xcentric technology & consulting
Oberwiesenstrasse 5
CH - 8057 Zürich

+41 (0)43 255 01 69
www.xcentric.ch
info@xcentric.ch

JAXFront - XUIEditor Manual

V2.0

http://www.jaxfront.com
info@jaxfront.com

Document description

Author(s)	M.Leber, S.Portmann info@jaxfront.com						
Version	2.0						
Web link	http://www.jaxfront.com/download/_JAXFront-XUIEditor-Manual-V2.pdf						
Classification	☒	not classified	☐	internal	☐	confidential	☐ secret
Processing status	☐	draft/in processing	☐	Ready for acceptance	☒	definitive version	☐

Conventions used in this document

Image	Meaning
	Points to an example-syntax in XML.
	Points to a code example in Java.
	Refers to important information.
	Describes a previously presented example..

TABLE OF CONTENTS

Table of Figures

1 Getting started with XUI–Editor

Welcome to XUI-Editor.

This section introduces XUI–Editor, describes its purpose and registration process, and lists the basic system requirements. In addition, the section describes how to install or uninstall the XUI-Editor software.

1.1 Purpose of XUI–Editor

XUI-Editor enables you to effortlessly produce XUI definitions. In addition, you can easily modify user interfaces produced by JaxFront, to adapt them to your needs.

1.2 Registration of XUI–Editor

To register the XUI-Editor, you need a valid license key, which has been issued at the time of purchase. Please contact us immediately if you did not receive the software activation key or if the key does not function as intended.

1.3 System requirements

The following minimum system requirements are recommended:

Type of requirements	**Description**
Hardware requirements	• Desktop: Pentium II 600 MHz or equivalent product • Notebook: Pentium III 700 MHz equivalent product • At least 256 MB RAM, 512 MB RAM recommended (depending upon the size of the visualized XML schema)
Software requirements	JRE/JDK 1.4.x

1.4 Installing XUI–Editor

To install XUI Editor follow these steps:

1. Go to www.jaxfront.com and download the current version of the install file (`JAXFront-Install-XX.jar`, where XX stands for the current version, e.g. V1.60). Alternatively, insert the installation CD in your computer and navigate to the installation file.

2. Double-click the installation file: `JAXFront-Install-XX.jar`. If the executable jar does not start, switch to your command line and type in: java –jar JAXFront-Install-XX.jar.

3. Follow the Installation Wizard, which leads you through the installation process.

When first starting the XUI-Editor you are prompted to enter and/or load your license key. The key has been issued to you with the license of XUI Editor. See Section 1.2, *Registration of XUI-Editor*.

1.5 Uninstalling XUI–Editor

To uninstall XUI-Editor from your Windows system:

1. Locate the folder on your system, where the XUI-Editor was installed. The file `Uninstaller.jar` is located in the **JAXFront** Uninstaller folder.
2. Double-click the `Uninstaller.jar` file, to run it. The originally installed **JAXFront** folder is deleted. If the executable jar does not start, switch to your command line and type in: java –jar Uninstaller.jar.

2 XUI – language

This section describes the basic concepts related to XUI-language, its logical structure, component-spreading definitions, rules, formula expressions and global variables.

2.1 Introduction to XUI language

To adapt a generically produced user interface to different needs, the produced representation types need to be declared in parameters. For more details see *Chapter Representation types* in the guide *JAXFront Concepts*.

For this purpose, the entire visualization tree of a XML schema has been transferred into a more user friendly format.

This format has the following characteristics:

- Simple to understand
- As expandable as possible, because the parameters for the more exact specification of graphic elements are very extensive

The produced specification is kept as abstract as possible, independent of a certain output format or a programming language. A particular GUI specification language may be sketched for every output format that can be produced. However, the following concept concentrates on the production of Java Swing user interfaces. For this kind of GUI specification, XML is best suited, because it is.

- Readable
- Expandable
- Easily changed with multiple XML editors (such as the JAXFront generic layout rendering engine).

An XML schema was produced for the definition of a GUI specification language, e**X**tensible **U**ser **I**nterface (XUI). This schema describes how to parameterize each representation type.

An XUI description is always written for a certain XML instance (Instance), or for the quantity of all XML instances that belong together (in regards to one of XML schema).

Note: This XML instance can be processed by the rendering engine only if it has a valid reference to an XML schema.

Both the XML schema referred by an XML instance and the schema for the GUI descriptions (XUIs) are written in the XML schema language, standardized by the W3C (World Wide Web Consortium). Thus, they are instances of the official W3C schema specification.

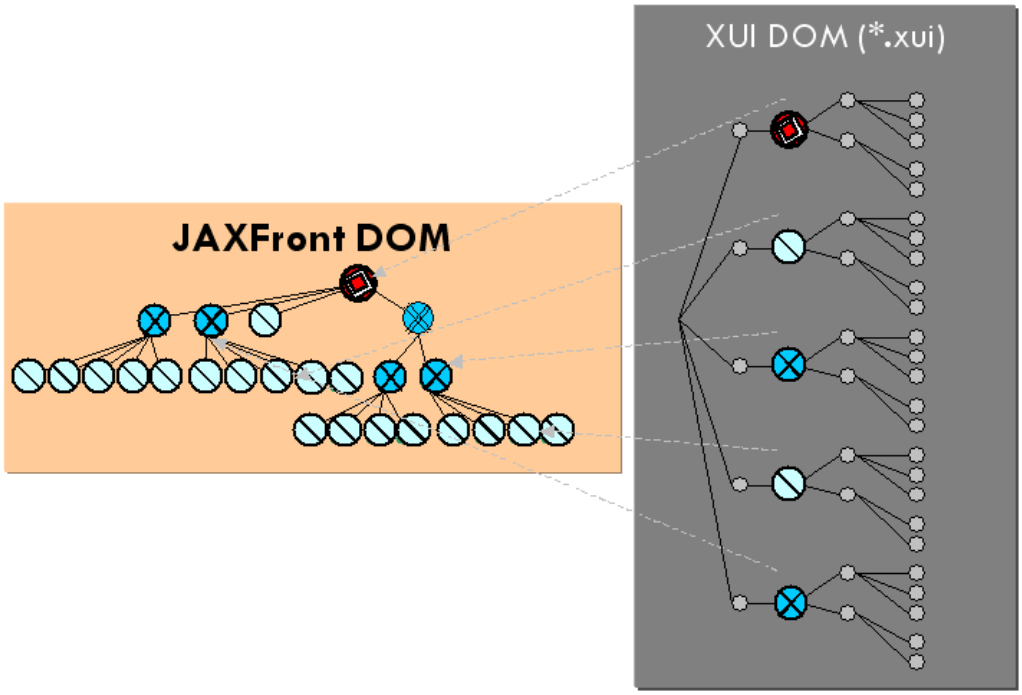


Figure 1: Addressing XML-nodes in XUI

Parts or sub-ranges of the XML instance are addressed with XPath[1] and enriched with layout and rule specific definitions.

The possibilities for this GUI refinement are described in an XML schema. Thus, the XUI pattern represents *all* possible adaptations of the graphical interface.

2.2 Logical structure

The XUI specification consists of a quantity of absolutely addressed visualization components (component), which connects an element from an XML instance with a representation type and the appropriate parameterizing possibilities for this type. In addition, global rules (rule) can be defined.

These rules can be used in different contexts. These rules can be used in different contexts. See the relevant section, further in this document.

[1] Path expressions can be formulated with XPath, which localize parts of an XML document.

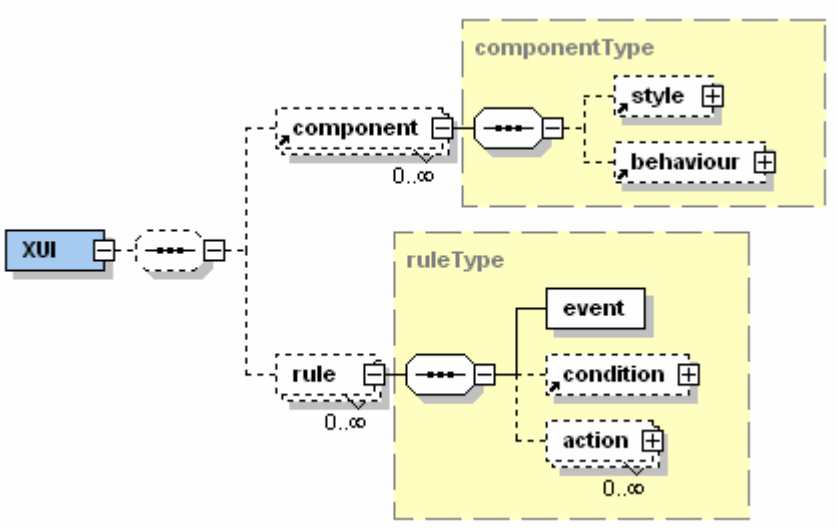


Figure 2: XUI schema - Logical structure

Each XUI component (component) in the XUI document is identified by *clear addressing*. This addressing corresponds to the XPath in the target instance (XML instance). For details see

Addressing by means of XPath.

This described nesting of the XML schema representation types is not repeated in the XUI specification, in order to obtain *loose coupling* (as small as possible) between the structural and the visual representation nesting, expressed in the XML schema. Instead, each representation type is identified only by *absolute addressing (XPath)*.

The XML syntax for an individual XUI component is specified below:

```
<?xml version="1.0" encoding="UTF-8"?>
<?jaxfront version=1.6;xui=none;time=Mon Sep 09 17:32:12 GMT+02:00 2002;?>
<XUI>
    <component xpath="xpath statement">
        <style/>
        <behaviour/>
    </component>
</XUI>
```


Individual nodes or whole quantities of nodes of the XML instance are addressed with an XPath expression (XPath statement). The parameterizing possibilities within the <style> and <behaviour> section vary from one type to another.

All representation-and behavior-relevant XUI parameters are described further in this document.

2.2.1 Addressing by means of XPath

XPath performs the addressing of knots in an XML instance. The following concepts are relevant for this addressing:

- The XML instance
- The JAXFront® -type-tree (JAXFront® DOM).

It is important to know what is addressed with an XPath statement:

- A list –or-
- All the elements.

JAXFront uses its own XPath extension, because the normal XPath syntax does not distinguish between list types (container) and the contents of their element.

- /../**NodeA*[list]***
- /../**NodeB*[text]***
- /../NodeA*[3]*

2.2.2 An example of addressing by means of XPath

In this example, we want to address all items of a *purchaseOrder*.

The XPath expression reads as follows:

/purchaseOrder/item

To address the JAXFront® list type *item*, the XPath expression reads as follows:

/purchaseOrder/item[list]

To address individual *items, apply the* following completely normal XPath expression:

/purchaseOrder/item[3].

The *[text]* predicate identifies the contents of a leaf element, which has at least one attribute defined. This leaf element becomes a simple group for JAXFront® (see *Representation Types* in the *JAXFront Whitepaper*).

However, with the normal XPath expression, you cannot differentiate what is meant: the whole group or only the text contents of this node.

- /../NodeB identifies a SimpleGroup named NodeB
- /../NodeB[text] identifies a SimpleType, the text contents of NodeB
- /../NodeB/@A identifies a SimpleType, the attribute A of the NodeB

2.3 Component-spreading definitions (globals)

Global XUI definitions are global XUI settings that affect the entire user interface and are not bound to a certain target node (XPath addressing).

All the global settings in the element 'global' are optional. An example of two global XUI settings is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>
<?jaxfront version=1.60;time=2005-04-25 13:23:04.703?>
<XUI>
    <global>
        <style>
            <editorPanel>
                <showStatusBar>false</showStatusBar>
            </editorPanel>
        </style>
        <errorDisplay>
            <errorColor>51,51,255</errorColor>
        </errorDisplay>
    </global>
</XUI>
```

In this case the status bar (useStatusBar) is not displayed and the error marking color (errorMarkingColor) is set to a certain color value (RGB).

For more information about global setting possibilities, see the relevant section further in this document. .

2.4 Logical structure of a XUI component

In the logical structure, there is a difference between representation- (*style*) and behavior-relevant (*behaviour*) information units for each XUI component. The settings defined under *style* affect the appearance of the visual component, while the *behavior* specifications define the rules that govern it.

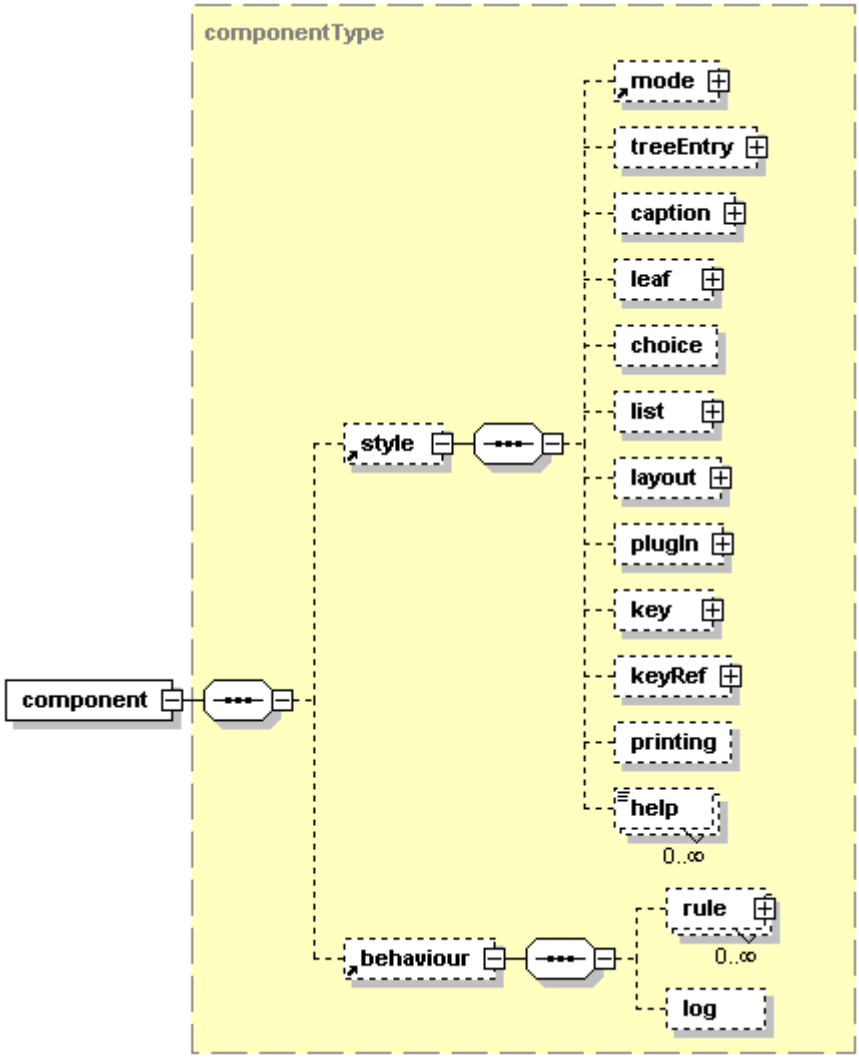


Figure 3: Logical structure of a XUI component

The definition of rules extends the W3C XML schema syntax, which is not capable of defining field spreading validating rules.

The following example of the XUI syntax for a single XUI component (component) illustrates the possibilities of the representation and behavior relevant settings.

The entries specified in this example briefly show that the number of parameterizing possibilities can be extended or adapted at any time.

```
<component xpath="/purchaseOrder/deliveryOption">
    <style>
        <leaf>
            <component>
                <value>
                    <enumeration>
                        <codeSet name="deliveryOption">
                            <url>po.codes</url>
                        </codeSet>
                    </enumeration>
                </value>
            </component>
```

```
            </leaf>
            <help language="de">3</help>
            <help language="fr">31</help>
            <help language="it">32</help>
            <help language="en">33</help>
        </style>
        <behaviour>
            <rule>
                <event type="changeAndInitialization"/>
                <condition useInverse="true">
                    <formulaExpression>nodeValue(&quot;.&quot;) == &quot;A&quot;</formulaExpression>
                </condition>
                <action>
                    <mainAction target="../email">
                        <uiAction methodName="setVisible">
                            <param name="param (0)" type="boolean" value="false"/>
                        </uiAction>
                    </mainAction>
                </action>
            </rule>
        </behaviour>
    </component>
```

For the leaf element (leaf), for example, „deliveryOption“, an option (enumeration) is offered by the indication of a code set ‘po.codes’. If the value changes and equals ‘A’, the field ‘email’ will not be visible. If not, the field ‘email’ is visible (this is the inverse action).

For the binding of a Help system, an anchor index (3,31,32,33) is set additionally to support German, French, Italian and English (de, fr, it, en).

2.4.1 Style definitions

Style definitions specify the visual appearance of a GUI component. The type addressed in the XUI component (via XPath) is crucial for the representation on the Graphical User Interface. Therefore, set options are relevant for a certain type only, for example, leaf value, selection, list or Key-KeyRef relations.

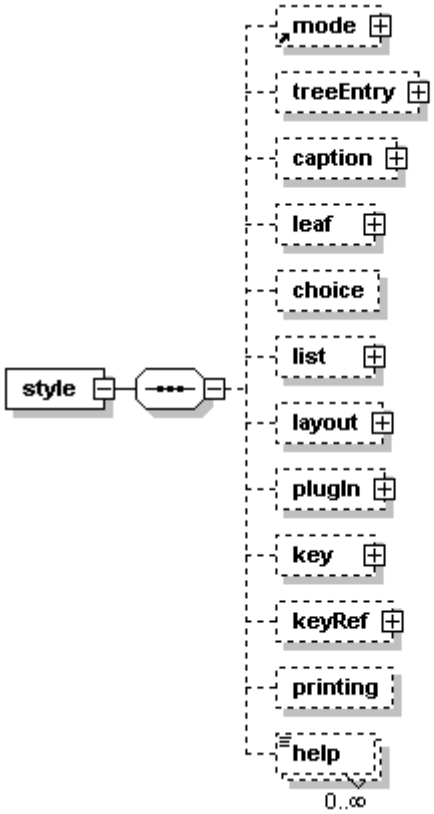


Figure 4: XUI schema – Style definition

The following tables briefly describe the sub-elements of a style definition. See the section *XUI-Editor* (3.4.2), for the detailed content descriptions as well as explanations of the parameterizing possibilities.

This table shows all style characteristics and their descriptions.

Style characteristic	Description
mode	Allows diverse settings, such as visibility, editing mode, and so on…
treeEntry	Defines whether a node is drawn in the navigation tree, as well as the exact representation type in the tree.
caption	Allows diverse settings for the appearance of a heading (Border, Header, and so on…)
leaf[1]	Defines the setting of leaf components and their values, and editing possibilities (default values, options, and so on…)
choice[2]	Defines the appearance of a selection type and its selection elements.
list[3]	Defines settings for nodes with cardinality bigger than 1 (lists).
layout	Defines the layout, in particular the arrangement of sub elements, for a visual component.
plugIn	Defines a plug-in through the specification of a JavaBean (Java class), which is to be displayed instead of the standard visualization component.
key[4]	Defines key specifications for a key-keyref relationship.
keyref[5]	Defines reference indications for a key-keyref relationship.
printing	Defines settings for the PDF printout of this component
help	Defines anchor points for the binding to an external help system for languages to be supported.

[1] This information is only relevant for the addressing of a leaf element (JAXFront: SimpleType).

[2] This information is only relevant for the addressing of a complex selection type (JAXFront: ComlexGroup/SimpleGroup).

[3] This information is only relevant for the addressing of a type with the cardinality bigger than one (maxOccurs facet in the XSD) (JAXFront: ListType).

[4] This information is only relevant for the addressing of a simple or complex type with a XML-schema-key-definition (JAXFront: Identity Constraint/Key).

[5] This information is only relevant for the addressing of a simple type with a XML-schema-KeyRef-definition (JAXFront: Identity Constraint/KeyRef).

The following table illustrates the dependence between style characteristic and the underlying JAXFront types[1] . See also *JAXFront Whitepaper*.

Name	ST	STL	SG	SG*	SGL	CG	CG*	CGL
mode	x	x	x	x	x	x	x	x
treeEntry	x	x	x	x	x	x	x	x
caption	x	x	x	x	x	x	x	x
leaf	x							
choice				x			x	
list		x			x			x
layout			x			x		
plugIn	x	x	x	x	x	x	x	x
key	x							
keyRef	x							
printing	x	x	x	x	x	x	x	x
help	x	x	x	x	x	x	x	x

[1] ST = SimpleType, STL = SimpleTypeList, SG = SimpleGroup,SG* = SimpleGroup as Choice, SGL = SimpleGroupList, CG = ComplexGroup, CG* = ComplexGroup as Choice, CGL = ComplexGroupList

2.4.2 Behavior definitions

These definitions specify the behavior of a GUI component as well as the logging of changes made to the XML instance. In this way, you can specify field-spreading dependencies or even more complex business rules.

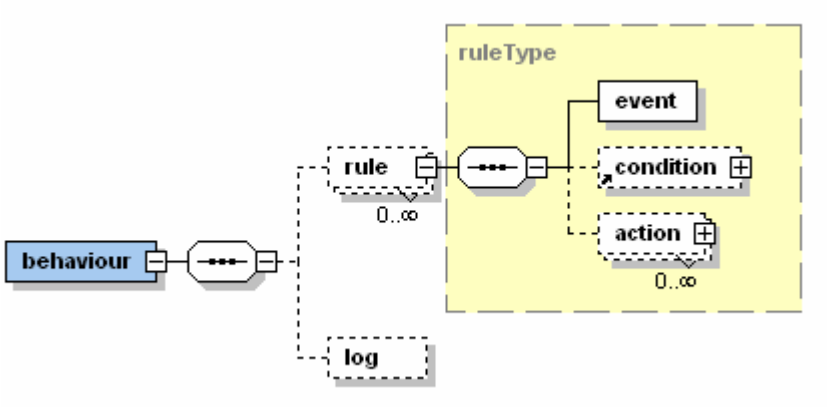


Figure 5: XUI schema – Behavior-Definition

The following table briefly describes the sub elements of a behavior definition. See the *XUI-Editor* section (3.4.3) for the detailed contents descriptions as well as explanations of the parameterizing possibilities.

Element	Description
rule	Defines situation-referred behavior rules for the examination of field-spreading dependencies.
log	Logs the changes made to the underlying XML instance that can be worked on.

2.5 Rules

The „rule“ entries serve for the realization of a situation-referred behavior. Often, business rules are also designated as situation action rules. [1]

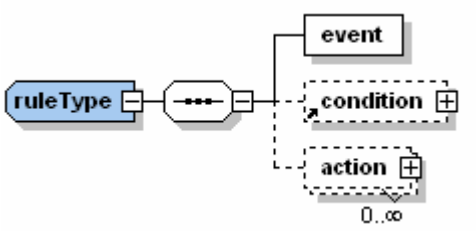


Figure 6: XUI schema – Rule definition

A rule consists of three components:

Component	Description
Event	Specifies when the rule is released.
Condition	Defines which conditions must be fulfilled.
Action	Describes the reaction.

Therefore, situations are described by events and conditions. Action components of the rules specify the reactions to be executed, if the events occurred and all conditions were met.

During the release of such an event message all „condition“-entries of this rule are examined. The return value of a condition is a Boolean one (*true* or *false*). If all the conditions are met, all the actions are released, sequentially.

The following example defines a rule for the leaf element „deliveryOption“:

```
<component xpath="/purchaseOrder/deliveryOption">
    <behaviour>
        <rule>
            <event type="changeAndInitialization"/>
            <condition useInverse="true">
                <formulaExpression>nodeValue(&amp;quot;.&amp;quot;) == &amp;quot;A&amp;quot;</formulaExpression>
            </condition>
            <action>
                <mainAction target="../email">
                    <uiAction methodName="setVisible">
                        <param name="param (0)" type="boolean" value="false"/>
                    </uiAction>
                </mainAction>
            </action>
        </rule>
    </behaviour>
</component>
```

During a change of value of the „deliveryOption“-node the field „email“ is displayed, if the node value corresponds to the letter A (mainAction). If this is not the case, it is not shown but faded out (inverseAction).

[1] For a formal description, we were inspired by the Event-Condition-Action (ECA)-rules, originating from the database theory

2.5.1 Event

An event consists of

- An event type. See *Event types*.
- Execution mode (client or server).
- The operating range belonging to it. See *Operating range*.
- The setting which specifies whether this rule is to be executed if its source is no longer serializable (storable).

2.5.1.1 Event types

Name	Description
1. change	Some change occurred (value or structural change).
2. initialization	Component was just produced (initialized).
3. changeAndInitialization	Combination of 1 + 2.
4. propertyChange	A value change of a leaf component occurred.
5. structureChange	A structural change (choice selection or list manipulation). Combination of 6 + 7.
6. structureChange (add)	New element was added to the list.
7. structureChange (remove)	An element was removed form a list.
8. save	The document (XML-instance) was just stored.

2.5.1.2 Execution mode

This defines where the rule should be executed. Either client or server. By default the rule will be executed in any case (client&server), never mind if JAXFront is running as a server or not.

One can use JAXFront on server side to validate incoming XML streams. Therefore only server relevant rules can be processed by setting the JAXFront engine as 'server instance' (call the method: `JAXFrontProperties.getInstance().setIsRunningAsServer(true)`) .

2.5.1.3 Operating range

Name	Description
1. sourceOnly	Only the current node is affected.
2. sourceAndRefNodes	The current and all referred nodes within the condition expression.
3. directDescendants	All direct descendants of the current node.
4. allDescendants	All descendants of the current node.
5. sourceAndDirectDescendants	The current node and all direct descendants of this node.
6. sourceAndAllDescendants	The current node and all descendants of this node.
7. all	All nodes.

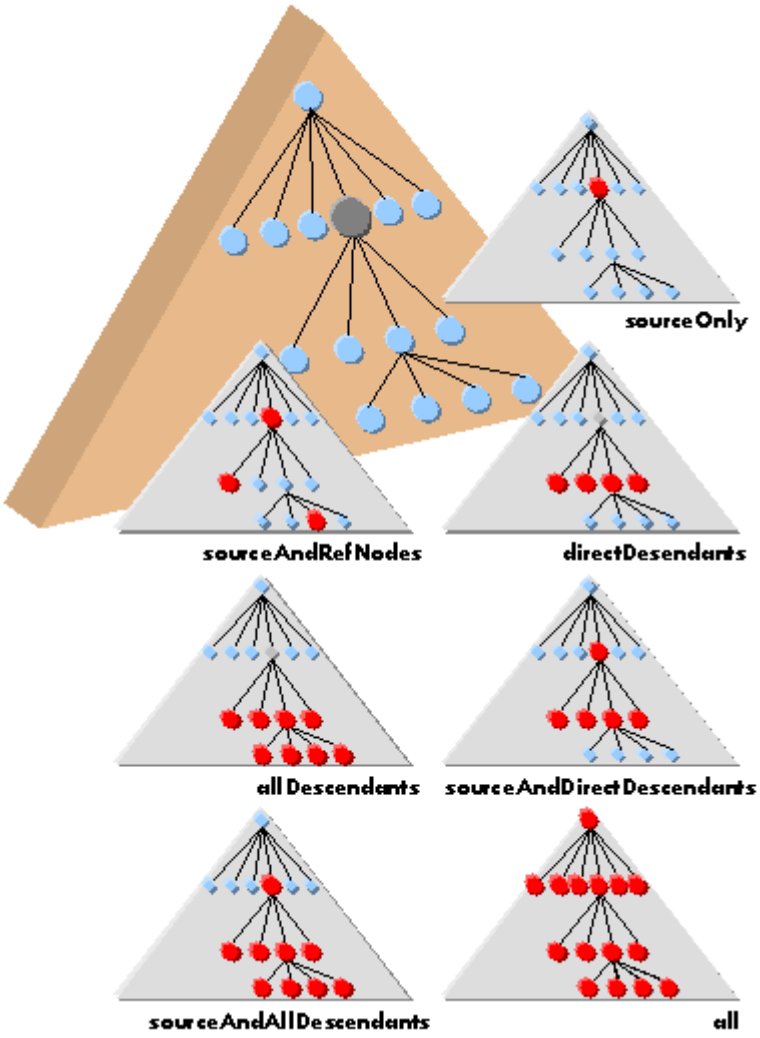


Figure 7: Event – Types of operating range

2.5.1.4 Restriction of the rule processing

A rule is processed if the occurrence of a change corresponds to:

- The defined type of event –and-
- The defined sphere of activity.

However, by the explicit *fade in* or *fade out* of GUI components, you can set single nodes or a group of nodes to *not storable*, "not serializable".

For example, if a whole group of single nodes is faded out, the tag names of the group and its sub-elements no longer need to be visible in the target instance (serialized XML data stream). Therefore, the whole group is removed from the target instance.

The rule can become irrelevant if rules are defined on nodes which are no longer visible at run-time (setVisible). To express this, set the event settings *doesConcernNoneSerializableSource* to „false".

2.5.2 Condition

A condition definition is a formula expression that returns a Boolean value. In addition, the condition definition may define a message, which is to be indicated if the defined condition occurs.

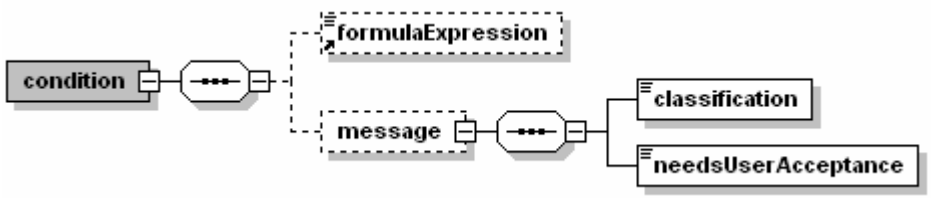


Figure 8: XUI schema – Condition

Name	Description
formulaExpression	Defines the condition as formula expression.
message	Defines a message, indicated if the examined formula expression is positive (that is, if the defined condition occurs).

To learn the exact syntax and the function range of a formula expression *see Formula expressions*.

2.5.3 *Action*

An *action* defines an act implemented when an event occurred and a relevant condition was examined. This action is meant for a firmly determined target, which concerns a single node or multiple nodes. To indicate the target, you need to formulate the XPath expression.

What needs to be defined for each action?

For each *action*, define:

- Main action (mainAction)
- Reverse action (inverseAction).

Optionally, you may define the reverse action. It is implemented only if the formulated condition has not been met and the condition permits that the reverse action may be implemented (useInverse = true).

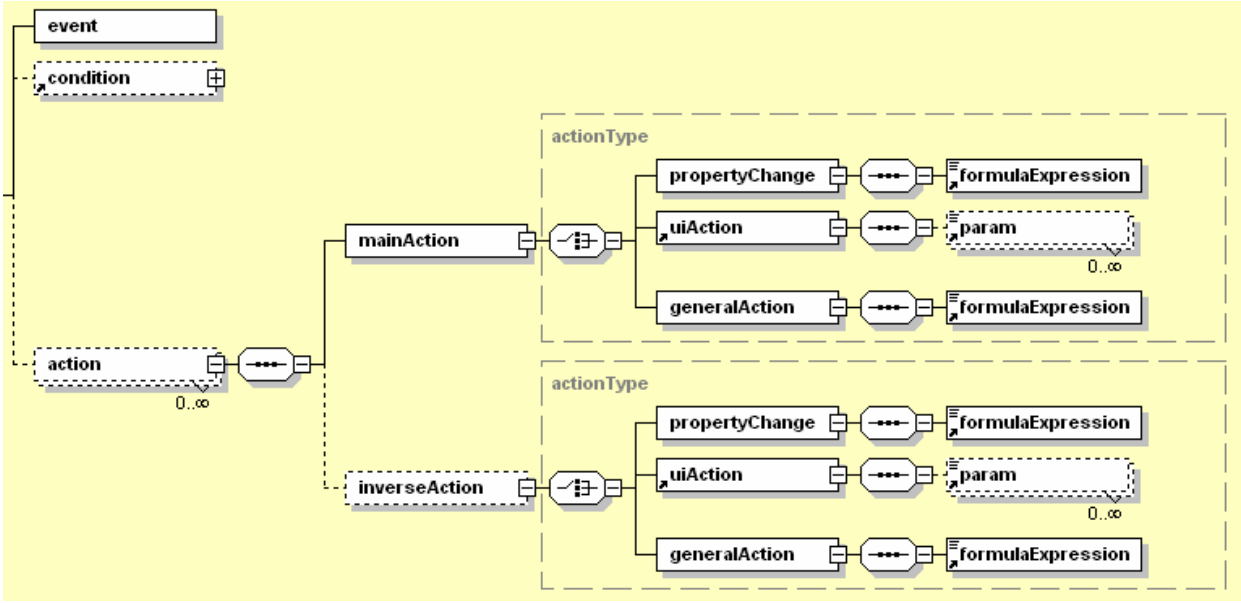


Figure 9: XUI schema - Action

Action	Description
mainAction	Main action, executed with the positive examination of the condition.
inverseAction	Reverse action (opposite), implemented with the negative examination of the condition..

Two types of actions

There are three different types of actions:

Type of action	Description
Change of model (propertyChange)	Sets the value of a data element anew. With the indication of a target in XPath statements, refers to a data element of a simple type. Thus, the new value is set. The model changes can be made on leaf nodes only. If the XPath addresses a complex node, the action is ignored.
Visual event (uiAction)	Triggers visual actions. That is, it displays (fades in) or hides (fades out) components, color changes and so on. A target is indicated by an XPath statement. The statement refers to a data element and consequently its representation types. Then, the method (methodName) is executed. The definition of the method name is compliant with the *Java Beans Definition*
General Action (generalAction)	Triggers any kind of action.

The following example shows an action with a change of model:

```
<propertyChange>
   <formulaExpression>"my change of model"</formulaExpression>
</propertyChange>
```

To set new model values, use a formula expression. With this expression, you can use complex XPath statements or extended functions. In the above example, the new value "my change of model" is set on the target node.

The following example shows an action with a visual event:

```
<uiAction methodName="setVisible">
  <param name="visible" type="boolean" value="true"/>
</uiAction>
```

With an UI-action the method name and the parameters, which will be handed over, are required. In the above example the target element is hidden.

2.5.3.1 Supported parameter types

A parameter consists of a name, a type and a value. The following parameter types specify delivery parameters during the execution of actions (uiActions):

Typ	Example
String	„anyString“
String[]	„9,9,9,9“ (by comma separated enumeration)
Integer	„999“
Int	„999“
int[]	“9,9,9” (by comma separated enumeration)
URL	“file:///c://test/test.xml” or “http://www.jaxfront.com/test.xml”
boolean	„true“ or „false“
Boolean	„true“ or „false“
Color	„50,60,70“ (by comma separated enumeration of the RGB values)
Visualizer	„/purchaseOrder/billTo/name“ (XPath expression)
Type	„/purchaseOrder/billTo/name“ (XPath expression)

If the delivery parameter of a calling method consists only of Boolean values, the reverse action must not be defined separately, because the parameters are toggled.

2.6 Formula expressions

This sections provides an overview of formula expressions, and lists the available operators and functions.

2.6.1 Overview

With XUI-Editor you can define complex mathematical expressions as well as simple stringer operations, in order to execute them at run-time.

You can use a set of prefabricated functions, listed in this document, and, if necessary, extend them with your „own“ functions.

2.6.2 Operators

Function name	Use
Power	^
Boolean Not	!
Unary Plus, Unary Minus	+x, -x
Modulus	%
Division	/
Multiplication	*
Addition, Subtraction	+, -
Less or Equal, More or Equal	<=, >=
Less Than, Greater Than	<, >
Not Equal, Equal	!=, ==
Boolean And	&&
Boolean Or	||

2.6.3 Functions

- This section describes the available functions. The functions may be String functions, Node functions, Mathematical functions and Cast functions.

2.6.3.1 Getting started with functions

Function name	Use
Sine	sin()
Cosine	cos()
Tangent	tan()
Arc Sine	asin()
Arc Cosine	acos()
Arc Tangent	atan()
Hyperbolic Sine	sinh()
Hyperbolic Cosine	cosh()
Hyperbolic Tangent	tanh()
Inverse Hyperbolic Sine	asinh()
Inverse Hyperbolic Cosine	acosh()
Inverse Hyperbolic Tangent	atanh()
Natural Logarithm	ln()
Logarithm base 10	log()
Angle	angle()
Absolute Value / Magnitude	abs()
Random number (between 0 and 1)	rand()
Modulus	mod()
Square Root	sqrt()
Sum	sum()

2.6.3.2 String functions

The following string functions have been provided:

- Concat
- Contains
- EndsWith
- Format
- LowerCase
- StringLength
- SubStringBefore
- SubStringAfter
- Starts With
- Substring

- Translate
- Trim
- UpperCase

Concat

Concat puts n character strings into one string.

Syntax	Return value
concat(*String 1 ,.. String [n]*)	String
Example	
Expression:	Result:
concat(„A“,“B“)	„AB“
concat(nodeValues("//item/USPrice"))	0.039.98148.95
concat(nodeValue("./name")," at home")	“xcentric technology & consulting at home”

Contains

The Contains function examines whether the argument *String1* exists in *String2*.

Syntax	Return value
contains(*String1*, *String2*)	Double (0.0/1.0)
Example	
Expression:	Result:
contains("XML","X")	1.0

EndsWith

EndsWith examines whether the character string *String1* ends with the argument *String2*.

Syntax	Return value
endsWith(*String1*, *String2*)	Double (0.0/1.0)
Example	
Expression:	Result:
endsWith('XML','X')	0.0
endsWith('XML','L')	1.0

Format

Format formats a string in accordance with a pattern definition.

Syntax	Return value
format(*String1*, *pattern*)	String
Example	
Expression:	Result:
format(1.025,“990,00“)	“1.03“

Additional explanations to the format () function

With the format() instruction you can format a value on the basis of a text pattern. The following symbols within the pattern are allowed:

Symbol	Meaning
0	Number
#	Number, zero is not displayed
.	Decimal separator
-	Minus sign
,	Group separator
E	Separates mantissa from the exponent in the scientific representation. Prefix or suffix must be located in quotation marks.

(See also: http://java.sun.com/j2se/1.4/docs/api/java/text/DecimalFormat.html)

LowerCase

LowerCase converts all capital letters of the argument into lowercase.

Syntax	Return value
lowerCase(*String1*)	Double (0.0/1.0)
Example	
Expression:	Result:
lowerCase(„XCENTRIC“)	„xcentric“

StringLength

Returns the number of characters of the argument.

Syntax	Return value
stringLength(*String1*)	Double
Example	
Expression:	Result:
stringLength(„JAXFront“)	8

SubStringBefore

SubStringBefore cuts the string argument at the place *String2* or at the indicated position respectively, and returns the substring before the separation point.

Syntax	Return value
subStringBefore(*String1*, *String2*)	String
subStringBefore(*String1*, *integer*)	
Example	
Expression:	Result:
subStringBefore(„12/10“,“/“)	„JAX“
subStringBefore("Xcentric",3)	“Xc”

SubStringAfter

SubStringAfter cuts the string argument at the place *String2* or at the indicated position respectively, and returns the substring after the separation point.

Syntax	Return value
subStringAfter(*String1*, *String2*)	String
subStringAfter (*String1*, *integer*)	
Example	

Expression:	Result:
subStringAfter(„JAXFront“,“F“)	„JAX“
subStringAfter("Xcentric",3)	“ntric”

StartsWith

Examines whether the *String1* begins with the argument *String2*.

Syntax	Return value
startsWith(*String1*, *String2*)	Double (0.0/1.0)
Example	
Expression:	Result:
startsWith('XML','X')	1.0

Substring

Substring cuts the argument String1 beginning at the position *0* or at the position *start on* respectively, up to the end.

Syntax	Return value
substring (*String1*, *int end*)	String
substring (*String1*, *int start, int end*)	
Example	
Expression:	Result:
substring('Beatles',1,4)	'Beat'
substring('Christian',4)	'Chri'

Translate

Translate replaces all elements within *String1*, which have the value *String2* with the value *String3* and returns this one as string.

Syntax	Return value
translate(*String1*, *String2*,*String3*)	String
Example	
Expression:	Result:
Translate(“R2D2“,“2“,“3“)	„R3D3“

Trim

Trim deletes all leading and trailing blanks.

Syntax	Return value
trim(*String1*)	String
Example	
Expression:	Result:
trim(“ xcentric “)	“xcentric”

UpperCase

UpperCase converts all small letters of the argument *String1* into capital letters.

Syntax	Return value
upperCase(*String1*)	String
Example	
Expression:	Result:
upperCase(“xcentric“)	„XCENTRIC“

2.6.3.3 Node functions

The following node functions have been provided:

- ChoosenNode
- CodesetValue
- CodesetValues
- FullCaption
- KeyXPath
- Node
- NodeLabel
- NodeName
- Nodes
- NodeValue
- NodeValues
- Position
- ShortCaption
- TreeHierarchyContext
- TreeIconName

ChoosenNode

Returns the selected type below a choice element.

Syntax	Return value
choosenNode(*XPath*)	Type
choosenNode(*Type*)	
Example	
Expression:	Result:
choosenNode("/purchaseOrder/paymentMethod")	
choosenNode(node("/purchaseOrder/paymentMethod"))	

CodesetValue

Returns the codeSetValue of the referred type.

Syntax	Return value
codeSetValue(*XPath*)	String
codeSetValue(filePath,codesetName,codeId)	

Example	
Expression:	Result:
codeSetValue("/purchaseOrder/deliveryOption")	
codeSetValue("c:\\jaxfront_swing\\examples\\po.codes","products","l")	

CodesetValues

For each type, addressed by an XPath expression returns the appropriate code set value.

Syntax	Return value
codeSetValues(*XPath*)	Vector
Example	
Expression:	Result:
codeSetValues("/purchaseOrder/deliveryOption")	

FullCaption

Returns the path up to the highest father node (without the root node) including the label of the starting type. For father nodes, which are represented in the tree, the tree label is inserted in the path. If not, then their heading text is used. For list nodes the list index is additionally attached to the label. Labels are separated from each other through " - ". Additionally the length can be limited to a fix character length.

Syntax	Return value
fullCaption(*XPath*)	String
fullCaption(Type)	
Example	
Expression:	Result:
fullCaption("/purchaseOrder/items/prize/currency")	ite... - items[1] - prize – currency
fullCaption("/purchaseOrder/items/prize/currency",3)	ite... - ite... - pri... - cur...
fullCaption(node("/purchaseOrder/items/prize/currency"))	ite... - items[1] - prize – currency

KeyXPath

Returns the key XPath of a selected type within a reference definition.

Syntax	Return value
keyXPath(*XPath*)	String
Example	
Expression:	Result:
keyXPath(".")	

Node

The node function returns exactly one type element, to which the TypePath applies. If the reference consists of several elements, then the first element of the list is always returned.

Syntax	Return value
node(*XPath*)	Type
Example	
Expression:	Result:
node(„/purchaseOrder/name“)	

NodeLabel

Returns the label, which was referred over an XPath expression. If necessary the label is translated into the current language. If the addressed type is about a “Complex Type”, then the heading text is returned.

Syntax	Return value
nodeLabel(XPath)	String
nodeLabel(Type)	
Example	
Expression:	Result:
nodeLabel“/purchaseOrder/@date“)	Datum (example for a NLS label in German)
nodeLabel(“/purchaseOrder/items”)	Artikel (example for a NLS caption title in German)
nodeLabel(node("/purchaseOrder/shipTo"))	

NodeName

Returns the name of a type, referred by the XPath.

Syntax	Return value
nodeName (*XPath*)	String
nodeName (*XPath*, *expression*)	
Example	
Expression:	Result:
nodeName("/purchaseOrder/@orderDate")	“orderDate”

Nodes

The Nodes function returns all by a type-path referred values as vector of types. The Nodes function can be used within the count function! Beyond that, you can limit the selection of the referred types over a simple condition. See the following example:

Syntax	Return value
nodes (*typepath*)	Vector
nodes(*typepath*, *expression*)	
Example	
Expression:	Result:
nodes("/purchaseOrder/item/productName")	
nodes("/purchaseOrder/item/productName", "=='C'")	

NodeValue

NodeValue returns the value of an element. The data type defined in the XML schema is considered and returned as this type. If several types are referred, then the value of the first hit is returned. With NodeValue all defined functions and operations in this section can be carried out.

Syntax	Return value
nodeValue(*XPath*)	String
nodeValue(*Type*)	Double
	Float
	Boolean
Example	
Expression:	Result:
node(„/purchaseOrder/name“)	

NodeValues

NodeValues returns all by a type-path referred values as vector. The data types defined in the XML schema are considered and returned. With NodeValues only a few functions and operations can be carried out, that is, Sum, Count, Min, Max, and Average. Beyond that, it is possible to limit the selection of the referred types over a simple condition (See the example).

Syntax	Return value
nodeValues (*XPath*)	Vector
nodeValues(*XPath*, *expression*)	
Example	
Expression:	Result:
nodeValues(“/purchaseOrder/items/name“)	
nodeValues(“/purchaseOrder/items/name”,” =='Test'”)	

Position

Position returns the position of the type within a list. If the type is not a part of a list, then the value 1.0 is returned. As argument only the node function can be used!

Syntax	Return value
position (*Type*)	Integer
Example	
Expression:	Result:
position(node("."))	3

ShortCaption

Returns the path up to the highest father node (without the root node) including the label of the starting type. Only the elements displayed in the tree are included in the path. Labels are separated by " - ".

Syntax	Return value
shortCaption(XPath)	String
shortCaption(Type)	
Example	
Expression:	Result:
shortCaption("/purchaseOrder/shipToAdress/name")	shipToAdress - name

TreeHierarchyContext

Returns the names of a tree entry and the label of the referred node.

Syntax	Return value
treeHierarchyContext(XPath)	String
treeHierarchyContext(Type)	
Example	
Expression:	Result:
treeHierarchyContext(“./anyType“)	
treeHierarchyContext(node("/purchaseOrder"))	purchaseOrder

TreeIconName

Returns the icon name of a tree node, addressed over an XPath expression. Optionally, it can indicate whether the result of the referred node contains an XPath expression. If so, this XPath can be used for the resolution of the „treeIconName“.

Syntax	Return value
treeIconName(XPath)	String
treeIconName(XPath, Boolean)	
Example	
Expression:	Result:
treeIconName("/purchaseOrder/email")	email.gif
treeIconName("/purchaseOrder/items/context/path",true)	productA.gif

2.6.3.4 Mathematical functions

The following mathematical functions have been provided:

- Avg
- Count
- Max
- Min
- Round
- Summation

Avg

Average determines the average value of all arguments and returns this one. The function NodeValues can also be used as argument.

Syntax	Return value
avg(*arg[1],.. arg[n]*)	Double
Example	
Expression:	Result:
avg(1,2,3,4,5)	3

Count

Count returns the number of all arguments (indicating how many arguments there are). The functions NodeValues and Nodes can also be used as arguments.

Syntax	Return value
count(*arg[1],.. arg[n]*)	Integer
Example	
Expression:	Result:
count(1,1,2,2)	4
count(nodes("/purchaseOrder/item/quantity"))	3

Max

Max returns the maximum of all arguments. The function NodeValues can also be used as argument.

Syntax	Return value
max(*arg[1],.. arg[n]*)	Double
Example	
Expression:	Result:
max(1,2,3,4,5)	5
max(nodeValues("/purchaseOrder/item/quantity"))	100

Min

Min returns the minimum of all arguments. The function NodeValues can also be used as argument.

Syntax	Return value
min(*arg[1],.. arg[n]*)	Double
Example	
Expression:	Result:
min(1,2,3,4,5)	1
min(nodeValues("/purchaseOrder/item/quantity"))	0.0

Round

The Round function rounds the argument, after the right-of-comma position Up or Down, indicated in the second argument, are rounded. (Values < 5 are rounded down and Values >= 5 are rounded up!)

Syntax	Return value
round(*arg, decimal places*)	Double
Example	
Expression:	Result:
round(1.2345,3)	1.235

Summation

Summation determines the total value of all arguments, and returns this one. The function NodeValues can also be used as an argument.

Syntax	Return value
sum(*arg[1], .. arg[n]*)	Double

Example	
Expression:	Result:
sum(2,3,4,5)	14.0
sum(nodeValues("/purchaseOrder/item/quantity"))=	102.0

2.6.3.5 Cast functions

The Cast function changes the resulting data type. For example, this is helpful if a function returns a value of the data type Integer and you want to change it with a substring() function

Name	*Syntax*	*Example*
Boolean	boolean(*arg*)	boolean(1) = true
Integer	integer(*arg*)	integer(1.2) =1
Number	number(*arg*)	number("123") = 123.0
String	string(*arg*)	string(0815) = 815.0

A peculiarity of the formula parsers is that Boolean values are not represented as TRUE or FALSE, but as 1.0 or 0.0. Consequently, an explicit Boolean cast has to be executed for expressions which are to return a string result TRUE or FALSE. An example is shown below:

Expression	Result
1>2	0.0
boolean(1>2)	FALSE

2.7 Definition and Use of global variables

Global variables can be stored as follows:

- Per JAXFront®-DOM –or-
- Per JavaVM (for an entire application).

A global variable is a substitute symbol, which can be set respectively and deleted at any time by an application that uses JAXFront®.

A variable is always defined by a key and string value.

Variables, held per DOM, can be set on the JAXFront®-DOM-instance by *com.jaxfront.dom.Document.setUserProperty(String key, String value)*.

To use globally defined variables in the XUI definition:

- End the defined key of the variable with the prefix **$(start** and the suffix).

Example:

For a variable *USER_EMAIL* end the defined key of the variable as follows *$(USER_EMAIL)*.

