StreamServe Persuasion SP5 XMLIN

User Guide

Rev A

StreamServe Persuasion SP5 XMLIN User Guide
Rev A
© 2001-2010 STREAMSERVE, INC.
ALL RIGHTS RESERVED
United States patent #7,127,520

StreamServe Web Site
http://www.streamserve.com

Contents

About XMLIN

The StreamServer can receive and process XML formatted input. The instructions to the StreamServer for handling XML input are created using the XMLIN tool.

Samples

When you create the XMLIN configuration, you can use XML documents as samples. See *XML document samples* on page 9.

Configuration

When you create an XMLIN configuration, you specify which information to select from the XML input, and how to label and structure this information as blocks and fields. See *Creating an XMLIN configuration* on page 15.

How StreamServer reads and processes XML documents

The StreamServer reads an XML document starting from the top. In the collection phase, the StreamServer continuously reads, parses, and saves input data until it comes to a specific check point in the document. The check point can be an end tag, a comment, a pattern, etc. What it is depends on the current collection mode – Document, Node, or Message. When a check point is reached, the StreamServer swaps to the process phase, and processes data that is currently available in the memory. The StreamServer continues to swap between collection phase and process phase until it reaches the end tag in the XML document.

Multiple Events using the same XML document as input

The StreamServer may use different collection modes for different parts of the input XML document. It all depends on the number of Events that use the same XML document as input, and which collection modes you have specified for the Events.

Available data

When the StreamServer processes data, it has access to the data that is currently available in the memory – nothing more and nothing less.

Releasing memory

The more data occupying the memory, the less performance. The StreamServer will therefore continuously release memory when operating in Message and Node collection mode. Consequently, all removed data will be unavailable to the StreamServer in the subsequent process phases.

Collection modes

Document mode	
Scope	This collection mode overrides any other collection mode, i.e. the whole XML document will be collected using the Document collection mode. The collection phase continues all the way through the XML document to the end tag for the root element. This means that the whole XML document will be available during the process phase.
Memory	No memory is released until the StreamServer has finished processing the whole XML document.
Availability vs. performance	This mode provides the best data availability since the whole XML document is available during the entire process phase. Performance will be affected because of the high level of memory usage.

Node mode	
Scope	This is the default collection mode. The collection phase continues until the StreamServer finds an end tag, a comment, or processing instruction.
Memory	**Text nodes:** If an element contains more than one text node at the same level, only one of them is kept in the memory. If a second text node is found while parsing the document, the first text node is removed. For example, while parsing `<A>billy<B/>bob<A/>`, `billy` is removed when `bob` is parsed. **Child elements:** Child elements will be removed when the parent element has been processed. For example, when the StreamServer has processed the elements `<A><B/><C/></A>`, the child elements `<B/>` and `<C/>` will be removed.
Availability vs. performance	This mode provides the best performance because of the low level of memory usage. Data availability will be affected since only a small portion of the XML document is available during each process phase.

Message mode	
Scope	The StreamServer runs in Node collection mode until it finds a matching pattern. When it finds a matching pattern, it swaps to Message collection mode – provided the corresponding Event is configured to use Message as collection mode. The collection phase extends from [*pattern start*] to [*pattern end*] according to the following example: `<[`*`pattern start`*`]A><B/><C>text</C></A[`*`pattern end`*`]>`
Memory	When the StreamServer has finished a process phase, only the main element is kept – everything else is removed. For example, after processing `<A><B/><C>text</C></A>`, only `<A></A>` is kept.
Availability vs. performance	Data availability and performance depends on the amount of data enclosed by [*pattern start*] and [*pattern end*]. For example, if the pattern is defined for the root element, the Message mode is the same as the Document mode. On the other hand, if the pattern encloses one single element, the Message mode is the same as the Node mode.

Nested Events

Multiple Events can retrieve input from the same XML document. Which parts of the document an Event will retrieve is defined by patterns. If one Event's pattern is enclosed by another Event's pattern, this is known as nested Events.

Example 1 *Nested Events*

```
...
<EventA>text
   <EventB>text</EventB>
</EventA>
...
```

If `EventA` operates in collection mode Message, `EventB` will also be forced to operate in collection mode Message. Everything enclosed by the parent pattern – in this example `EventA` and `EventB` – will be available during the process phase. This means that the StreamServer will have access to all data that belongs to `EventA` when it processes `EventB`, and vice versa.

XML document samples

You can load one or more XML document samples into the XMLIN tool, and use these samples when you configure XMLIN Events. You can use an existing XML document, or let the XMLIN tool generate an XML document based on an DTD or XSD (XML schema). See *Generating samples from DTDs and XSDs* on page 10.

Resources

DTSs, XSDs, and existing XML document samples are all resources that you load to the XMLIN tool. Before you load a resource, you must import the corresponding file to a resource set connected to the corresponding Message.

To load a sample

1. Select **File** > **Open Sample**. The Select Resource dialog box opens.
2. Browse to the sample folder and select the resource.

To specify whether or not to unload

1. Select **Tools** > **Options**. The options dialog box opens.
2. On the Samples tab, specify whether or not to **Save samples on exit.**

Generating samples from DTDs and XSDs

If you have the appropriate DTD/XSD, you can use the XMLIN tool to generate a sample based on the DTD/XSD. The generated sample is a temporary file that is loaded into the XMLIN tool.

Methods

You can use three methods to generate an XML document sample based on a DTD/XSD:

- **Generate on load**. With this method, you cannot configure anything. The XML document will be created when the DTD/XSD is loaded. See *Generate on load* on page 11.
- **Manually – typical**. With this method, you can specify the number of instances and recursion level for the elements. What you specify here applies to all elements. You can also specify which element is the root element, and where to store the temp file. See *Typical configuration* on page 11.
- **Manually – custom**. With this method, you can customize the number of instances and recursion level per element. You can also specify which element is the root element, and where to store the temp file. See *Custom configuration* on page 12.

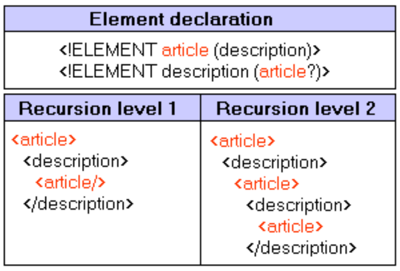
Recursion level and Number of instances

The DTD/XSD contains information about the number of allowed instances of an element type within another element type. This information does not explicitly describe “five type A elements, ten type B elements” etc. Instead it describes “one type A element, one or more type B elements, zero or more type C elements” etc. If no more information was provided, the XMLIN tool would not know when to stop generating the XML sample. To know when to stop, the XMLIN tool needs the following input:

- **Recursion level** – specifies how many instances of itself an element type may contain. See *Recursion level*.
- **Number of instances** – specifies how many instances of other element types an element type may contain. See *Number of instances*.

