><strong>Variable</strong></td>
<td>Name of a field variable. See <em>Field variables</em> on page 56.</td>
</tr>
<tr>
<td><strong>Variable type</strong></td>
<td>Field class that can assist formatting in a PageOUT Process. For example, if you specify a font for a class in the Process, the font will be used for all fields belonging to this class. <strong>Label</strong> – For fields containing static data. <strong>Dynamic</strong> – For fields containing dynamic data. <strong>Header</strong> – For fields containing static header data.</td>
</tr>
<tr>
<td><strong>Alignment</strong></td>
<td>Specify alignment of data in the Process tool.</td>
</tr>
<tr>
<td><strong>Input format</strong></td>
<td>See <em>Numeric and date formats</em> on page 56.</td>
</tr>
<tr>
<td><strong>Keep spaces</strong></td>
<td>Select whether or not to keep leading spaces and trailing spaces defined in the field when the field is used in the output data.</td>
</tr>
<tr>
<td><strong>Job ID</strong></td>
<td>Select whether or not to assign an index to the content of the field to make it searchable in a Job ID repository.</td>
</tr>
</tbody>
</table>