```
<propertyChange>
  <formulaExpression>"The current email will be inserted here: $(USER_EMAIL)"
</formulaExpression>
</propertyChange>
```

At run-time, the variable USER_EMAIL is replaced with the set value.

If the variables are DOM independent, they can be defined with the same method, *JAXFrontProperties.getInstance().setUserProperty(String key, String value)*.

3 XUI - Editor

This section describes the XUI-Editor.

3.1 Starting the XUI-Editor

To start XUI–Editor, you need to start a javaVM. Be sure your JAVA_HOME directory is set. If so, just double click batch (XUIEditor.bat) file for windows or the shell script (XUIEditor.sh) for unix systems.:

To start the XUI-Editor using the command line interface

1. Click the Start button, then select Run >cmd. See the following figure.

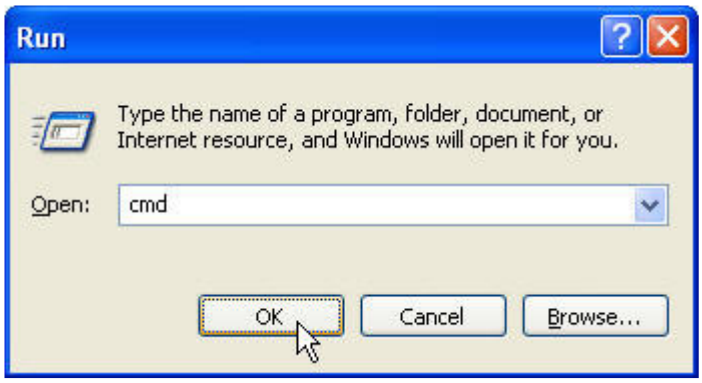


Figure 10: Opening XUI-Editor through the command line

2. Switch to the JAXFront installation directory.
3. Enter ***java –jar .\lib\jaxfront-xuieditor.jar***
4. Press the ENTER key. The XUI-Editor is started

If no other target was indicated at the beginning of the installation, the standard directory reads as „c:\Program Files\XUIEditor“

3.1.1 Language selection

When the XUI-Editor is initially started, you need to select a language you want to use within the XUI–Editor. In this release, the XUI–Editor is offered in English and German.

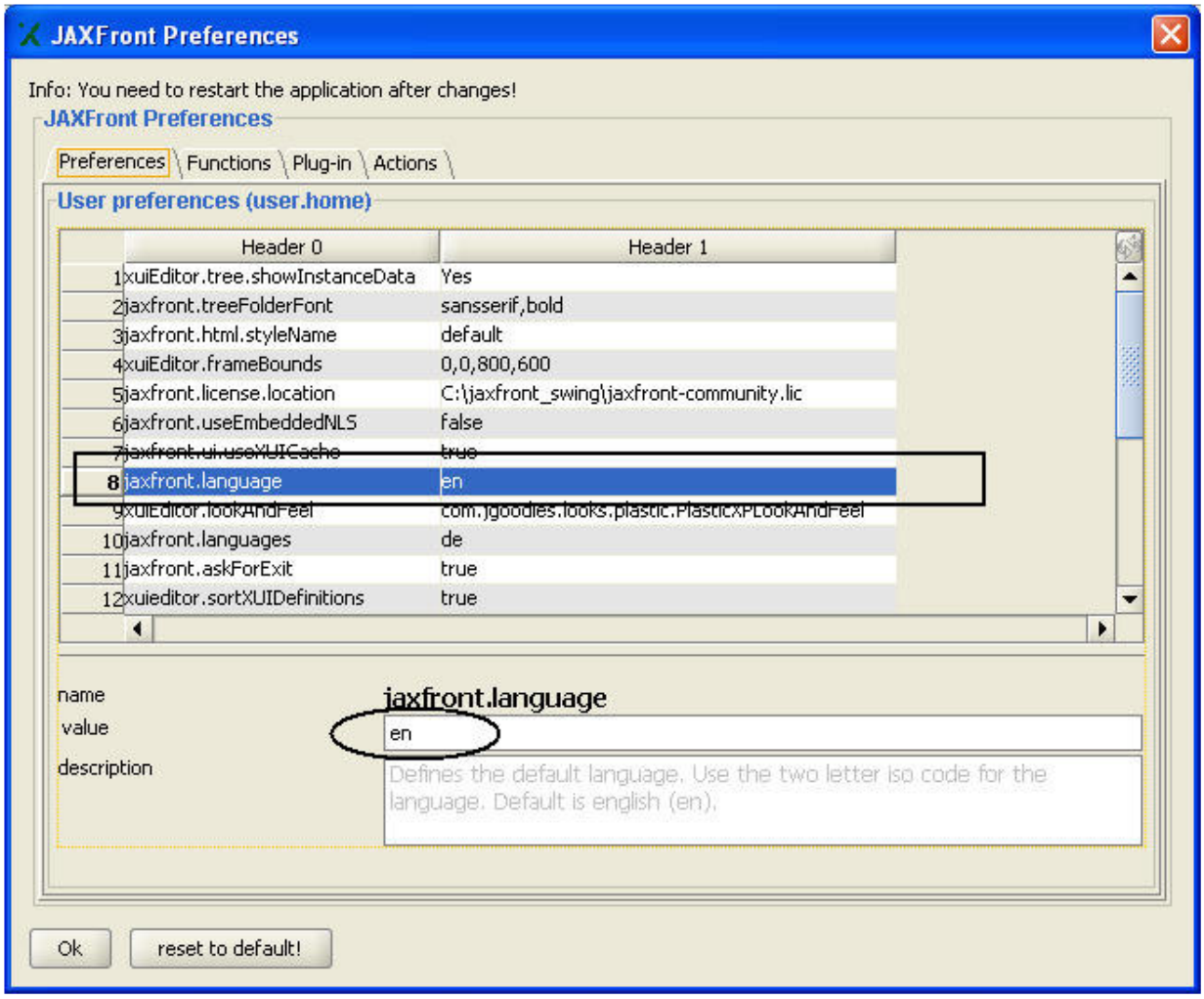


Figure 11: Language selection

3.1.2 How to enter the license key

First time you start the XUI-Editor, you need to enter a valid license file. Note that you cannot start the XUI-Editor without a valid license.

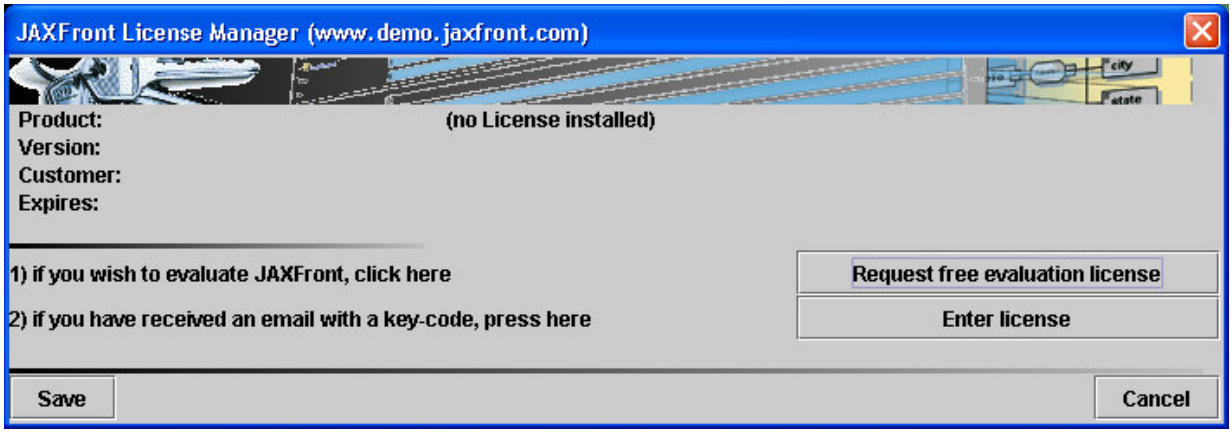


Figure 12: JAXFront License Manager

If you have a JAXFront **demo** version without a valid license file, send an email to info@jaxfront.com to get a free license key. If you have a **valid license**, click the "Enter license" button. The Select dialog box appears. Select the appropriate license file in this dialog box.

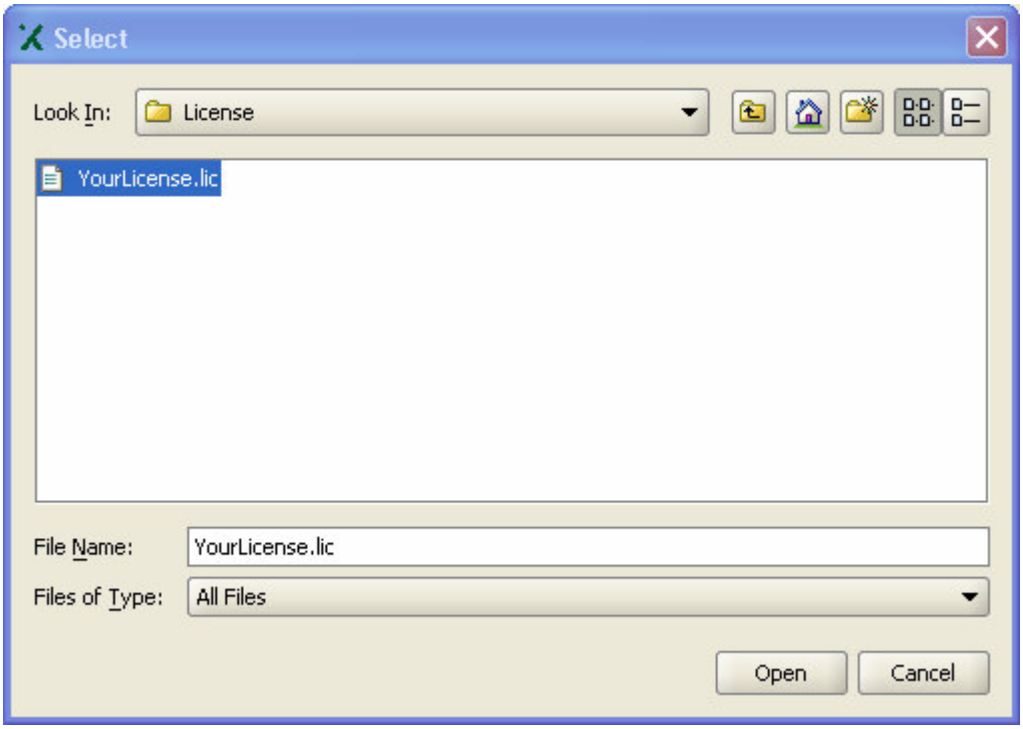


Figure 13: Select a license file

JAXFront will only keep the location of the provided license file. Be sure that you do not move or delete the .lic file. If so, JAXFront will ask you again for the license file location the next time you start the XUI editor.

If the selected file contains a valid license, the „License Agreement“ dialog box appears. To accept the terms and conditions of this License Agreement, click the „Yes“ button. Then, click “Save”. The activation process is completed and the main program launched. If the selected file does not contain a valid license not, an error message is displayed.

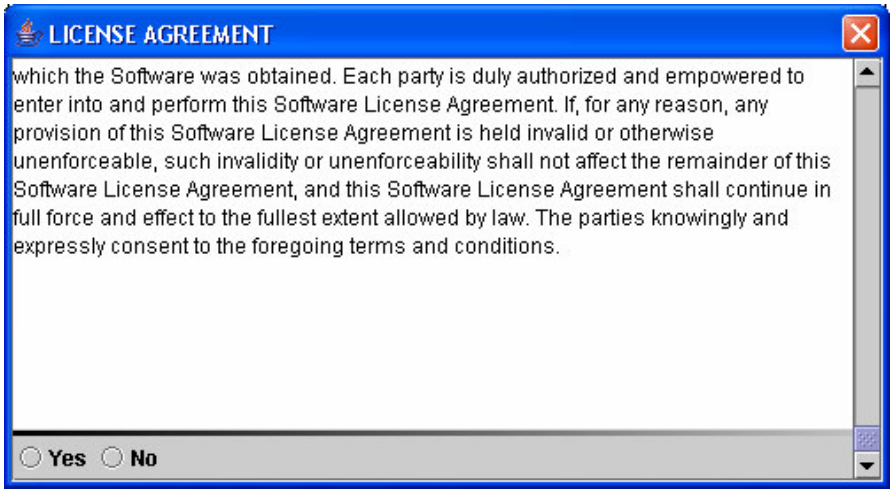


Figure 14: License Agreement terms and conditions

3.1.3 Selection of a XML schema

To work within the XUI-Editor, you must select an XML schema. The appropriate dialog box is displayed. Select the desired XML Schema file and the Root Node (as a starting point). All global elements defined in the XML schema can be used as starting point.

If the pattern cannot be loaded, for example, there is „ no schema conformity“, the selected node is locked.

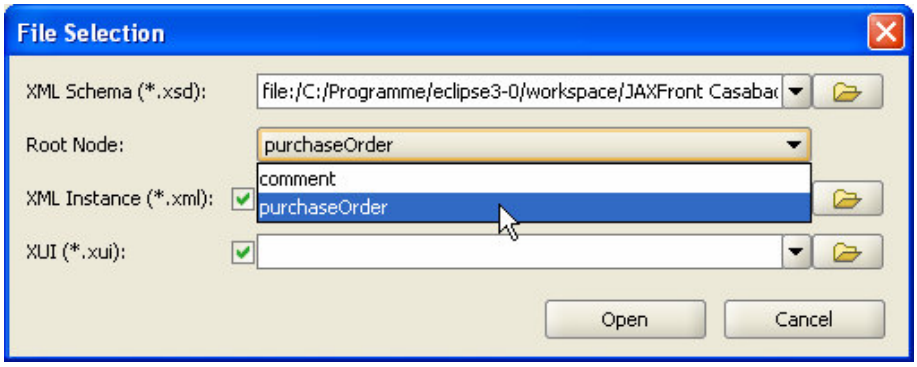


Figure 15: Open a XML schema file

The selection of an XML instance file and/or a XUI file is optional. These files are needed only if you want to adopt and/or modify an already produced XUI definition.

The most recently used files are listed in the Combo-box and may be selected.

Select the files, and then click “OK” to confirm the selection. The XUI-Editor loads the selected file(s) and displays the work area.

3.2 The work area

The XUI-Editor work area is divided into the following five areas:

1. Menu bar
2. Toolbar
3. Navigation area
4. Editor area
5. Status bar

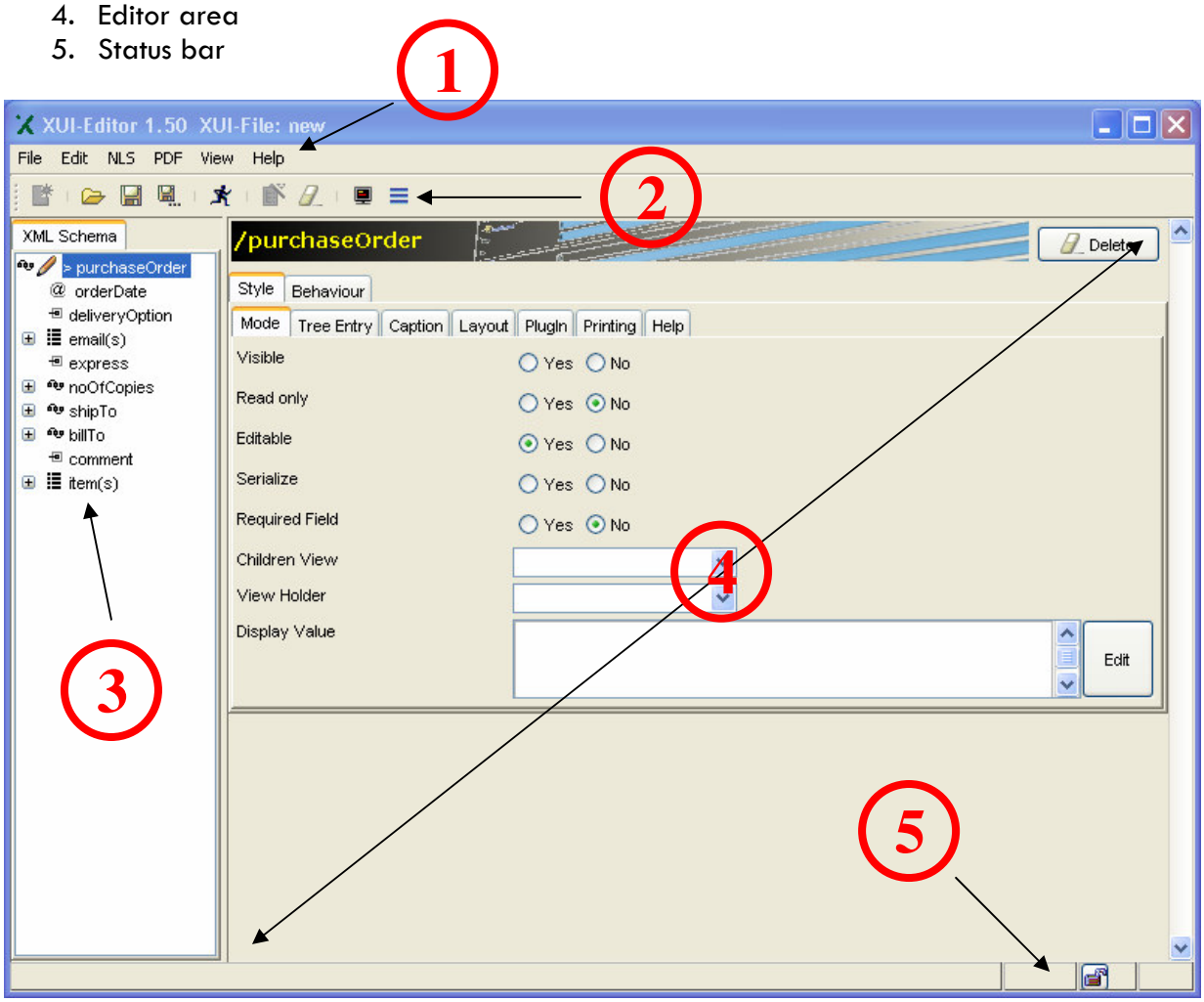


Figure 16: The XUI-Editor - Work area

3.2.1 Menu bar

3.2.1.1 File menu

Name	Shotcut key	Meaning
New	CTRL-N	Produces a new, empty XUI file.
Open	CTRL -O	Opens an existing file.
Reload		Loads the last saved version of the active XUI file.
Save	CTRL -S	Saves the active XUI file under its current name.
Save as		Saves the active XUI file under a different name.
Exit	CTRL -Q	Terminates the XUI–Editor.

3.2.1.2 Edit menu

Name	Shotcut key	Meaning
Cut	CTRL -X	Removes the currently selected text and stores it on the clipboard.
Copy	CTRL -C	Copies the currently selected text.
Paste	CTRL -V	Pastes the text from the clipboard, at the current text position.
Delete	Del	Deletes the marked text.
Create XUI Component		Creates a XUI component for the nodes selected in the tree.
Delete XUI component		Deletes a XUI component for the nodes selected in the tree.

The functions Cut, Copy, Paste and Delete apply to text elements only. To access the additional editing functions, press the right mouse button (right-click).

3.2.1.3 NLS menu

Name	Meaning
Maintain NLS	Opens the NLS admin dialog.
NLS Preview	Produces a NLS GUI preview based on the current settings.
Export NLS	Opens the NSL export dialog.
Import NLS	Opens the NLS import dialog.

3.2.1.4 PDF menu (if available)

Name	Meaning
Designer	Opens the PDF Designer dialog.
Preview	Opens the PDF preview, based on the current definitions.
Save	Save the current PDF settings.
Reload	Reloads the PDF settings from file.
Delete	Delete the current PDF settings.

3.2.1.5 HTML menu (if available)

Name	Meaning
Start Tomcat	Starts the tomcat web server on default port 8080.
Stop Tomcat	Stops the tomcat web server.
Preview HTML	Produces an HTML GUI preview, based on the current XUI settings.

3.2.1.6 View menu

Name	Shotcut key	Meaning
Preview	F12	Produces a GUI preview, based on the current settings.
Preview with file…	F11	Produces a GUI preview, based on the current settings and for any XML document to choose from a file selection.
Show XML syntax		Shows the XUI definition in its XML representation.
Global settings		Opens the screen for the global XUI settings.
Show Global Types		Opens/closes the folder for editing global types, in the navigation area.
Show Rules		Opens/closes folder for editing global rules, in the navigation area the
Debug		Activates a simple debug console to track events.
Preferences	CTRL -P	Opens the XUI–Editor program settings (see the relevant section further in this document).

3.2.1.7 Help menu

Name	Meaning
About	Displays the information about the XUI-Editor, such as the version, runtime environment, and so on.
Check Latest Version	Opens the JAXFront Web page, in order to check for the most recent version.
Online Forum	Opens the JAXFront online forum.
License Manager	Opens the License Manager dialog.
General Help	Opens JAXFront Online Support Help.

3.2.2 Toolbar

Name of the toolbar icon	Toolbar Icon	Meaning
File new		Produces a new, empty XUI file.
File open		Opens an existing file.
Save		Saves the active XUI file under its current name.
Save as…		Saves the active XUI file under a different name.
Preview		Produces a GUI preview, based on the current settings.
Create XUI Component		Creates a XUI component, for the nodes selected in the tree.
Delete XUI component		Deletes the XUI component, for the nodes selected in the tree.
Global settings		Opens the screen for the global XUI settings.
Show XML source		Display the XUI definition in its XML representation.

3.2.3 *Navigation area*

In the navigation area, you can select individual schema elements in order to define or modify their settings. The elements in the navigation area are represented in a tree-like structure.

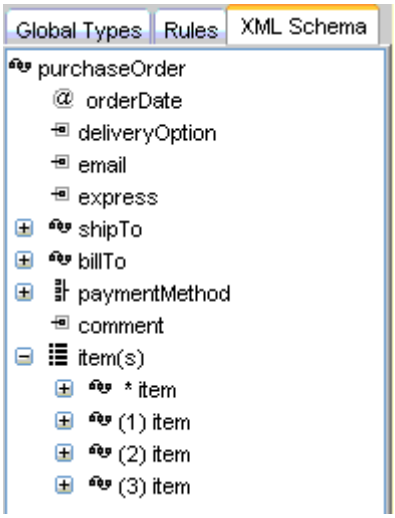


Figure 17: Navigation area

The navigation area has the following three tabs, providing three different structural views:

- XML Schema
- Rule
- Global Types

XML Schema

The XML Schema tab enables you to navigate within the XML schema definition, and to modify its individual elements (see 3.2.4). You can expand or collapse the hierarchical levels of the XML schema. To expand the XML Schema tree click the plus sign (+). To collapse it, click the minus sign (-).

Rule

The Rule tab enables you to navigate within the "global rules"- definitions, and to modify them. Global rules reduce the number of rules definitions, by defining reusable rules and applying them on a global level. Global rules can be reused (referenced) at any position within the XUI.

Global Types

The Global Types tab is similar to the XML Schema tab. However, at the Global Types you can view and/or modify only the "global schema types". See http://www.w3.org/TR/xmlschema-0/#Globals). To display „not global elements", switch to the XML Schema view (by clicking the XML Schema tab).

Existing XUI definitions are marked within the tree, under each tab. For example, modified nodes are marked with a pencil . 

The following table describes the icons (pictograms) used to mark the XUI definitions within a tree:

Pictogram	Meaning
	Local XUI definition.
	Global (global schema type) XUI definition
	XUI definition, which originates from another (included) XUI file.
	Global (global schema type) XUI definition, which originates from another (included) XUI file.
>	Indicates whether a XUI definition exists for a subcomponent.
~	A global XUI component exists, serving as a template (definition supplier) for the local component.

Pictogram	Meaning
	A group, in which all sub-elements are arranged in a simple sequence (sequence).
	A group, in which one of the sub-elements can be chosen (choice).
	Defines a list (cardinality greater than 1) of elements (choice).
	Designates a leaf component (leaf).
	Designates an attribute (attribute).
*	Designates a template component. XUI settings on this level apply to all entries in this list (template).
(1)	Designates a specific component within a list (indexed component).

If the XML schema is rather complex and you want to search for a certain node, click on the search icon in the navigation tree.

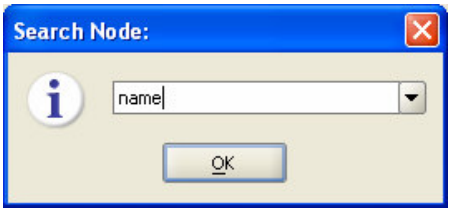


Figure 18: Search for XSD nodes.

Type in any node name and click 'OK'. A new dialog pops up with all the target nodes. The selected node will be displayed in the underlying XML schema tree.

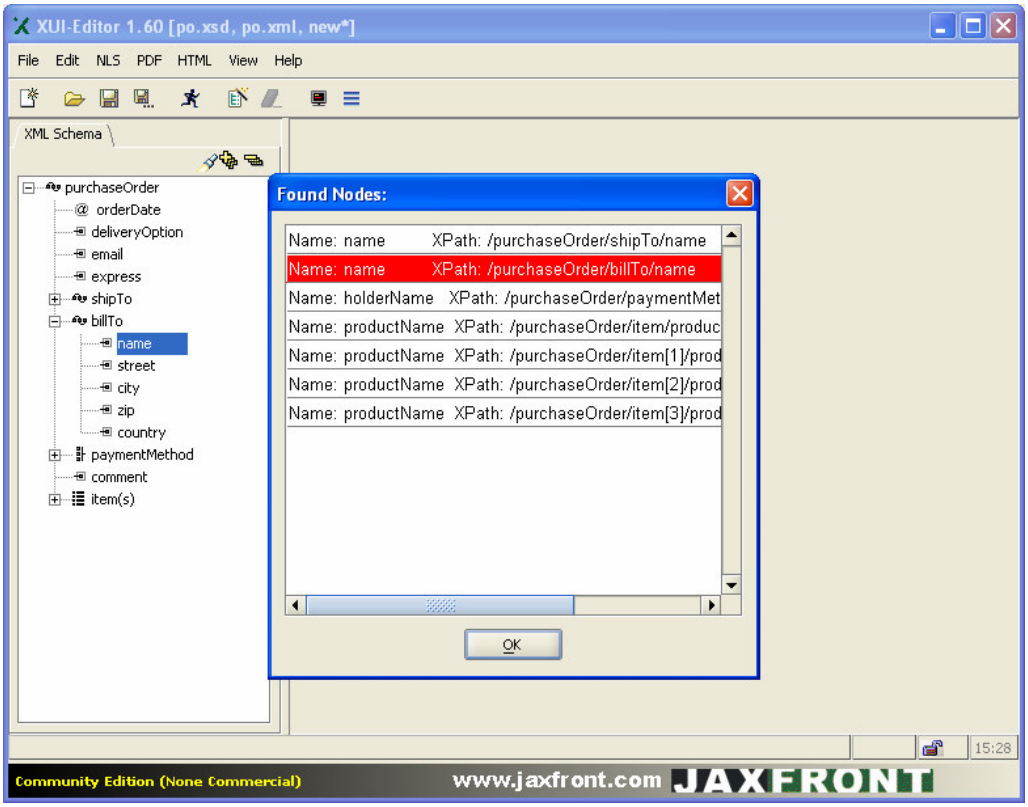


Figure 19: Found nodes dialog.

Clicking on the "+" icon will expand the selected XML schema node. Clicking on the "-" icon will collapse the selected node.

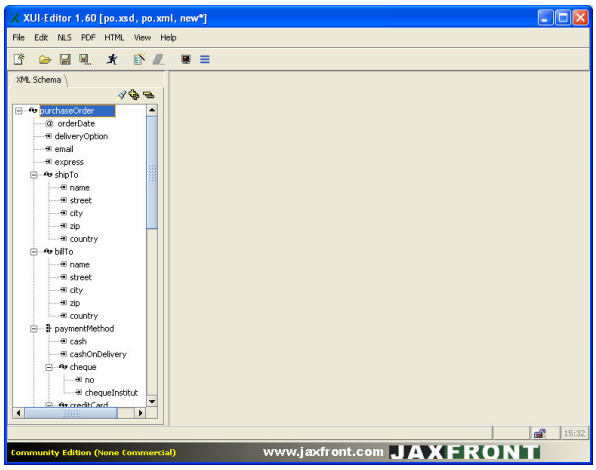


Figure 20: Expand tree node.

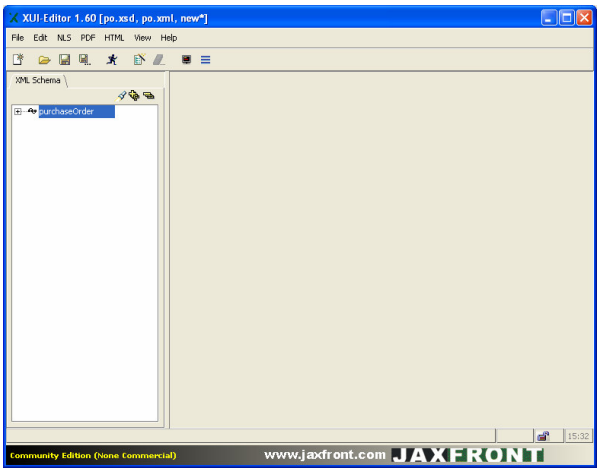


Figure 21: Collapse tree node.

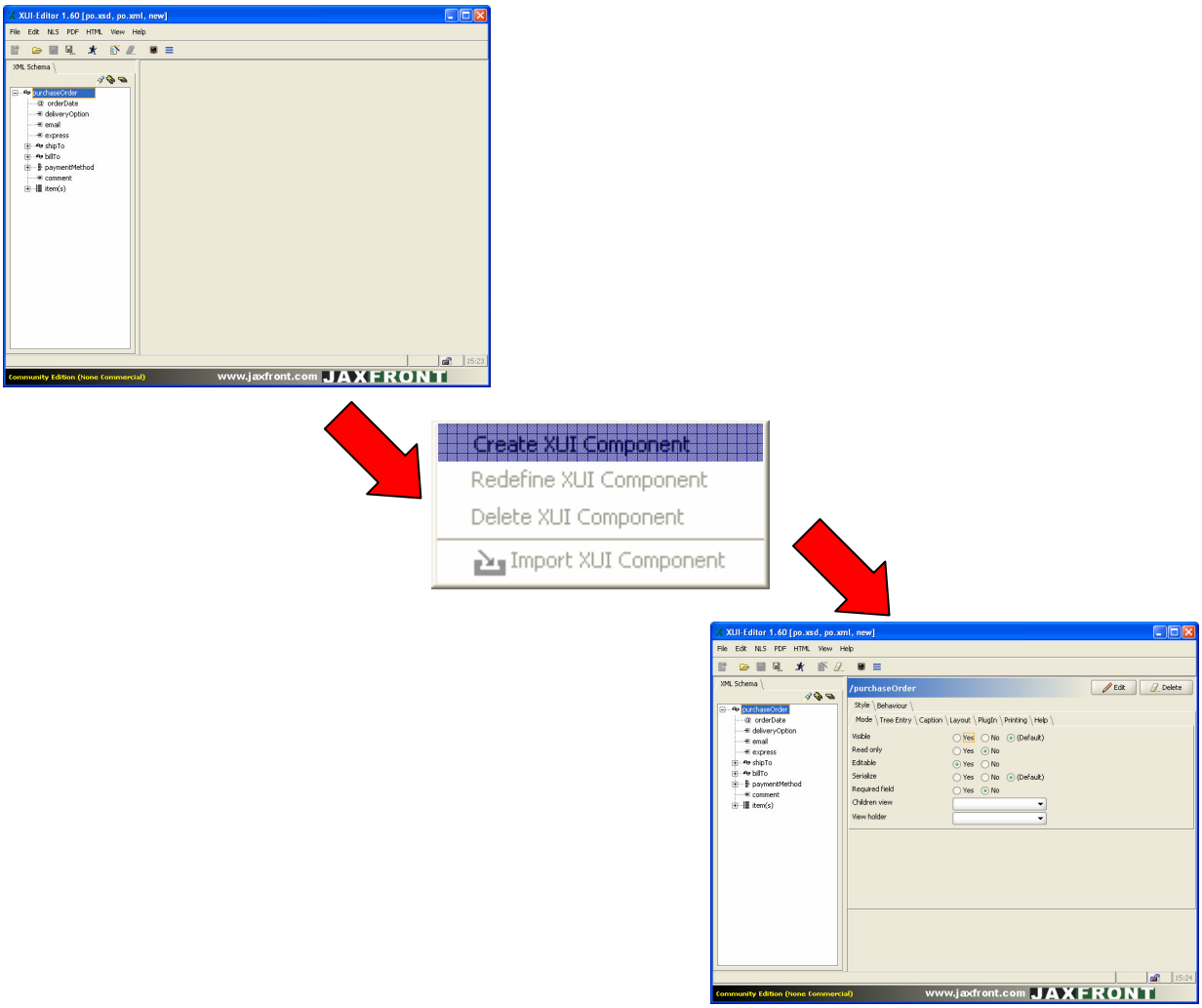
3.2.4 XUI Component Editor

Using the Component Editor, you can adjust the appearance and behavior of the components. If a modified node is selected in the navigation area, it appears within the Component Editor where it can be edited.

3.2.4.1 Create a new XUI component

To create a new XUI component:

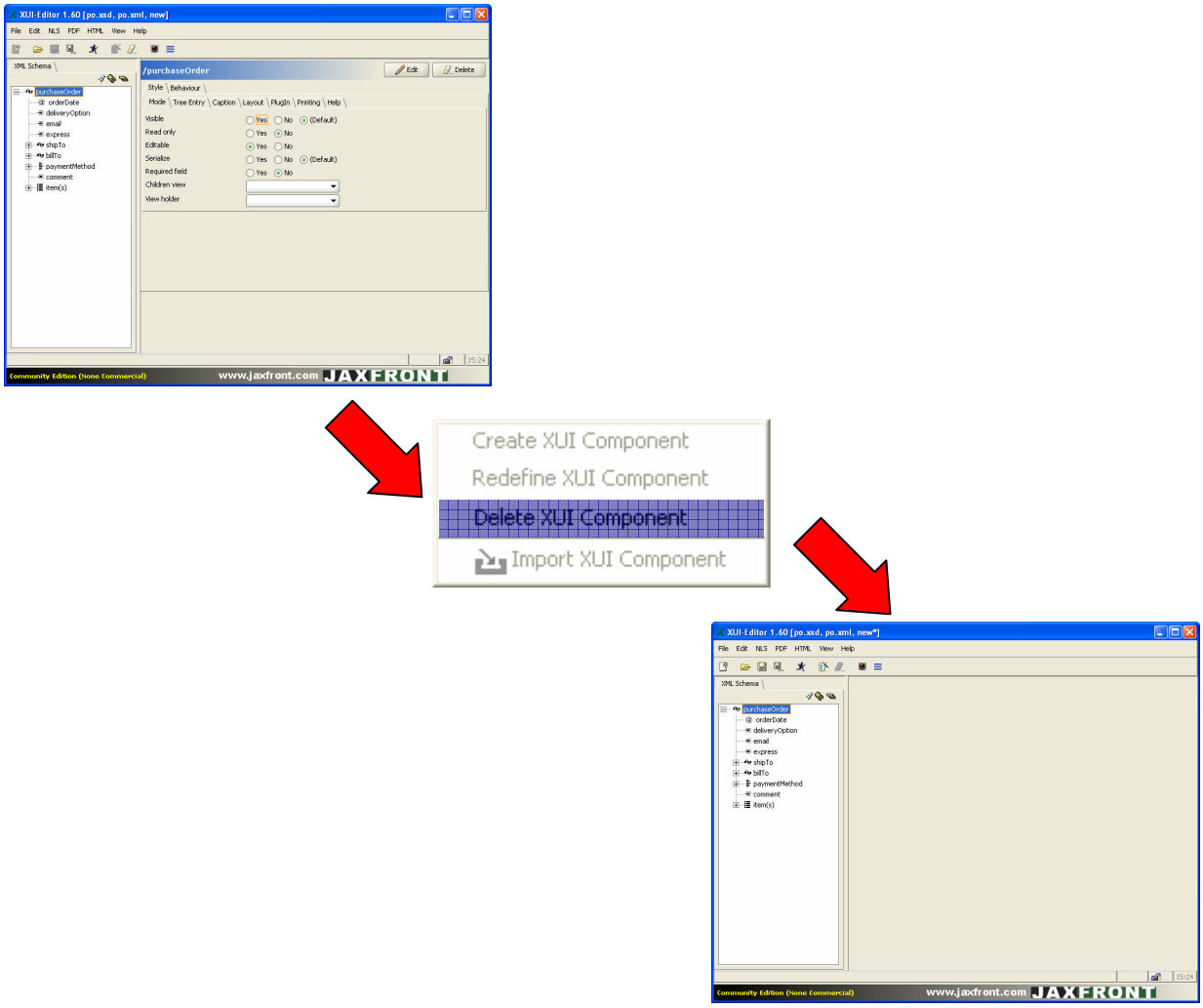
1. Select a reference node under one of the three available navigational views (XML Schema, Rule or Global Types).
2. To create the component, use any of the following four methods:
   - Click the  icon on the toolbar.
   - From the Edit menu, select „Create XUI Component“.
   - Position the mouse cursor on the selected tree element, and press the right mouse button. Then, select „Create XUI Component“, in the right-click menu. See the following figure.



3.2.4.2 Delete an XUI component

To delete an existing XUI component, use any of the following four methods:

- Click the  icon on the toolbar.
- From the Edit menu, select „Delete XUI Component“.
- Position the mouse cursor on the selected tree element, and press the right mouse button. Then, select „Delete XUI Component“, in the right-click menu. See the following figure.



3.2.4.3 Redefine an XUI component

You can redefine XUI components only for the elements of a global type. However, once redefined, all definitions are no longer of a global type, but defined locally on the node itself. Thus, local settings overwrite the global ones.

To redefine an XUI component:

- Position the mouse cursor on the selected tree element, and press the right mouse button. Then, select „Redefine XUI Component".

3.2.4.4 Import XUI component

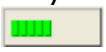
This function is similar to redefining a component (section3.2.4.3). However, the XUI definition has been created outside the currently used XUI file.

To import an XUI component:

- Position the mouse cursor on the selected tree element, and press the right mouse button. Then, select „Import XUI Component".

3.2.5 Status bar

The status is located along the bottom edge of the application screen. The status bar provides the following information:

- **Information row:** Informs you of the current task performed by the system
- **Progress indicator:** Indicates the progress of the task processing 
- **License state:** Informs you when the license will expire:

Ausdruck	Ergebnis
	License will expire in 9 days
	License expired
	Unrestricted license

- **Time:** Tells the current time. 

3.3 Global XUI settings

The global settings apply to the entire Graphic User Interface. They are defined in the following screens:

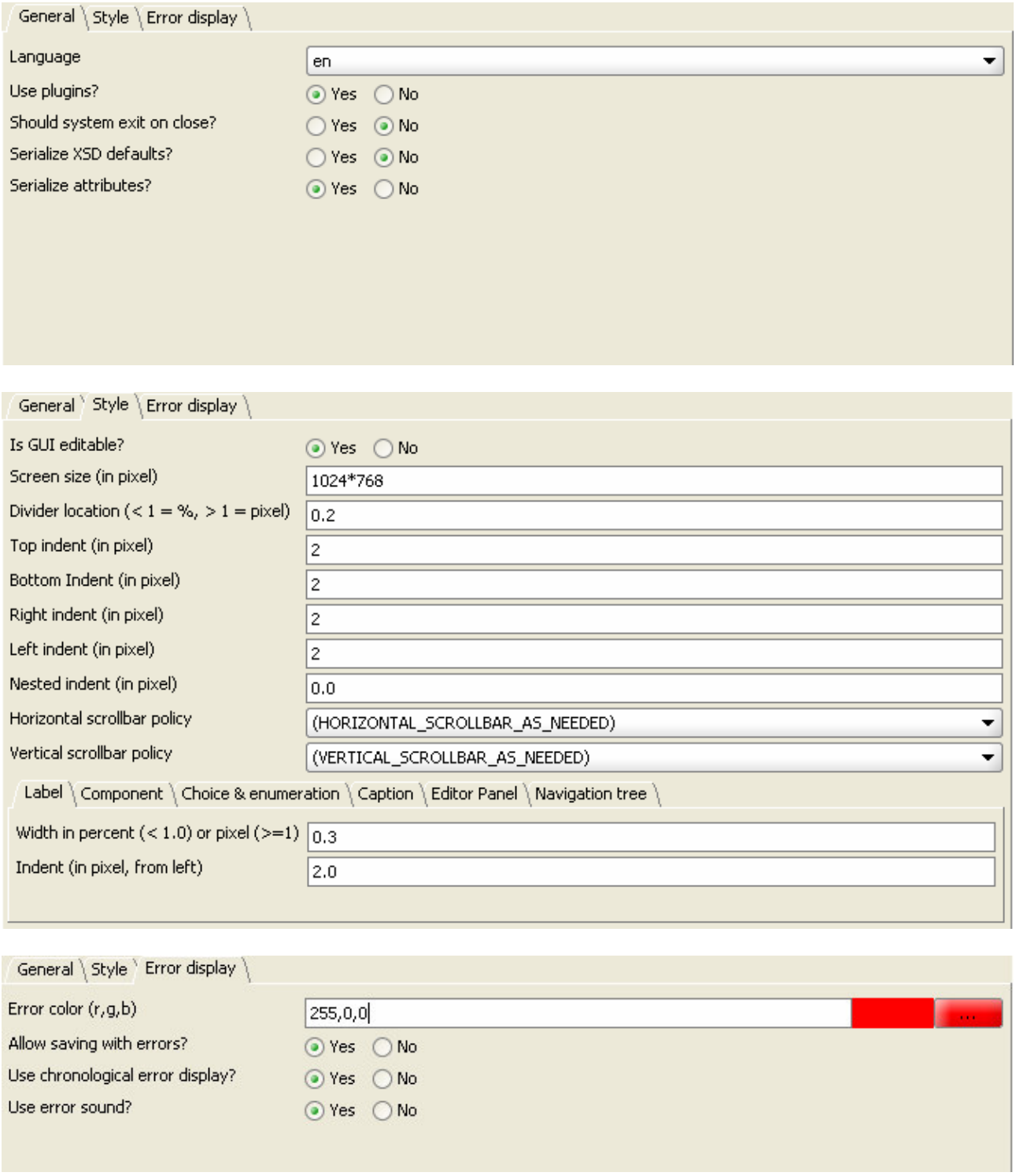


Figure 22: Global XUI definitions

The settings are divided into different categories:

- **General**
- **Style**
  - **Label**
  - **Component**
  - **Choice & enumerations**
  - **Caption**
  - **Editor Panel**
  - **Navigation tree**
- **Error display**

Field name *XUI schema name*	Default	Description
General		
Language language	en	Defines the language to be used.
Use plugins? usePlugins	Yes	Defines whether JavaBeans can be used as a plug-in class.
Should system exit on close? systemExitOnClose	No	Defines whether the application is to be closed when the object editor is closed.
Serialize XSD defaults? serializeDefaults	No	Defines if the xsd defaults (for primitive types) should be serialized.
Serialize attributes? serializeAttributes	Yes	Defines if the xsd attributes should be serialized.
Style		
Is GUI editable? editable	Yes	Defines whether the document is editable or not.
Screen size (in pixel) screenSize	800*600	Defines the standard screen size.
Divider location dividerLocation	0.2	Defines the position of the split-bar, either in pixels (> 1) or percents (< 1.0).
Top indent (px) globalTopIndent	2	Defines the top indent for the entire workspace. The setting is expressed in pixels.
Bottom indent (px) globalBottomIndent	2	Defines the button indent for the entire workspace. The setting is expressed in pixels.
Left indent (px) globalLeftIndent	2	Defines the left indent for the entire workspace. The setting is expressed in pixels.
Right indent (px) globalRightIndent	2	Defines the right indent for the entire workspace. The setting is expressed in pixels.
Nested indent (px) globalNestedIndent	0	Defines the indent from the left border. The indent increases with each increasing nesting level. The indent can be set in pixels only.
Horizontal scrollbar policy horizontalScrollbarPolicy	asNeeded	Defines the horizontal scrollbar behavior.
Vertical scrollbar policy verticalScrollbarPolicy	asNeeded	Defines the vertical scrollbar behavior.
Style - Label		
Width in percent or pixel	0.3	The width of the component label. The setting is

labelColumnWidth		expressed in pixels or percents.
Indent (in pixel, from left) globalLabelIndent	2	Defines the indent from the left border, for the label of a component. The setting is expressed in pixels.

Style – Component

Width in percent or pixel componentColumnWidth	0.7	The component width. The setting is expressed in pixels or percents. The Width setting makes sense only in combination with the label.
Indent (in pixel, from left) globalComponentIndent	0	Defines the left indent for the component. The setting is expressed in pixels.