Example 2 *Recursion level*

This DTD example illustrates how many instances of itself the `<article>` element can contain when Recursion level is set to 1 and 2.



Element declaration

```
<!ELEMENT article (description)>
<!ELEMENT description (article?)>
```

Recursion level 1

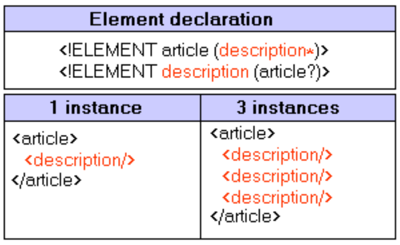
```
<article>
  <description>
    <article/>
  </description>
```

Recursion level 2

```
<article>
  <description>
    <article>
      <description>
        <article>
      </description>
```

Example 3 *Number of instances*

This DTD example illustrates how many instances of the `<description>` element the `<article>` element can contain when Number of instances is set to 1 and 3.



Element declaration

```
<!ELEMENT article (description*)>
<!ELEMENT description (article?)>
```

1 instance

```
<article>
  <description/>
</article>
```

3 instances

```
<article>
  <description/>
  <description/>
  <description/>
</article>
```

Generate on load

The XMLIN tool can automatically generate an XML document sample when a DTD/XSD is loaded. The generated sample will include all possible combinations of the elements declared in the DTD/XSD. The document element will be the element that is most likely to be the root element, and both Number of instances and Recursion level will be set to 1.

To enable automatic sample generation

1 Select **Tools** > **Options**. The options dialog box opens.

2 On the Samples tab, select **Automatically generate DTD/XSD-based instance document on load.**

Typical configuration

With this method, you can specify the number of instances and recursion level for the elements. What you specify here applies to all elements. The generated sample will include all possible combinations of the elements declared in the DTD/XSD.

To generate and load the sample

1. Load the DTD/XSD.
2. Right-click the DTD/XSD and select **Generate Instance Document**. The XML Builder – Settings dialog box opens.
3. Select **Typical**, configure the settings, and click **OK**. The XML document sample is loaded.

Settings	
Number of instances	Defaults to 1. You can increase the number.
Recursion level	Defaults to 1. You can increase the number.
Document element	Defaults to the element that is most likely to be the root element.
XML result file	Path to the result file (generated sample).

Custom configuration

With this method, you can customize the number of instances and recursion level for each element.

The generated sample will by default include the elements that are unambiguously declared in the DTD/XSD. You must insert all other elements manually.

XML Builder initial settings	
Document element	Defaults to the element that is most likely to be the root element.
XML result file	Path to the result file (generated sample).

To generate and load the sample

1. Load the DTD/XSD.
2. Right-click the DTD/XSD and select **Generate Instance Document**. The XML Builder – Settings dialog box opens.
3. Select **Custom**.
4. Specify the **Document element** (root element) and click **OK**. The XML Builder – Add Element Children dialog box opens.
5. For each active element, you can add new sub-elements. See *Figure 1* on page 13. Click **OK** to continue with the next element in turn.

6 When you have reached the last element, and clicked **OK**, a preview of the final XML document sample is displayed in the XML Builder – Result Tree dialog box. You can edit the attribute values.

7 Click **OK**. The XML document sample is generated and loaded.

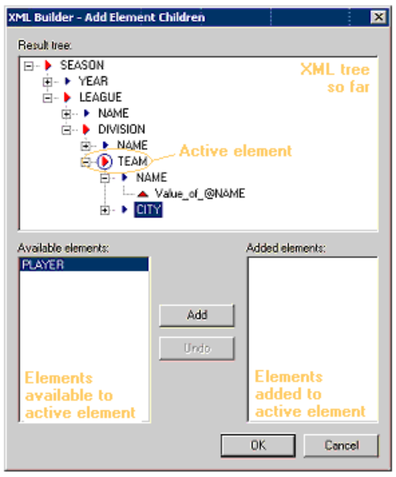


Figure 1 Add element children

Validating samples

You can validate an XML document sample against the corresponding DTD or XSD (XML Schema).

DTDs

The `<!DOCTYPE...>` tag in the sample determines which DTD to validate against. For example:

```
<!DOCTYPE Music SYSTEM "C:\DTDS\musichits.dtd">
```

XSDs

In the root element, you must use the schema location attributes `*prefix*:noNamespaceSchemaLocation` or `*prefix*:schemaLocation` to specify which XSD to validate against. For example:

```
<root xsi:schemaLocation="C:\DTDS\musichits.xsd">
```

The schema location attributes have the namespace URI `http://www.w3.org/2001/XMLSchema-instance`.

See the W3C XML schema recommendation for more information.

Validating on load

You can configure the XMLIN tool to validate a sample each time it is loaded:

1 Select **Tools** > **Options**. The Options dialog box opens.

2 On the Samples tab, select **Validate instance document on load**.

Validating manually

You can also validate a sample manually by right-clicking the sample and selecting **Validate**.

Creating an XMLIN configuration

An XMLIN configuration determines what to extract from an XML input document. How to create the XMLIN configuration depends on the type of data you have available.

Collection mode

The collection mode is an important parameter for all XMLIN configurations. It determines how much of the XML input is available to the StreamServer when it processes the data. Which mode to use depends on what is the most important – availability or performance. See *Selecting collection mode* on page 32 and *How StreamServer reads and processes XML documents* on page 5.

Namespace declarations

For each namespace declared in the XML input, you must add a corresponding namespace object to the XMLIN configuration. See *Namespaces in an XMLIN configuration* on page 31.

Automatic generation of an XMLIN configuration

You can use the XMLIN tool to automatically generate a configuration based on a sample. You can then edit the configuration by adding fields, reorganizing blocks, etc. See *Generating an XMLIN configuration automatically* on page 16.

Creating an XMLIN configuration manually

You can also create the XMLIN configuration manually – completely from scratch or by picking nodes from a sample. See *Creating an XMLIN configuration manually* on page 17.

Default settings suggested by the XMLIN tool

You can configure the default settings for how the XMLIN tool should generate match criteria for patterns and fields etc. when it picks nodes from a sample. See *Default match criteria for fields and patterns* on page 41 and *Field ID expressions* on page 44.

Generating an XMLIN configuration automatically

You can use the XMLIN tool to pick nodes from an XML document sample and automatically generate an XMLIN configuration with namespaces, fields, blocks, sub-blocks, etc. See *XML document samples* on page 9 for information about how to create and load samples.

To generate an XMLIN configuration

1. Load the XML document sample.
2. Right-click the sample and select **Pattern Tool**.
3. In the sample, click the node you want to use as top node in the XMLIN configuration.
4. Select **Tools** > **Extract Message**. An XMLIN configuration based on data from nodes below the top node is generated.

To edit patterns

See *Patterns* on page 18.

To edit blocks

See *Blocks* on page 21.

To edit fields

See *Fields* on page 24.

To edit namespaces

See *Namespaces in an XMLIN configuration* on page 31.

To select collection mode

See *Selecting collection mode* on page 32.

Creating an XMLIN configuration manually

You can create an XMLIN configuration manually – completely from scratch or by picking nodes from a sample.

To add and configure patterns

See *Patterns* on page 18.

To add and configure blocks

See *Blocks* on page 21.

To add and configure fields

See *Fields* on page 24.

To add and configure namespaces

See *Namespaces in an XMLIN configuration* on page 31.

To select collection mode

See *Selecting collection mode* on page 32.

Managing patterns, blocks, and fields

You will most likely have to edit at least some parts of an automatically generated XMLIN configuration – add/remove fields, reorganize blocks, edit pattern properties, etc. If you do not generate the XMLIN configuration automatically, you must create and configure all patterns, blocks, and fields manually. This section describes how to manage patterns, blocks, and fields in any XMLIN configuration – both automatically generated and manually created.

Patterns

There must be at least one pattern in the XMLIN configuration. Any type of node – element, text, etc. – can be used as a pattern. The main pattern determines which portions of the input XML document this specific XMLIN Event will use. You can also define one or more additional patterns that you use to set up rules for triggering the XMLIN Event. See *Triggering an Event using multiple patterns* on page 20.

To edit pattern properties

1 In the Message view, select the pattern. The properties are displayed in the Properties view.

2 Edit the pattern properties. See *Pattern properties* on page 56.

Adding a new pattern

You can add a new pattern to an XMLIN configuration in two ways:

- Pick the pattern from an XML document sample.
- Add the pattern manually.

To pick the pattern from a sample

1 Right-click the sample and select **Pattern Tool**.

2 In the sample, click the node you want to map as pattern. The pattern is added to the Message view.

3 Edit the pattern properties. See *Pattern properties* on page 56.

The default match criterion for the pattern depends on the type of node. Elements will have the path to the node as default match criteria. Text nodes, attribute values, comments and processing instructions will also include a condition (predicate) in the match criterion,

To add a pattern manually

1 In the **Message** view, right-click the Message node and select **New** > **Pattern**. The pattern is added to the Message view.

2 Edit the pattern properties. See *Pattern properties* on page 56.

Match criteria for patterns

Any type of node – element, text, etc. – can be used as a trigger pattern for an XMLIN Event. A match criterion for a trigger pattern is specified as an XSLT Pattern. The XSLT Pattern describes the path to a node, and if required, additional conditions.

To specify a match criterion

1 In the Message view, select the field. The properties are displayed in the Properties view.

2 In the **Match** field, enter the XSLT Pattern. You can use all valid XSLT patterns. See www.w3.org/TR/xslt.

Examples

These examples have the following XML input:

```
<?xml version="1.0"?>
  <music>List
    <CD type="western">Classic
        <Composer>Brahms</Composer>
    </CD>
    <CD type="eastern">Classic
        <Composer>Quing Mao</Composer>
    </CD>
    <CD type="Africa">Classic
        <Composer>Outou Badou</Composer>
    </CD>
 </music>
```

Example 4 *Match criterion specified as "/music"*

The match criterion is specified as “`/music`”.

The StreamServer scans the input and finds a match in `<music>`. The rule is fulfilled, and the Event is triggered. The Event is triggered once in this scenario.

Example 5 *Match criterion specified as "/music/CD"*

The match criterion is specified as “`/music/CD`”.

The StreamServer scans the input and finds a match in `<CD>`. The rule is fulfilled, and the Event is triggered. The StreamServer continues to scan the input, and finds two more matches for `<CD>`. The rule is fulfilled, and the Event is triggered two times more. The Event is triggered three times in this scenario.

Example 6 *Match criterion specified as "/music/CD[2]"*

The match criterion is specified as “`/music/CD[2]`”.

The StreamServer scans the input and finds a match in the second occurrence of `<CD>`. The rule is fulfilled, and the Event is triggered. The Event is triggered once in this scenario.

Example 7 *Match criterion specified as "/music/CD[@type=’western’]"*

The match criterion is specified as “`/music/CD[@type=’western’]`”.

The StreamServer scans the input and finds a match in `<CD type="western">`. The rule is fulfilled, and the Event is triggered. The Event is triggered once in this scenario.

Supported XSLT and XPath functions

In the XSLT Patterns, you can use the XSLT and XPath functions listed in *Supported XSLT and XPath functions* on page 45.

Triggering an Event using multiple patterns

You can define multiple patterns that you use to set up rules for triggering the XMLIN Event. A rule can be an AND or OR statement, or a combination of both. For example:

```
Pattern1 OR (Pattern2 AND Pattern3)
```

Valid characters for the pattern IDs in a rule
A-Z
a-z
0-9
_ (underscore)
. (dot)

To create a rule

1 In the **Message** view, select the Message node. The Message properties are displayed in the Properties view.

2 In the **Rule** field, enter the rule.

Example 8 *Pizzas and drinks*

The patterns `pizzas` and `drinks` match the elements `<pizzas>` and `<drinks>` in the sample below.

```
...
<pizzas>
  <pizza>Calzone</pizza>
  <pizza>Primavera</pizza>
</pizzas>
<drinks>
  <drink>Manhattan</drink>
</drinks>
...
```

Rule 1:

```
pizzas AND drinks
```

The StreamServer scans the input and finds a match in `<pizzas>`. The StreamServer continues to scan the input and finds a match in `<drinks>`. The rule is fulfilled, and the Event is triggered. The Event is triggered once in this scenario.

Rule 2:

```
pizzas OR drinks
```

The StreamServer scans the input and finds a match in `<pizzas>`. The rule is fulfilled, and the Event is triggered. The StreamServer continues to scan the input and finds a match in `<drinks>`. The rule is fulfilled, and the Event is triggered once more. The Event is triggered twice in this scenario.

Blocks

Recurring data must be defined as fields within a block, and each family of recurring data must have its own block. A block in the XMLIN configuration can contain sub-blocks.

A block corresponds to an element node in the input document. When you generate an XMLIN configuration automatically, the XMLIN tool concludes which elements to use as blocks.

Example 9 *Blocks in an XML structure*

```
...
<pizzas>
  <pizza>Calzone</pizza>
  <pizza>Primavera</pizza>
</pizzas>
<drinks>
  <drink>Manhattan</drink>
  <drink>Moonshine</drink>
</drinks>
...
```

In the sample file above, `<pizzas>` corresponds to one block, and `<drinks>` to another block.