Top indent (px) topSpaceComponent	1	Defines the top margin for the component.
Bottom space indent (px) bottomSpaceComponent	1	Defines the bottom margin for the component.
Show attributes? serializeAttributes	Yes	Defines whether attributes will be displayed
Show fixed leaf values? showFixedValue	Yes	Defines, if elements, for which a fixed value was assigned in the XML schema, are displayed or not.
Use optional check box? useOptionalCheckBox	No	Defines whether optional data elements can be displayed or removed via checkbox.
Optional field color (r,g,b) optionalFieldColor	255, 255, 255	Defines the background color of the optional fields. Standard: white.
Required field color (r,g,b) requiredFieldColor	255, 225, 204	Determines the background color of components, for which a value must be entered. This is a mandatory field, according to XML schema. Standard: light blue.
Application required color (r,g,b) applicationRequiredFieldColor	155, 100, 255	Determines the background color of components, for which a value must be entered. This is a mandatory field, in accordance with XUI definition. Standard: violet.
No. of visible list items visibleRowCount	5	Defines the number of visible list items within a table.
Boolean visualizer booleanVisualizerType	radio button	Defines how primitive Boolean types (Yes/No) are displayed.
Boolean button sequence booleanVisualizerButtonSequence	yes, no	Defines the sequence of Yes/No radio buttons.

Selection & enumerations

Visualizer globalChoiceType	combo box	Defines how the choice components are displayed, as a combo box or radio-buttons.
Orientation globalChoiceOrientation	horiz.	Defines the orientation (horizontal or vertical) of the choice components, if these are displayed as radio buttons.
Enumeration Visualizer globalEnumerationType	combo box	Defines, how the enumerations components are displayed (as a combo box or radio buttons).
Enumeration Orientation globalEnumerationOrientation	horiz.	Defines the orientation (horizontal or vertical) of the enumerations components, if they are displayed as radio buttons.
No reset if choice made? singleChoiceAndSerialize	No	Defines whether after a successful selection of an optional choice the empty selection no longer

		appears. If this option is activated, the successful (done) selection is serialized in any case.

Style-Caption

Type globalCaptionType	border	Defines the kind of heading to be displayed (frame, title, dividing lines).
Indent (in pixel, from left) globalCaptionIndent	0	Defines the indent from the left border for the component heading (frame, title, dividing lines).
Apply indent on globalApplyCaptionIndentOn		Define if the nested indent should apply for the label, component only or both.

Style-Caption-Border

Top indent (in pixel) topSpaceBorder	0	Defines the upper distance to the preceding visual component. This setting is expressed in pixels.
Inner bottom indent (in pixel) innerBottomSpaceBorder	0	Defines the distance from the last component within the border to the actual border. This setting is expressed in pixels.
Outer bottom indent (in pixel) outerBottomSpaceBorder	0	Defines the distance from the bottom border to the next visual component. with optional borders. This setting is expressed in pixels.
Show optional border control? showOptionalBorderControl	Yes	Defines whether the arrow for the unfolding and folding is displayed with optional borders.
Collapse control by default? collapseOptionalControl	No	Defines if the optional border control should be collapsed by default.

Style-Caption-Header

Top indent (in pixel) topSpaceHeader	0	Defines the upper distance from the dividing line to the preceding visual component. This setting is expressed in pixels.
Bottom indent (in pixel) bottomSpaceHeader	0	Defines the lower distance from the dividing line to the first visual component within this section. This setting is expressed in pixels.
Start color (r,g,b) headerStartColor	44,73,135	Defines the starting color of the verlauf for a title. Standard color : dark gray
End color (r,g,b) headerEndColor	-1, -1, -1	Defines the final color of the Verlauf for a title. Standard color: background color of the panel
Text color headerTextColor	255, 255, 255	Defines the title color. Standard color: white
Make tree entry as default? defaultTreeEntryForHeaders	No	Defines whether all elements, for which a header caption was defined, automatically receive a tree entry.

Style-Caption-Separator (Classic)

Top indent (in pixel) topIndent	0	Defines the upper distance to the preceding visual component. This setting is expressed in pixels.
Bottom indent (in pixel) bottomIndent	0	Defines the lower distance to the next visual component. This setting is expressed in pixels.

Style-Caption-Separator (Modern)

Top indent (in pixel) topIndent	0	Defines the upper distance to the preceding visual component. This setting is expressed in pixels.
Bottom indent (in pixel) bottomIndent		0

Style-Caption-Tab(register)

Top indent (in pixel) topIndent	0	Defines the upper distance to the preceding visual component, in pixels.
Bottom indent (in pixel) bottomIndent	0	Defines the lower distance to the next visual component. This setting is expressed in pixels.

Style-Editor Panel

Show status bar? useStatusBar	Yes	Defines whether a status bar is displayed, to show error messages.
Show button bar? useButtonBar	Yes	Defines whether a button bar is used, containing the standard buttons „Save“ and „Cancel“.
Show forward button? useForwardButton	No	Enables the forward navigation to the next node within the navigation tree. One can navigate through navigation buttons or by pressing the F8 key.
Show backward button? useBackwardButton	No	Enables the backward navigation to the next node within the navigation tree. One can navigate through navigation buttons or by pressing the F7 key.

Style-Navigation tree

Visible? visible	Yes	Defines if the navigation tree should be displayed.
Root node (xpath) treeRoot		Designates any node of the tree as the root. The statement is done with an XPath expression and must address at least one node.
Selected node (xpath) treeSelection		Displays the contents of the selected node in the workspace. The statement is done with an XPath expression and must address at least one node.
Show container item for list? showTreeListContainer	Yes	For lists, defines whether an intermediate element in the tree is displayed, summarizing all the list entries.
Ending for list tree items globalTreeListEnding	(s)	Defines the ending for the list container.
Show help panel? showHelpPanel	No	Defines whether a Help page should be displayed in the north where the XML Schema annotation text of the selected tree netry will be displayed.
Use navigation keys (up,down)? useNavigationKeys	Yes	Defines whether the navigation keys (F7, F8, cursor Up & Down) are active.

Error display

Error color (r,g,b)	255,	Defines the color for error messages.

errorMarkingColor	0, 0	Standard color is red
Allow save with errors? allowSavingWithErrors	Yes	Defines whether an XML instance may be saved if errors are detected.
Use chronological error display? useChronErrorSequence	Yes	Allows the display of the error messages chronologically (true) or in the order of the node definition from the XML schema (false).
Use error sound? useErrorSound	Yes	Enables a beep tone if, for example, an error was validated during the input.

The current background color of the Java container can be addressed with the RGB values -1, -1, -1. This depends on the current look & feel, in each case.

3.3.1 Global Fingerprint Information

A fingerprint provides additional information about the currently loaded XUI file.

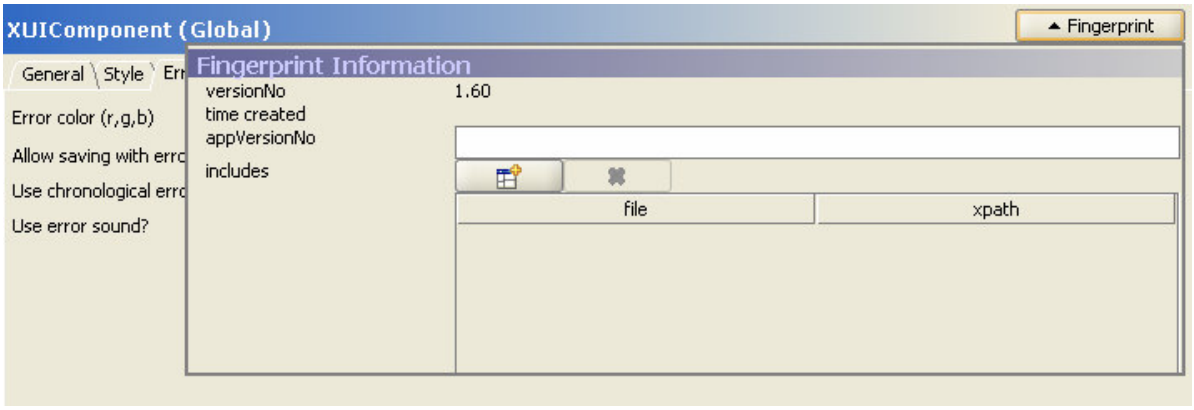


Figure 23: XUI fingerprint information.

Fieldname	Description
versionNo	The XUI version number of the loaded XUI file.
timeCreated	The time when this XUI file was last saved.
apVersionNo	Application version number. Freely selectable.
includes	Includes other XUI files. They are freely selectable, and can be added or deleted. Only relative XUI file specifications are processed. Add an optional xpath for every include to reduce the xui components to be included.

3.4 Editing an XUI component

This section describes how to edit an XUI component.

3.4.1 General concepst

You can delete an already produced XUI component, using the corresponding menu option or the "Delete" button.

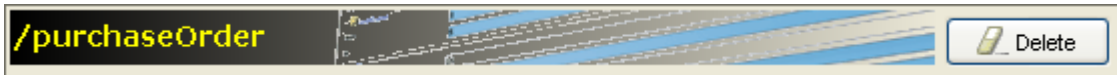


Figure 24: Delete a XUI component

You can copy the contents of list entries or entire groups to the clipboard, as well as reinsert them at a later time.

The right-click menu enables you to copy the content or paste it. The menu can be activated within lists or tabs (registers), for headers or dividing lines.

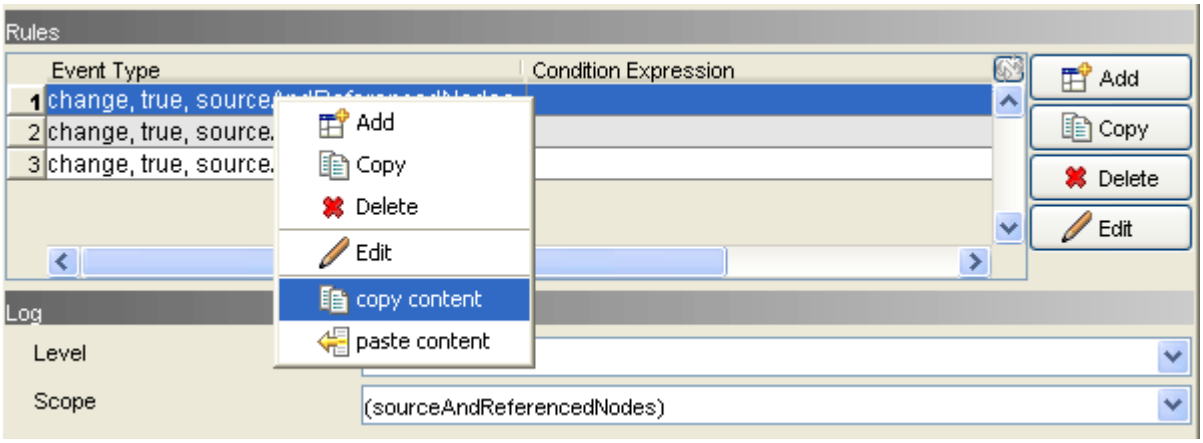


Figure 25: Copy/insert content into lists

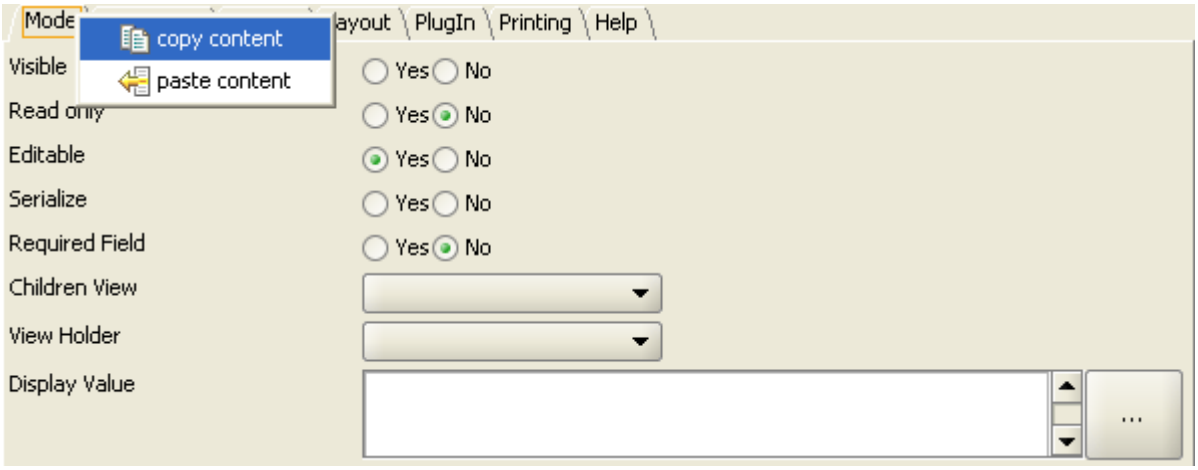


Figure 26: Copy/insert content into tabs

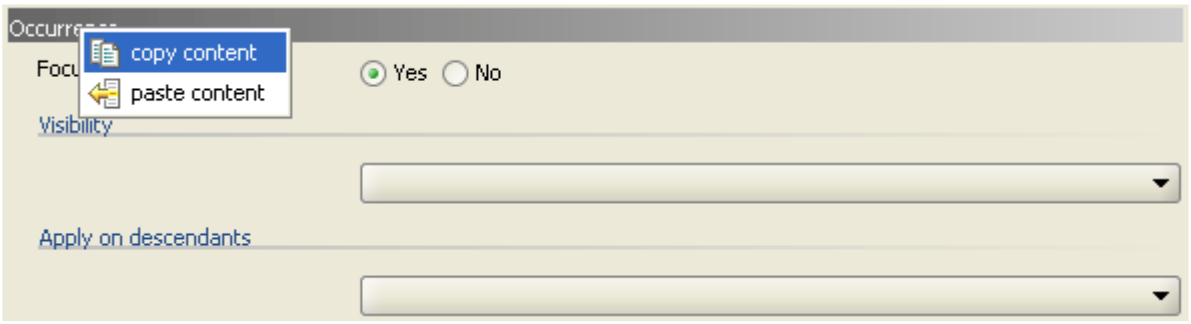


Figure 27: Copy/insert contents into titles

Figure 28: Copy/insert content into dividing lines

3.4.1.1 Editors

To simplify the configuration of a XUI component, XUI-Editor provides the following tools:

- Formula Editor
- Class specification-editor
- File specification-editor
- Color specification-editor

3.4.1.1.1 Formula *Editor*

The Formula Editor enables you to quickly add a formula expression.

Figure 29: Reduced view of the Formula Editor

To display the Formula Editor, click the "Edit" button. The text field to the left of the button indicates the current formula expression. This field is not editable.

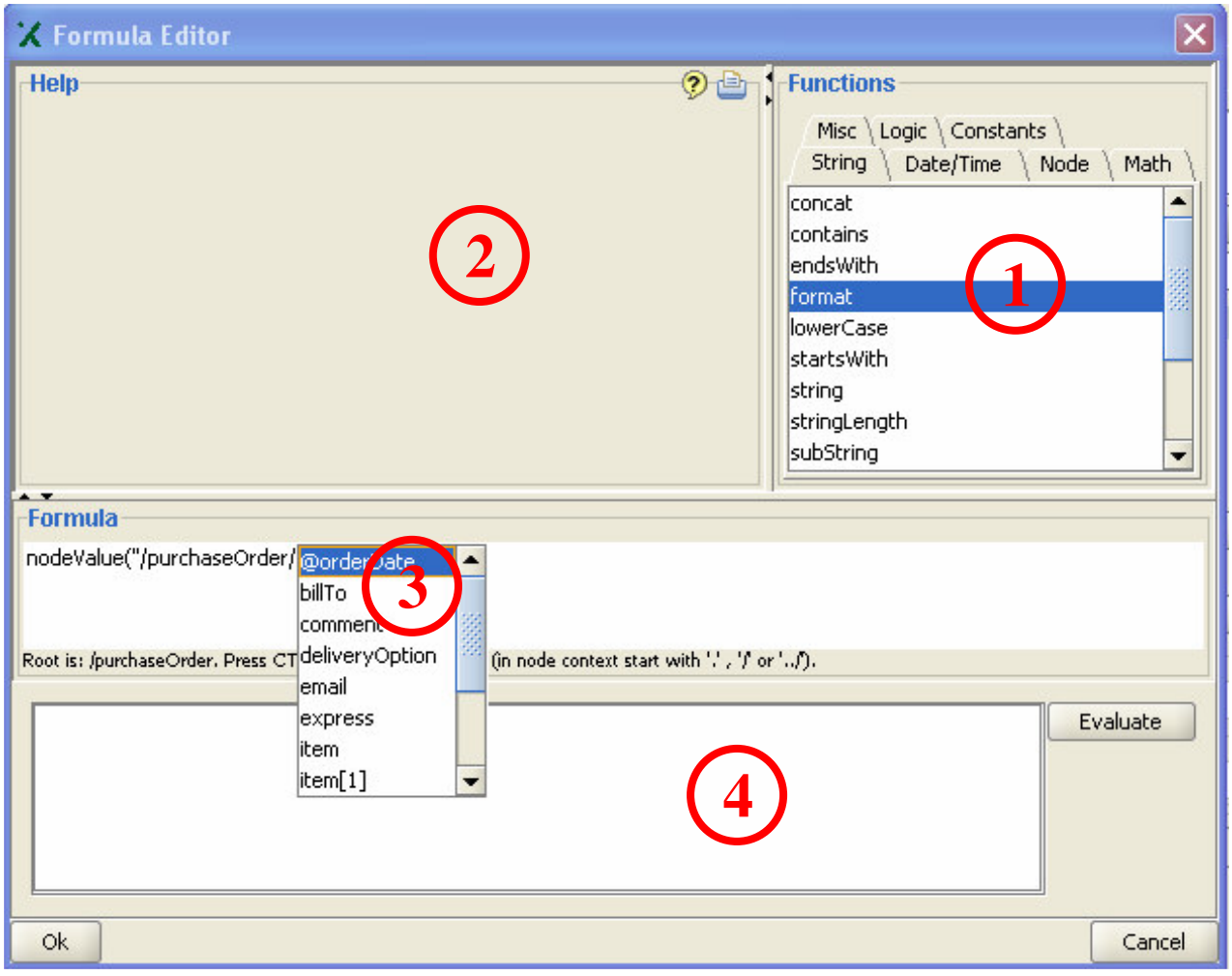


Figure 30: Full view of the Formula Editor

Features of the Formula Editor

The Formula Editor provides the following features to support creation of formula expressions:

Formula Editor feature	Description
(1) Function list	Displays all available functions. Double-click a function, to insert it into the Formula field. The functions are separated in categories (see tabs).
(2) Help area	Describes the function or operator selected in the function list and/or In the popup menu. The Help area includes examples.
(3) „Type-ahead"-functionality for functions and XPath expressions	Here you can enter the formula expression. You can press the *CTRL + Space* key combination to display a popup menu, from which you can select a function and/or a type path selection within an addressing function (node functions).
(4) Syntax check and execution of the formula expression	Checks the syntax used in the formula expression. Click the "Evaluate" button. The syntax is examined and a potential error message or the formula result displayed. The XML file selected when you started XUI_Editor serves as the basis of the formula. Any error messages are displayed within this evaluation area.

3.4.1.1.2 Class specification-editor

Often, you can write your own implementation of a certain task or action. The class specification-editor enables you to specify and reference your own Java class.

Figure 31: The class specification-editor

In the input field, you must enter the entire class name, including the indication of the Java packages (full qualified class name).
When you click the "J" button the following is examined:

a) Whether the class is reachable over the class path,
b) Whether it contains a default constructor, and
c) Whether it implements the indicated Java interface.

If the class can be resolved, respectively instantiated, and the mentioned Java interface implemented, the light indicator changes from red to green. The text below input field talks about the Java interface, which must implement the specified class.

3.4.1.1.3 File selection editor

To indicate a file, use the File selection editor.

Figure 32: Reduced view of the File selection editor

The file name can be entered manually. The indication of the file path is not required.
However, the file has to be resolved over the defined Java class path. The File selection editor is displayed with the button „Select“.

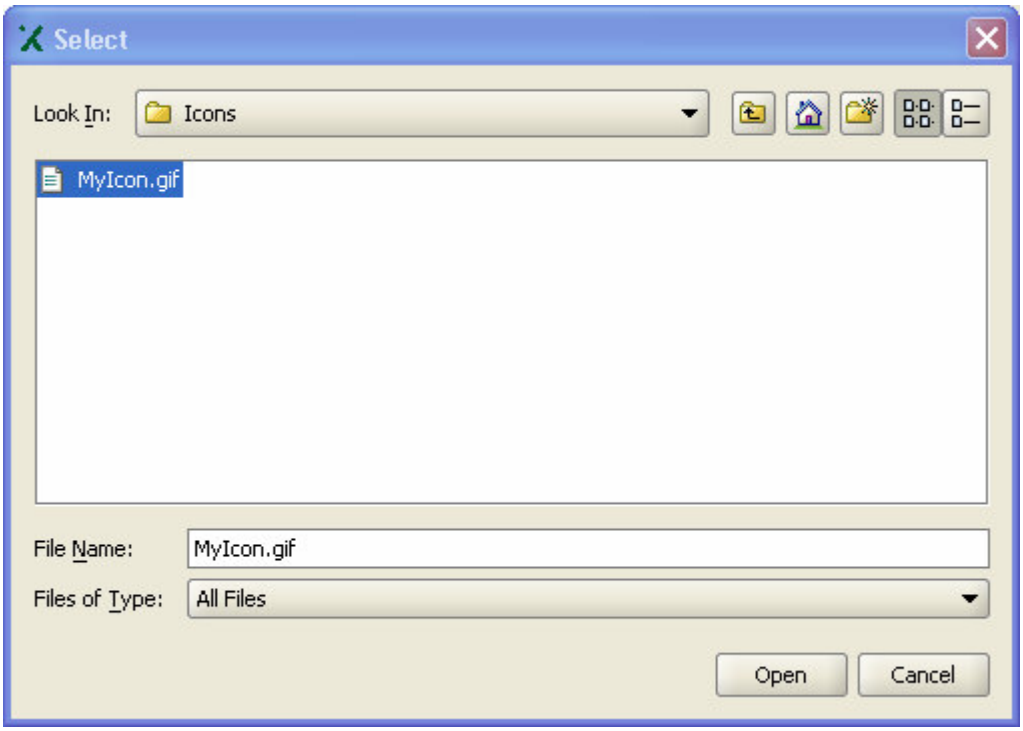


Figure 33: The File selection editor

3.4.1.1.4 Color selection editor

To define a color for a simple component, use the Color selection editor.

Figure 34: Reduced view of the color selection-editor

A color is defined by its RGB value (red, green, blue). The RGB value can be entered manually (separated by comma). The current color is the background color of the button.

With the color value -1, -1-1, you can define the standard background color of the current Look & Feel.

The graphical selection of the color is activated through the button (red button).

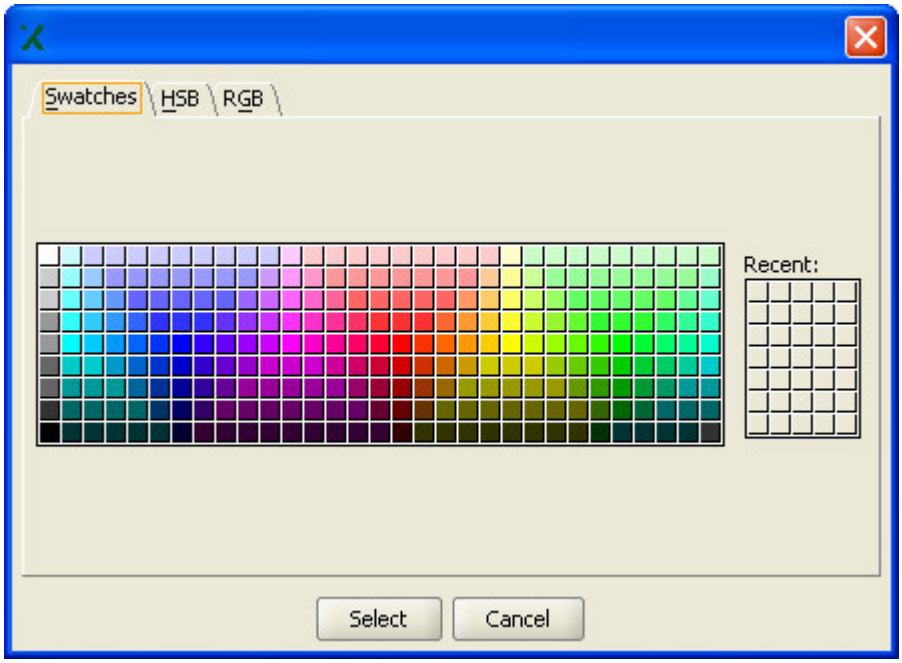


Figure 35: Color selection-editor

3.4.2 Representation (Style)

This section describes how a representation of a component may be controlled.

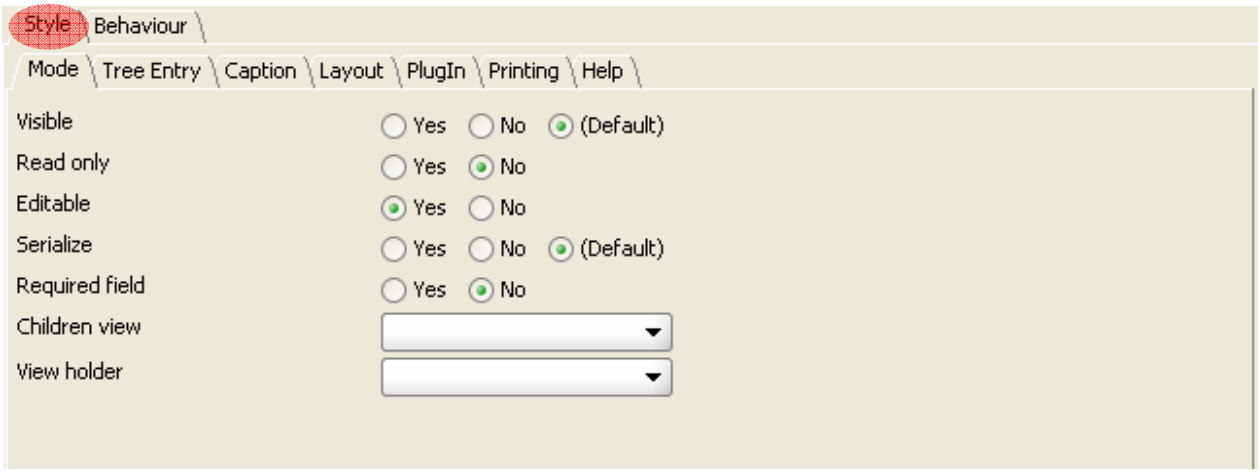


Figure 36: XUI component - Representation

The following sections talks about the representation parameters and how they can be set.

3.4.2.1 Mode

The Mode tab enables you to define the visibility, display of sub-elements and the general display criteria.

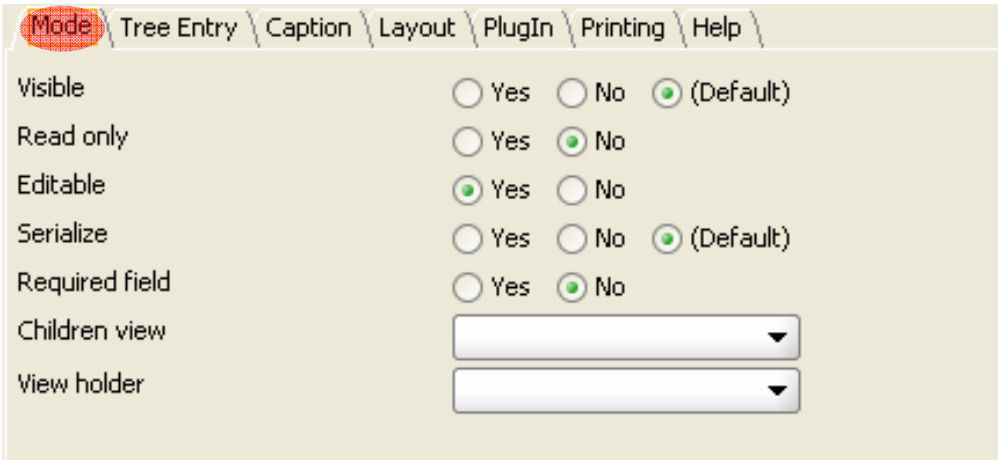


Figure 37: Mode-display in the XUI-Editor

Fieldname	Default	Description
Visible		Defines whether this component is visible or not. Default: if no selection is present the internal display mechanism is used (for example, global settings).
Read only	No	Defines whether this component is active (enabled) or not.
Editable	Yes	Defines whether this component is editable.
Serialize		Defines whether the contents value of this component is to be serialized (saved into an XML instance) or not. Default: if no selection is present the internal serialization mechanism is used (for example, global settings).
Required field	No	Defines whether this component has a mandatory input field. If it does, it is colored with the designated color or the controllability is changed (with compositions).
Children View (see 3.4.2.1.1)		Defines how direct sub-elements are displayed.
View Holder (see 3.4.2.1.2)		Defines the place (context) where the component is to be represented. Default: if this is a complex list (CGL, ComplexGroupList), the visualization context is set internally to „self“. Otherwise, „parent“ is applied.

If the selected node is a primitive type, the error message can be specified as follows.

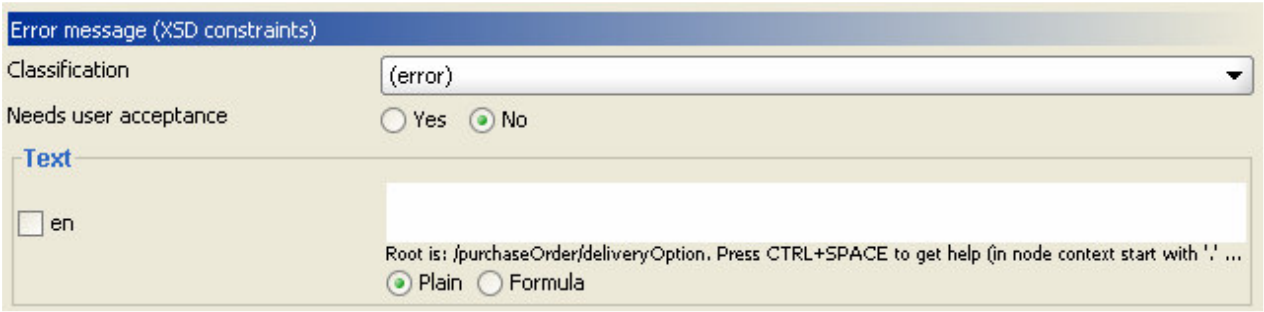


Figure 38: Error message of primitive types.

Fieldname	Default	Description
Classification	Error	Defines the category of the error.
Needs user acceptance	No	Defines whether the occurred errors needs a user acceptance to disappear.
Text		Defines the error message text for a certain language (e.g. en – English). All this text information will be stored in the NLS file.

3.4.2.1.1 Children view

The Children view enables you to define how to represent the direct children of an element (defined in the XML schema). Each child that does not correspond to a leaf may be displayed as

- A separate tab,
- A horizontal bar (header) – or –
- A horizontal line (separator).

Name	Description
(default)	By default, all sub-components are placed below each other on a panel. At the same time, a border is drawn around each component.
tab	Sub-components are represented as tabs.
header	Sub-components are partitioned by a horizontal bar with a color gradient and heading text.
separator_classic	Sub-components are divided by a horizontal line and the heading text uses the classical look (line below the text).
separator_modern	Sub-components are divided by a horizontal line and the heading text uses the modern look (line right beside the text).

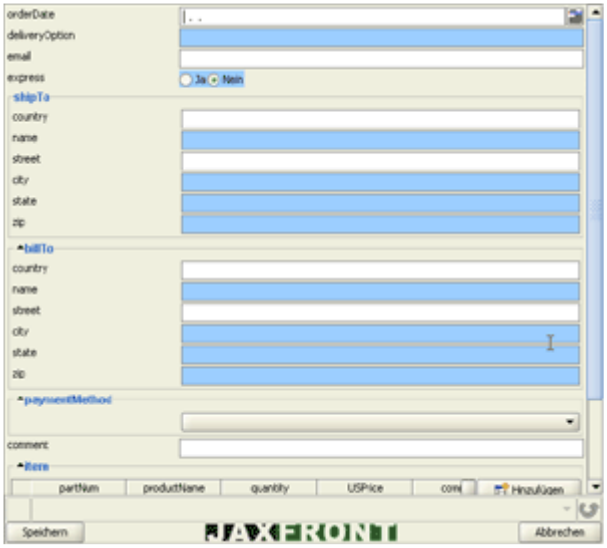


Figure 39: Standard view of sub-elements

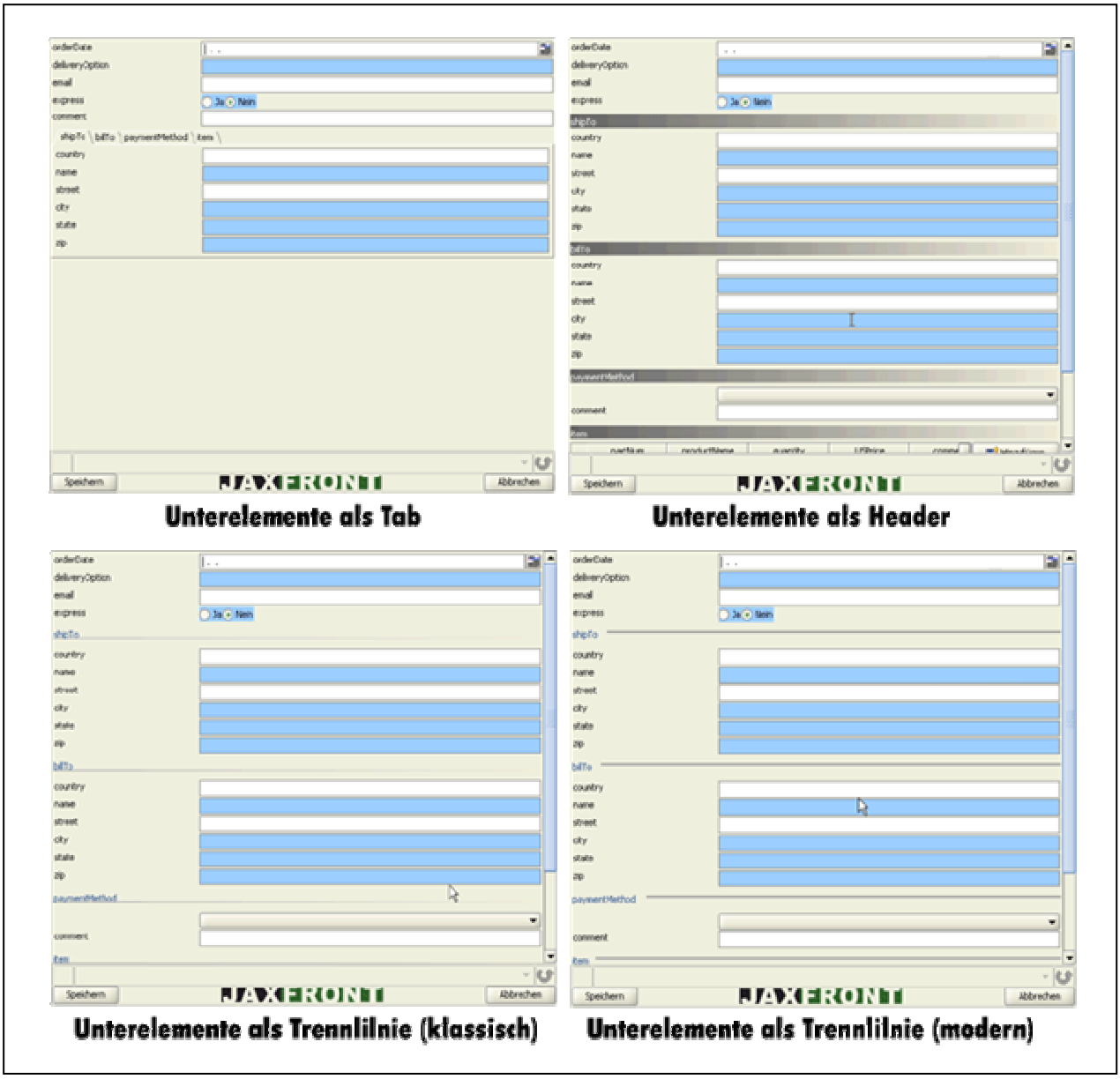