To edit block properties

1. In the Message view, select the block. The properties are displayed in the Properties view.
2. Edit the block properties. See *Block properties* on page 57.

Adding a new block

You can add a new block to an XMLIN configuration in two ways:

- Pick the block from an XML document sample.
- Add the block manually.

To pick the block from a sample

1. Right-click the sample and select **Block Tool**.
2. In the sample, click the node you want to pick. The block is added to the Message view.
3. Edit the block properties. See *Block properties* on page 57.
4. Add the applicable fields and sub-blocks.

To add a block manually

1. In the **Message** view, right-click the Message node and select **New** > **Block**. The block is added to the Message view.
2. Edit the block properties. See *Block properties* on page 57.
3. Add the applicable fields and sub-blocks.

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. You can specify sort criteria for all main blocks and sub-blocks in the XMLIN configuration.

To specify sort criteria for the main 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 main 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 main 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.

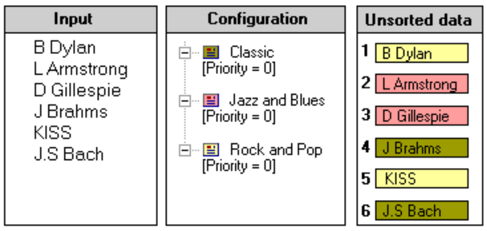
5 Repeat steps 3 and 4 for all sub-blocks.

Sorting examples

The following examples illustrate how the order in which data will be delivered changes when sort criteria is specified for the blocks.

Example 10 *No sort criteria is used*

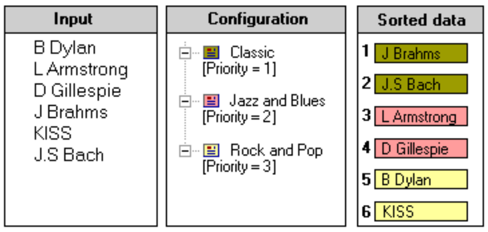
All blocks have sort priority set to 0. Output is delivered in the same order as it arrived.



Input	Configuration	Unsorted data
B Dylan	Classic [Priority = 0]	1 B Dylan
L Armstrong	Jazz and Blues [Priority = 0]	2 L Armstrong
D Gillespie	Rock and Pop [Priority = 0]	3 D Gillespie
J Brahms		4 J Brahms
KISS		5 KISS
J.S Bach		6 J.S Bach

Example 11 *Sort criteria is used*

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



Input	Configuration	Sorted data
B Dylan	Classic [Priority = 1]	1 J Brahms
L Armstrong	Jazz and Blues [Priority = 2]	2 J.S Bach
D Gillespie	Rock and Pop [Priority = 3]	3 L Armstrong
J Brahms		4 D Gillespie
KISS		5 B Dylan
J.S Bach		6 KISS

Fields

A field in the XMLIN configuration can correspond to any type of node in the XML input document. You must use XSLT patterns to specify the path to a node, and XPath expressions to specify what to extract from the XML input.

To edit field properties

1. In the Message view, select the field. The properties are displayed in the Properties view.
2. Edit the field properties. See *Field properties* on page 57.

Adding a new field

You can add a new field to an XMLIN configuration in two ways:

- Pick the field from an XML document sample.
- Add the field manually.

To pick the field from a sample

1. Right-click the sample and select **Field Tool**.
2. In the sample, click the node you want to map. The field is added to the Message view.
3. Edit the field properties. See *Field properties* on page 57.

To add a field manually

1. In the **Message** view, right-click the Message node and select **New** > **Field**. The field is added to the Message view.
2. Edit the field properties. See *Field properties* on page 57.

Mapping XML nodes to fields in the XMLIN configuration

You use XSLT Patterns as match criteria to map nodes in the XML input document to fields in the XMLIN configuration. A match criterion describes the path to the node and, if required, additional conditions. XPath expressions define which value to add to the field.

Before you start mapping nodes to fields, you can edit the default match criteria suggested by the XMLIN tool. See *Default match criteria for fields and patterns* on page 41.

To map XML nodes to fields

1. In the Message view, select the field. The properties are displayed in the Properties view.
2. In the **Match** field, enter the XSLT Pattern. You can use all valid XSLT Patterns. See www.w3.org/TR/xslt.

3 In the **Value** field, enter the XPath expression. You can use all valid XPath expressions. See www.w3.org/TR/xpath.

Match criteria examples

These examples have the following XML input:

```
<?xml version="1.0"?>
  <music>List
    <CD>Classic
        <Composer type="western">Brahms</Composer>
        <Composer type="western">Grieg</Composer>
    </CD>
   <CD>Classic
        <Composer type="eastern">Quing Mao</Composer>
        <Composer type="eastern">Han Zuy</Composer>
    </CD>
 </music>
```

Example 12 *Match criterion for <Composer> set to "CD/Composer"*

The match criterion is specified as "`CD/Composer`". There will be a match for all <`Composer`> nodes in the input.

Example 13 *Match criterion for <Composer> set to "CD/Composer[2]"*

The match criterion is specified as "`CD/Composer[2]`". There will be a match for the second `<Composer>` node below each `<CD>` node.

Example 14 *Match criterion for <Composer> set to "CD/Composer[@type='eastern']"*

The match criterion is specified as `CD/Composer[@type='eastern']`. There will be a match for all <`Composer`> nodes with attribute `type="eastern"`.

Field value examples

These examples have the following XML input:

```
<?xml version="1.0"?>
  <music>List
    <CD>Classic
        music
        <Composer>Brahms</Composer>
    </CD>
 </music>
```

The nodes `<CD>` and `<Composer>` correspond to the fields `CD` and `Composer` respectively. The match criteria are specified as `/music/CD` and `/music/CD/Composer`.

All examples use the Document collection method. See *How StreamServer reads and processes XML documents* on page 5 for information about collection methods.

Example 15 *Value for field CD specified as "."*

The value for the field `CD` is specified as "." (dot). The following value is extracted from the XML input:

```
Classic
music
   Brahms
```

All characters, including tabs, line-break, etc., between `<CD>` and `</CD>` are extracted.

Example 16 *Value for field CD specified as "normalize-space(.)"*

The value for the field `CD` is specified as `normalize-space(.)`. The following value is extracted from the XML input:

```
Classic music Brahms
```

The text nodes between `<CD>` and `</CD>` are extracted. Line-breaks, tabs, etc. are ignored.

Example 17 *Value for field CD specified as "text()"*

The value for the field `CD` is specified as `text()`. The following value is extracted from the XML input:

```
Classic
music
```

All characters, including tabs, line-break, etc., between `<CD>` and `<Composer>` are extracted.

Example 18 *Value for field CD specified as "normalize-space(text())"*

The value for the field `CD` is specified as `normalize-space(text())`. The following value is extracted from the XML input:

```
Classic music
```

The text nodes between `<CD>` and `<Composer>` are extracted. Line-breaks, tabs, etc. are ignored.

Example 19 *Value for field Composer specified as "../../text()"*

The value for the field `Composer` is specified as `../../text()`. The following value is extracted from the XML input:

```
List
```

Example 20 *Value for field Composer specified as "normalize-space(../../.)"*

The value for the field `Composer` is specified as `normalize-space(../../.)`. The following value is extracted from the XML input:

```
List Classic music Brahms
```

Example 21 *Value for field Composer specified as "concat(name(.), ':', ' ', .)"*

The value for the field `Composer` is specified as

```
concat(name(.), ':', ' ',  .).
```

The following value is extracted from the XML input:

```
Composer: Brahms
```

Supported XSLT and XPath functions

In the XSLT Patterns and XPath expressions, you can use the XSLT and XPath functions listed in *Supported XSLT and XPath functions* on page 45.

Field variables

You can create field variables, and later on refer to the variables instead of static values.

You can configure the XMLIN tool to always define a variable for each field extracted from an XML document sample. Variables affect performance, so only use them when necessary.

To manually define a variable

1 Select the field you want to configure. The field properties are displayed in the Properties view.

2 In the **Variable** field, enter the name of the variable and press `Enter`.

You cannot enter the `$` prefix in the Variable field. The `$` prefix will be added when you export the Project.

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 In the **Input Format** field, click **Select**. The Formats dialog box opens.

3 Select the **Numeric** | **Date** category.

4 Double-click the **Format** that corresponds to the input format.

Example 22 *Numeric formats*

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

Example 23 *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.

Namespaces in an XMLIN configuration

For each namespace declared in the XML input, you must add a corresponding namespace object to the XMLIN configuration. The namespaces are used in the match criteria for fields and patterns.

When you load an XML document sample, all declared namespaces are automatically added to the XMLIN configuration.

To edit namespace properties

1. In the Message view, select the namespace. The properties are displayed in the Properties view.
2. Edit the namespace properties. See *Namespace properties* on page 58.

To add a new namespace manually

In the **Message** view, right-click the Message node and select **New** > **Namespace**. The namespace is added to the Message view.

Renaming namespace prefixes

If you rename prefixes for automatically generated namespaces, and reload the sample, the sample will be updated with the new prefixes.

Note: Rename the prefixes before you generate the XMLIN configuration.

Selecting collection mode

The collection mode determines how much of the XML input is available to the StreamServer when it processes the data. Which mode to use depends on what is the most important – availability or performance. See *How StreamServer reads and processes XML documents* on page 5.

To select collection mode Message or Node

1. Make sure that Document collection mode is disabled (see below).
2. Select **Tools** > **Event Options**. The Event Options dialog box opens.
3. On the Message tab, select **Node** | **Message** and click **OK**.

To select collection mode Document

1. In the Design Center Message window, right-click the XMLIN Event and select **Settings**. The Event Settings dialog box opens.
2. On the Agent Settings tab, select **Collect entire documents** and click **OK**.

Note: Document mode overrides any other collection mode.

Handling references to external files in the XML input

The XML input can contain references to external files. To be able to extract data from the external files, you must:

- Add the applicable parts of the external document to the XMLIN configuration.
- Enable extraction of data from external files.

To enable extraction of data from external files

1. In the Design Center Message window, right-click the XMLIN Event and select **Settings**. The Event Settings dialog box opens.
2. On the Agent Settings tab, select **Expand external entities** and click **OK**.

Caching external files

The StreamServer can cache the external file instead of downloading it each time. You specify the file cache settings in the Configure Platform dialog box.

1. Right-click the Platform window and select **Configure Platform**. The Configure Platform dialog box opens.
2. On the File Cache tab, configure the file cache settings.

File cache settings	
File cache base directory	The file cache base directory. You can use an absolute path, or a path relative to the StreamServer's working directory. XML and DTD documents will be cached in a sub-directory called XML.
Enable XML cache	Select this option if you want the StreamServer to cache XML and DTD documents.
XML cache size	The maximum number of documents in the XML cache directory. When the limit is exceeded, the oldest document will be removed.
XML cache time-out	The "best before date" for a cached document. The StreamServer checks this date each time it retrieves a document from the cache. If the document is older than the cache time-out, it will be updated.

Validating XML input

The StreamServer can be instructed to validate the XML input before processing the data. The input can be validated against a DTD or XSD (XML Schema).

Validating against DTDs

The `<!DOCTYPE...>` tag in the XML input determines which DTD to validate against. For example:

```
<!DOCTYPE Music SYSTEM "C:\DTDS\musichits.dtd">
```

Validating against XSDs

You can use an XSD mapping table to determine which XSD to validate against. See *XSD mapping table* on page 36.

You can also use the schema location attributes *`prefix`*`:noNamespaceSchemaLocation` or *`prefix`*`:schemaLocation` in any element. For example:

```
xsi:schemaLocation="C:\DTDS\musichits.xsd"
```

The schema location attributes have the namespace URI `http://www.w3.org/2001/XMLSchema-instance`.

See the W3C XML schema recommendation for more information.

Relative paths to DTDs and XSDs

The path to a DTD or XSD can be relative, for example `<!DOCTYPE Music SYSTEM "musichits.dtd">` or `xsi:schemaLocation="musichits.xsd"`. If the XML input is polled using HTTP(S) Poll input connectors, the path is relative to the polled server's URL. For all other types of input connectors, the path is relative to the Project's export directory.

Validation levels

There are five levels of validation. To select which level to use, you must:

1. In the Design Center Message window, right-click the XMLIN Event and select **Settings**. The Event Settings dialog box opens.
2. On the Agent Settings tab, select **Validation level** and click **OK**.

Validation level	
DTD + XSD strict validation	Validate and preprocess incoming XML against a DTD or XSD. This option decreases performance, but prevents the StreamServer from processing XML documents that are not well-formed or valid. When validating against an XSD, a root element that cannot found in any XSD will be treated as an error.
DTD + XSD lax validation	Same as above. The difference is that a root element that cannot found in any XSD will NOT be considered to be an error.
DTD validation	Validate and preprocess incoming XML against a DTD. This option decreases performance, but prevents the StreamServer from processing XML documents that are not well-formed or valid.
Preprocess	The StreamServer will preprocess the XML input. This option slightly decreases performance, but prevents the StreamServer from processing input that is not well formed. Documents that are not valid will be processed.
No validation/ preprocessing	The StreamServer will not validate or preprocess the XML input. This option gives the best performance, but does not prevent the StreamServer from processing input that is not well formed or valid.

XSD mapping table

An XSD mapping table contains namespace to schema location mappings.