Figure 40: Different views of sub-elements

3.4.2.1.2 View holder

Using the view holder, you can define the context in which the component is to be drawn. By default, all the flat components are drawn in the context of their parental component.

Name	Description
parent (default)	By default, the component is drawn in the context of its parental (superordinate) node, if the flat representation is possible.
self	Component is drawn in its own context. The option is meaningful only if the component has its own tree node -> see TreeEntry).

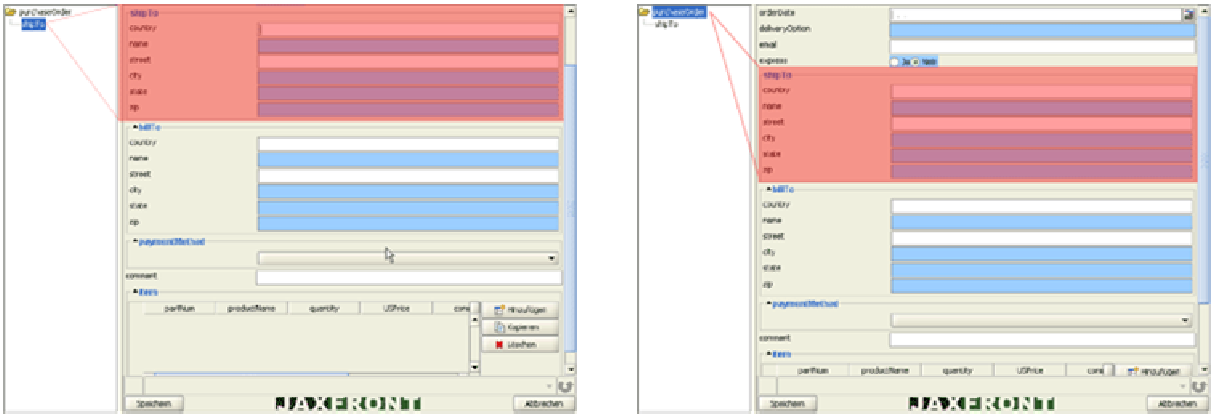


Figure 41: Visualization context - Parent

Illustration on the left:
When the tree entry („shipTo“) is clicked, the component to be indicated (see the red area) is displayed (in this case „purchaseOrder“). The component is displayed in the context of its parental element (parent). Scroll to the desired component.

Illustration on the right:
When the root node („purchaseOrder“) is clicked, the component with the „parent“ visualization context is still displayed. See the red range.

By default, the complex list (ComplexGroupList is displayed in its own context (self). This is an exception.

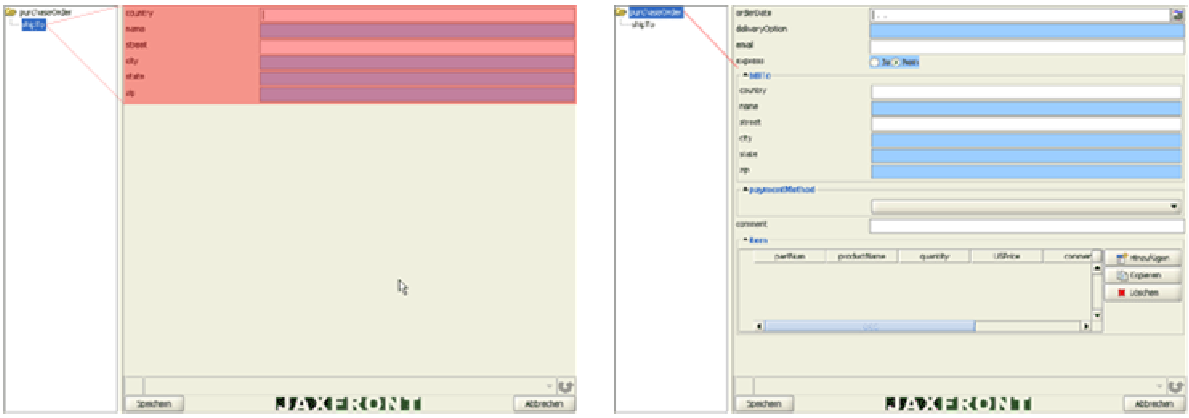


Figure 42: Visualization context - Self

Illustration on the left:
With the visualization context „self“, only the component to be indicated (see the red area) is displayed when a tree entry is clicked. In this case, the component is„shipTo“.

Illustration on the right:
When the root node is clicked („purchaseOrder“) the component with the visualization context „self“ is no longer displayed. This component now lives in its own context (self).

3.4.2.2 Tree Entry

The Tree Entry tab enables you to define whether a component appears in the navigation tree. By default, only the root element and complex lists are inserted into the tree. However, each component can be entered into the tree.

A tree entry enables you to provide a better overview of complex and deeply interlocked XML schema structures. The tree is to be understood as a navigation component.

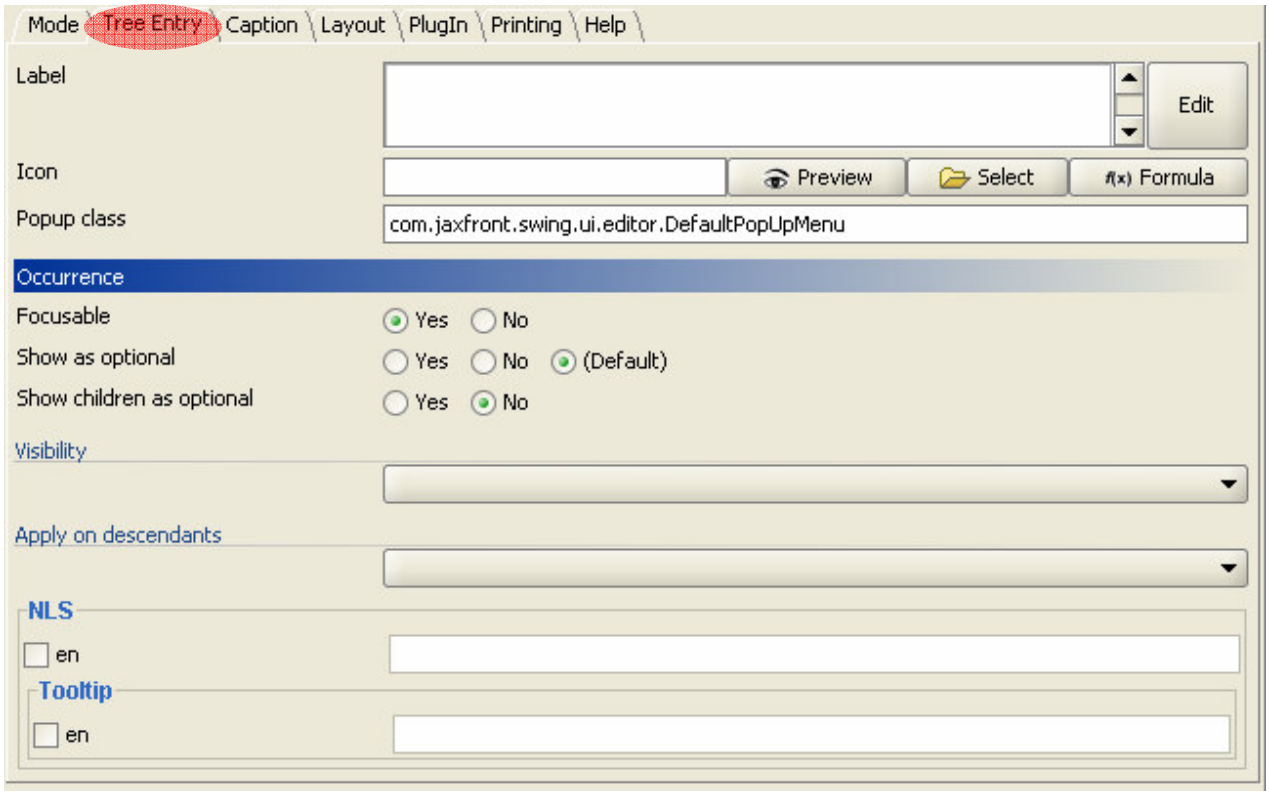


Figure 43: XUI-component - TreeEntry

Fieldname	Default	Description
Label		Formula expression that designates the tree entry. Default: if no name is indicated, the element name defined in the XML schema is used.
Icon		Defines a picture (icon) displayed in the tree, for this tree entry. Absolute or relative picture names are permitted (resolved over the class path)
Popup class	Default PopUp Menu	Defines a popup implementation class. This class needs to be a subclass of: com.jaxfront.swing.ui.editor.AbstractPopUpMenu.
Focusable	Yes	Defines whether the current tree entry can be traversed with the navigation keys (F7, respectively F8).
Visibility (see 3.4.2.2.1)		Defines whether the current component appears in the tree. By default, only complex list types (ComplexGroupList) are displayed as a tree entry.
Apply on descendants? (see 3.4.2.2.2)		Defines the visibility operating range.

3.4.2.2.1 Visibility

There are three kinds of visibility:

Name	Description
(default)	By default, only complex lists tree entries are produced.
always	Component is always displayed as a node in the navigation tree.
never	Component is never displayed in the navigation tree.
dependent on	Component takes over the tree settings of the node referred through XPath.

3.4.2.2.2 Visibility – Operating range

There are two kinds of operating range:

Name	Description
Direct sub-elements	All *direct* descendants are affected by the defined visibility.
all sub-elements	All descendants are affected by the defined visibility.

3.4.2.3 Caption (Heading)

The Caption tab enables you to define a heading for the current component.

By default, only *group* elements, both simple and complex, are drawn within the frame. In this way, you can better visualize the grouping. However, you can define a frame or another kind of heading for each individual component.

The cardinality definition of the current component from the XML schema specifies whether a frame is drawn as optional or not. (0 = optional, 1 = mandatory).

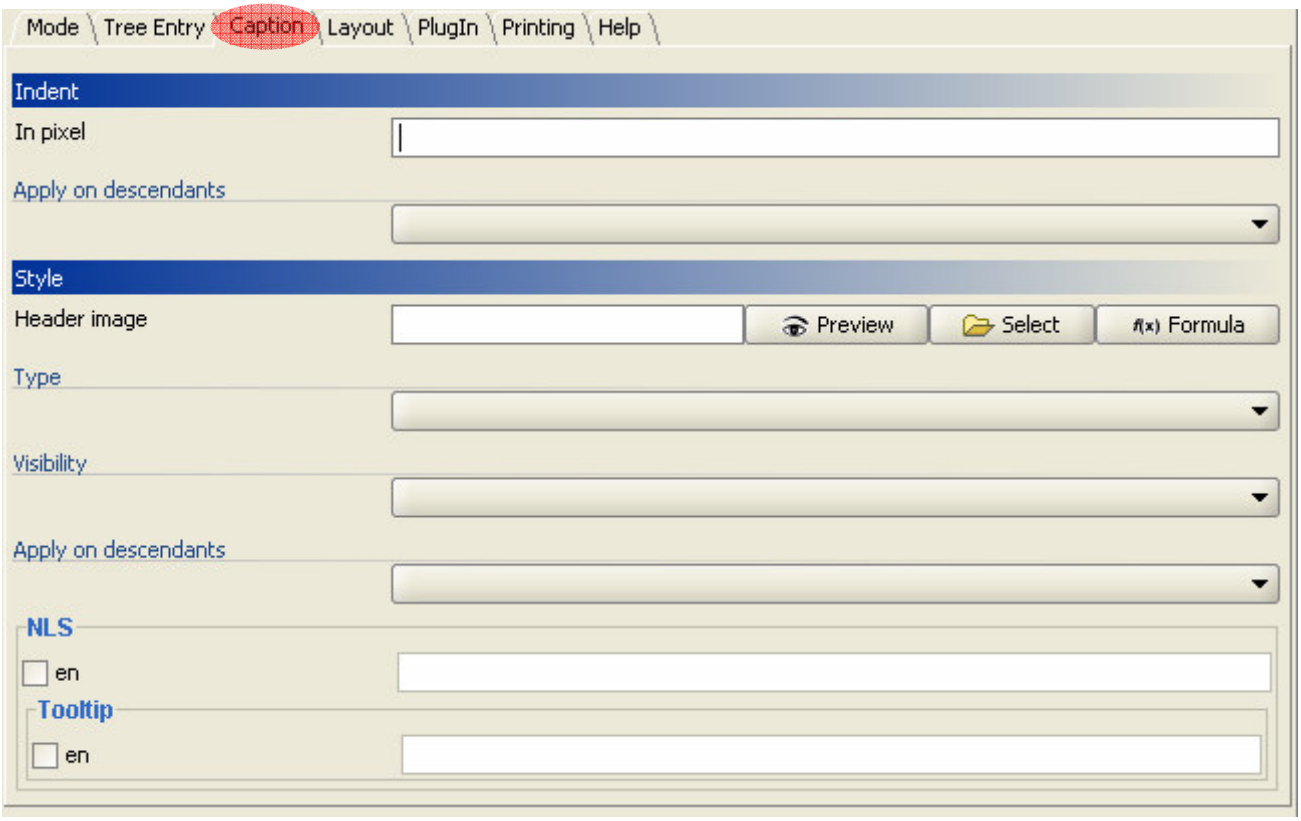


Figure 44: XUI-component - Heading

Fieldname	Default	Description
Indent in pixel		Distance in pixels, from the left border (indent).
Apply on descendants? (see 3.4.2.3.1)		Defines the operating range of the indent.
Header image		Defines a picture displayed above the heading.
Type (see 3.4.2.3.2)		Defines the kind of heading.
Visibility (see 3.4.2.3.3)		Defines the visibility of the kind of visualization.
Apply on descendants?		Defines the operating range of both the kind of visualization and the visibility.

3.4.2.3.1 *Indent – Operating range*

The following two operating ranges exist:

Operating range	Description
Direct sub-elements	Only *direct* descendants are affected by the defined indent..
all sub-elements	All descendants are affected by the defined indent.

3.4.2.3.2 *Categories of visualization*

The following visualization categories exist:

Visualisation category	Description
(default)	By default, all groups (simple or complex) are displayed with a border. However, the border is removed if a group is represented in the tree and has its own visualization context (*self*).
Border	Draws a border around the entire component.
Header	Draws a bar with color gradient over the entire width of this component..
Separator (modern)	Draws a thin line over the entire width of this component (to the right of the heading text).
Separator (classic)	Draws a thin line below the heading text, over the entire width of this component.

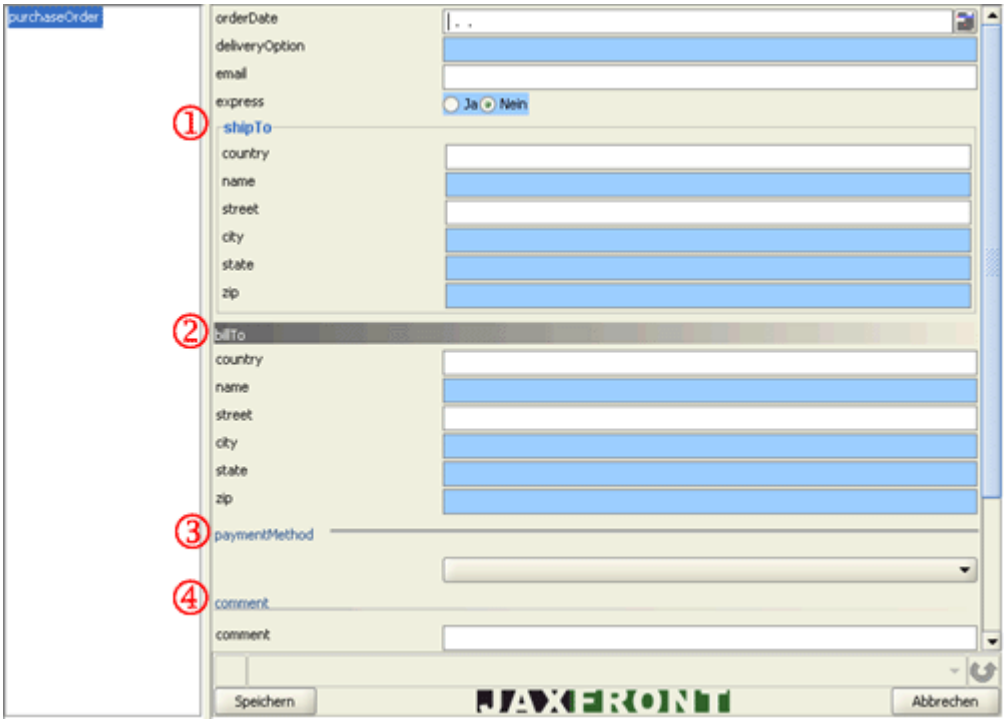


Figure 45: XUI component - Different headings

In the above example, all the visualization categories have been used, producing the expected and distinct visual effect. For the composition „shipTo“ (1) the border was selected. The element „billTo“ (2) is represented as a title. Finally, the option „paymentMethod“ (3) is shown as a modern dividing line, and the input field „comment“ (4) as a classical dividing line.

3.4.2.3.3 Visibility

The following three categories of visibility exist:

Visibility category	Description
(default)	By default, only the group headings (simple or complex) are produced.
Always	Defined category of overview is always displayed.
Never	Defined category of overview is never displayed.
dependent on	Component takes over the kind of heading belonging to the node referred by xpath.

3.4.2.3.4 Visibility – Operating range

The following two operating ranges exist:

Visibility-Operating range	Description
direct sub-elements	Only the direct descendants are affected by the defined visibility.
all sub-elements	All descendants are affected by the defined visibility.

3.4.2.4 Layout

Using the Layout Manager, you can arrange individual visual components.. The Layout Manager allows free positioning of individual visualization components within an imaginary grid field.

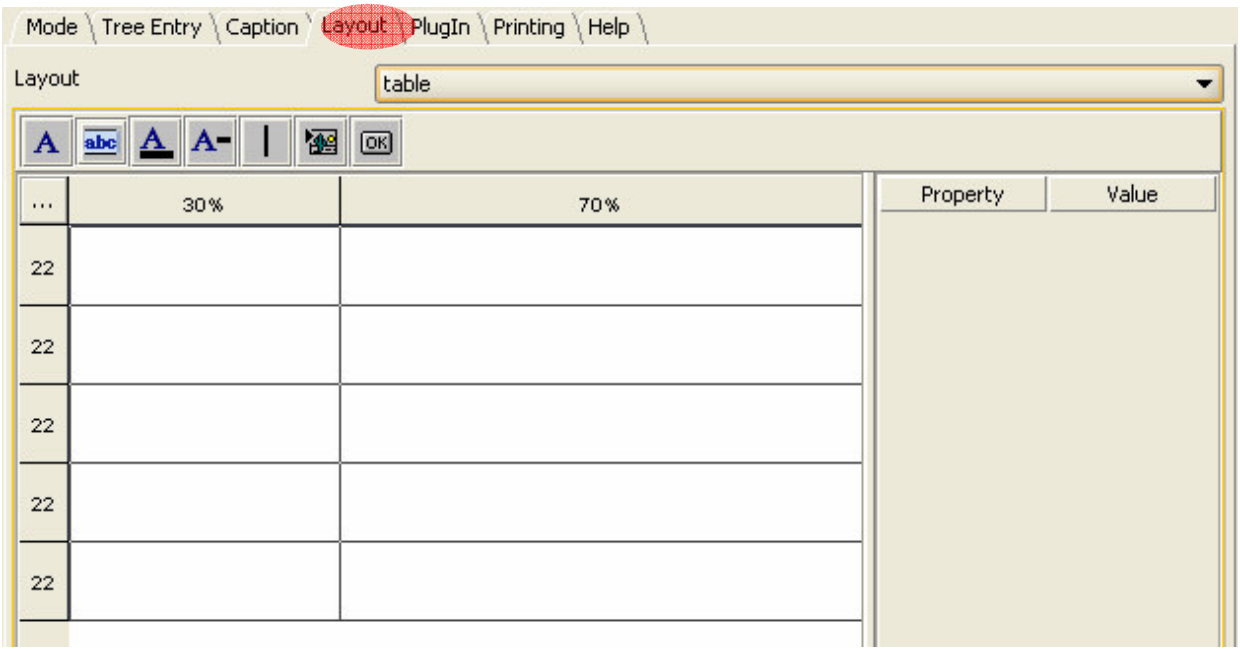


Figure 46: XUI component - Layout

Select the Table option under the Layout tab. The Form Layout editor is displayed.

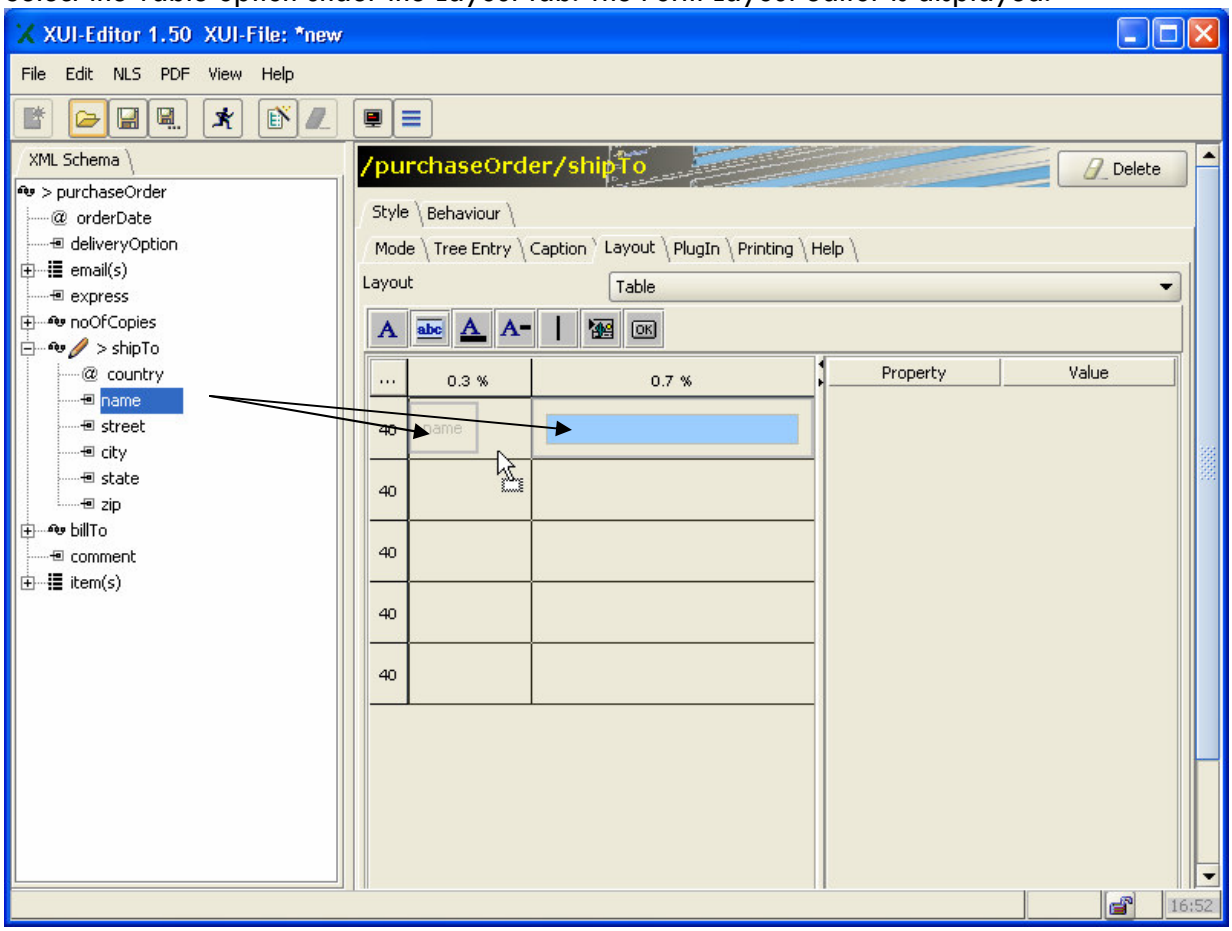


Figure 47: Drag & drop of components on the Form Layout editor

Click a component in the left pane of the Editor, drag it and then drop onto the grid. The visual component is now produced and inserted into the grid.

By positioning a simple component (simpleType), the following two cells are produced:

- Cell for the label
- Another cell for the input field (in this case a text field).

If a PlugIn is defined or any visual characteristics are changed for a component that has already been placed onto the grid, these characteristics are not updated on the grid.

3.4.2.4.1 Structure of the Form Layout editor

The Form Layout Editor is divided into four areas:

1) Toolbar (lists all graphic GUI widgets for the placement on the grid)
2) Main menu (used to produce/define columns and rows)
3) Grid area (represents cell contents)
4) Cell Editor

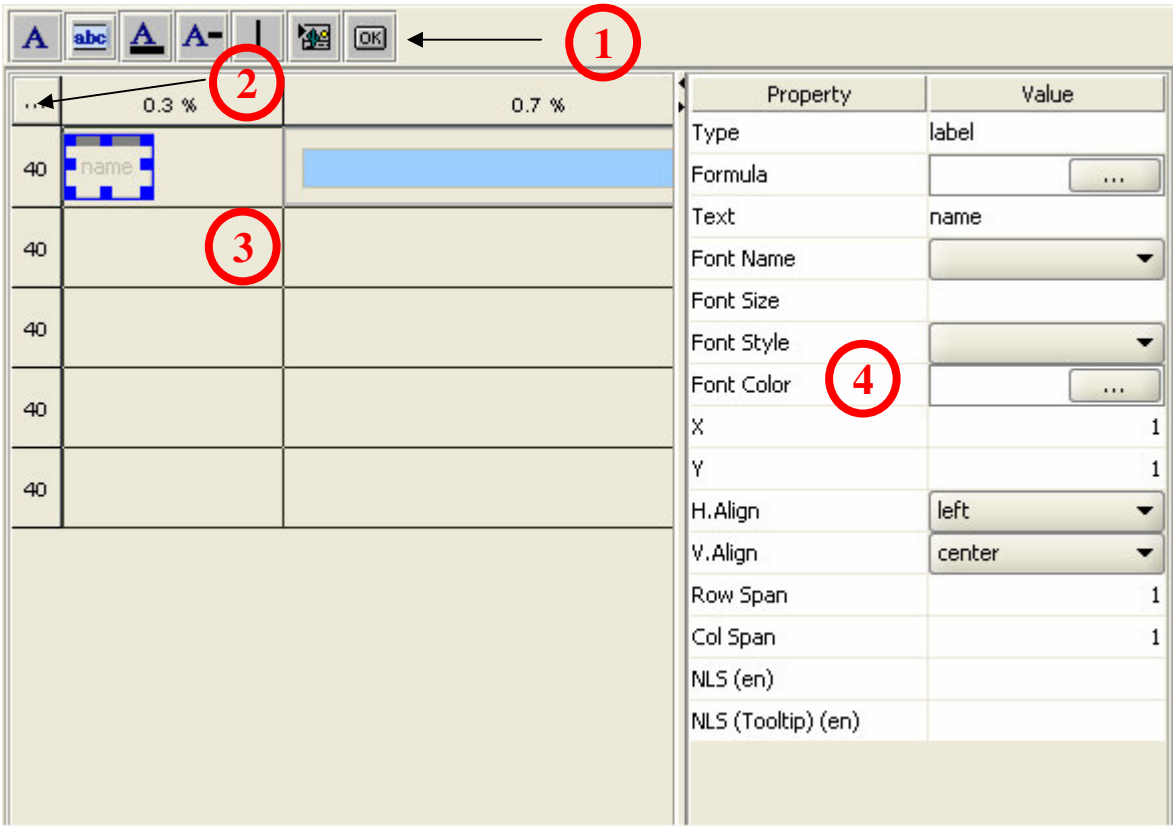


Figure 48: Structure of the form layout editor

3.4.2.4.2 Toolbar

Using the drag & drop technique, you can move all GUI widgets from the toolbar to the grid area. The following seven GUI elements exist:

Icon	Description
	Produces a simple label
	Produces a header.
	Produces a header as a simple separator (classical).
	Produces a header as a modern separator.
	Produces a vertical separator.
	Displays a picture (*.jpg or *.gif).
	Produces a push button. The indication of the Java Action class is needed.

3.4.2.4.3 Definition of columns and rows

The imaginary grid divides the container into several cells. The container can cover several columns and rows. The size of a cell can be adapted to your particular needs, using different strategies.

To produce a new grid, click the push button (number 2, connecting point of column- and row header in the left upper corner of the grid area).

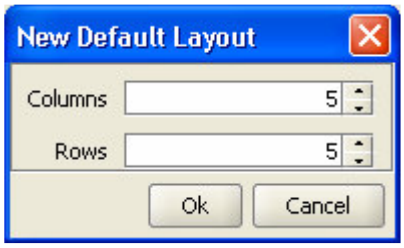


Figure 49: Definition of columns and rows

Fieldname	Description
Columns	Indication of the number of columns.
Rows	Indication of the number of rows.

To indicate different proportions, click on a column (1) or a cell.

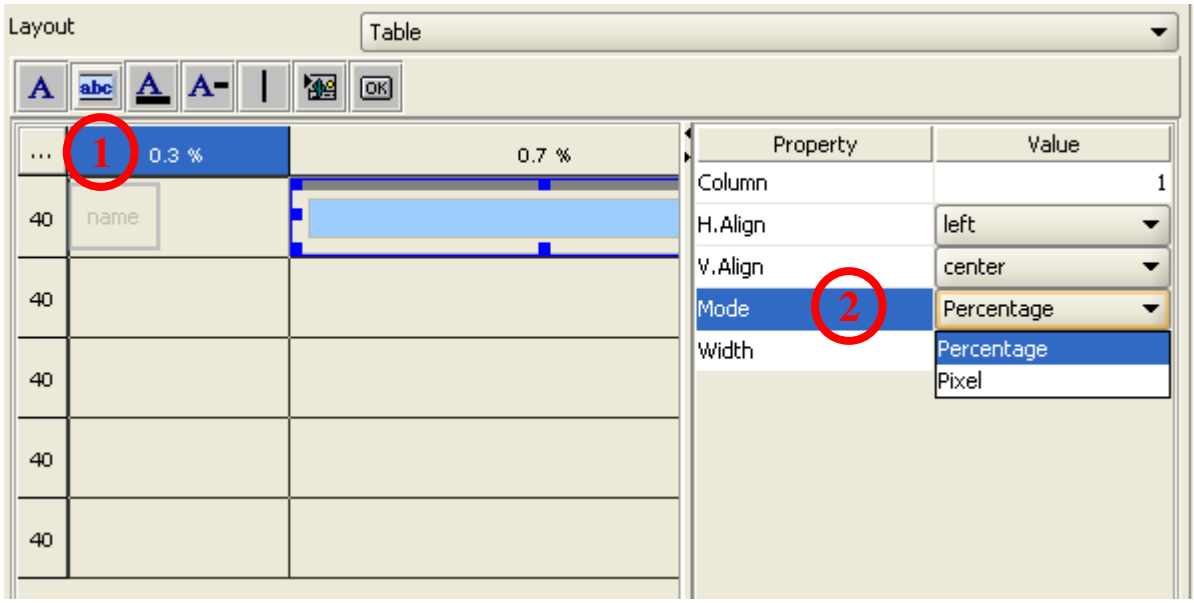


Figure 50: Column-/row definition with the cell editor

In the Field Mode (2) within the Cell Editor, you can select Pixel or Percentage :

Field Mode	Description
Pixel[1]	Defines the width/height in number of pixels.
Percentage	Defines the width/height in per cent.

Initially, the size of a container is divided between the columns and rows, which were defined as „Pixel“ and/or as „preferredSize“. Then, the remaining surface is assigned to the scalable („Percentage“) columns and rows. If the sum of the scalable columns and/or rows is less than 100%, the remainder is distributed to the columns and rows that were defined as „fill“.

The horizontal and vertical alignment of a column, and then a row, can be defined in the „H.Align“ and „V.Align“ fields.

Mode	Description
default	Standard setting: left (horizontal), center (vertical).
fill	Fills the entire cell.
center	Centers the component within a cell.
top	Sets the component to the top border of a cell.
bottom	Sets the component to the lower border of a cell.
left	Aligns the component to the left border of the cell.
Right	Aligns the component to the right border of the cell.

3.4.2.4.4 The virtual grid area

While dividing the container into columns and rows, you can drag and drop the components into the container, and consequently assign them to a logical cell. The component can then use the cell area or extend over several cells (row/column span). In the former case, the component can be further adjusted within the cell. In the latter case, this is not possible.

[1]If the indication of pixel is set to 0, then the row height, respectively the column width, arranges itself after the largest component (preferred size) within this row/column. However , the 0-pixel indication is permitted only if at least one component was placed on the row/column. Otherwise the indication of pixel jumps back to the default value of 40.

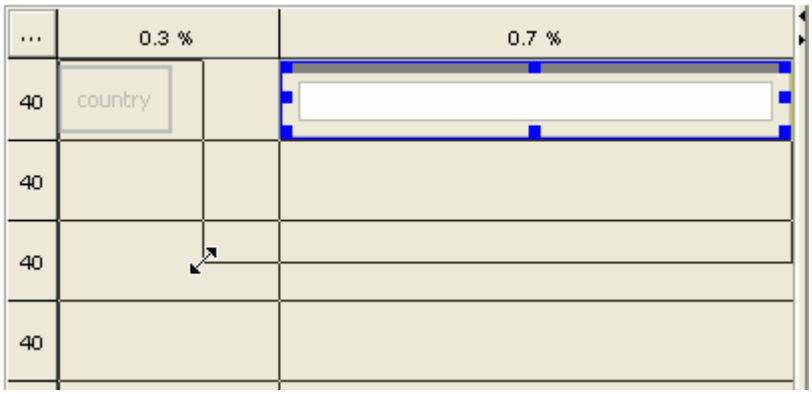


Figure 51: Enlarge a cell

Using the mouse, you can change the cell range and its expansion over several rows, respectively columns. Position the mouse over the blue rectangles (the cursor changes) and hold the left mouse button down. Move the mouse to indicate the cell range.§

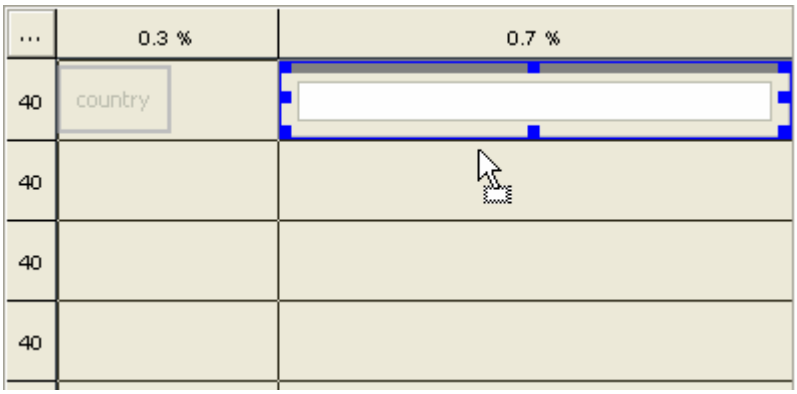


Figure 52: Moving a cell

To assign the component to another cell, grab the blue frame and then pull the component to the desired cell.

3.4.2.4.5 The Cell Editor

With the Cell Editor, all cell data can be mutated.

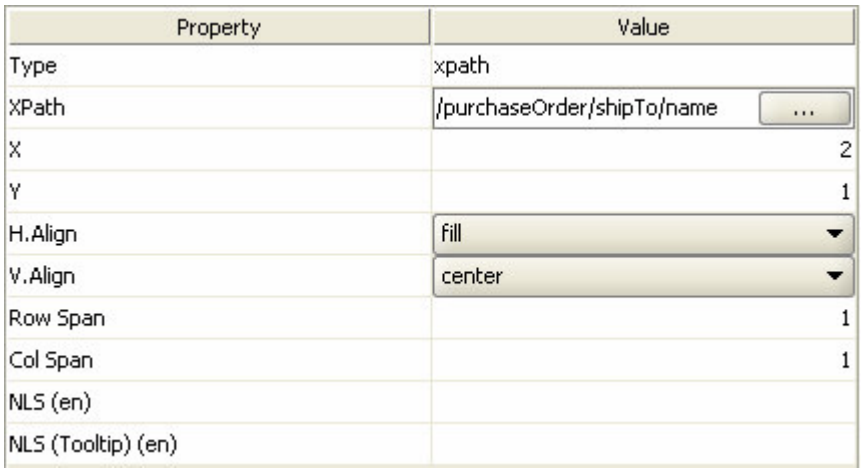


Figure 53: Cell Editor for components

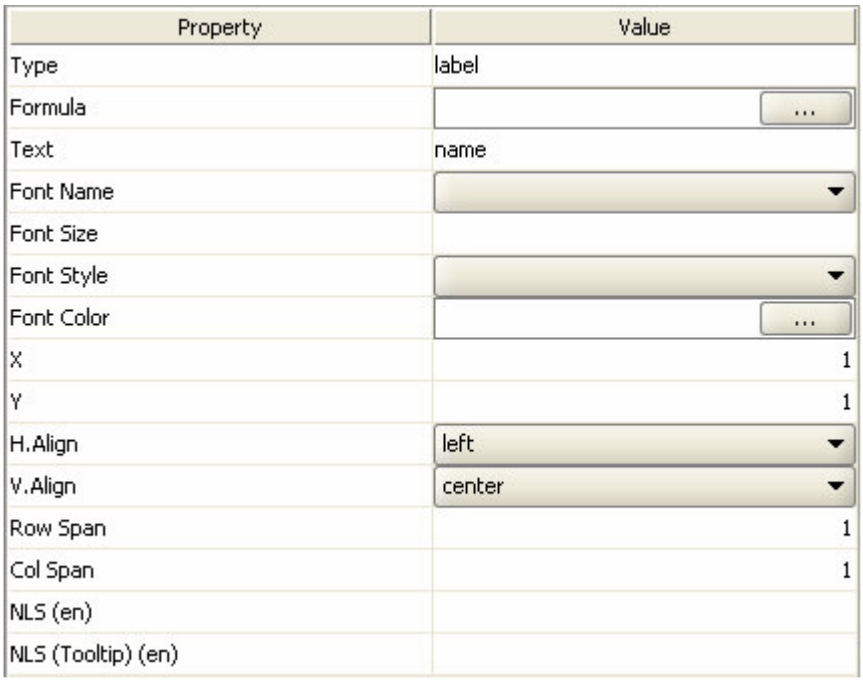


Figure 54: Cell Editor for labels

Property	Description
Type*	Defines the cell type.
XPath*	Displays the XPath of the addressed component. You can click the push button „...“ to jump to the addressed XUI component (if one is available).
Formula	Defines a formula expression for a label.
Text	Defines the standard text for a label (see also NLS)
Character font	Defines the character font used for a label (not yet implemented)
Character size	Defines the character size used in a label (not yet implemented)
Writing color	Defines the label color (not yet implemented)
X*	X coordinate.
Y*	Y coordinate.
H.Align	Defines the horizontal alignment.

V.Align	Defines the vertical alignment.
Row span*	Defines the row span.
Column span*	Defines the column span.
NLS (de)	Defines the language dependent display text.
NLS (Tooltip) (de)	Defines the language dependent tooltip.

* These fields are not editable.

3.4.2.5 PlugIn

The appearance of a component may be changed with the indication of a Plugin class (a special possibility).

The self compiled JavaBean is selected at run-time, instead of the standard component for the visualization of the representation type.

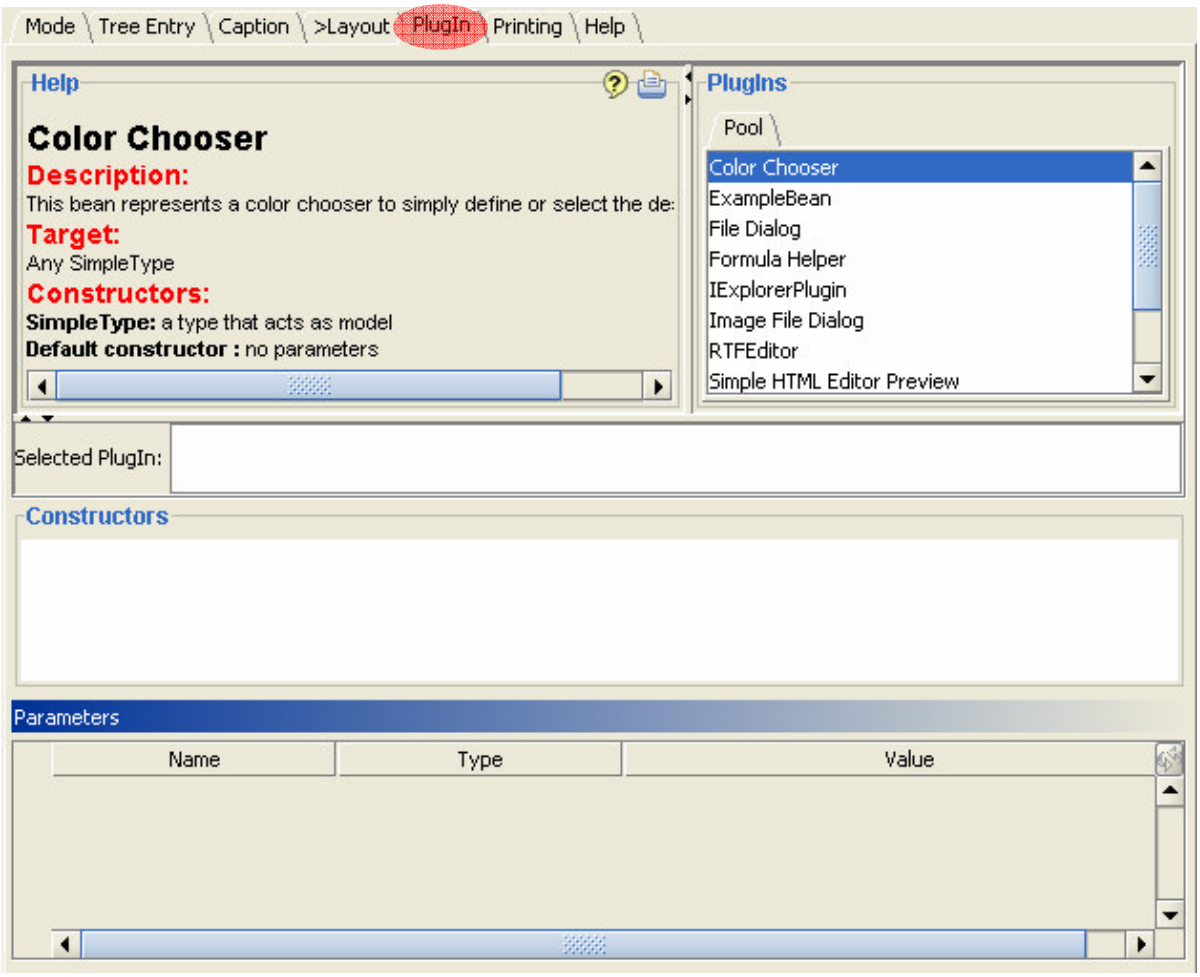


Figure 55: XUI component - PlugIn

The validation and representation of the data is up to the respective plugin component. A Java interface (visualizer) allows the communication between the rendering engine and the PlugIn component.

The following figure shows the definition of the „FileDialog“PlugIn.

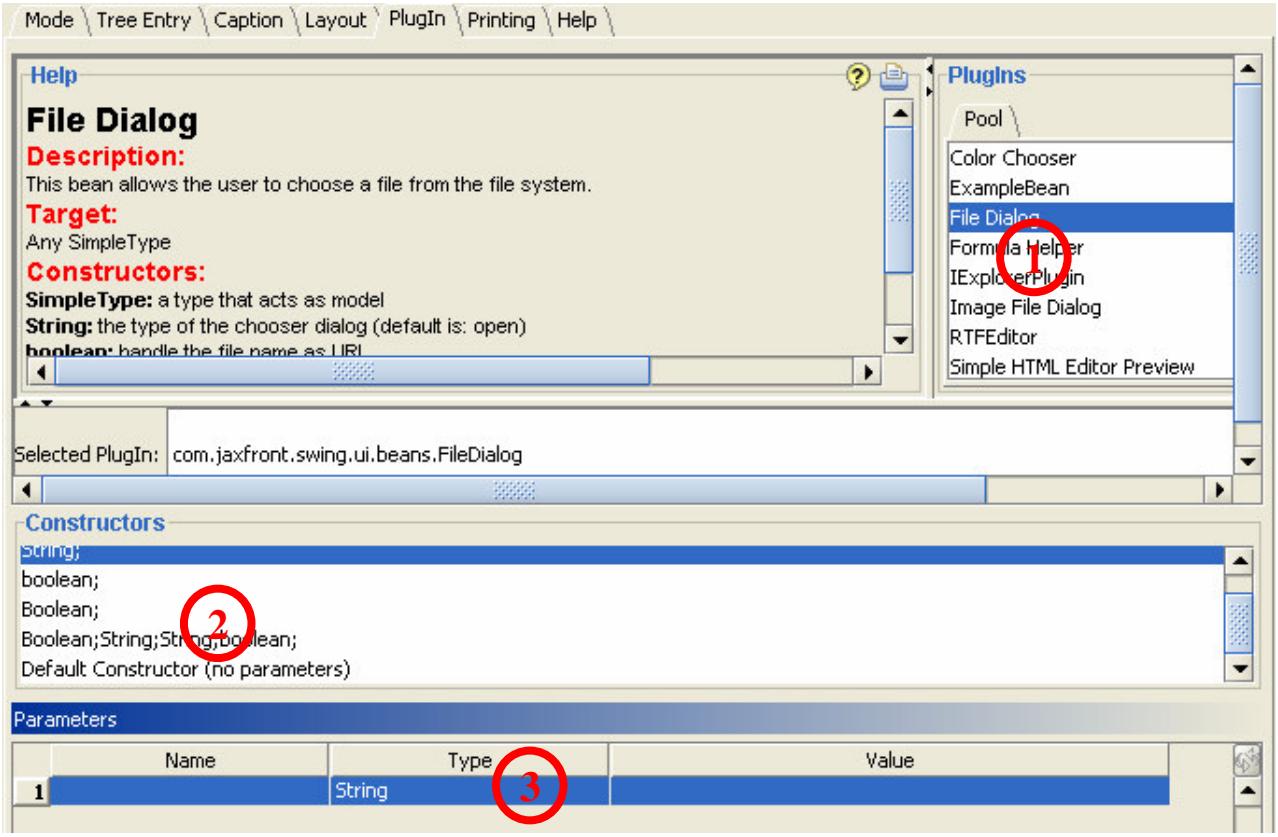


Figure 56: Definition of a PlugIn

(1) Double-click one of the installed PlugIns. The selected class is taken over, and all the configuration options listed under the label „Constructors“.

The PlugIn classes are instantiated with the Java Reflection API. A list of all constructors of this class appears, with a combination of parameter types to be processed by JAXFront. (see section 2.5.3.1). If no constructors are displayed, a *Java Default Constructor* exists for the selected class.

(2) Select the desired configuration (constructor) with a double-click. All the parameters to be declared are listed.

(3) Fill out the contents of the displayed parameter entries

A precise specification of the parameter types and their expected values is delivered by the JavaDoc of the respective PlugIn class.

3.4.2.6 Printing

Using the Printing tab, you can specify print setting per component, for the PDF output. These settings have no effect if the JAXFront PDF rendering engine is not installed.

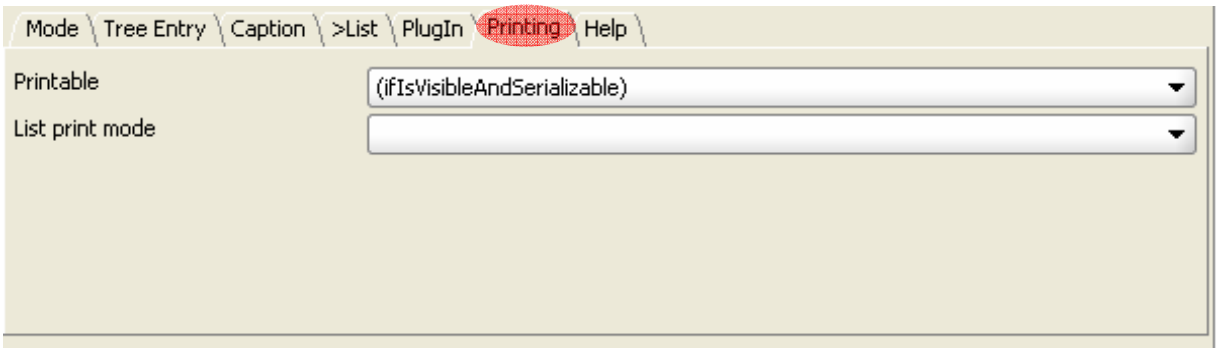


Figure 57: XUI component - Printing

Fieldname	Default	Description
Printable		Defines whether the component is to be printed. Applies to all types. The following options are available: • always (the current type will be printed) • never (the current type will not be printed) • ifIsVisible (the current type will be printed if it is visible) • ifIsSerializable (the current type will be printed if it is serializable) • ifIsVisibleAndSerializable (the current type will be printed if it is visible and serializable)
List print mode		For the list type, you can define how to print the list. The following options are available: • serial (list entries are represented serially interlocked), • table (values of the list entries are represented in a table) –or- • tableAndSerial (combination of table and serial).

3.4.2.7 Help

To connect to an external Help system, for each component you can deposit any identification for all supported languages.

When a component with deposited Help index moves into the visible area, the Help system is informed through event notification. Implement the HelpListener Interface (class: com.jaxfront.core.help.HelpListener) to react on any help notification event. Add the help listener to the EditorPanel - addHelpListener(HelpListener newListener).