XSD mapping table description	
Table syntax	`namespace_URI   schema_location   load_option`
`namespace_URI`	The namespace URI to map against a schema.
`schema_location`	The schema location. Relative to the export directory

XSD mapping table description	
load_option	Specifies when to load a schema. Can be either `0` or `1`. If *load_option* is not specified, you must use the argument `-xsdimport <0|1>` to define when to load schemas. **0:** The schema will be loaded when input related to the schema is received. The schema is read and parsed before validation. **1:** The schema will be loaded at StreamServer start-up. The schema is always available, but occupies memory.

Example 24 *XSD mapping table*

```
//!multicolumn!
http://www.streamserve.com/xsd/schema/one   schemaone.xsd     1
http://www.streamserve.com/xsd/schema/two   schematwo.xsd     0
http://foo.com                              ./grb/xsd001.xsd  1
```

If schema locations are specified both in the mapping table and by schema location attributes in the XML document (`xsi:schemaLocation` etc.), the following apply:

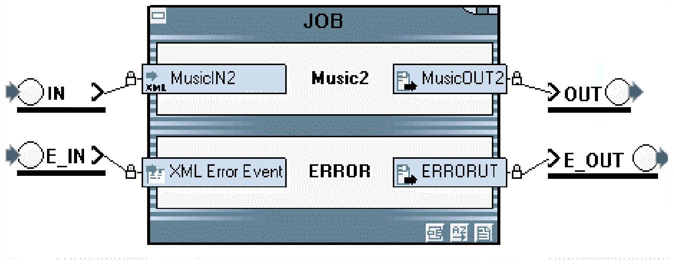
- If the XSD is loaded at start-up, the mapping table will override the schema location attributes.
- If the XSD is not loaded at start-up, the schema location attributes will override the mapping table.

Sending error messages

You can create a Message that sends an error message each time a validation error occurs. See *Creating an XMLIN validation error Message* on page 38.

Creating an XMLIN validation error Message

You can create a specific Message that extracts error information from the StreamServer when a validation error occurs, and sends the information to any type of destination (PDF file, SMS, email, etc.). The figure below shows an example of a Runtime configuration containing one main XMLIN Message and a corresponding *validation error Message*.



Enable validation

You must first enable validation for the main XMLIN Message. You do this by selecting one of the DTD or XSD validation levels. See *Validation levels* on page 35.

Configure connectors for the error Message

The *error Message* must receive input using an Internal input connector. The output connector can be any type of connector depending on the output format and destination.

Associate the XMLIN Event with the error Message

You must configure the XMLIN Event to send error information to the *error Event* via the Internal input connector:

1 In the Design Center Message window, right-click the XMLIN Event and select **Settings**. The Event Settings dialog box opens.

2 In the **XML error connector** field, enter the name of the input connector for the *error Message* and click **OK**.

Configure the error Event

The *error Event* is a MessageIN Event using the SXD file `XML_ErrorEvent.sxd`. See the *MessageIN* documentation for more information on how to configure a MessageIN Event.

Configure the error Process

The Process in the *error Message* can be any type of Process depending on the output format.

Troubleshooting

The XMLIN Event is not triggered

- Check that the XSLT Patterns in the pattern matches are valid.
- Check that all namespaces are declared.
- Check that the collection method makes all required nodes available to the StreamServer.
- If you use multiple patterns, check that any rules can be fulfilled by the incoming data. For example, if you use an AND operator between two patterns, the input data must be able to match both patterns in the same document.

The XMLIN Event is triggered, but not all fields are extracted

- If the field values use data from several nodes, check:
  - that the required data is included in the node-set selected by the XSLT Pattern in the pattern match
  - that the collection method makes all required nodes available to the StreamServer.
- If the fields contain recurring data, check:
  - that the data is defined as fields in blocks
  - that no path indexes are used in the XSLT Patterns in the field matches
  - that the blocks are created at correct level.
- For fields, check:
  - that the XSLT Patterns in the field matches are valid and unique
  - that the XPath expressions in the field values are correct.
- Check that all namespaces are declared.
- If the Event contains nested Events, check:
  - that the collection method makes all required nodes available to the StreamServer
  - that the nested Events do not match fields in the parent Event
  - that the parent Event does not match fields in the nested Events.

The field is extracted, but the value is not correct

- Check that the collection method makes all required nodes available to the StreamServer.
- Check that the XPath expression in the field value is correct.

XSLT and XPath in XMLIN

You use XSLT Patterns when you define fields and patterns in the XMLIN tool. Which value to add to a field is defined by XPath expressions.

Default match criteria for fields and patterns

When you let the XMLIN tool pick fields and patterns from a sample, the XMLIN tool suggests which XSLT Patterns to use in the match criteria definitions. To change the default settings you must select **Tools** > **Event Options** and edit the settings.

See www.w3.org/TR/xslt for more information about XSLT patterns.

Ancestor axis level

On the Message tab you can set the Ancestor axis level which specifies the number of ancestor nodes included in the path for a match criterion. See the examples below.

Path indexes

On the Message tab you can specify whether or not to include path indexes in the path for a match criterion. Path indexes are used to distinguish one branch in the XML tree from another. See the examples below.

Including attributes in the match criteria for field paths

By default, the match criterion suggested for field paths includes element names, but not attributes. On the Custom Field Paths tab, you can configure the XMLIN tool to include a specified attribute in the path. Note that if multiple instances of an element contains this attribute, only first instance, but not its siblings, will have the attribute included in the match path. Elements that do not contain the specified attribute will not be affected. To enable the use of attributes in the path you must:

1. Select **Match element name or attribute**.
2. Enter the name of the attribute to include.

See the examples below.

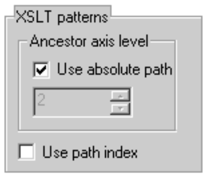
Examples

The XML sample below is used in all examples. The match criteria in all examples refer to the highlighted node.

```
<?xml version="1.0"?>
<!DOCTYPE music SYSTEM "DTDS/musichits.dtd">
  <music>
    <CD>
        <Composer type="western">Brahms</Composer>
   </CD>
   <CD>
        <Composer type="eastern">Quing Mao</Composer>
   </CD>
  </music>
```

Example 25 *Use absolute path, no path index*

XSLT patterns settings:



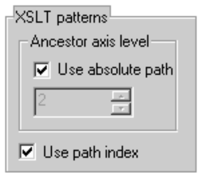
XSLT patterns
Ancestor axis level
[✓] Use absolute path
2
[ ] Use path index

Suggested match criterion:

`music/CD/Composer`

Example 26 *Use absolute path, path index*

XSLT patterns settings:



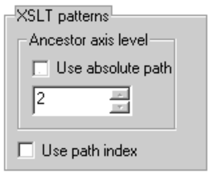
XSLT patterns
Ancestor axis level
[✓] Use absolute path
2
[✓] Use path index