```
com.jaxfront.core.dom.Document dom;
com.jaxfront.core.help.HelpListener myHelpListener = new MyHelpListener();
dom.getEditor().addHelpListener(myHelpListener);
```

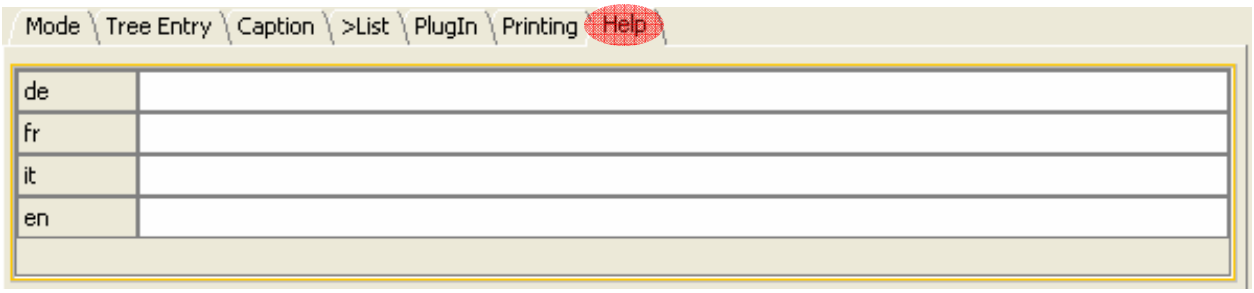


Figure 58: XUI component - Help

3.4.2.8 Leaf

This section defines all the settings for the leaf element (simple type). A simple component always consists of a label and an input component.

The Leaf tab appears only if the selected component is a primitive leaf type.

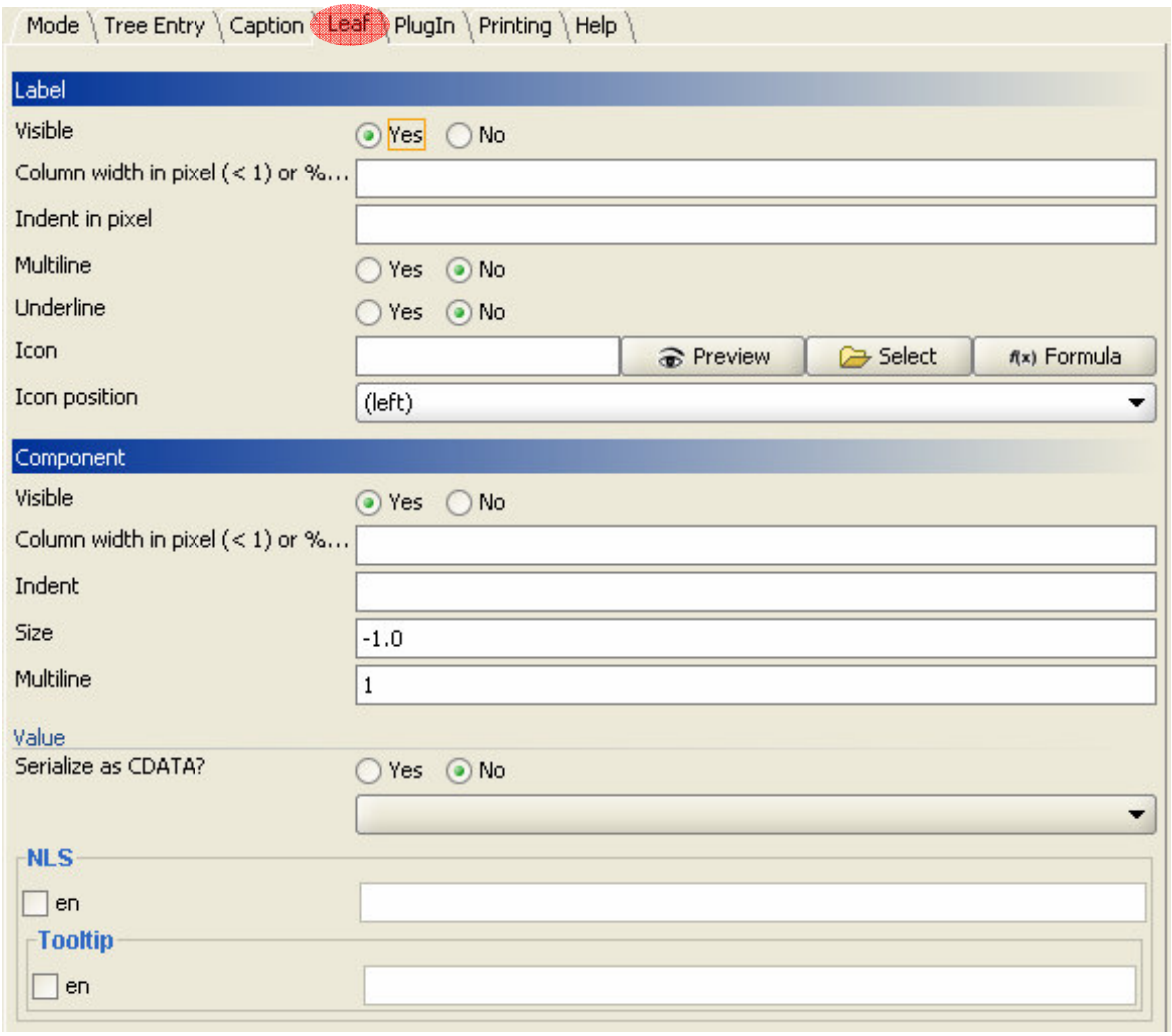


Figure 59: XUI component - Leaf

3.4.2.8.1 Label

Fieldname	Default	Description
Visible	Yes	Defines whether the label is indicated or not.
Column width in pixel (<1) or % (>1)		Defines the width of the label in per cent (if value is less than 1) or in absolute pixels. Default comes from global settings (Style-Label) → 0.3
Indent in pixel		Defines the distance from the left-hand margin (indent) in pixels.
Multiline	No	Defines whether multiple lines (Textarea) are to be displayed if label is longer than space available.
Underline	No	Defines whether the label is to be underlined.
Icon		Defines an icon for the label. You can define an absolute path specification or the icon name only (must be found in the class path).
Icon Position	(left)	Defines the icon position.

3.4.2.8.2 Component

Fieldname	Default	Description
Visible	Yes	Defines whether the component is indicated or not.
Column width in pixel (<1) or % (>1)		Defines the width of the component in per cent (if value is less than 1) or in absolute pixels. Default comes from global settings (Style-Component) → 0.7
Indent		Defines the distance from the right margin of the component (indent) in pixels.
Size	-1.0	Defines the size in pixel of this component.
Multiline	1	Defines whether and how many lines of a string type are to be displayed. If more than one line is defined, a TextArea with the number of defined lines is displayed.
Value		Defines the kind of the input field (simple value or enumeration, see below).
Single Value		Pre-defined value of this component.

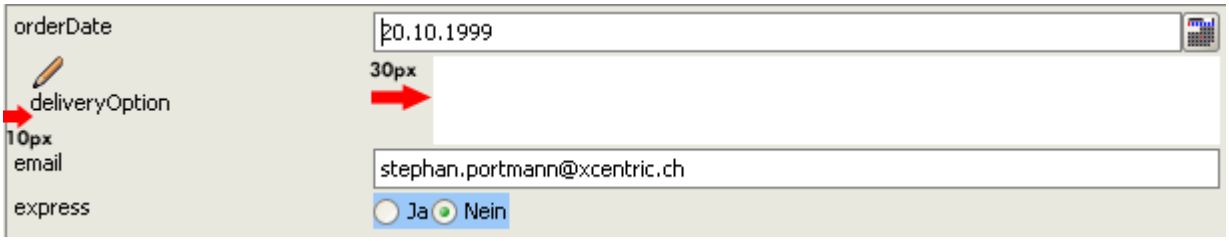


Figure 60: Example: Indent with a label and component

The example above shows an indent of 10 pixels on the label, respectively 30 pixels for the component. In addition, an icon was defined for the label. The icon is displayed above the label. The input field for the „deliveryOption" was set to three (3) lines (number of lines).

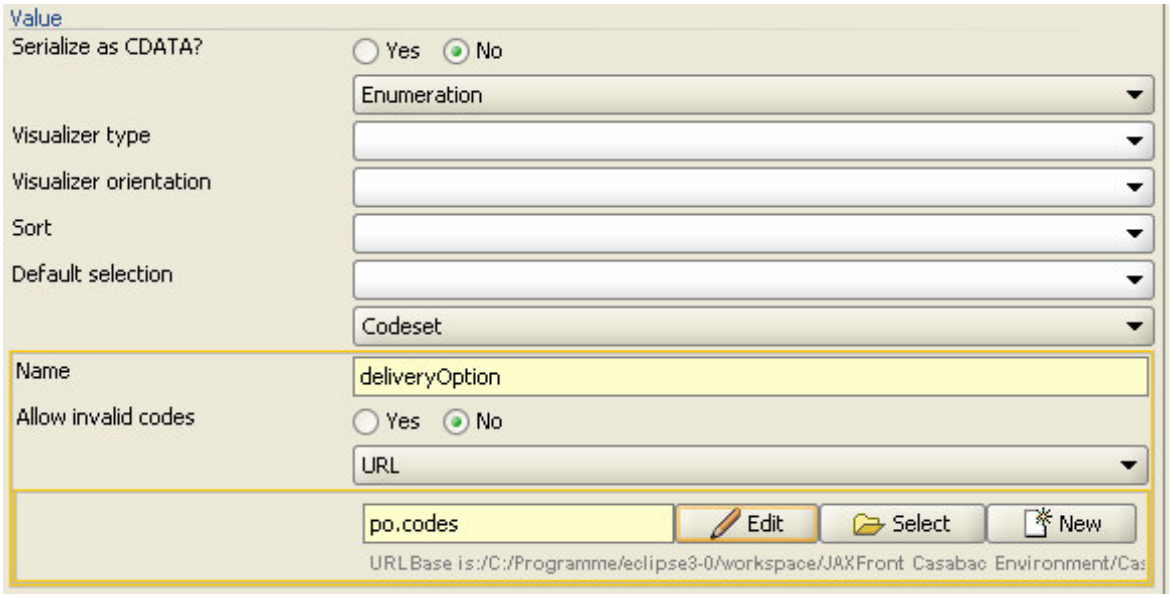


Figure 61: Settings for the type of value

3.4.2.8.3 Value – Enumeration

Fieldname	Default	Description
Serialize as CDATA	No	Defines if the content of this component should be serialized as CDATA.
Visualizer type		Defines the type of indicator (comboBox or radioButton) for the representation of the selection.
Visualizer Orientation	horiz.[1]	Defines the alignment (horizontal or vertical) for the representation of „radioButtons“.
Sort		Defines whether and how the option is to be sorted.
Default Selection		Defines the standard selection.

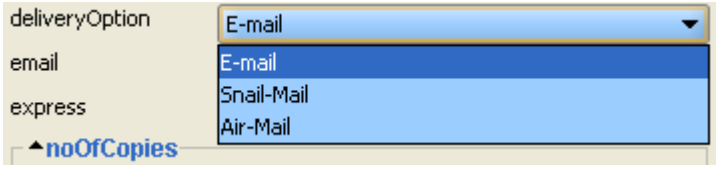


Figure 62: Value enumeration as ComboBox

Figure 63: Value enumeration as RadioButtons (horizontal)

Figure 64: Value enumeration as RadioButtons (vertical)

3.4.2.8.4 CodeSet

Fieldname	Default	Description
Name		Defines the name of a code table within the referred CodeSet.
Allow invalid codes	No	Defines whether codes marked as invalid (displayed) are accepted.
URL*		Defines the localization of the CodeSet. Here, you can indicate a complete URL or only a CodeSet name. If only the name is indicated, the file will be searched in the directory where the XUI file is located.
Class		Instead of the indication of an URL, you can also indicate your Java class definition.

*The indicated CodeSet file (if an URL has been indicated) is about an XML file, which must correspond to a pre-defined format (codes.xsd).

[1] This setting has an effect only if the display component is set to "radioButton".

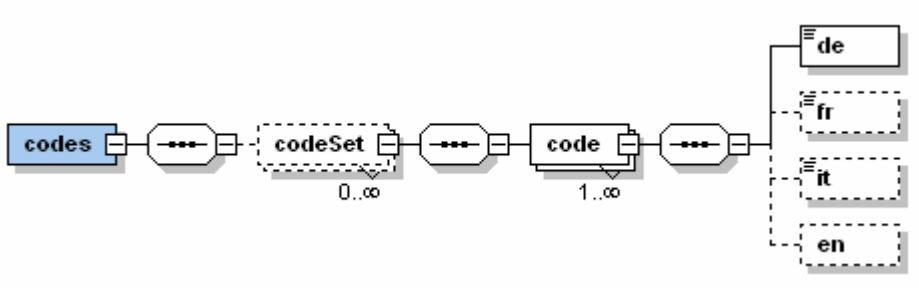


Figure 65: Structure of a CodeSet (XML schema)

XML schema definition: codeSet

Attribute	Default	Description
name		Defines the name of a code table.

XML schema definition: code

Attribute	Default	Description
id		Defines the name of a code table.
status	valid	Defines whether the code is still valid. If the code is invalid , it is marked as invalid in the selection box.

Example of a CodeSet file.