Suggested match criterion:

`music[1]/CD[2]/Composer[1]`

Example 27 *Relative path = 2, no path index*

XSLT patterns settings:

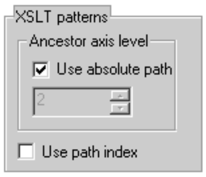

XSLT patterns
Ancestor axis level
☐ Use absolute path
2
☐ Use path index

Suggested match criterion:

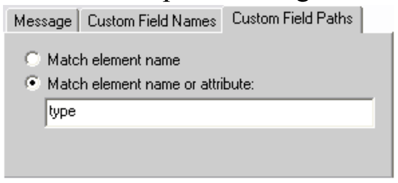
```
CD/Composer
```

Example 28 *Use absolute path, no path index, and match attribute*

XSLT patterns settings:


XSLT patterns
Ancestor axis level
☑ Use absolute path
2
☐ Use path index

Custom field paths settings:



○ Match element name
● Match element name or attribute:
type

Suggested match criterion:

```
music/CD/Composer[@type='eastern']
```

Field ID expressions

When you let the XMLIN tool pick fields from a sample, the XMLIN tool suggests which XPath expressions to use in the field ID definitions. The default XPath expressions for the field IDs are as follows:

Node type	Default XPath expressions for the field ID
Element	`name(.)`
Attribute	`name(.)`
Text	`name(..)`
Comment	`'Comment'`
Processing instruction	`name(.)`

To edit the default settings

1 Select **Tools** > **Event Options**. The Event Options dialog box opens.

2 On the Custom Field Names tab, select **Custom field name for <*node type*>** and edit the expression.

You can select an option from the list or enter your own expression. See www.w3.org/TR/xpath for more information about XSLT patterns.

Supported XSLT and XPath functions

The StreamServer is compliant with XSLT and XPath version 1.0 functionality. See www.w3.org/TR/xpath and www.w3.org/TR/xslt. If you try to use functions that are not supported, the following will be returned:

Function type	Returns
String	`empty string`
Node-set	`null`
Number	`0.0`
Boolean	`false`

Fully implemented XPath functions

Node-set functions

`id()`

`local-name()`

`namespace-uri()`

`name()`

`position()`

String functions

`string()`

`concat()`

`starts-with()`

`contains()`

`substring-before()`

`substring-after()`

`string-length()`

`normalize-space()`

`translate()`

Boolean functions

`boolean()`

`not()`

`true()`

`false()`

`lang()`

Number functions

`number()`

`floor()`

`ceiling()`

Fully implemented XSLT functions

`function-available()`

`current()`

`generate-id()`

Partly implemented XPath functions

Node-set functions

`last()`
Not fully implemented if you use Node collection, or Message collection outside the Event. In these cases, the function will not return the correct value, unless the current node is the last node.

`count()`
Not fully implemented if you use Node collection, or Message collection outside the Event. In these cases, the function will return 0 or 1 depending on whether the argument resulted in a node set or not.

String functions

`substring()`
Unicode characters that cannot be unambiguously represented, are not handled correctly.

Number functions

`sum()`
Not fully implemented if you use Node collection, or Message collection outside the Event. In these cases, StreamServe only supports node sets that contain one node.

`round()`
The value -0.0 is not handled correctly.

Partly implemented XSLT functions

`format-number()`
StreamServe does not use the second string argument (custom decimal formats).

XMLIN tool GUI reference

Main window

The Main window contains three views:

- Message view
  This is where you navigate in, and configure, the XMLIN structure.
- XMLIN sample view
  This is where you load XMLIN document samples and DTDs.
- Properties view
  Select a field or block in the Message view, and configure the corresponding properties in the Properties view.

Menus and menu commands

File menu

New	Clear the existing XMLIN configuration and start with a new, empty Message view.
Open	Open an existing XMLIN configuration. The XMLIN configuration must have been saved as a `*.xin` file.
Save	Save the XMLIN configuration as data embedded in the corresponding Message file in the Design Center Project.
Save As	Save the XMLIN configuration as a separate `*.xin` file.
Open sample	Open a sample in the XMLIN sample view. You can open as many samples as you need.
Reload Sample	Reload the active sample file.
Close Sample	Close the selected sample or DTD.
Close All Samples	Close all samples and DTDs.
Exit	Exit the XMLIN tool.

Edit menu

Find	Find text in fields. You can search for specific property values.
Find Next	Find the next occurrence of the text.
Find Previous	Find the previous occurrence of the text.

View menu

Patterns	In the sample, highlight all nodes that are picked as patterns in the XMLIN configuration.
Fields	In the sample, highlight all nodes that are picked as fields in the XMLIN configuration.
Expand/ Collapse All Subnodes	In the Message view, expand/collapse all nodes below the selected node.
Hide selected instance	In the sample, hide the selected node instance.
Hide multiple instances	In the sample, hide multiple instances of the selected node.

Insert menu

Namespace	Insert a namespace in the XMLIN configuration.

Tools menu

Selection Tool	Select this option and click a node on the sample – the properties for the selected node are displayed in the Properties view.
Block Tool	Select this option and click a node on the sample – the selected node is added as a block to the XMLIN configuration.
Field Tool	Select this option and click a node on the sample – the selected node is added as a field to the XMLIN configuration.

Pattern Tool	Select this option and click a node on the sample – the selected node is added as a pattern to the XMLIN configuration.
Event Options	Open the Event Options dialog box. See *Event Options dialog box* on page 49.
Options	Open the Options dialog box. See *Options dialog box* on page 51.
Customize	Open the Customize dialog box. See *Customize dialog box* on page 52.
Extract Message	Automatically generate an XMLIN configuration from the active sample.
Validate	Validate the selected sample.
Edit Sample	Select to open and edit the active sample resource.
Generate Instance Document	Generate an XML sample from the active DTD or schema.

Dialog boxes

Event Options dialog box

Message tab

XSLT patterns settings	
Used for: Setting default match criteria for patterns and fields. See also *Default match criteria for fields and patterns* on page 41.	
Ancestor axis level	The number of ancestor nodes to include in a match path.
Use path index	Specifies whether or not to use path indexes.

Data collection level settings	
Used for: Selecting collection mode. See also *How StreamServer reads and processes XML documents* on page 5 and *Selecting collection mode* on page 32.	
Node	Specifies collection mode Node.
Message	Specifies collection mode Message.

General settings	
Include attributes when extracting Message	Specifies whether or not to pick attributes from the sample, and add them as fields to the XMLIN configuration.
Set sample data	Specifies whether or not to pick sample data (text nodes) from the sample. The sample data will be displayed in the Process tool.
Create field variable	Specifies whether or not to create variables for the fields. If you select this option, the XMLIN tool will create a variable for each and every field it picks from the sample.

Custom Field Names tab

Used for: Customizing the field IDs suggested by the XMLIN tool. See also *Field ID expressions* on page 44.

Settings	
Custom field names for elements	Suggested XPath expression for elements.
Custom field names for attributes	Suggested XPath expression for attributes.
Custom field names for texts	Suggested XPath expression for texts.
Custom field names for comments	Suggested XPath expression for comments.
Custom field names for processing instructions	Suggested XPath expression for processing instructions.

Custom Field Paths tab

Used for: Specifying whether or not to include attributes in the match path for fields. See also *Default match criteria for fields and patterns* on page 41.

Settings	
Match element name	Include element names, and not attributes, in the suggested path.

Settings	
Match element name or attribute	Include the element name and the specified attribute in the suggested path. If multiple instances of an element contains this attribute, only first instance, but not its siblings, will have the attribute included in the match path. Elements that do not contain the specified attribute will not be affected.

Options dialog box

Preferences tab

Settings	
Save window layout	Save the window layout when you close and open the XMLIN tool. Applies to the size of views, expanded/ collapsed nodes, etc.
Set focus to new item...	Automatically display the properties for a new item when it is added to the XMLIN configuration.

Samples tab

Settings	
Automatically generate...	Automatically generate, and load, an XML document sample when you load a DTD/XSD. The XML document is stored as a temporary file, and will be removed when you exit the XMLIN tool.
Validate instance document...	Automatically validate all XML samples that you load.
Save samples on exit	Save loaded samples and DTDs/XSDs when you exit the XMLIN tool. Not applicable to automatically generated XML documents.

Settings	
Filter input before opening sample	Select to filter the sample file before opening the sample in the XMLIN tool. If the sample is a *.xsf file (SAP connectivity pack only), an XSF filter is used as filter. The XSF filter GUI is launched when you open the sample file, and in this GUI you must specify the appropriate filter parameters. If the sample is any other type of file, you must use a custom filter executable, e.g. a Pearl script. The External filter is launched when you open the sample file, and in this filter you must specify the external command to use. This command is either the full path to the filter executable, or the name of the filter executable if this executable is included in the Path environment variable.

Customize dialog box

Toolbars tab

Turn toolbars on and off, turn tooltips on and off, and modify the appearance of the toolbars.

Command tab

Display information about the toolbar buttons.

Appearance tab

Customize colors and fonts displayed in samples and DTDs.

XML Builder – Settings dialog box

Used for: Generating a sample from a DTD/XSD.

Settings	
Mode	**Typical** – Generate a typical XML document. This document will contain all possible combinations specified in the DTD/XSD. **Custom** – Generate a custom XML document. See *XML Builder – Add Element Children dialog box* on page 53.
Number of instances	The number of instances of other element types an element type can contain. See *Generating samples from DTDs and XSDs* on page 10.

Settings	
Recursion level	How many instances of itself an element type can contain. See *Generating samples from DTDs and XSDs* on page 10.
Document element	The root element in the generated sample.
XML result file	The location of the temporary file for the sample. If you want to keep the generated sample, you can import it as a resource to the appropriate resource set, and later load it to the XMLIN tool.

XML Builder – Add Element Children dialog box

Used for specifying which elements to include in a customized sample.

XML Builder – Result Tree

Used for displaying a preview of the final XML document sample. You can edit the attribute values.

Message view

Used for creating and configuring the XMLIN Event structure. The Message tree structure shown here will be displayed in the corresponding Process tool.

XMLIN sample view

XML samples, DTDs, and XSDs are displayed in separate windows in this view.

Properties view

Used for viewing and editing properties for patterns, blocks, and fields. Select the object (field etc.) in the Message view, and edit the properties in this view.

Message properties

Used for editing Message properties.

Properties	
ID	Message name. Will be displayed in the Process tool.
Description	Textual description of the pattern.
Comment	Additional comment.
Rule	Rule defining how to handle multiple patterns specified for the Message. See *Triggering an Event using multiple patterns* on page 20.
Use block sort priority	Select to enable sorting of data.

Pattern properties

Used for editing pattern properties. Select the pattern in the Message view, and edit the properties in this view.

Pattern properties	
ID	Pattern name.
Description	Textual description of the pattern.
Comment	Additional comment.
Match	Match criterion for the pattern. See *Match criteria for patterns* on page 19. Long paths and path indexing will affect performance. Use the shortest path possible, and use path indexing only when necessary.
Enabled	Select whether or not to use the pattern. If you disable the pattern, make sure that it is not used in any rules.

Block properties

Used for editing block properties. Select the block in the Message view, and edit the properties in this view.

Block properties	
ID	Block name. In the Process tool, you can select whether to display this label, or the Description.
Description	Textual description of the block. In the Process tool, you can select whether to display this description, or the block name.
Comment	Additional comment.
Block sort priority	Set the sort criterion for this block. See *Sorting* on page 22.
Use block sort priority	Select to enable sorting of data in sub-blocks. See *Sorting* on page 22.
Array type	Select to create an array of field instances within the block. Only fields defined as variables can be used to create arrays. **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.
Enabled	Select whether or not to use the block.

Field properties

Used for editing field properties. Select the field in the Message view and edit the properties in this view.

Field properties	
ID	Field name. In the Process tool, you can select whether to display this label, or the Description.
Description	Textual description of the field. In the Process tool, you can select whether to display this description, or the field name.
Comment	Additional comment.

Field properties	
Sample data	An example of field content.
Match	Match criterion for the field. See *Mapping XML nodes to fields in the XMLIN configuration* on page 24. Long paths and path indexing will affect performance. Use the shortest path possible, and use path indexing only when necessary. Do not use path indexes when you specify match criteria for fields in blocks.
Value	Value to extract from the XML input. See *Mapping XML nodes to fields in the XMLIN configuration* on page 24.
Variable	Name of a variable that refers to the field. See *Field variables* on page 29.
Class	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. **Label** – For fields containing static data. **Dynamic** – For fields containing dynamic data. **Header** – For fields containing static header data.
Alignment	Specify alignment of data in the Process tool.
Input format	See *Numeric and date formats* on page 29.
Keep spaces	Select whether or not to keep leading spaces and trailing spaces defined in the field, when the field is used in the output data.
Enabled	Select whether or not to use the field.
Job ID	Select whether or not to assign an index to the content of the field to make it searchable in a Job ID repository.

Namespace properties

Used for editing namespace properties. Select the namespace in the Message view, and edit the properties in this view.

Properties	
Prefix	The prefix (name) of the namespace. The prefix will be used in the match criteria for fields and patterns. Must be unique within an XMLIN configuration.
URI	The URI that identifies the namespace.