```
<?xml version="1.0" encoding="UTF-8"?>
<codes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="codes.xsd">
  <codeSet name="deliveryOption">
    <code id="A">
      <de>EMail</de>
    </code>
    <code id="C">
      <de>Postweg</de>
    </code>
    <code id="B" status="invalid">
      <de>Luftweg</de>
    </code>
  </codeSet>
  <codeSet name="deliverTo">
    <code id="1">
      <de>Lieferadresse</de>
    </code>
    <code id="2">
      <de>Rechnungsadresse</de>
    </code>
  </codeSet>
</codes>
```

The above shown example delivers two code tables, for „deliveryOption“ and „deliverTo“. The code with the identification „B“ (Luftweg) was marked as invalid. Consequently, it is marked as invalid in the selection box. If the user selects this code, an error message is displayed.

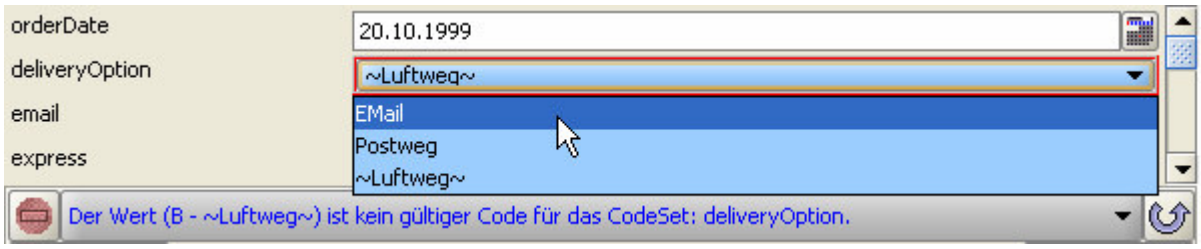


Figure 66: Example of an invalid code from the CodeSet

3.4.2.9 Choice

Using the Choice tab, you can configure the selection elements (choices), defined in the XML schema.

The Choice tab appears only if the selected component is a selection.

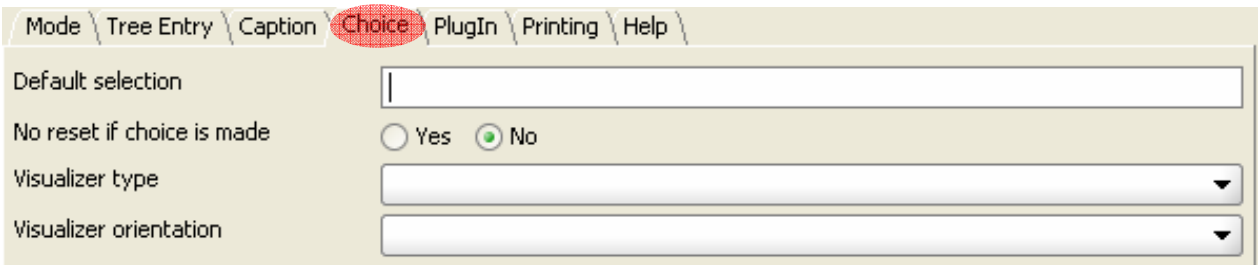


Figure 67: XUI component - Choice

Fieldname	Default	Description
Default selection		Defines the initially selected choice.
No reset if choice is made	No	In case of an optional Choice , defines whether the empty selection no longer appears after a selection occurred. If this option is activated, the selection which took place is serialized in any case.
Visualizer type		Defines whether the selection is displayed in the ComboBox or as a RadioButton.
Visualizer orientation	[1]	Defines how the selection is displayed (horizontally or vertically). The „vertical“ orientation is used to display „radioButtons“. Otherwise, the selection is displayed horizontally

[1].

3.4.2.10 List

Using the List tab, you can configure the list elements defined in the XML schema (cardinality > 1, maxOccurs > 1).

The List tab appears only if the selected component is a list.

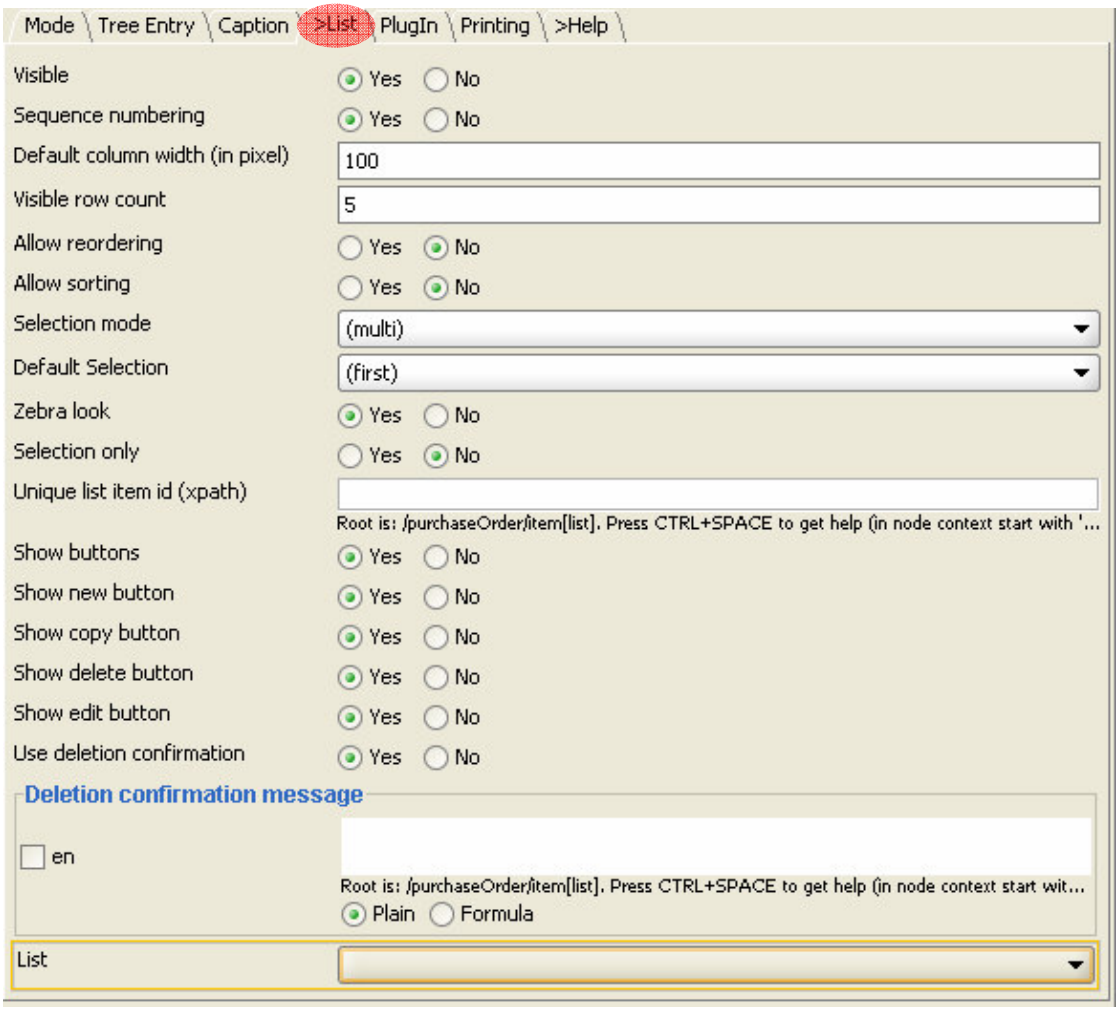


Figure 68: XUI component - List

Fieldname	Default	Description
Visible	Yes	Defines if the table is visible.
Sequence Numbering	Yes	Defines whether the line numbering is to be indicated.
Default column width (in pixel)	100	Defines the standard width for all columns within the table.
Visible row count		Defines the number of visible rows.
Allow Reordering	No	Defines whether the sequence of the list entries can be changed.
Allow Sorting[1]	No	Defines whether columns can be sorted.
Selection mode	single	Defines the selection behavior of rows (single or multi).
Default Selection	none	Defines which row/column should be selected and visualized by default.
Zebra look	Yes	Defines whether the Zebra Look is used. If it is, all the odd rows are colored.

[1] The sort function is possible for complex lists only.

Selection only	No	Defines whether the list is protected (may not be modified). In this case, all the control buttons are automatically removed.
Unique list item id (xpath)		Defines the element of each new list entry for which the unique ID, generated by the system, is administered. The ID consists of the schema name of the list element and a number assigned by the system. Example: items_1
Show buttons	Yes	Defines whether the control buttons are to be displayed.
Show new button	Yes	Defines whether the button „new“ is displayed.
Show copy button	Yes	Defines whether the button „copy“ is displayed.
Show delete button	Yes	Defines whether the button „delete“ is displayed.
Show edit button[1]	Yes	Defines whether the button „edit“ is displayed.
Use deletion confirmation	Yes	Defines whether a confirmation is displayed before a list entry is deleted.
List	Table	Defines how the list entries are represented. By default, each list generates a table. The serial representation shows all the list entries vertically in a row.

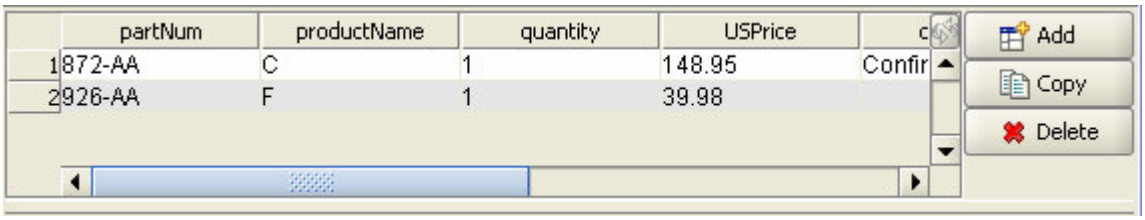


Figure 69: Display a list as table

The table above shows the standard view of a list element defined in the XML schema.
Now you can add, copy or delete new list entries. You can select the number of columns and their sequence, by specifying the column information (Line Infos).

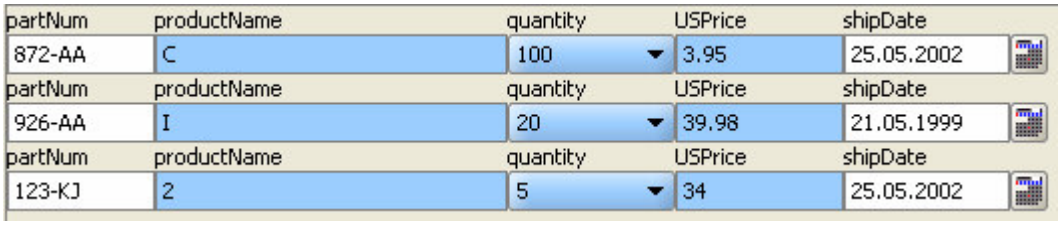


Figure 70: Display a list as serial view

You can define the layout for the Editing type of the list (master for a list entry, in this case „item“) (see section 3.4.2.4). The layout represents the entire view of the list entry in a row. Thus, the serial representation is more attractive and compact.

[1] The „Edit“ button is indicated only if the list runs in the "dialog" editing mode.

3.4.2.11 Table definition

Figure 71: Table definition for a list element

Fieldname	Default	Description
Direction	AS ROW	Defines whether the rows are represented vertically below each other (AS_ROW) or horizontally next to each other (AS_COLUMN).
Allow table flipping?	Yes	Defines whether, at run-time, the mode can be changed from row oriented to column oriented.
Position	north	Defines the position of the table (north, south, east, west).
Column header class		Defines the implementation class for the column heading. The following class is the standard one: : com.jaxfront.ui.model.PrimitiveRowHeaderLabellingStrategy
Edit Mode[1]		Defines how to edit list entries. In the dialog mode, the "editing" button is displayed, enabling you to edit the selected list entry over a dialog window.
Bounds (x,y,w,h)	0,0	Defines the position (x,y) of the dialog window relative to the current window. Also, it defines the exact size of the window (width, height in pixels).[2]
Button Sequence		Defines the sequence of both buttons „ok“ and „cancel“.

[1] The values of a simple list (consisting only of simple types) may be edited directly within the table (inbound editing). For more complex lists or your own column definitions, the detailed view of the selected node is displayed at the defined position (north, south, west east).

[2] With the indication of a comma-separate string, e.g., 20,20,300,500 , the dialogue window is opened at x=20, y=20 with width of 300 and a height of 500.

If nothing is specified (that means 0,0), the dialog window appears in standard size (preferredSize), centered relative to the current window.

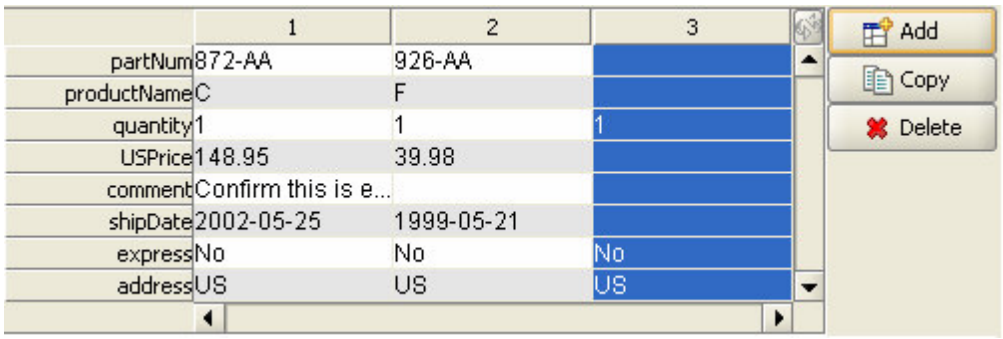


Figure 72: Display of the table in the mode: AS_COLUMN

To change the view at run time, click the button in the upper right corner of the table (if „Allow Table Flipping“ is activated).

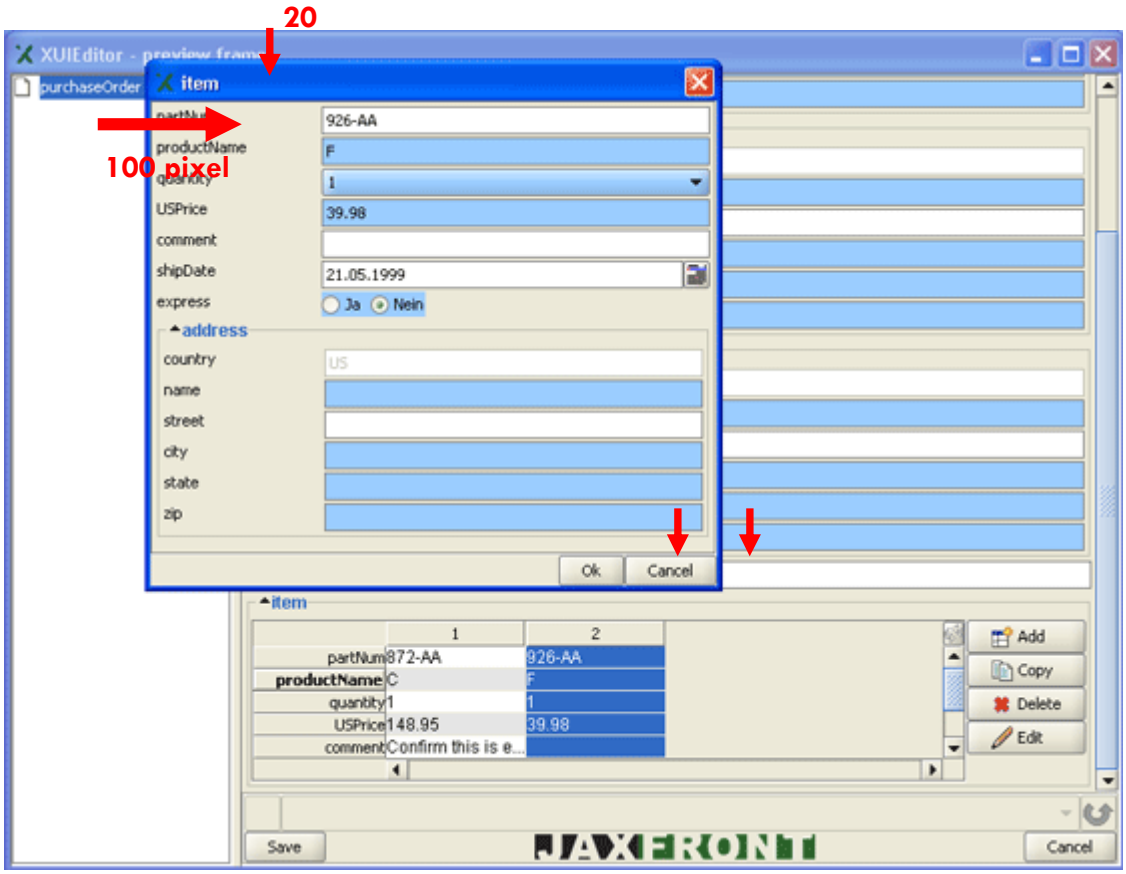


Figure 73: Table in the Edit Mode "Dialog"

In the previous example, the following settings were used:

- The table editing mode was set to „Dialog“.
- The Position of the dialog window was set to 100,20,400,250.

The window opens accurately at the coordinates x=100 y=20, and it has firmly defined width of 400 and a height of 250 pixels. The buttons „OK“ and „Cancel“ are aligned in the right corner of the dialog box.

3.4.2.12 Definition of own columns

To determine the number of columns and their sequence, use the Line Info list.

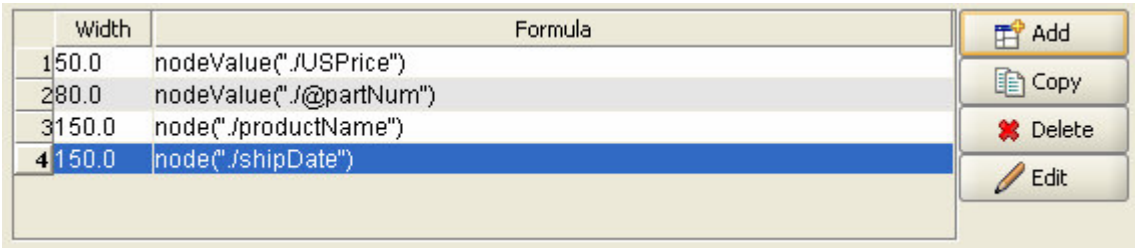


Figure 74: Definition of own columns

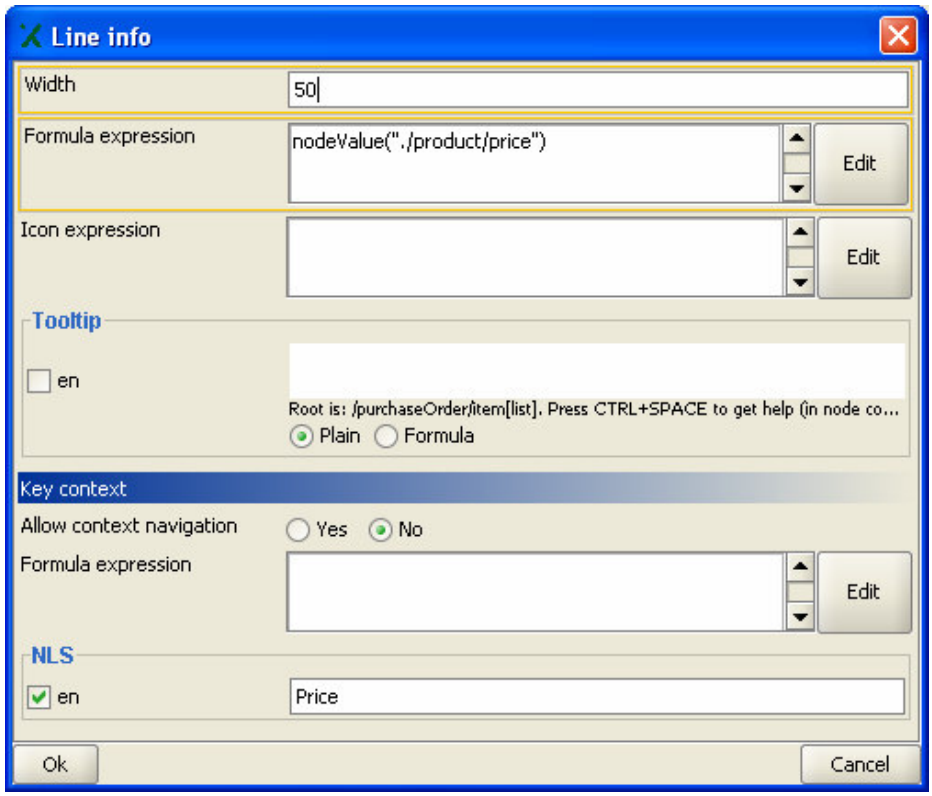


Figure 75: Definition of a single column

Fieldname	Default	Description
Width	100	Defines the column width in pixel.
Formula expression		Defines the content of this column over a formula expression.
Icon expression		Defines a formula expression that returns the icon name as a result. The icon is displayed above the name within the class path.
Allow context navigation	No	Defines whether context navigation for a key element (Key) is active. The context navigation permits to return to the place, where a key (Key) was selected as reference (KeyRef).
Formula Expression		Defines a formula expression for the display of the reference value (KeyRef) within the context cell.

Four different line information with varying column widths were created here. The first two are a simple output of a node value (function nodeValue). However, the third and the fourth line information refer to the entire node (function: node).

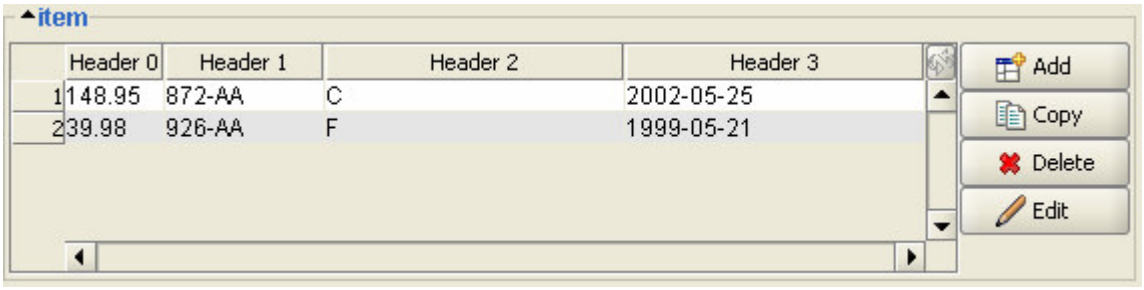


Figure 76: Table with own column definitions

The column headings „Header 0 – 3“can be translated into any language (see chapter 4).

Only the nodes referred with the function node (in this case, the column „Header 2“ and „Header 3“) may be edited in the table. The first two columns are not editable. They represent the value of a node only.

3.4.3 Behavior

This section describes how to define the component's behavior (in contrast to its visual representation).

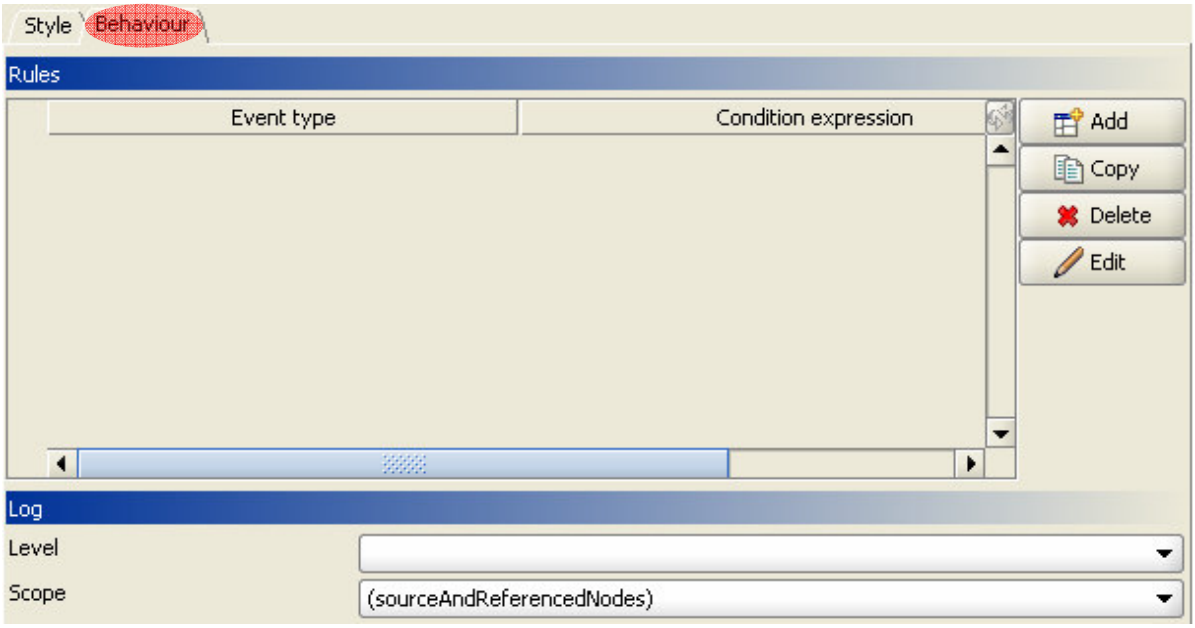


Figure 77: XUI component - Behavior

Fieldname	Default	Description
Rules		Defines a list of rules for this component. . .
Level		Defines the logging stage (1 or 2). These stages can be extended at will and are meant as context information for the application to be processed.
Scope	sourceAndReferencedNodes	Defines the operating range of the logging (see section 2.5.1.2)

3.4.3.1 Rules

A rule consists of an event, a condition and an action. For more details see chapter 2.5.

A rule may be defined in the following two ways:

- Create a rule in the current context (the source component of the XUI definition)
- Refer to a global rule.

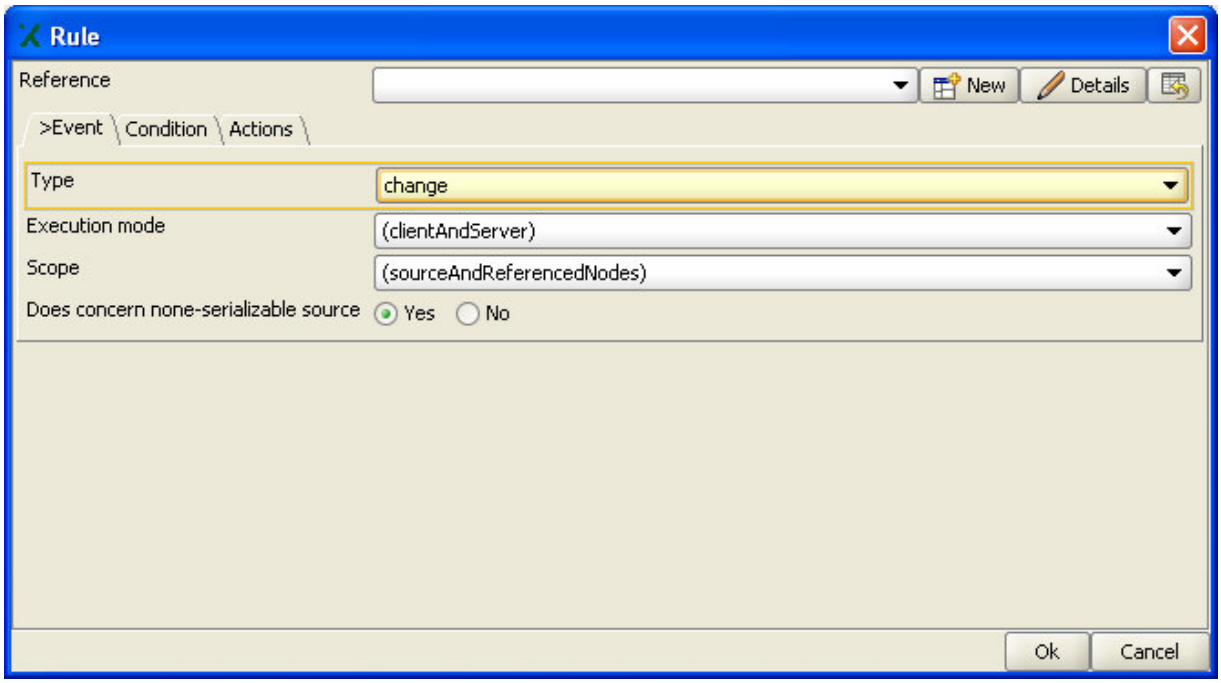


Figure 78: Definition of a rule

Fieldname	Default	Description
Reference		References a globally defined rule. In this case, the local context rule is faded out. In addition, new global rules can be created (using the New button), or the existing, already referenced rules can be modified (click the „Details“ button).

3.4.3.1.1 Event

An event defines the kind of trigger and the relevant range (scope) for the processing of the condition, and respectively all actions. A rule is processed in these circumstances:

- The occurrence of a change corresponds to the defined kind of event
- The occurrence of a change affects the defined scope.

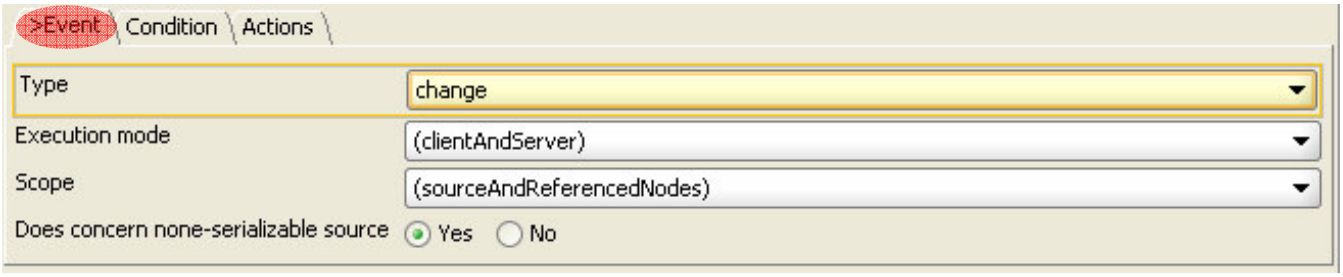


Figure 79: XUI component - Event

Fieldname	Default	Description
Type	change	Defines the type of event. See section 2.5.1.1.
Execution mode	clientAndServer	Defines if this rule should be executed on client only, server only or both.
Does concern none-serializable source [1]	Yes	Defines whether the rule is to be executed, if the underlying node is no longer serializable.
Scope	sourceAndReferencedNodes	With the specification of its own Java class, the condition itself can be implemented. The class must implement the condition interface. If a formula expression and its own class are indicated, both conditions must be fulfilled, in order that the actions be executed.

For example, to inform yourself about the entry of a list element or the selection from a choice (modeled Choice), you must select the following type and scope for the list element: Type=„structureChange (addNode)“ and Scope=„sourceOnly“.

[1] If this setting is deactivated, as soon as the underlying node is no more serialized the rule stops being executed. In this case, a previously indicated validating message is removed, regardless of its category (incomplete, hint, info, error).

3.4.3.1.2 Condition

A condition defines the formula expression for its evaluation, as well as data concerning a display of a possible error message. The actions are executed only when the condition is fulfilled.

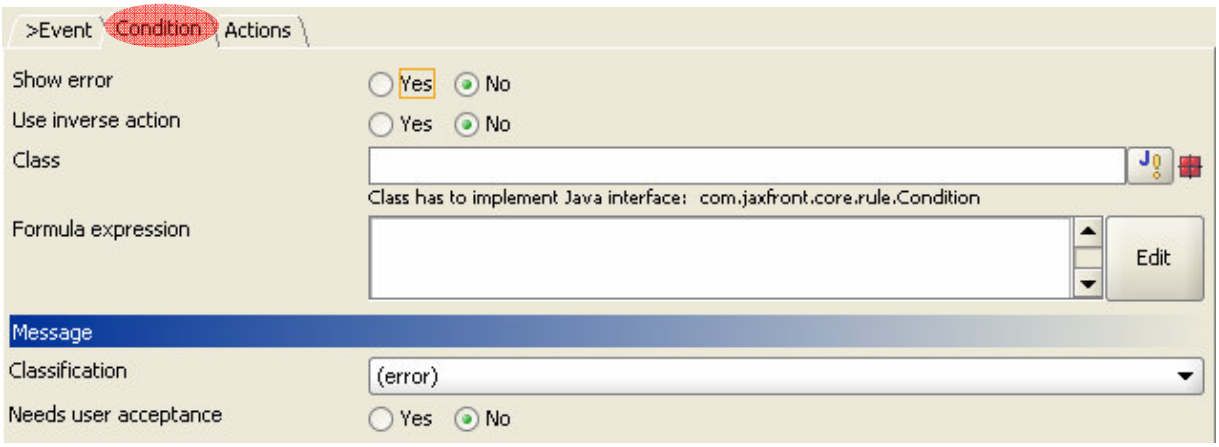


Figure 80: XUI component - Condition

Fieldname	Default	Description
Show error[1]	No	Defines whether an error message should be displayed when a formula expression occurs.
Use inverse action?[2]	No	Defines whether the reverse action was released for the non-incidence of the formula expression.
Class		With the indication of its own Java class, the condition can be implemented. The class must implement the Condition interface. If a formula expression and its own class are indicated, both conditions must be fulfilled, so that the actions are executed.
Formula expression		Defines the condition expression. This must return either Yes or No.
Classification	error	Defines the category of the error message.
Needs user acceptance	No	Defines whether the user confirms the message by a mouse-click.

If no formula expression is indicated, the condition is fulfilled and the actions are executed.

[1] An error message is displayed with a red frame (or the color specified for it) of the relevant component. In addition, the error message is displayed in the error list. If no error message was entered for this condition, the formula expression is indicated as text.

[2] The reverse action (inverseAction) must be declared when an action is defined. However, the reverse action need not be indicated if the action parameters of the method to be executed consist of Boolean values only. The parameters are automatically inverted in the case of reversal (flip).

3.4.3.1.3 Action

An action defines a change of the model or the user interface (visual event).

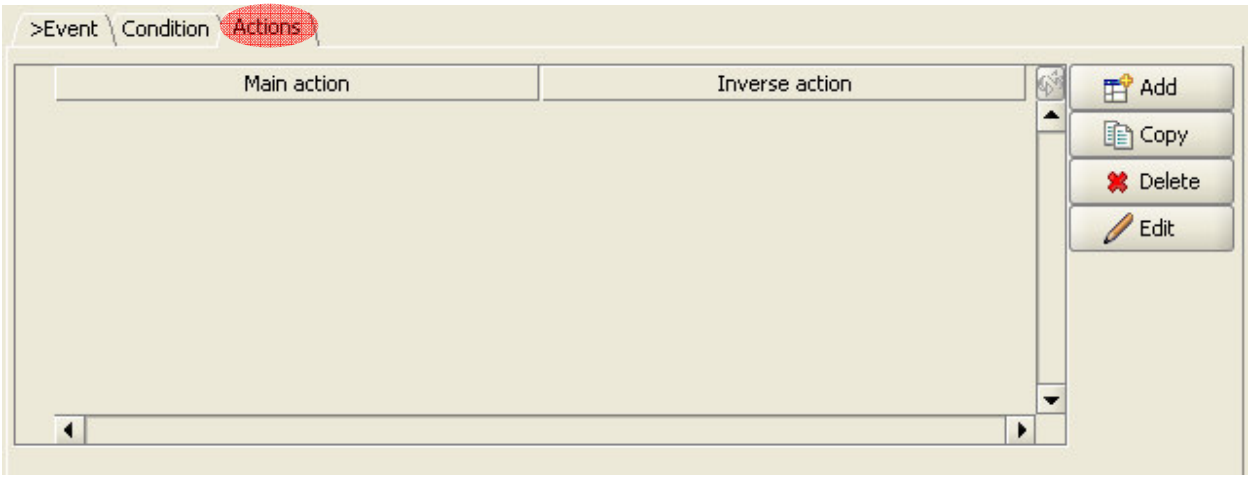


Figure 81: XUI component – Actions

The actions are sequentially released if the condition was positive on examination.

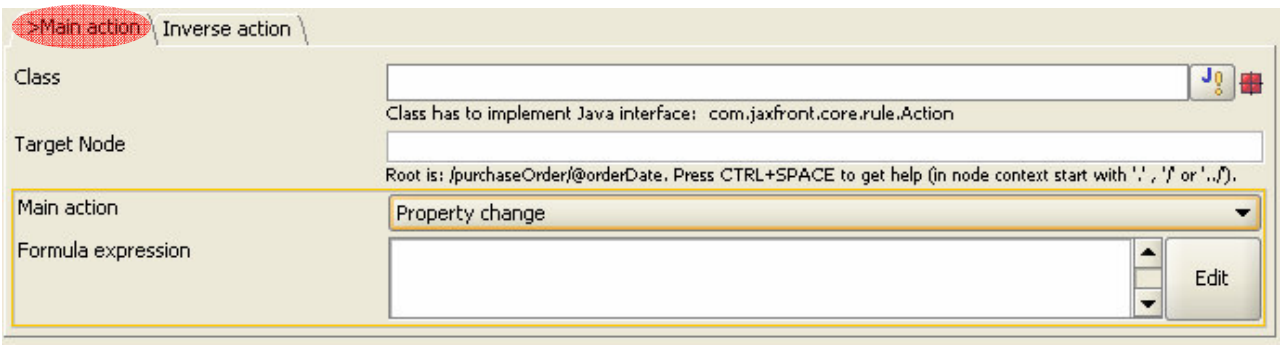


Figure 82: XUI component – Main Action

Fieldname	Default	Description
Class		With the indication of its own Java class, the condition can be implemented. The class must implement the Action interface. If a standard GUI action or a change of model is indicated, in combination with its own class, both actions are implemented. The own action implementation is executed last.
Target Node		Every action requires the indication of a target node, respectively target component. If no target is indicated, the action is executed on the node of the rule definition.
Main action		Determines the kind of the action: Model, GUI or general.
Formula expression		Defines the value to be set at the target node. Note: Thereby the target node must be a leaf (simple type).

If the target of an action refers to a component, which is represented by a Plugin, you can call all visible methods, which define a combination of the permitted parameter values (see section 2.5.3.1). The methods may be called over the Java Reflection API.

The GUI action causes a visual change in the user interface.

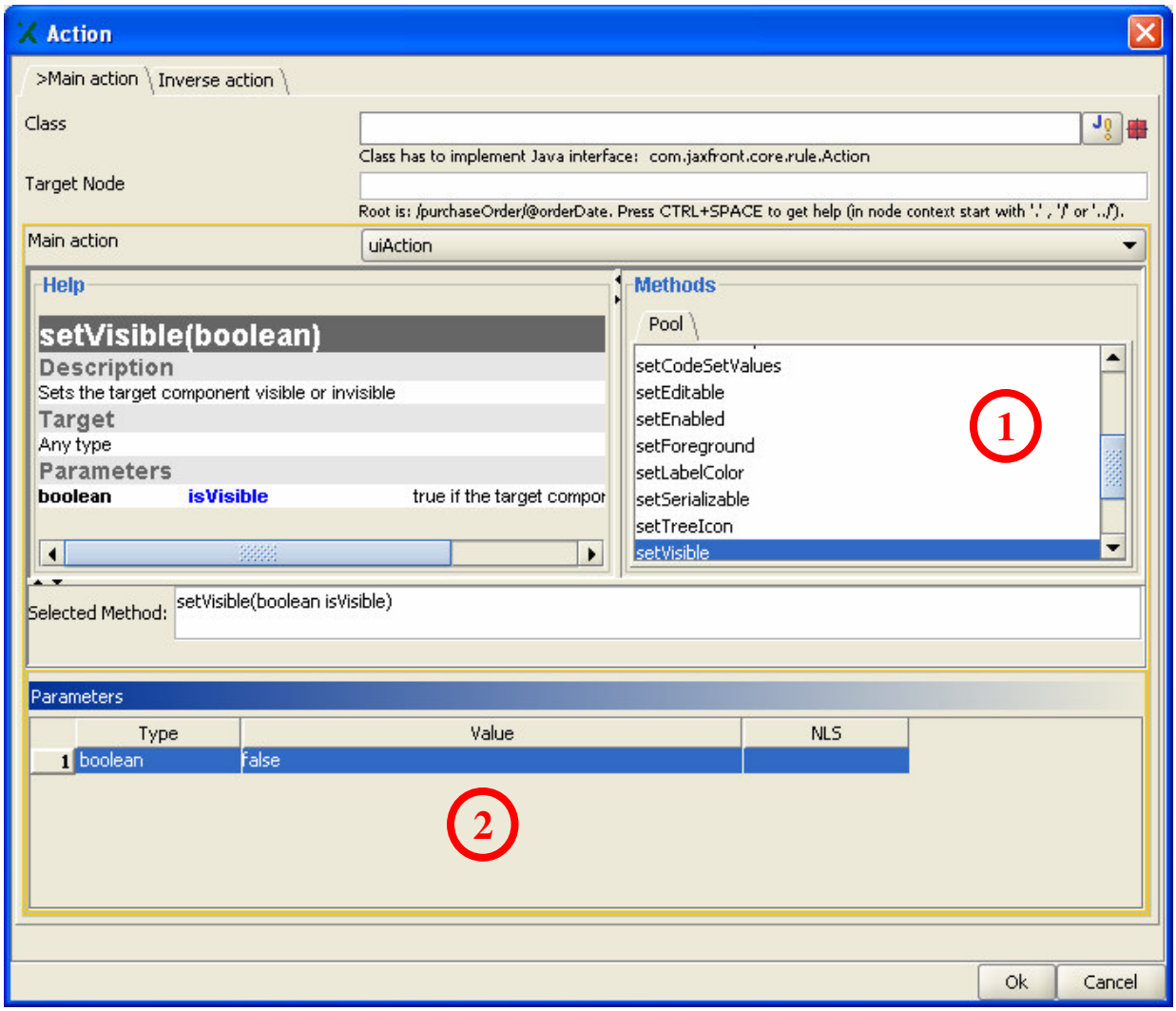


Figure 83: XUI component – GUI Action

You can double-click one of the specified Methods to select it and list its parameter values. See 1 in the above Figure. . Then, you can define a suitable value for each parameter. See 2, in the above Figure.

If a parameter is of the type „String“, you can enter a text for the language-dependent translation., into the „NLS“ field
Listing all standard GUI actions.

Name Inverse-Action Name	**Description**
addTreeNode **removeTreeNode**	Inserts the target node as a new tree entry. *Boolean* — *Defines whether the inserted node is to be serialized or not.*
jumpTo	Jumps to any target node and brings its visualizer to the front.
removeHint	Removes a previously inserted reference to the target node.
removeMessage	Removes a previously introduced message for the target node.
removeTreeNode	Removes a previously imported tree entry for the target node. *Boolean* — *Defines whether the node, which is being removed, is to be serialized or not. .*
setApplicationRequired	Sets the target node to be mandatory or optional, for the application. This changes the background color of the addressed input element. *Boolean* — *Defines whether the target node is mandatory (true) or optional (false).*
setBackground	Defines the background color for the target input element. *Color* — *RGB color value (e.g. 210,220,120)*
setChoosenKeyContextValue	Sets any context data relative to the referred context (provided XPath) of a Key-KeyRef relationship. *String* — *XPath expression relative to the current context of the target node and its Key-KeyRef relationship.* *String* — *Value to be newly set.*
setCodeSetValues	Sets a number of CodeSet values on the target node. *String* — *URL of the CodeSet.* *String* — *Name of the CodeSet.* *String[]* — *Codes to be set for the new selection (comma separated).* *String* — *Default selection value.*
setEditable	Make the target editable or not.
setEnabled	Locks or unlocks the input element of the target node. *Boolean* — *Lock or unlock.*
setForeground	Sets the text color for the input element of the target node. *Color* — *RGB color value (e.g. 210,220,120)*
setLabelColor	Sets the label color for the target input element. *Color* — *RGB color value (e.g. 210,220,120*
setSerializable	Defines whether the target node is to be serialized or not. *Boolean* — *Should the target node be serialzed or not.*

setTreeIcon	Defines the icon for the tree entry of the target node.	
	String	*Icon name of the tree entry (gif or jpg). Thereby, you can define the entire path indication or the icon name alone (resolution via class path).*
setVisible	Defines whether the target input element is to be indicated or not.	
	Boolean	*Should the target element be displayed or not.*
setVisibleAndSerialize	Defines whether the target input element should be displayed respectively serialized or not.	
	Boolean	*Should target element be displayed or not.*
	Boolean	*Should target node be serialized or not.*
showHint **removeHint**	Displays a dynamic reference on the target element.	
	*String**	*Reference text.*
	Boolean	*Defines whether the reference text concerns a formula or not.*
	Color	*RGB color value of the dynamic reference.*
	Boolean	*Defines whether the reference text should be displayed in bold font.*
showLabel	Controls the visibility of the target input label.	
	Boolean	*Defines whether the label is visible or not.*
showMessage **removeMessage**	Displays a message, which is freely selectable for the target node.	
	*String**	*Text of the message.*
	String	*Classification of the message* („error“, „incomplete“,„info“,„warning“,„hint“, or „question“)
	Boolean	*Defines whether the message is displayed as a dialog (true) or added to the error list (false).*
	Boolean	*Defines whether the end user should confirm the message, so that it disappears.*

* When these values are indicated, the text can be translated. .

3.4.3.2 Logging

All changes to an entire DOM (Document Object Model are logged. They logs may be viewed at any time over the Java API.

You can extend the logging levels according to your preferences. . The logging levels provide the context information for the application to be processed. For the operating range see *Operating range*, in this document.

3.5 PDF – Designer

This section describes the PDF Designer.

3.5.1 Introduction

In addition to a Java Swing and HTML Renderer, JAXFront offers a PDF-Renderer, which enables you to display documents in the Portable Document Format (PDF), as well as print them in the desired page layout.
The JAXFront PDF-Renderer is not included in the standard package and needs to be licensed as an additional component. If you do not have a licensed PDF-package and you are interested in it, please contact us. JAXFront will provide a demo package and examples to demonstrate the power of PDF-Renderer.

3.5.1.1 PDF-Renderer

The JaxFront PDF-Renderer is used for printing of XML-instance files The PDF-Renderer visually prepares the data contained in an XML document and permits no manipulation of the data.
The PDF-Renderer is based on the Open-Source-Framework iText and offers visualization of the XML data, which may be generic as well as adaptable by the user.

3.5.1.2 PDF-XUI

Data representation may be user specific. For this purpose, JAXFront provided a special XUI (`pdf.xui`) with its own XML schema (`pdf.xsd`).
The PDF-Renderer has its own XUI definition. This definition differs from the general XUI, due to the fact that only its global settings can be changed. Layout- and behavior specific definitions on component level are not available for the PDF-Renderer.
However, when the the PDF-Visualizer is produced, the general XUI definitions for the respective component are read and converted. The visual result is similar to the one produced by the Java Swing Renderer. Thus, you do not need to redefine the XUI definition for the component. You can achieve an attractive visual result with the help of global settings only.

3.5.2 Edit a PDF-XUI

If the PDF Renderer package was installed successfully, its PDF menu appears in the XUI-Editor.
Using the XUI_Editor, you can create, edit or delete the XUIs for the PDF Renderer. In addition, you can preview the resulting PDF files.

3.5.2.1 Create a new PDF-XUI

If no PDF XUI or general XUI exists and/or was not loaded, the PDF Creator Wizard dialog box appears.

In this dialog box, you can enter the file name of the new PDF XUI that you want to create.
The new XUI file is stored relative to the path of the currently loaded XML schema:

Figure 84: Creation of a new XUI for the PDF-Renderer

When a new PDF-XUI is created, an entry, which refers to this PDF-XUI, is set for the currently loaded general XUI in the footprint:

```
<?jaxfront version=1.42;time=2004-02-10 15:12:40.578;include-pdf=po_PDF.xui?>
```

The reference to a PDF-XUI is always stored in a general XUI. Therefore, ensure that this one is also stored. Otherwise the reference is lost when the current XUI is closed.

3.5.2.2 Save existing PDF-XUI

To save the currently loaded or newly created PDF-XUI, click the Save button in the PDF-Designer Editor.

Figure 85: Save current PDF-XUI

3.5.2.3 Reload existing PDF-XUI

You can reject the changes on a PDF-XUI and return to the originally stored file. To do so, use the Reload button.

Figure 86: Reload current PDF-XUI

3.5.2.4 Delete existing PDF-XUI

An existing PDF-XUI may be deleted. For example, delete an existing PDF-XUI if you want to create a new PDF-XUI.

To delete a PDF_XUI, use the Delete button.

Figure 87: Delete current PDF-XUI

3.5.3 Editor of the PDF-Designer

Using the PDF-Designer editing features, you can easily define user specific layout settings. The settings are discussed in more detail further in this document.

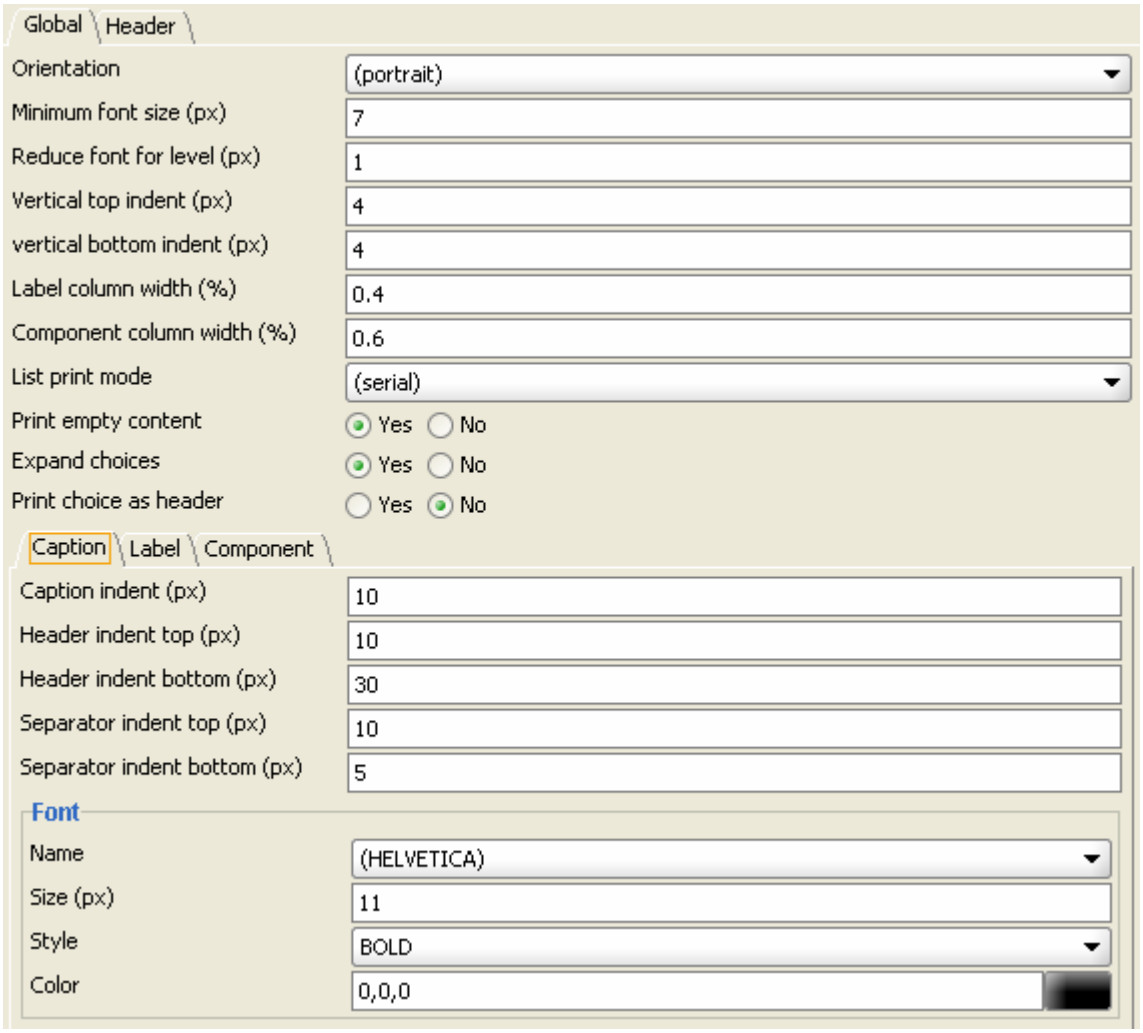


Figure 88: Editor of the PDF-Designer

3.5.4 Global settings

The global settings affect the entire PDF document to be generated. The global settings can be defined in the following two areas:

- Global document settings, such as page orientation
- Global settings for the PDF-text modules, such as headings, labels and components

3.5.4.1 Global document settings

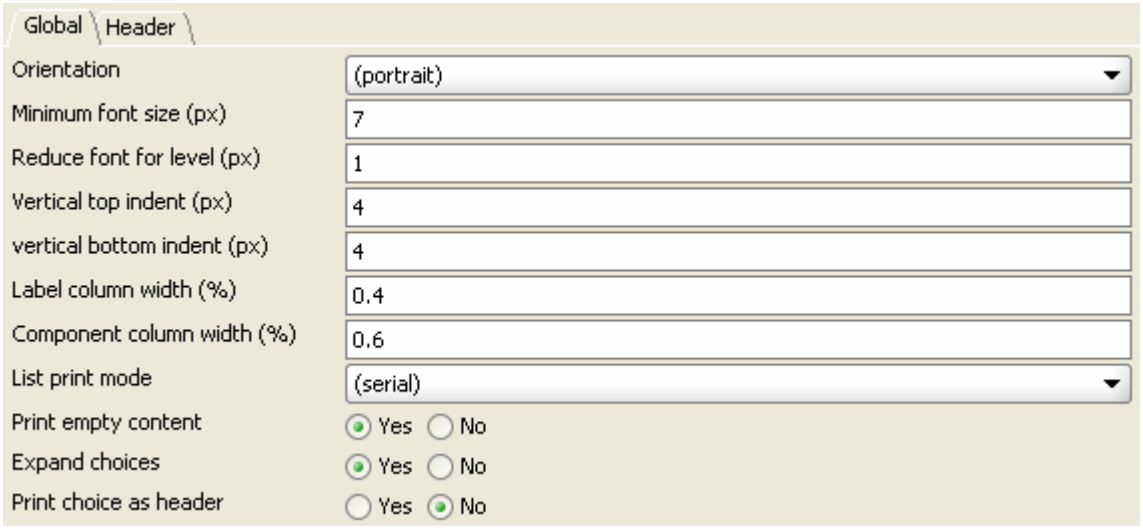


Figure 89: Global document settings

Fieldname	Default	Description
Orientation	portrait	Defines the orientation of the PDF-document to be created. Possible values are Portrait or Landscape.
Minimum font size	7	Defines the minimum font size for the PDF-document to be created. This parameter is used with list elements only. It reduces the font size for the nested headings, labels and components, by a certain pixel value, until this minimum font size is reached. See also the following parameter:„ Reduce font for level“).
Reduce font for level	1	This parameter defines the reduction of font size for headings, labels and components, displayed in a list. See also the following parameter::„Minimum font size“).
Vertical top indent	4	Defines the vertical spacing above each component.
Vertical bottom indent	4	Defines the vertical spacing below each component.
Label column width	0.4	Defines the width of the label in per cent. Pixel values are not supported.
Component column width	0.6	Defines the width of the components in per cent. Pixel values are not supported.

List print mode	serial	Defines how list elements should be displayed. Possible variations: **serial** (serial, vertical nesting of the individual list entries), **table** (table, in which the values of the list entries are registered) or **tableAndSerial** (combination of a table and serial representation).
Print empty content	No	Defines whether components, which do not have values, are to be displayed.
Expand choices	No	Defines whether Choices are to be displayed in their short form (String) or whether all child elements are to be displayed separately.
Print choice as header	No	Defines whether Choices should be displayed as headings or normal components. This parameter is useful if you need a compact expression.

3.5.4.2 Global settings for the PDF-text modules

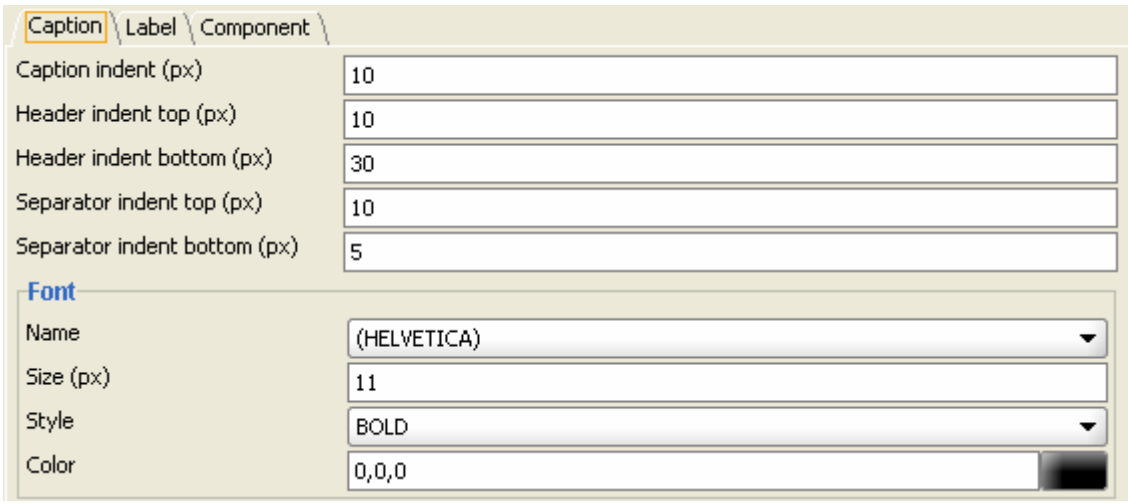


Figure 90: Global settings for headings, labels and components

Fieldname	Default	Description
Caption indent (px)	10	Defines the indent for the headings and/or labels in pixels. The defined indent is used per hierarchic level, if the heading and/or the label is visible. Otherwise, the indent is carried forward to the next visible heading and/or label. No indent can be defined for the components.
Header indent top (px)	10	Defines the vertical upper spacing for the heading titles.
Header indent bottom (px)	5	Defines the vertical lower spacing for the heading titles.
Separator indent top (px)	10	Defines the vertical upper spacing for the heading separators.
Separator indent bottom (px)	5	Defines the vertical lower spacing for the heading separators.

Name	HELVETICA	Defines the font names for the headings, labels or components. By default, the following fonts are available: • Helvetica • Courier • Times-Roman • Symbol • Zapf-Dingbats
Size	12	Defines font size for the headings, labels or components in pixels.
Style	NORMAL	Defines the style for the headings, labels or components. By default, the following styles are available: • Normal • Bold • Italic • Underline • StrikeThru • Bold-Italic (Bold and Italic)
Color	0,0,0 (black)	Defines the colors for the headings, labels or components as RGB values.

3.5.5 Header and footer

In your PDF-XUI you can define a header and footer for the PDF-document which you are about to create. The header and footer will be shown on each page.

3.5.5.1 Header

To define a header, use the Layout-Editor. For more details on the Layout-Editor, see the Layout section in this document.

The Layout-Editor, with which the header can be created, is similar to the Layout-Editor for the Java Swing Renderer. However the following restrictions have been identified:

Restriction	Details
Formula restriction	No Drag&Drop of elements from the XML schema navigation tree (left), references to field values can be made by means of formula
Classic separator only	No support for the heading separator modern (only classic)
No button definitions	Buttons cannot be defined.

Simple layout only	Only simple layouts may be defined, as the following dependencies exist on the PDF Renderer: • Column sizes are effective in percentages only • Row heights are effective in pixels only (otherwise Preferred-Size) • Alignment „fill“ is not supported. Instead, the component is centered (center) • Pictures are not scaled, but displayed in their original size The WYSIWYG conformance is not always achieved.

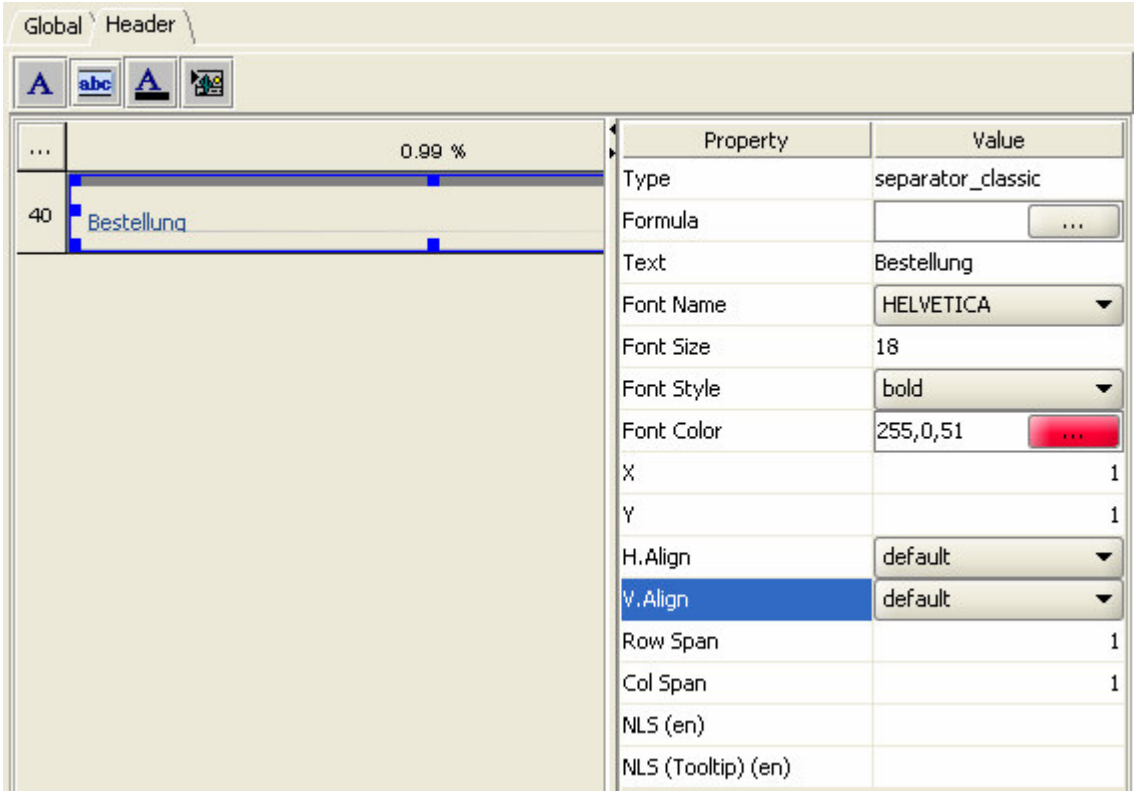


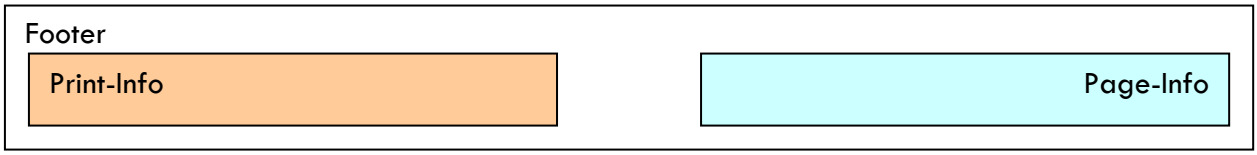
Figure 91: Table-Layout for Header

A header in a resulting PDF document may look like this:

Figure 92: Sample header

3.5.5.2 Footer

The footer layout may not be defined as a table, due to the complexity of the PDF-Rendering process. Therefore, a fixed footer was programmed as a Best-Effort-Solution.
The programmed footer consists of the following parts:



The Print-Info displays the following line by default:

Printed on: <**DATE**> <**TIME**>

The Page-Info displays the following line by default:

Page <**CURRENT PAGE**> of <**TOTAL PAGES**>

The default-footer may look like this:

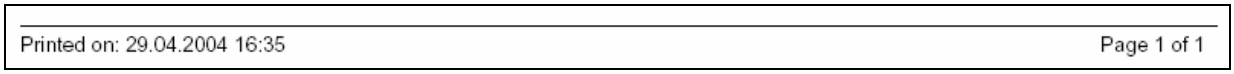


Figure 93: Sample footer

The footer may be customized in the following two ways:

- **Change Print-Info- and Page-Info-Text.** For example, do so to translate the text into another language.
  To do this, change the following entries in the respective NLS-file (for example, change xui_de.nls for the German language):

Fieldname	Description
pdf_print_info_printed_on	Placeholder for „printed on:“
pdf_page_info_page	Placeholder for „page“
pdf_page_info_of	Placeholder for “of”

- **Reprogram the footer**

  If you want to make extensive changes to the footer, reprogram it. For more details, see the PDF-Renderer section in the *JAXFront Developer* manual.

3.6 Configuration

The following sections describe how to modify the basic settings of JAXFront and the XUI-Editor, to adapt them to your needs.

You need to restart the application, in order for the changes to take effect. The functions, plugins and actions are stored in the folder “config” in your XUI Editor installation directory.

3.6.1 General settings (Preferences)

To modify the general setting at the Preferences tab:

1. From the View menu, select Preferences. The JAXFront Preferences dialog box appears. The Preferences tab is displayed by default.
2. Make the changes. See the table below, for more details.
3. When you have made the changes, click the Ok button. The preferences are now stored.
4. Restart the application, in order for the new preferences to take effect.

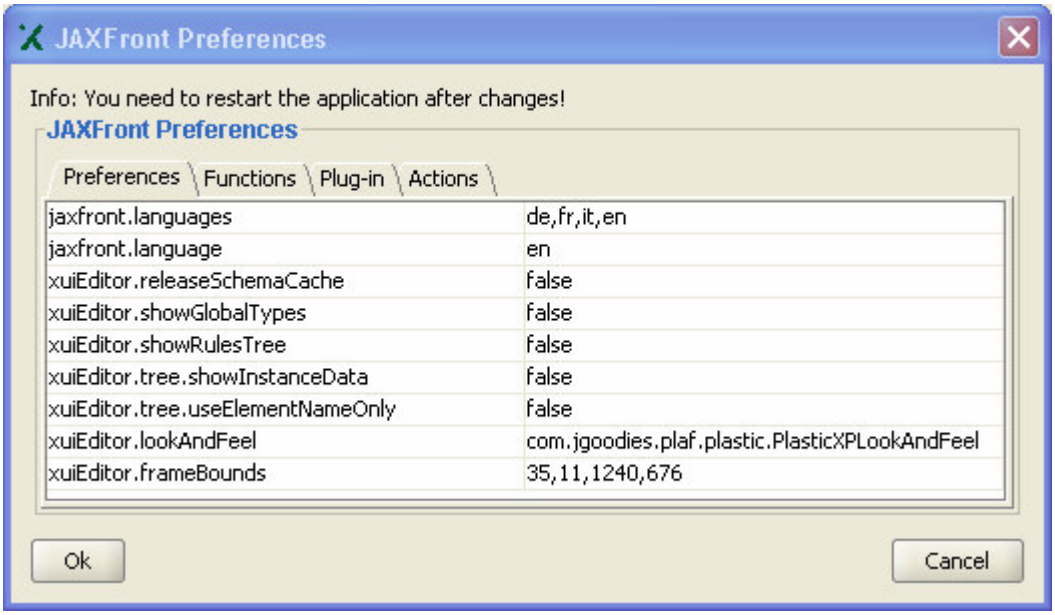


Figure 94: Global JAXFront-settings

Fieldname	Default	Description
jaxfront.languages	de,fr,it,en	Defines, which languages JAXFront supports and/or processes for the translation (National Language Support).
jaxfront.language	de	Defines the default language used by JAXFront and the XUI-Editor.
xuiEditor.showGlobalTypes	false	Defines whether the global types should be displayed in the XUI-Editor as a tree or not. This preference may also be set at a menu level. From the View menu, select *Display global types*.
xuiEditor.showRulesTree	false	Defines whether the global rules in the XUI-Editor should be displayed as a tree or not. This preference may also be set at a menu level. From the View menu, select *Display global rules*.
xuiEditor.tree.showInstanceData	false	Defines whether the XML schema navigation tree in the XUI-Editor should display data of the loaded XML instance or not. Effective only if an XML instance is loaded.
xuiEditor.tree.useElementNameOnly	false	Defines whether the XML schema navigation tree in the XUI-Editor should only display the element names according to the XML schema or whether possibly translated texts for the tree nodes should be displayed.
xuiEditor.lookAndFeel	User-dependent	Defines the Look & Feel of the XUI -Editor. This preference may also be set at a menu level. From the View menu, select *Look & Feel*.
xuiEditor.frameBounds	User-dependent	Defines the relative X/Y positioning to the left upper screen corner and the size of the XUI-Editor window. The syntax consists of: x-position, y-position, width, and height. If you change the window size of the XUI-Editor, the new coordinates are automatically stored in this field and retrieved for the next load.

3.6.2 Functions

You can edit JAXFront functions using the Functions tab in the JAXFront Preferences. The following editing functions for lists are available: Add, Copy, Delete and Edit.

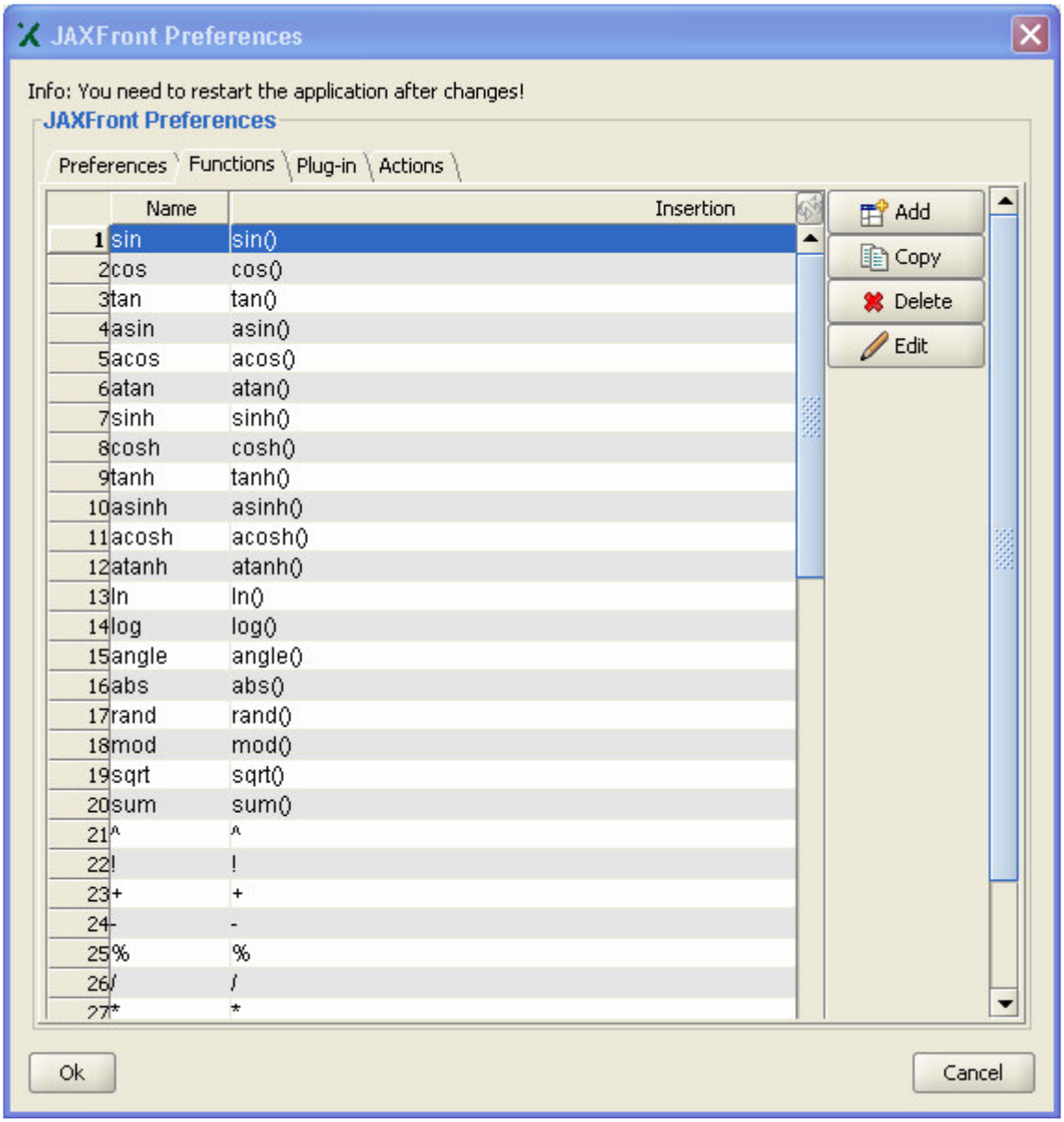


Figure 95: Functions for the Formula Editor

The implementation of a new function occurs in two stages:

- Implementation of the new function as Java class in the package `com.jaxfront.jep`. For more details, see the JAXFront Developer manual. .
- Adding and describing a new function in the JAXFront preferences. See the following figure.

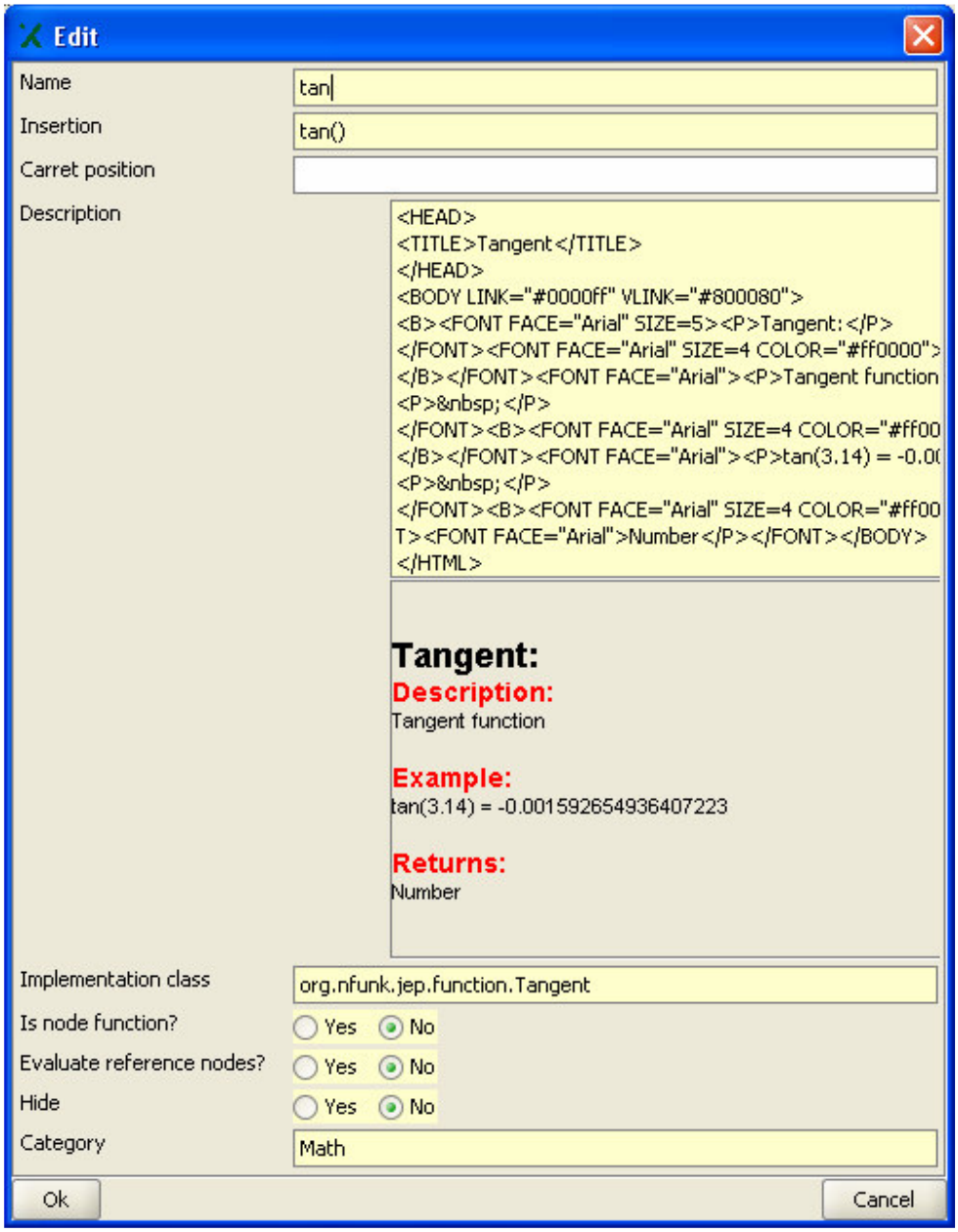


Figure 96: Editing a function

Fieldname	Description
Name	Defines the function name.
Insertion	Defines which text is inserted in the Formula Editor when CTRL+Space is pressed on the keyboard.
Caret position	Defines the position where the function parameter values can be entered. This is important for the navigation aid of the formula editor (CTRL+Space).
Description	In this field, the function description is inserted. This description then appears in the Formula Editor. It is defined as an HTML page.
Implementation class	Class which implement this functions (must implement the class org.nfunk.jep.function.PostfixMathCommandI).
Is node function?	Indicates if this function is a node (xpath) function.
Hide	Defines if functions should be hidden.
Category	Category of this function (each category is displayed in a tab).

3.6.3 PlugIns

The PlugIns used by JAXFront are listed in the JAXFront Preferences, under the Plug-in tab. They are available in the PlugIn Editor for a selected component. See also the section 3.4.2.5 PlugIn.

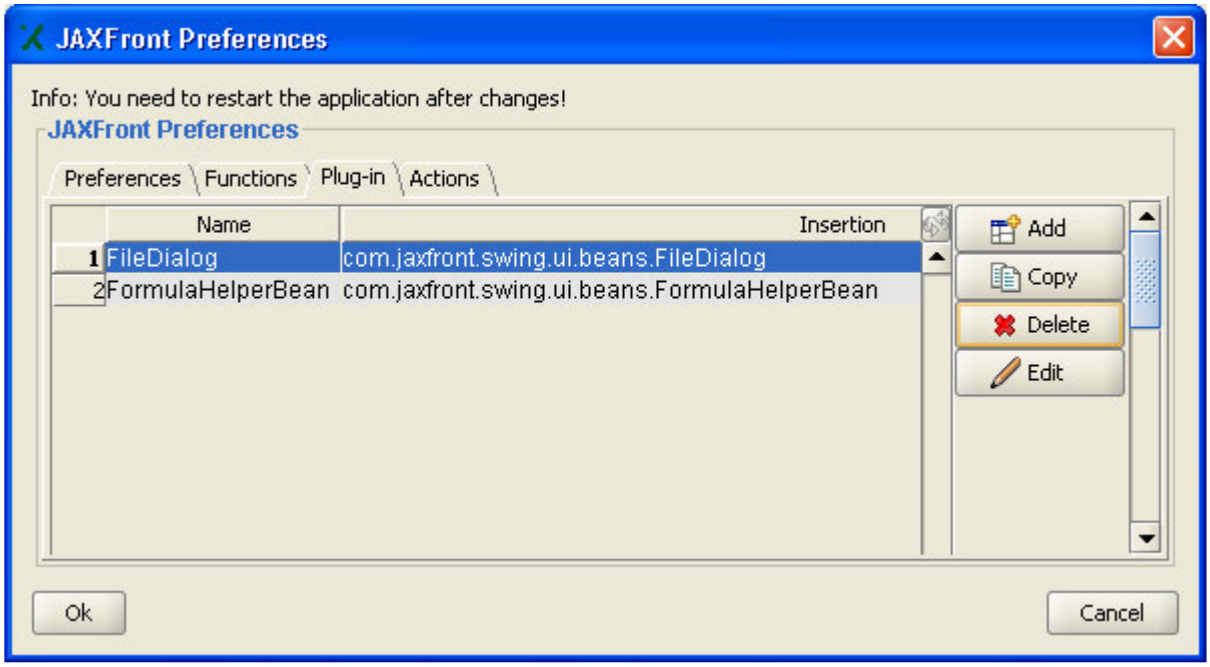


Figure 97: Overview of the PlugIns, used by JAXFront

The implementation of a new PlugIn occurs in two stages:

- Implementation of the new PlugIn as a PlugIn-class in any package, e.g., `com.jaxfront.ui.beans`. For more details, see the Programming Manual.
- Addition and description of the new PlugIn in the JAXFront Preferences. See the following figure.

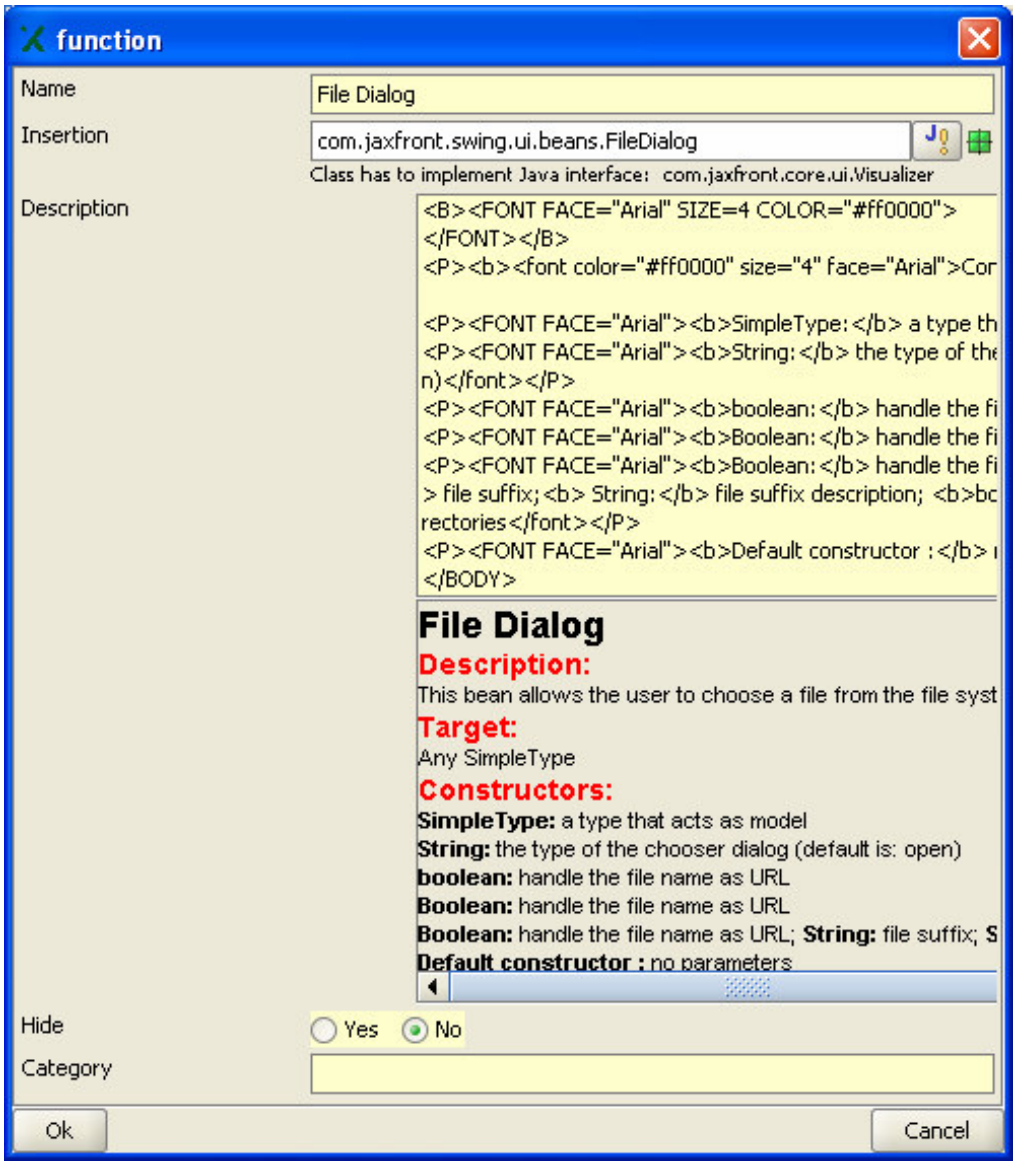


Figure 98: Editing a PlugIn-definition

Fieldname	Description
Name	Defines the PlugIn name.
Insertion	Defines which text is inserted into the PlugIn Editor as a class and addressed as a PlugIn-class.
Description	Describes the PlugIn. This description then appears in the PlugIn Editor, defined as an HTML page.
Hide	Defines if plugin should be hidden.
Category	Category of this plugin (each category is displayed in a tab).

3.6.4 Actions

In the JAXFront Preferences, you can also edit the GUI actions, used for the rules. The GUI actions cause visual changes to the interface. For details see the section 3.4.3.1.3 Action.

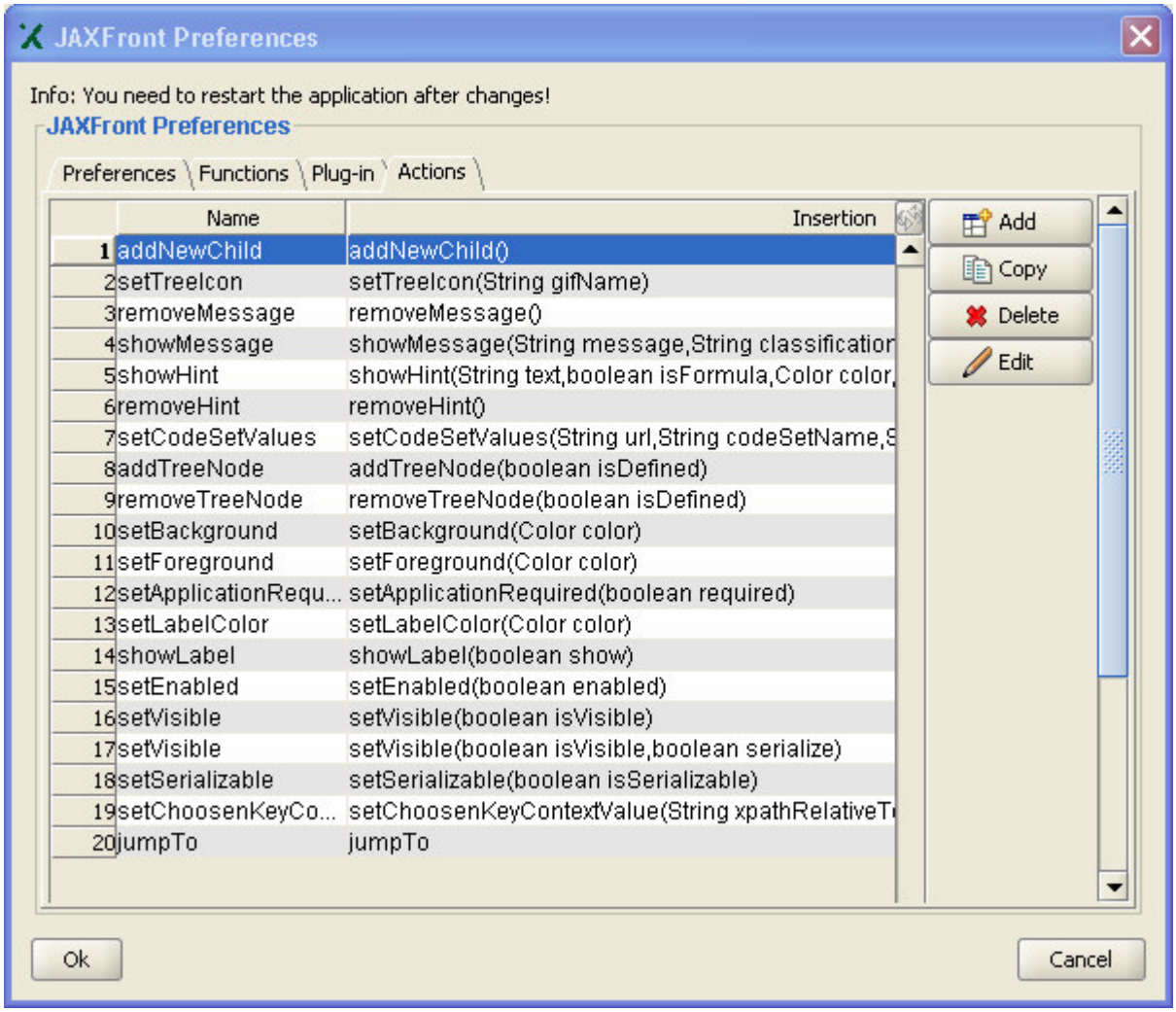


Figure 99: Overview of the GUI-Actions

The implementation of a new GUI-Action occurs in two stages:

- Implementation of a new method within a Visualizer-class (for example, AbstractView). For more details, see the *Programming Manual*.
- Addition and description of the new method as a GUI-action in the JAXFront Preferences. See the following figure.

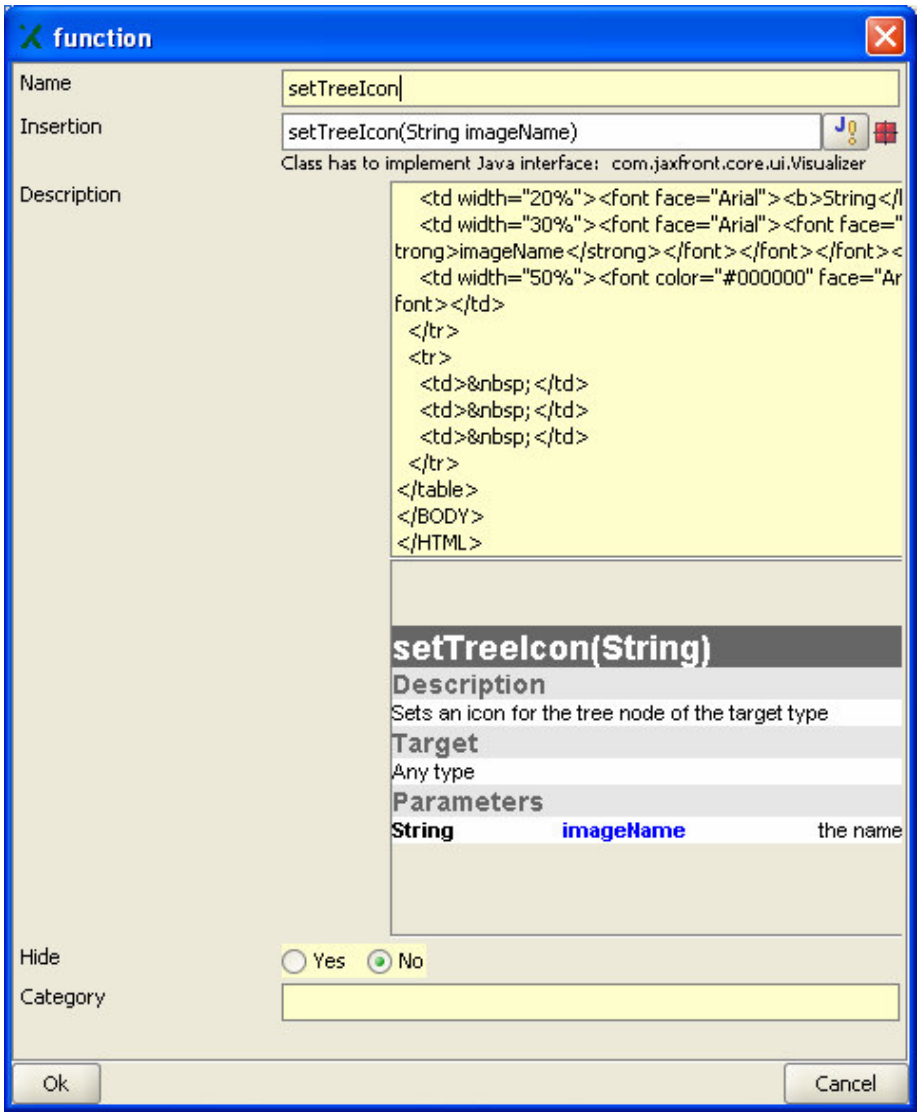


Figure 100: Edit a definition for a GUI-Action

Fieldname	Description
Name	Defines the name of the GUI action.
Insertion	Defines which method is called, that is valid for the Visualizer class of the current type.
Description	Describes the GUI action. This description then appears in the Editor for the selection of an action. It is defined as an HTML page.
Hide	Defines if action should be hidden.
Category	Category of this action (each category is displayed in a tab).

4 National Language Support (NLS)

In order to provide GUIs in different languages, JAXFront implemented the National Language Support (NLS) system. This NLS system can administer as many languages as desired per GUI, and activate them if necessary.
All language resources are stored in the „Property Files“. The file names must use the following syntax:

<*XUIName*> +“_“+ <*LanguagueCode*>+“.nls“

In addition, the file names must be accessible via class path, so that JAXFront can recognize them as a language definition. Alternatively to the class path, you can store the files in the same directory in which the XUI definitions are stored.

The translation creation, import and export always occur in the context of the current XUI definition.

4.1 Create NLS

By default, JAXFront uses the node names of the XML schema definition for the visualization of the schema nodes.
JAXFront offers the following two options to adapt a GUI to a certain language:

- Input over the Type Editor dialog
- Input over the NLS Preview

Input over the Type Editor dialog

You can define a translation for each XUI-Element, using the Editor area (see 3.2). Other elements, dependent on the respective element type (SimpleType, ListType, ComplexType, etc.), are available for the translation (see table below).

Name [1]	ST	STL	SG	SG*	SGL	CG	CG*	CGL
Tree entry	x	x	x	x	x	x	x	x
Heading	x	x	x	x	x	x	x	x
Leaf name	x							
Choice elements				x			x	
Lineinfo		x			x			x
TableLayout			x			x		
Error message	x							
Help text	x	x	x	x	x	x	x	x

[1]

A typical NLS entry is structured as follows:

- Translation text to be displayed
- Optional Tooltip text to appear if the mouse pointer remains on an element for a longer time.

The structure described above applies to tree entries, headings, leaf names and elements which can be used in a Table-Layout representation. In the following Figure, some typical NLS dialogues, which follow this schema, are specified.

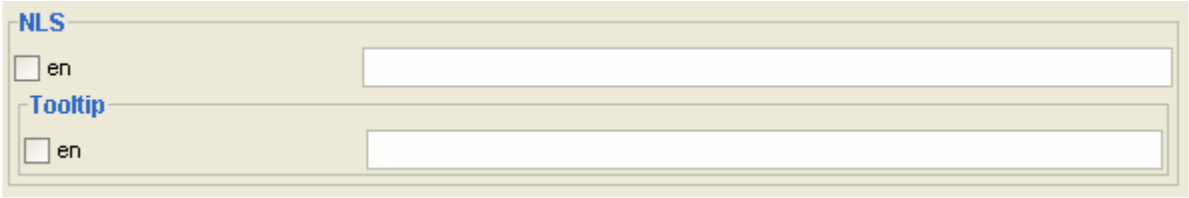


Figure 101: Typical NLS dialog

[1] [1] ST = SimpleType, STL = SimpleTypeList, SG = SimpleGroup,SG* = SimpleGroup as Choice, SGL = SimpleGroupList, CG = ComplexGroup, CG* = CompleGroup as Choice, CGL = ComplexGroupList

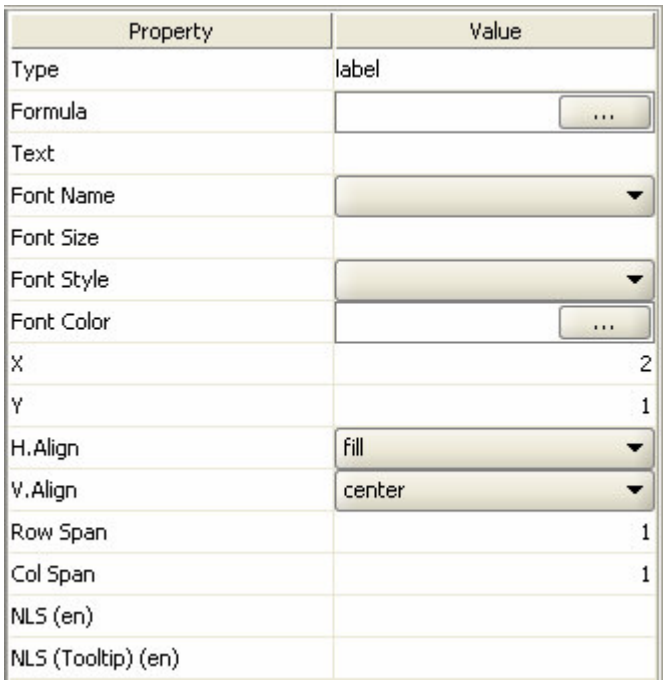


Figure 102: NLS dialog in the Table Layout

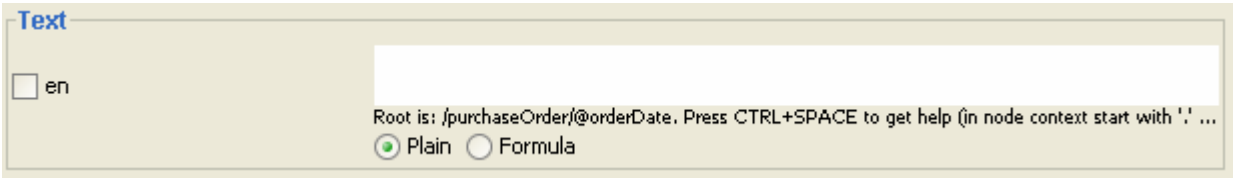


Figure 103: NLS input field for an error message (screenshot)

For a few elements, you cannot define a Tooltip text, and you need to use the special NLS dialogs. The elements without a Tooltip are:

- Elements within a selection list (Choice Type)
- Help texts
- Column/row names (LineInfos) of tables

The following figures provide an example of each element without a Tooltip:

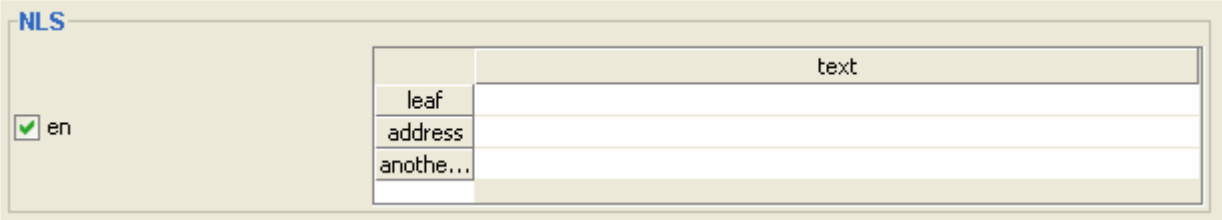


Figure 104: NLS for selection lists

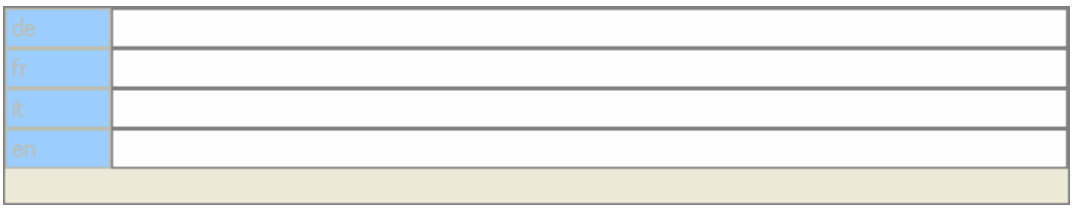


Figure 105: NLS anchor points for Help text

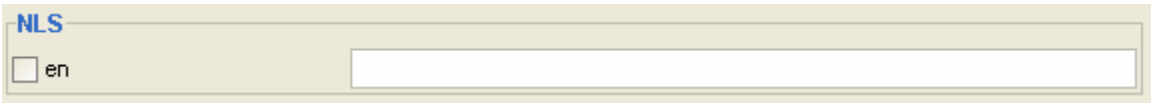
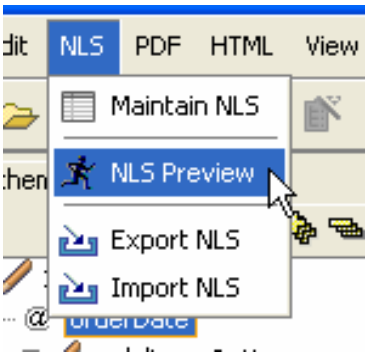


Figure 106: NLS- input element for LineInfos

Input over the NLS Preview

In principle, the structure of the NLS texts is similar to the one described in the previous section (Input over the Type Editor dialog). The difference lies in the presentation of the NLS dialogs and their distributed appearance within the Editor area.

To start the NLS preview, select the *NLS preview option from the* NLS menu.



In this GUI, all the elements for which a translation is possible have a yellow border and an *NLS Button.*

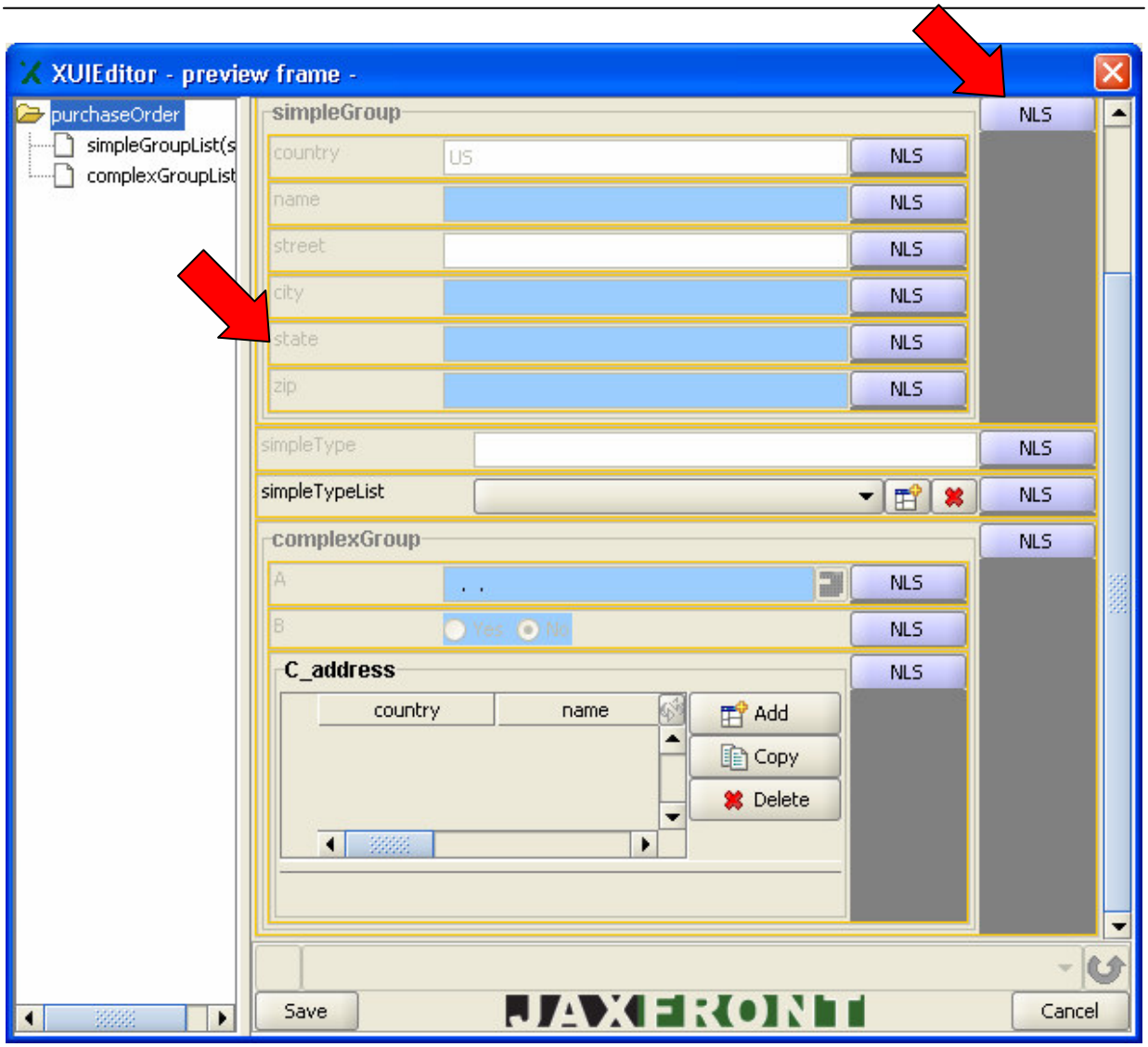


Figure 107: NLS Editor

If you click an NLS button, a corresponding *NLS dialogue* appears, depending on the element type.

These NLS dialogues offer all the elements for which a translation can be created in this context (see figure below). This differs from the previous method (Input over the Type Editor dialog).

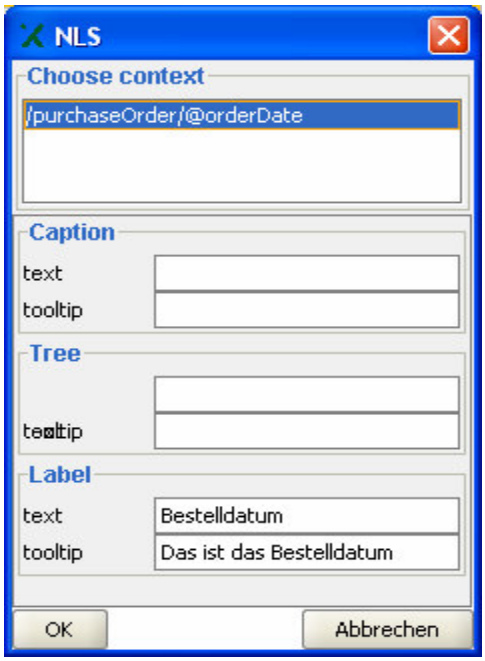


Figure 108: NLS-Preview-Dialog for SimpleTypes

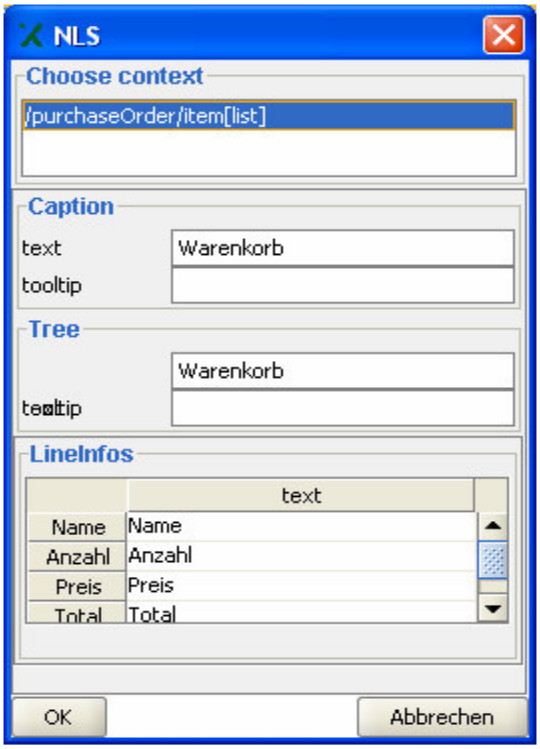


Figure 109: NLS-Preview-Dialog for ListTypes

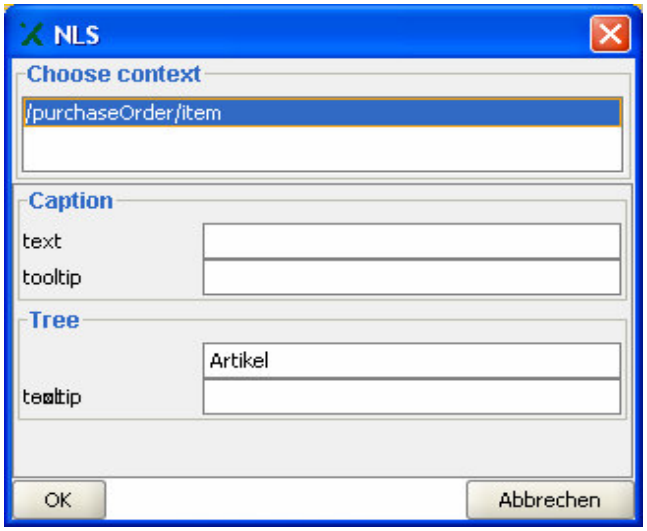


Figure 110: NLS-Preview-Dialog for ComplexTypes

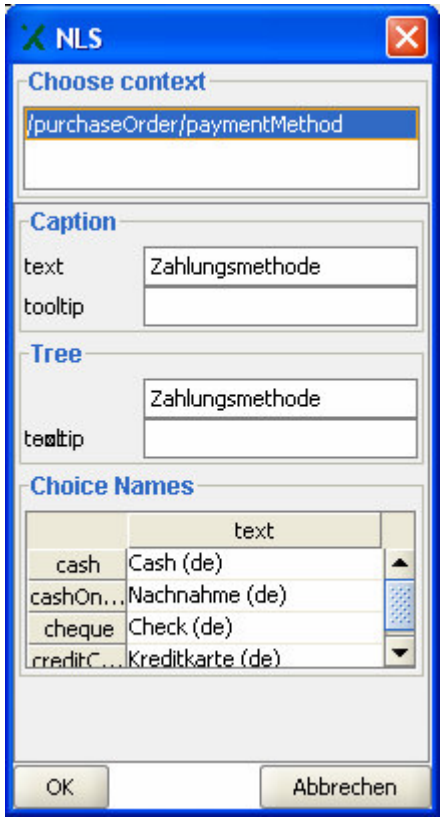


Figure 111: NLS-Preview-Dialog for Choices

The advantage of this preview is the proximity to the GUI in which it is used. You can quickly navigate between the individual translations.

A translation can be made for elements that were not defined via Plugin or TableLayout (see the figure below).

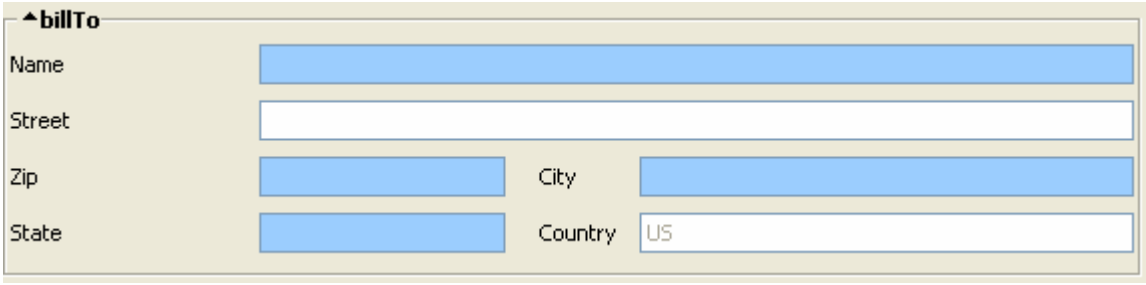


Figure 112: NLS within a TableLayout

4.2 NLS administration

JAXFront provides the basic functions that enable you to manage your NLS-definitions. You can export translations to a file, as well as import them again. In addition, you can use external, third-party programs, such as Microsoft Excel, OpenOffice, or Microsoft Access to create and/or modify your NLS-definitions.

4.2.1 NLS export

To start the NLS export, select the *NLS export* option from the NLS menu.

When NLS definitions are exported, the stored translations are saved in the CSV-format (comma separated file).
The CSV file has a certain structure.

In the beginning of the file, all the languages for which the GUI should be supported are specified. See the example (a). All the elements for which a translation text was already provided follow.


Empty translation elements and/or elements not translated within the XUI-Editor are not exported.

All the entries shown below have the following form (see also example b):

Reference (XPath)	**+**	**Context**	**=**	**de**	**+**	**en**	**+**	**…**
/purchaseOrder/comment	_	label	;	Kommentar	;	comment	;	…
/purchaseOrder/billTo	_	treeEntry	;	Rechnungsadr.	;	billto	;	…

Reference (XPath):
The XPath refers to a JAXFront-type for which a translation text was produced. However, multiple types existing on an element may be translated. Therefore, you must precisely define which element is concerned. The context serves for this purpose

The following contexts are possible:

Name	Meaning	Syntax
treeEntry	Tree entry	_ treeEntry
	Tool tip	_ treeEntry_tooltip
caption	Caption	_caption
	Tool tip	_caption_tooltip
leaf	Leaf	_label
	Tool tip	_ label_tooltip
lineInfo	LineInfo	_lineInfo_<*index*>[1]
tableLayout	Table-layout	_ tableLayout_<*X*>,<*Y*>,<*rows*>,<*cols*>[2]
condition	Condition	_ condition

[1] Index of the column
[2] X,Y = starting cell; rows,cols= Dimension of the covered rows, columns

Extracted from an export file:

```
id;de;fr;it;en a)
/purchaseOrder/express_label;Soll express geliefert werden?;<n.a.>;<n.a.>;<n.a.> b)
/purchaseOrder/comment_label;Kommentar;<n.a.>;<n.a.>;<n.a.>
/purchaseOrder/billTo_treeEntry;Rechnungsadresse;<n.a.>;<n.a.>;<n.a.>
/purchaseOrder/item_tableLayout_1,6,4,6;Lieferort/Datum;<n.a.>;<n.a.>;<n.a.>
```

Translation elements, which do not possess values, are exported as <n.a.>.

4.2.2 NLS import

To start the NLS-import, select the *NLS import* option from the NLS menu.

When importing NLS-files, consider the following points:

1. No empty elements may exist! Elements without a value must be stored as <n.a.>.
2. For each language, the import mechanism creates its own language-property-file. Thus, the existing data is overwritten.
3. A file locked by a process is saved in the „c:\temp“ directory. The file can be manually copied into the appropriate directory.

4.2.3 Edit NLS

To ensure a review of the translated components, select the Maintain NLS from the NLS menu. The *NLS administration* table is shown.
The first column shows the reference (XPath) (see 4.2.1) of the translated component. The second column shows the corresponding text. You can modify the text as desired.

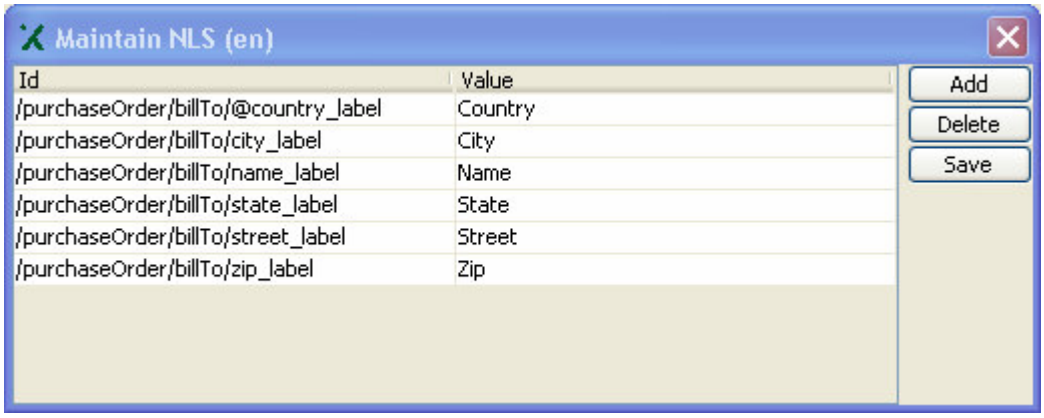


Figure 113: NLS administration

4.2.4 NLS example: Translation of a simple label

The label of the node "*email*" should no longer appear with the text „*email*" but with the modified text „*E-mail address*".

The NLS functionality is not available without a loaded XUI!

Method 1:

1. Start the NLS-preview.
   NLS -> NLS-preview -> de
2. Click the NLS button to the right of the „email"-entry.

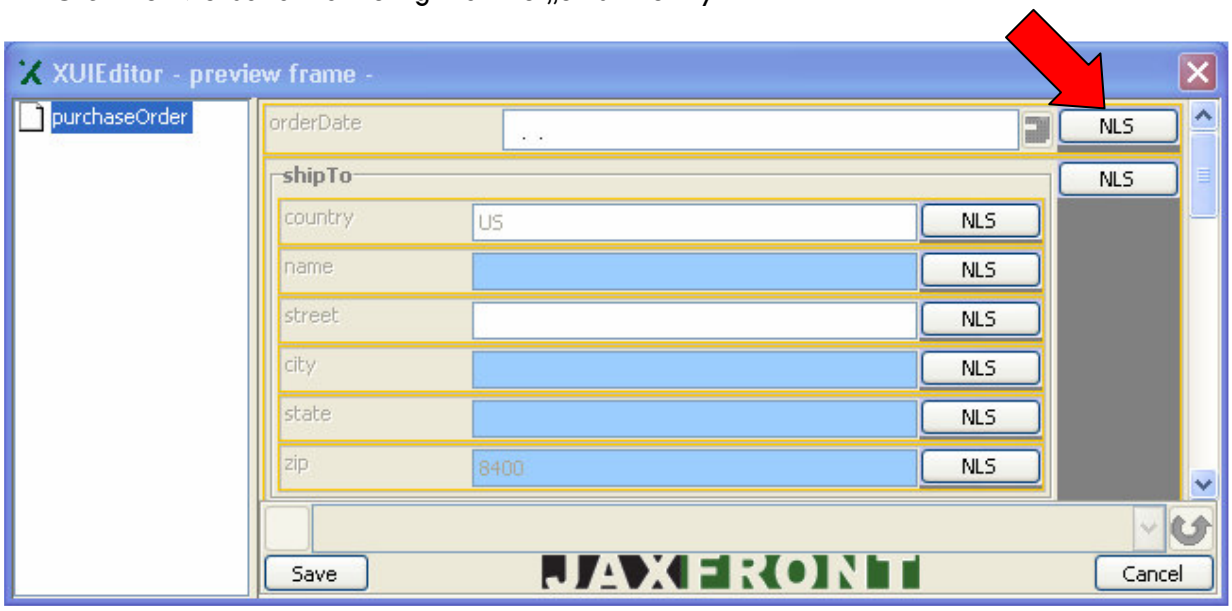


Figure 114: NLS-Editor for simple fields (screenshot)

The NLS Editor dialog for simple types appears.

3. Enter the new text ("E-mail address"), under Label->Text" . Optionally, you can also enter a Tooltip->Text.

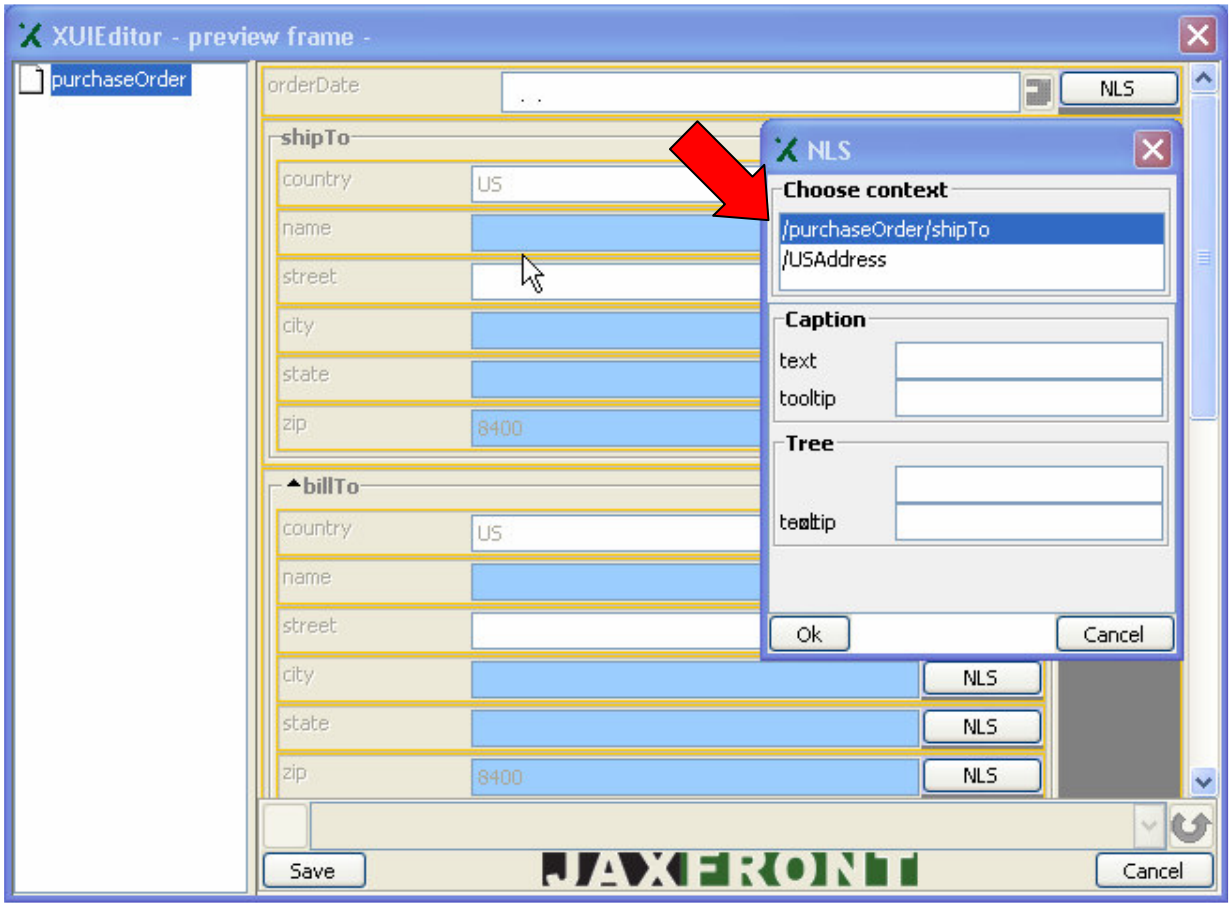


Figure 115: NLS-input-dialogue for simple fields

4. Click OK to confirm the change you made.
5. Click the X button in the upper right corner of the NLS Preview dialog box, to close it.
6. Start the „Normal“ preview and examine the result.

Method 2:

1. Select the tree element with the „email“.
2. Right-click on the „email“element, and then select „Produce XUI component“.
3. Click the Leaf tab marker in the Editor area .
4. Enter the text „E-mail address“ under the point „Label->Text“. Optionally you can also enter a Tooltip text.

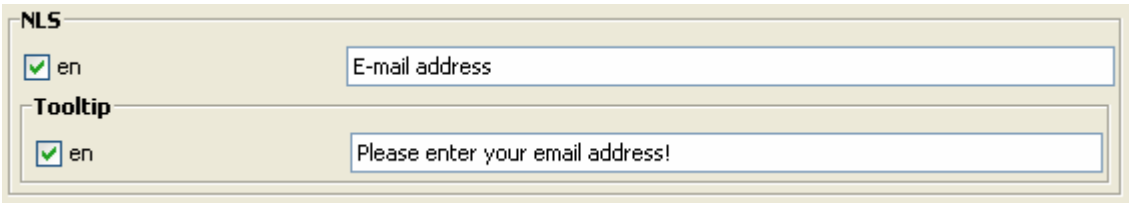


Figure 116: NLS-Tooltip for simple fields

5. Start the „Normal“preview and examine the result.

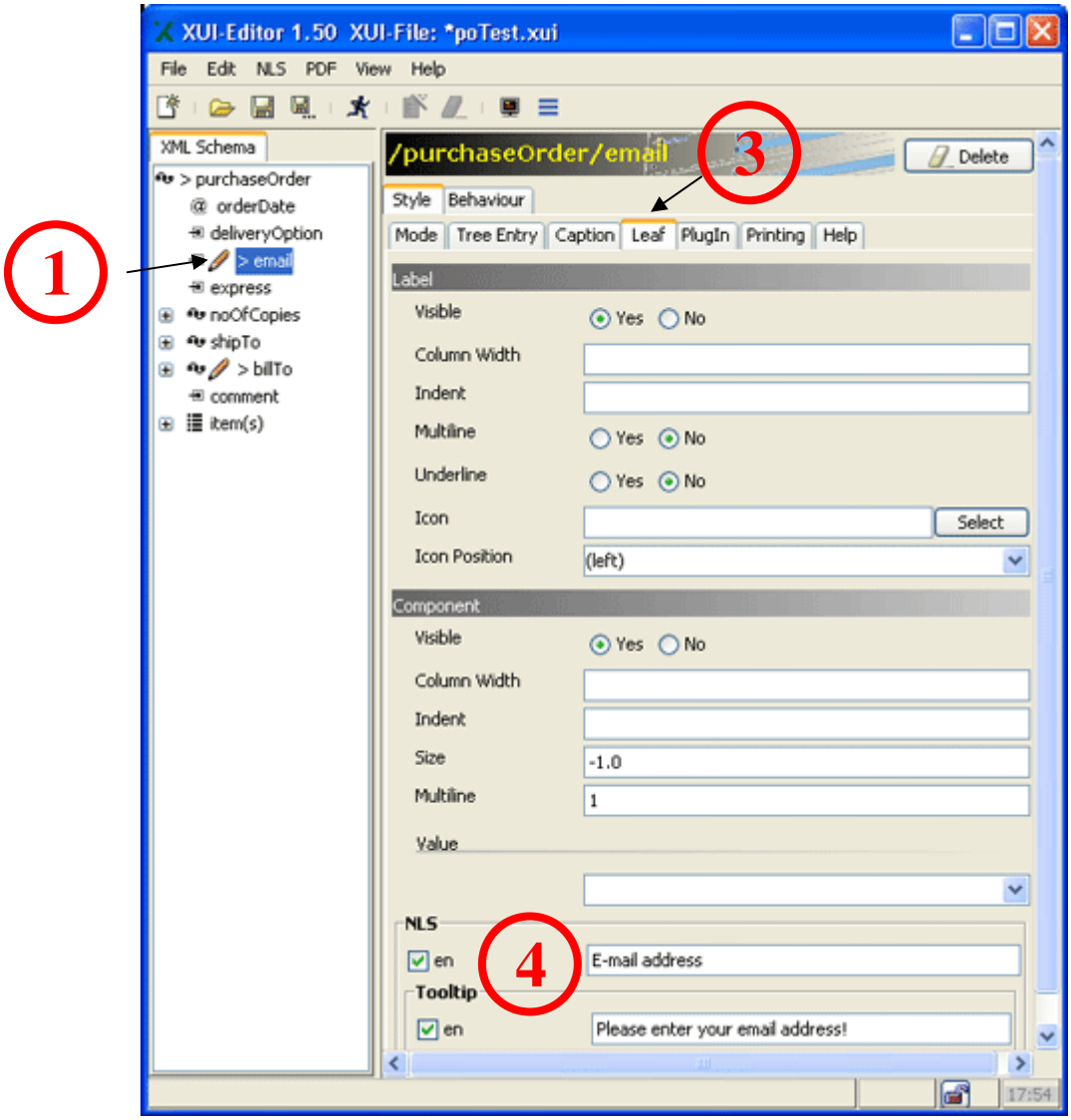


Figure 117: NLS input within the XUI component

The result:

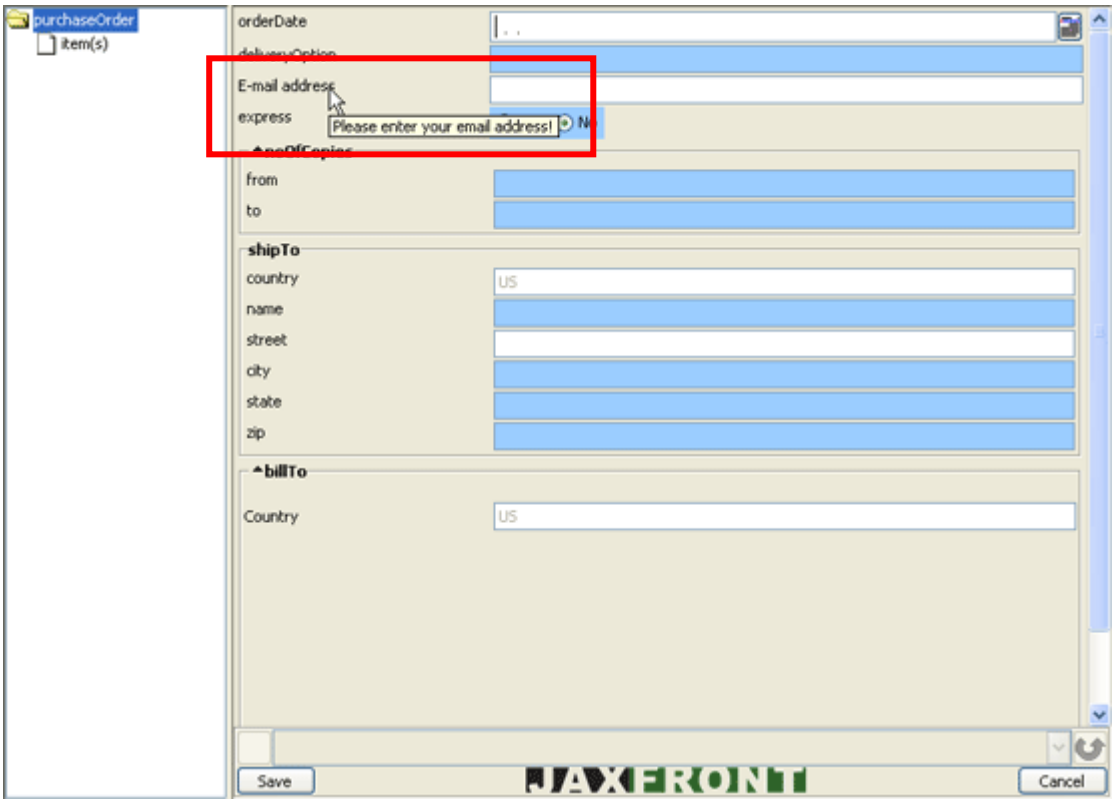


Figure 118: NLS-effects in the preview

5 XUI-schema-specification (xui.xsd)

This section talks about the XUI-schema specification.

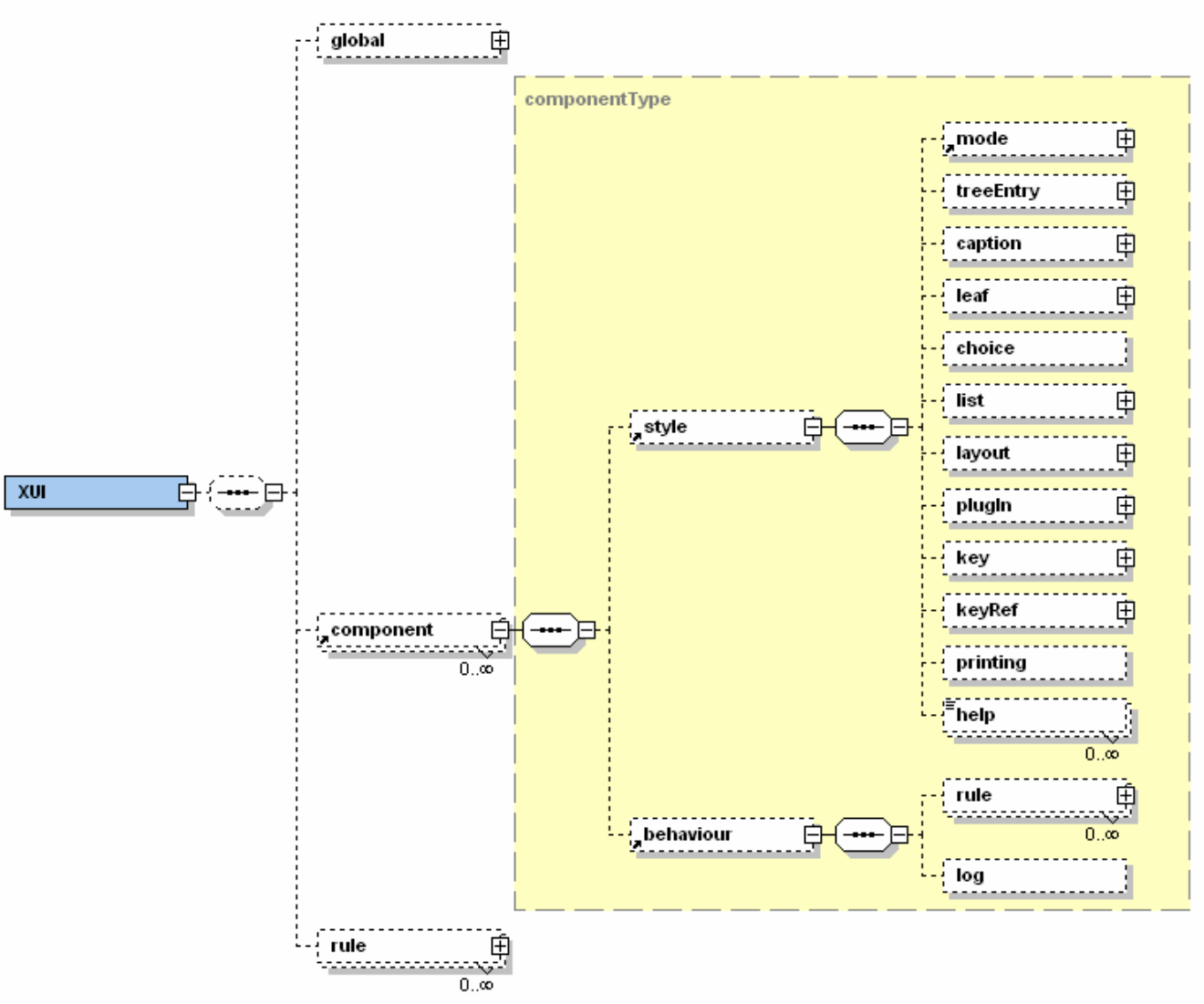


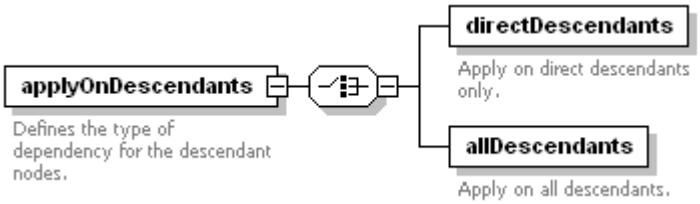
Figure 119: XUI-schema-specification (xui.xsd)

Schema **xui.xsd**

(Generated XML schema documentation)

element **applyOnDescendants**

diagram



children **directDescendants** **allDescendants**

used by elements **style/treeEntry/occurrence** **style/caption/style/occurrence** **occurrence**

annotation documentation Defines the type of dependency for the descendant nodes.

element **applyOnDescendants/directDescendants**

diagram

annotation documentation Apply on direct descendants only.

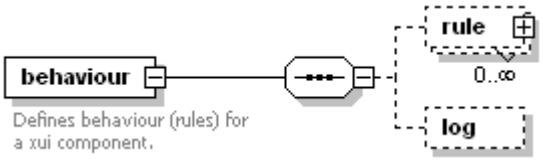
element **applyOnDescendants/allDescendants**

diagram

annotation documentation Apply on all descendants.

element **behaviour**

diagram



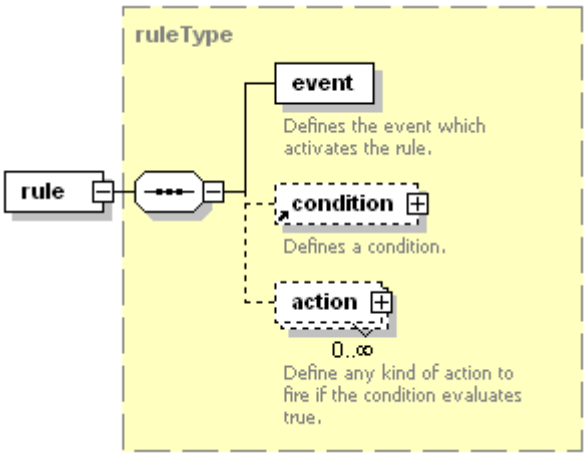
children **rule** **log**

used by complexType **componentType**

annotation documentation Defines behaviour (rules) for a xui component.

element **behaviour/rule**

diagram



type extension of **ruleType**

children **event** **condition** **action**

attributes

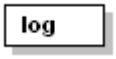
Name	Type	Use	Default	Fixed	Annotation
priority	xsd:int	optional			
ref	xsd:string	optional			

identity constraints

Name	Refer	Selector	Field(s)
keyref	ruleRef	ruleId	.	@ref

element **behaviour/log**

diagram

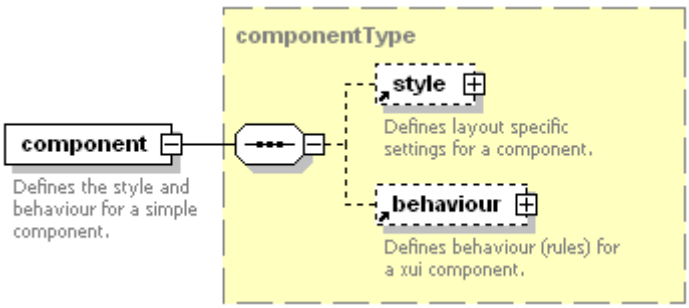


attributes

Name	Type	Use	Default	Fixed	Annotation
level	xsd:string	optional			
scope	xsd:string	optional	sourceAndReferencedNodes		

element **component**

diagram



type extension of **componentType**

children **style** **behaviour**

used by element **XUI**

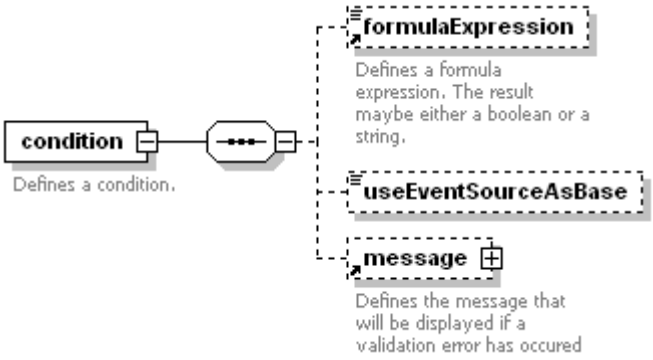
attributes

Name	Type	Use	Default	Fixed	Annotation
xpath	xsd:string	required			
id	xsd:string	optional			

annotation documentation Defines the style and behaviour for a simple component.

element **condition**

diagram



children **formulaExpression** **useEventSourceAsBase** **message**

used by complexType **ruleType**

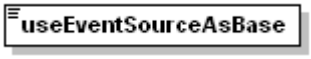
attributes

Name	Type	Use	Default	Fixed	Annotation
showError	xsd:boolean	optional	false		
useInverse	xsd:boolean	optional	false		
className	xsd:string	optional			

annotation documentation Defines a condition.

element **condition/useEventSourceAsBase**

diagram



type **xsd:boolean**

element **dialog**

diagram

used by elements **style/key/context** **tableListType/editMode**

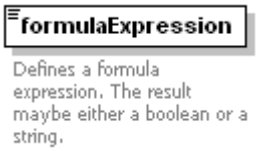
attributes

Name	Type	Use	Default	Fixed	Annotation
dialogBounds	xsd:string	optional	0,0		
dialogButtonSequence	xsd:string	optional	ok[left],cancel[right]		

annotation documentation This will show a dialog for editing the list entries.

element **formulaExpression**

diagram



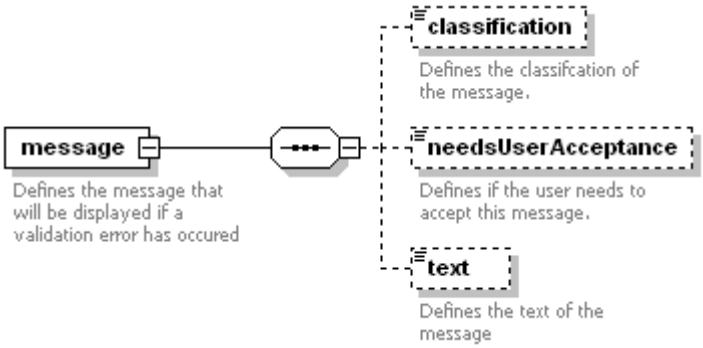
type extension of **xsd:string**

used by elements **style/keyRef/condition** **condition** **style/plugIn/formulaExpressions** **actionType/generalAction** **tableListType/lineInfo/keyContext** **tableLayoutType/cell/type/label** **tableListType/lineInfo** **actionType/propertyChange** **style/keyRef/selectionLineInfo**

annotation documentation Defines a formula expression. The result maybe either a boolean or a string.

element **message**

diagram



children **classification** **needsUserAcceptance** **text**

used by elements **condition** **mode**

annotation documentation Defines the message that will be displayed if a validation error has occured

element **message/classification**

diagram

type restriction of **xsd:string**

facets
enumeration error
enumeration incomplete
enumeration info
enumeration warning
enumeration hint
enumeration question

annotation documentation Defines the classifcation of the message.

element **message/needsUserAcceptance**

diagram

type **xsd:boolean**

annotation documentation Defines if the user needs to accept this message.

element **message/text**

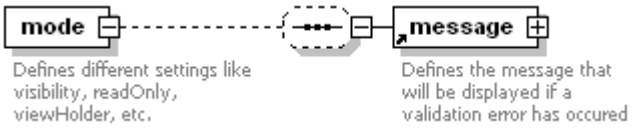
diagram

type **xsd:string**

annotation documentation Defines the text of the message

element **mode**

diagram



children **message**

used by element **style**

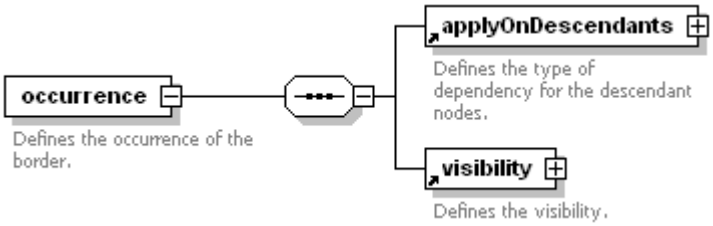
attributes

Name	Type	Use	Default	Fixed	Annotation
visible	xsd:boolean	optional			
readonly	xsd:boolean	optional	false		
editable	xsd:boolean	optional	true		
serialize	xsd:boolean	optional			
isRequired	xsd:boolean	optional	false		
childrenView	xsd:string	optional			
viewHolder	xsd:string	optional			
displayValue	xsd:string	optional			

annotation documentation Defines different settings like visibility, readOnly, viewHolder, etc.

element **occurrence**

diagram



children **applyOnDescendants** **visibility**

attributes

Name	Type	Use	Default	Fixed	Annotation
applyOnDescendants	xsd:boolean	optional	false		

annotation documentation Defines the occurrence of the border.

element **param**

diagram

type extension of **xsd:string**

used by elements **style/plugIn** **uiAction**

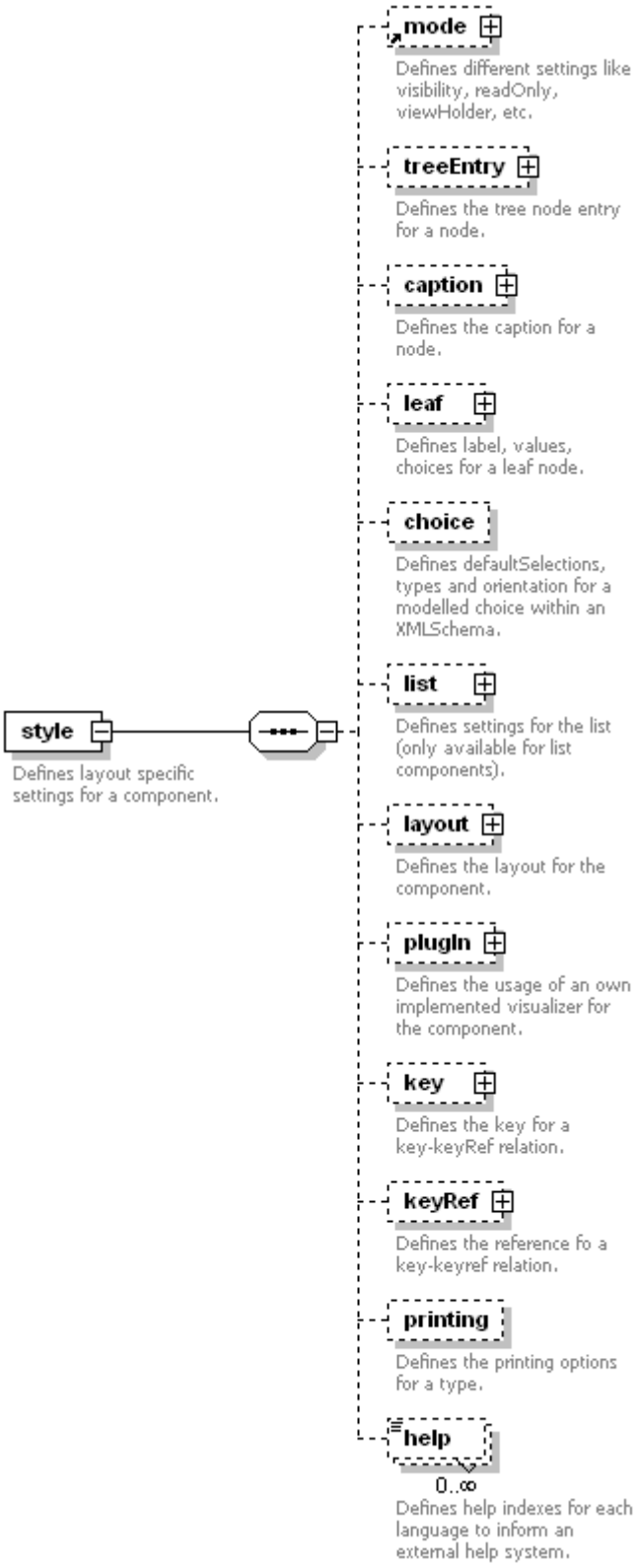
attributes

Name	Type	Use	Default	Fixed	Annotation
name	xsd:string	optional			
type	xsd:string	required			
value	xsd:string	required			

annotation documentation Defines a method paramter as an alias(name), type and a value.

element **style**

diagram



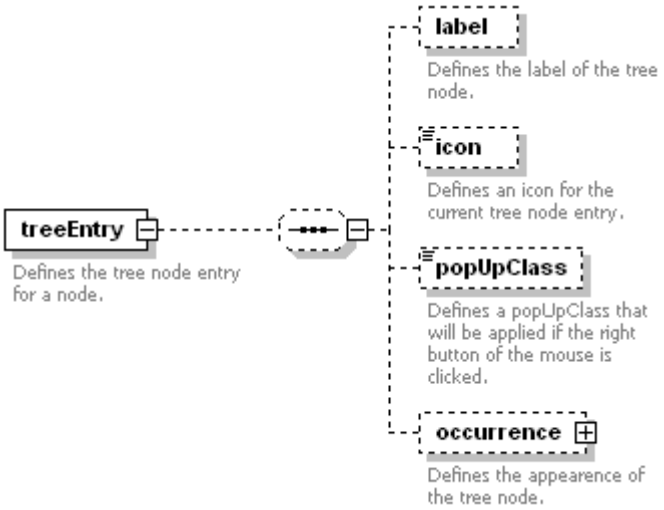
children **mode** **treeEntry** **caption** **leaf** **choice** **list** **layout** **plugIn** **key** **keyRef** **printing** **help**

used by complexType **componentType**

annotation documentation Defines layout specific settings for a component.

element **style/treeEntry**

diagram



children **label** **icon** **popUpClass** **occurrence**

annotation documentation Defines the tree node entry for a node.

element **style/treeEntry/label**

diagram

annotation documentation Defines the label of the tree node.

element **style/treeEntry/icon**

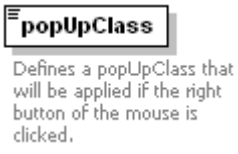
diagram

type **xsd:string**

annotation documentation Defines an icon for the current tree node entry.

element **style/treeEntry/popUpClass**

diagram

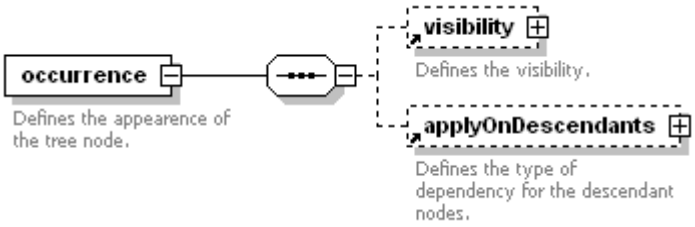


type **xsd:string**

annotation documentation Defines a popUpClass that will be applied if the right button of the mouse is clicked.

element **style/treeEntry/occurrence**

diagram



children **visibility** **applyOnDescendants**

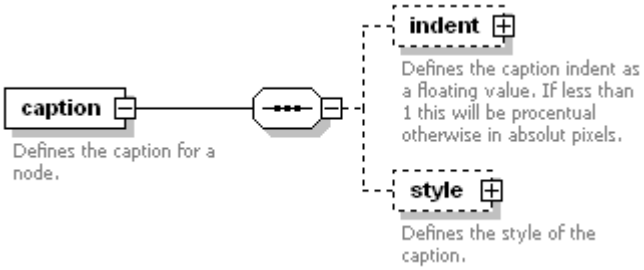
attributes

Name	Type	Use	Default	Fixed	Annotation
isFocusable	xsd:boolean	optional	true		
showAsOptional	xsd:boolean	optional			
showChildrenAsOptional	xsd:boolean	optional	false		

annotation documentation Defines the appearence of the tree node.

element **style/caption**

diagram

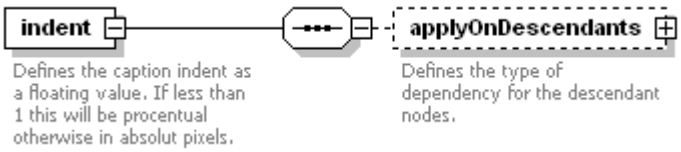


children **indent** **style**

annotation documentation Defines the caption for a node.

element **style/caption/indent**

diagram



children **applyOnDescendants**

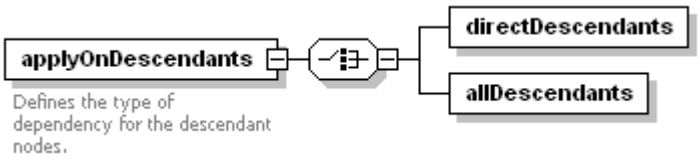
attributes

Name	Type	Use	Default	Fixed	Annotation
value	xsd:float	optional			

annotation documentation Defines the caption indent as a floating value. If less than 1 this will be procentual otherwise in absolut pixels.

element **style/caption/indent/applyOnDescendants**

diagram

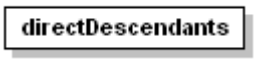


children **directDescendants** **allDescendants**

annotation documentation Defines the type of dependency for the descendant nodes.

element **style/caption/indent/applyOnDescendants/directDescendants**

diagram



type captionIndentType

attributes

Name	Type	Use	Default	Fixed	Annotation
applyCaptionIndentOn	xsd:string	optional	label		
nestedIndent	xsd:float	optional			

element **style/caption/indent/applyOnDescendants/allDescendants**

diagram

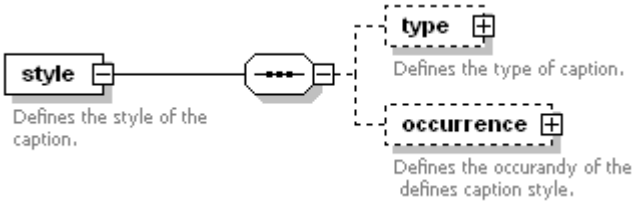
type captionIndentType

attributes

Name	Type	Use	Default	Fixed	Annotation
applyCaptionIndentOn	xsd:string	optional	label		
nestedIndent	xsd:float	optional			

element **style/caption/style**

diagram



children type occurrence

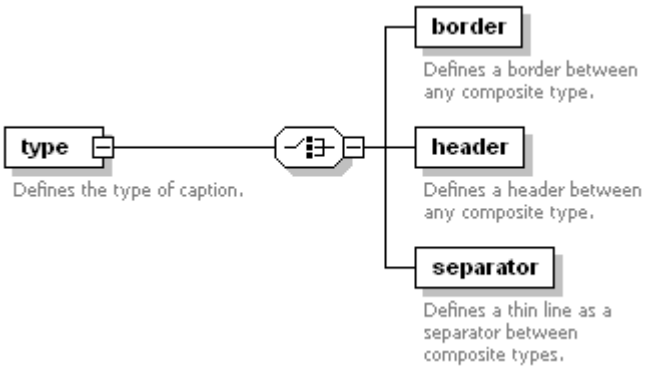
attributes

Name	Type	Use	Default	Fixed	Annotation
headerImage	xsd:string	optional			

annotation documentation Defines the style of the caption.

element **style/caption/style/type**

diagram



children border header separator

annotation documentation Defines the type of caption.

element **style/caption/style/type/border**

diagram

attributes

Name	Type	Use	Default	Fixed	Annotation
style	xsd:string	optional	line		
collapsed	xsd:boolean	optional			

annotation documentation Defines a border between any composite type.

element **style/caption/style/type/header**

diagram

annotation documentation Defines a header between any composite type.

element **style/caption/style/type/separator**

diagram

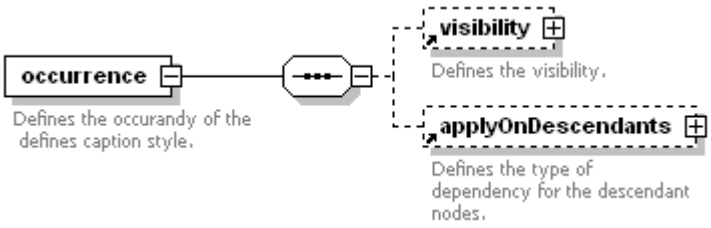
attributes

Name	Type	Use	Default	Fixed	Annotation
style	xsd:string	optional	modern		

annotation documentation Defines a thin line as a separator between composite types.

element **style/caption/style/occurrence**

diagram

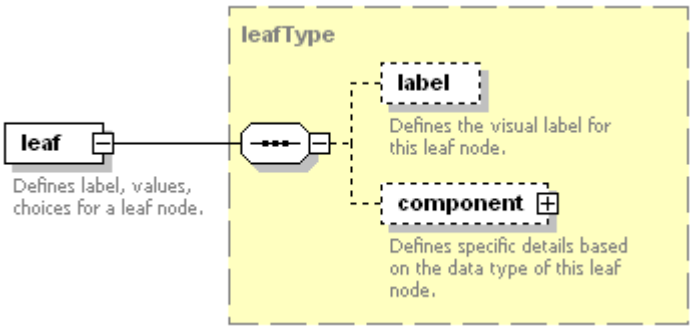


children **visibility** **applyOnDescendants**

annotation documentation Defines the occurandy of the defines caption style.

element **style/leaf**

diagram



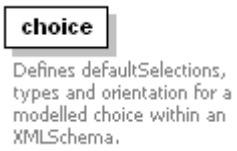
type **leafType**

children **label** **component**

annotation documentation Defines label, values, choices for a leaf node.

element **style/choice**

diagram



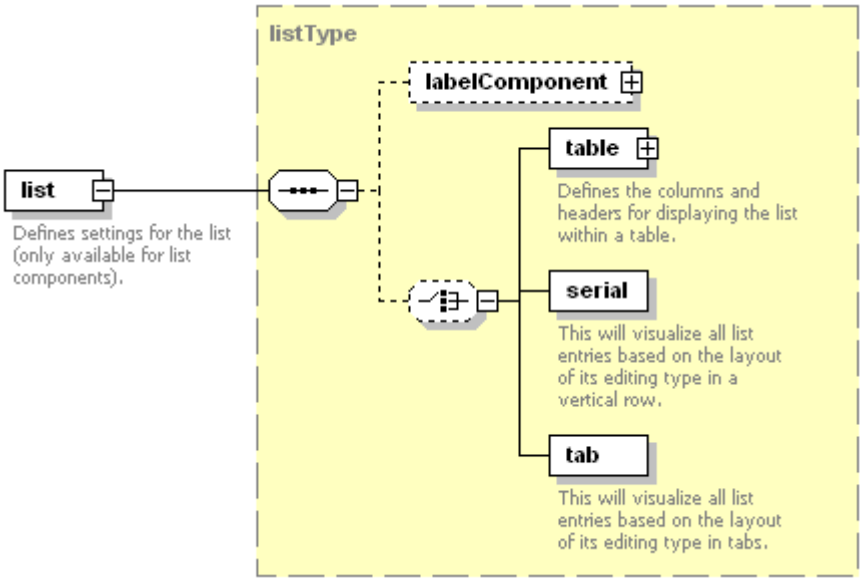
attributes

Name	Type	Use	Default	Fixed	Annotation
defaultSelection	xsd:string	optional			
singleChoiceAndSerialize	xsd:boolean	optional	false		
type	xsd:string	optional			
orientation	xsd:string	optional			

annotation documentation Defines defaultSelections, types and orientation for a modelled choice within an XMLSchema.

element **style/list**

diagram



type extension of **listType**

children **labelComponent table serial tab**

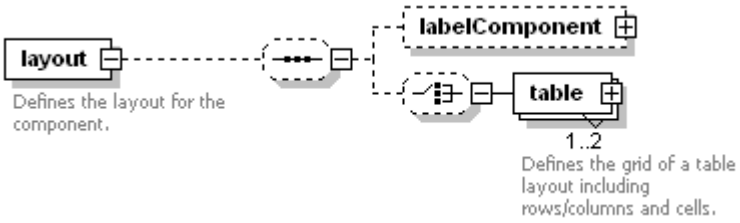
attributes

Name	Type	Use	Default	Fixed	Annotation
visible	xsd:boolean	optional	true		
autoCreateMinOccurListItems	xsd:boolean	optional			
showSequenceNumbering	xsd:boolean	optional	true		
columnWidth	xsd:int	optional	100		
visibleRowCount	xsd:int	optional	5		
allowReOrdering	xsd:boolean	optional	false		
allowSorting	xsd:boolean	optional	false		
selectionMode	xsd:string	optional	multi		
startupSelection	xsd:string	optional	first		
useZebraLook	xsd:boolean	optional	true		
selectionOnly	xsd:boolean	optional	false		
uniqueListItemIDPath	xsd:string	optional			
showButtons	xsd:boolean	optional	true		
showNewButton	xsd:boolean	optional	true		
showCopyButton	xsd:boolean	optional	true		
showDeleteButton	xsd:boolean	optional	true		
showEditButton	xsd:boolean	optional	true		
useDeletionConfirmation	xsd:boolean	optional	true		
deletionConfirmationMessage	xsd:string	optional			

annotation documentation Defines settings for the list (only available for list components).

element **style/layout**

diagram

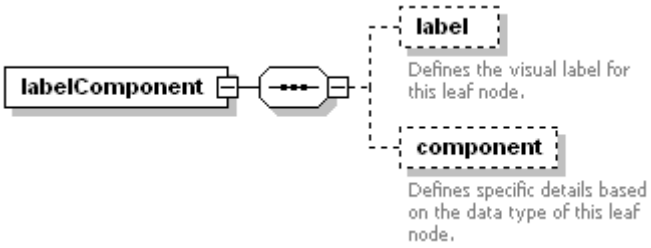


children **labelComponent** **table**

annotation documentation Defines the layout for the component.

element **style/layout/labelComponent**

diagram



type restriction of **leafType**

children **label** **component**

element **style/layout/labelComponent/label**

diagram

attributes

Name	Type	Use	Default	Fixed	Annotation
visible	xsd:boolean	optional	true		
columnWidth	xsd:float	optional			
indent	xsd:float	optional			
multiLine	xsd:boolean	optional	false		
useUnderline	xsd:boolean	optional	false		
icon	xsd:string	optional			
iconPosition	xsd:string	optional	left		

annotation documentation Defines the visual label for this leaf node.

element **style/layout/labelComponent/component**

diagram

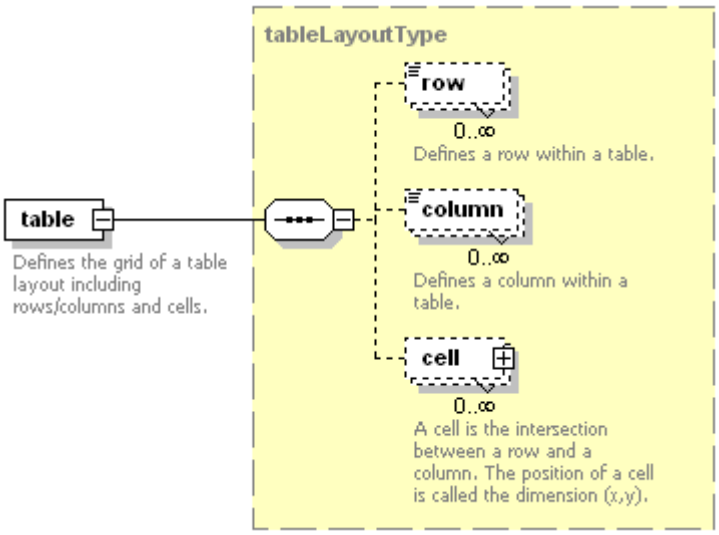
attributes

Name	Type	Use	Default	Fixed	Annotation
visible	xsd:boolean	optional	true		
columnWidth	xsd:float	optional			

annotation documentation Defines specific details based on the data type of this leaf node.

element **style/layout/table**

diagram



type **tableLayoutType**

children **row column cell**

attributes

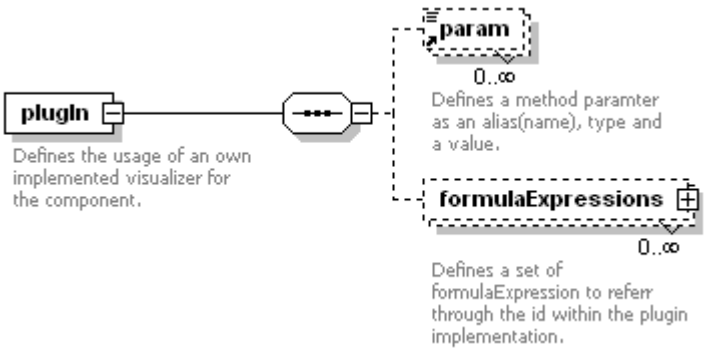
Name	Type	Use	Default	Fixed	Annotation
optional	xsd:boolean	optional	true		
id	xsd:string	optional	main		

annotation

documentation Defines the grid of a table layout including rows/columns and cells.

element **style/plugIn**

diagram



children **param formulaExpressions**

attributes

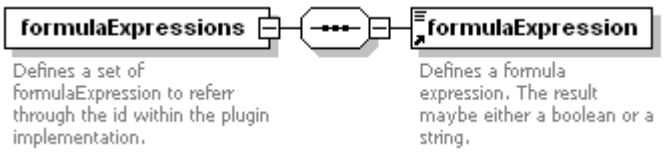
Name	Type	Use	Default	Fixed	Annotation
class	xsd:string	optional			

annotation

documentation Defines the usage of an own implemented visualizer for the component.

element **style/plugIn/formulaExpressions**

diagram



children **formulaExpression**

attributes

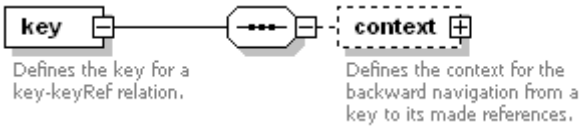
Name	Type	Use	Default	Fixed	Annotation
name	xsd:string	optional			
id	xsd:string	optional			

annotation

documentation Defines a set of formulaExpression to referr through the id within the plugin implementation.

element **style/key**

diagram

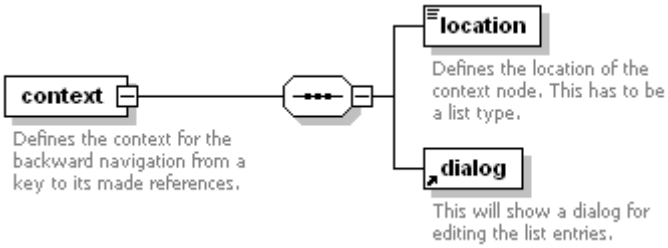


children **context**

annotation documentation Defines the key for a key-keyRef relation.

element **style/key/context**

diagram

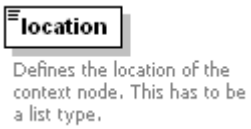


children **location dialog**

annotation documentation Defines the context for the backward navigation from a key to its made references.

element **style/key/context/location**

diagram

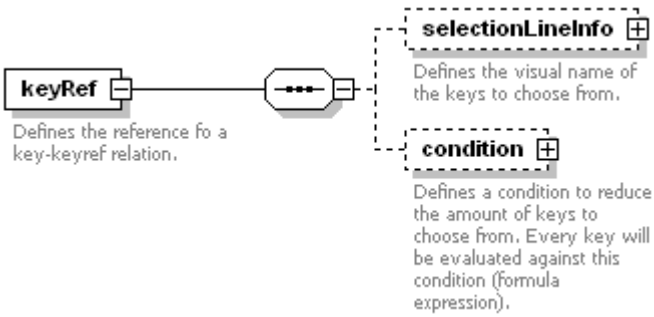


type **xsd:string**

annotation documentation Defines the location of the context node. This has to be a list type.

element **style/keyRef**

diagram

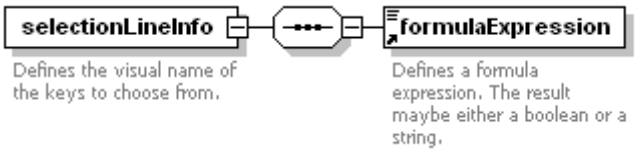


children **selectionLineInfo condition**

annotation documentation Defines the reference fo a key-keyref relation.

element **style/keyRef/selectionLineInfo**

diagram

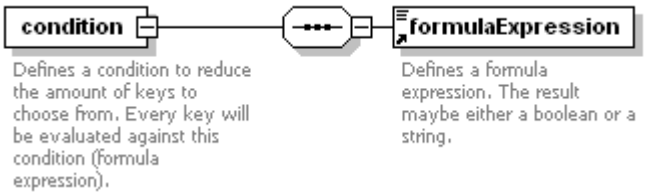


children **formulaExpression**

annotation documentation Defines the visual name of the keys to choose from.

element **style/keyRef/condition**

diagram

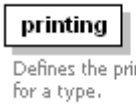


children **formulaExpression**

annotation documentation Defines a condition to reduce the amount of keys to choose from. Every key will be evaluated against this condition (formula expression).

element **style/printing**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
printable	xsd:string	optional	ifIsVisibleAndSerializable		
printChoiceAsHeader	xsd:boolean	optional			
listPrintMode	xsd:string	optional			

annotation documentation Defines the printing options for a type.

element **style/help**

diagram

type extension of **xsd:string**

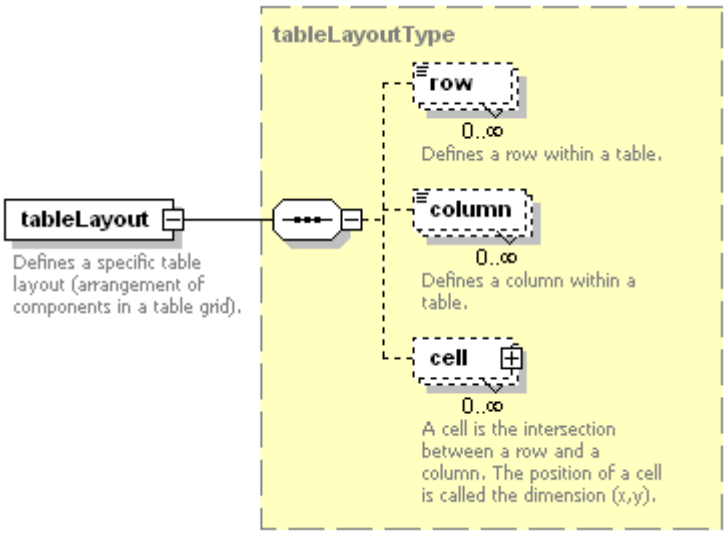
attributes

Name	Type	Use	Default	Fixed	Annotation
language	xsd:string	required			

annotation documentation Defines help indexes for each language to inform an external help system.

element **tableLayout**

diagram



type **tableLayoutType**

children **row** **column** **cell**

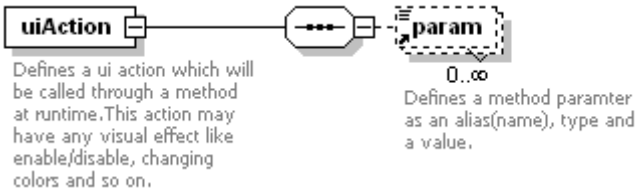
attributes

Name	Type	Use	Default	Fixed	Annotation
optional	xsd:boolean	optional	true		
id	xsd:string	optional	main		

annotation documentation Defines a specific table layout (arrangement of components in a table grid).

element **uiAction**

diagram



children **param**

used by complexType **actionType**

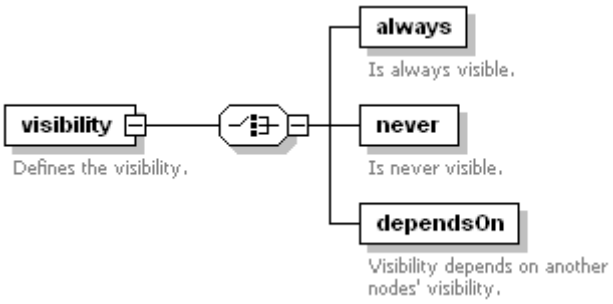
attributes

Name	Type	Use	Default	Fixed	Annotation
methodName	xsd:string	required			

annotation documentation Defines a ui action which will be called through a method at runtime.This action may have any visual effect like enable/disable, changing colors and so on.

element **visibility**

diagram



children **always** **never** **dependsOn**

used by elements **style/treeEntry/occurrence** **style/caption/style/occurrence** **occurrence**

annotation documentation Defines the visibility.

element **visibility/always**

diagram

annotation documentation Is always visible.

element **visibility/never**

diagram

annotation documentation Is never visible.

element **visibility/dependsOn**

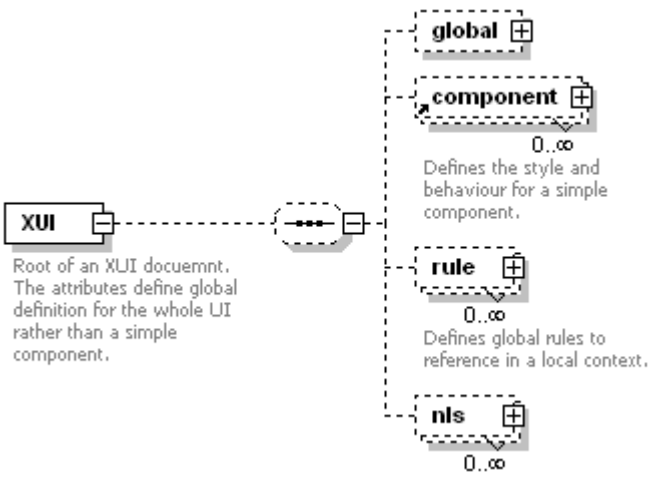
diagram

attributes	Name	Type	Use	Default	Fixed	Annotation
	xpath	xsd:string	optional			

annotation documentation Visibility depends on another nodes' visibility.

element **XUI**

diagram



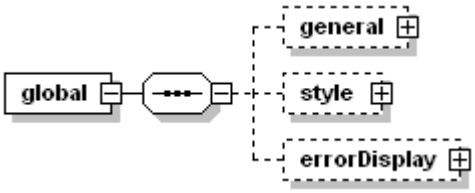
children **global** **component** **rule** **nls**

attributes	Name	Type	Use	Default	Fixed	Annotation
	version	xsd:string	optional		2.0	

annotation documentation Root of an XUI docuemnt. The attributes define global definition for the whole UI rather than a simple component.

element **XUI/global**

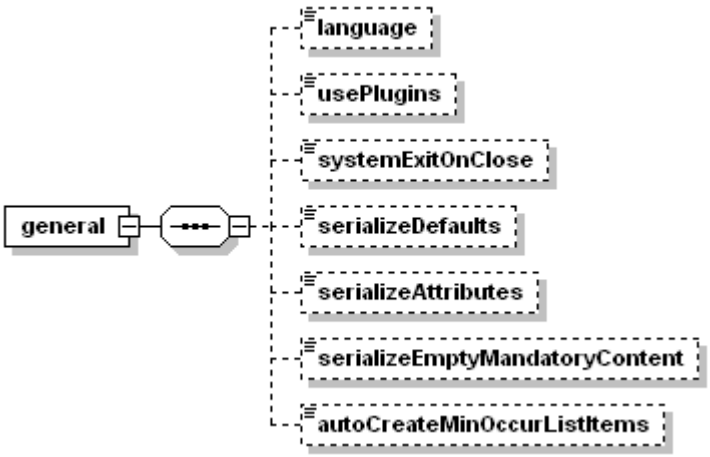
diagram



children general style errorDisplay

element **XUI/global/general**

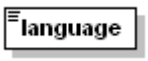
diagram



children language usePlugins systemExitOnClose serializeDefaults serializeAttributes serializeEmptyMandatoryContent autoCreateMinOccurListItems

element **XUI/global/general/language**

diagram



type **xsd:string**

element **XUI/global/general/usePlugins**

diagram

type **xsd:boolean**

element **XUI/global/general/systemExitOnClose**

diagram

type **xsd:boolean**

element **XUI/global/general/serializeDefaults**

diagram

type **xsd:boolean**

element **XUI/global/general/serializeAttributes**

diagram 

type **xsd:boolean**

element **XUI/global/general/serializeEmptyMandatoryContent**

diagram 

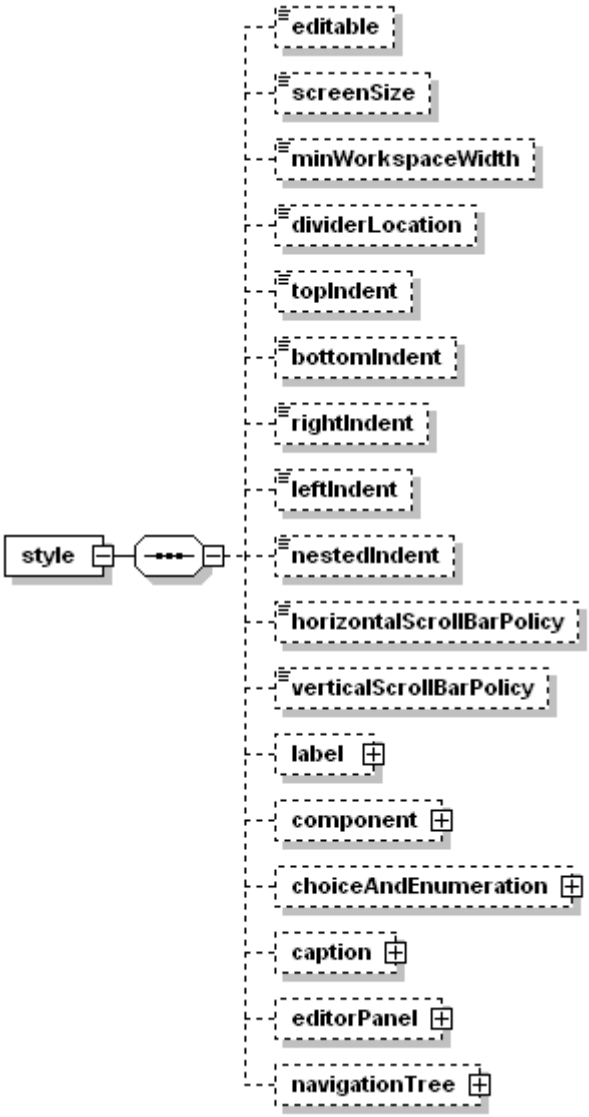
type **xsd:boolean**

element **XUI/global/general/autoCreateMinOccurListItems**

diagram 

type **xsd:boolean**

element **XUI/global/style**

diagram 

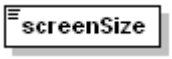
children editable screenSize minWorkspaceWidth dividerLocation topIndent bottomIndent rightIndent leftIndent nestedIndent horizontalScrollBarPolicy verticalScrollBarPolicy label component choiceAndEnumeration caption editorPanel navigationTree

element **XUI/global/style/editable**

diagram 

type **xsd:boolean**

element **XUI/global/style/screenSize**

diagram 

type **xsd:string**

element **XUI/global/style/minWorkspaceWidth**

diagram 

type **xsd:int**

element **XUI/global/style/dividerLocation**

diagram 

type **xsd:float**

element **XUI/global/style/topIndent**

diagram 

type **xsd:int**

element **XUI/global/style/bottomIndent**

diagram 

type **xsd:int**

element **XUI/global/style/rightIndent**

diagram 

type **xsd:int**

element **XUI/global/style/leftIndent**

diagram 

type **xsd:int**

element **XUI/global/style/nestedIndent**

diagram 

type **xsd:float**

element **XUI/global/style/horizontalScrollBarPolicy**

diagram 

type restriction of **xsd:string**

facets
enumeration HORIZONTAL_SCROLLBAR_AS_NEEDED
enumeration HORIZONTAL_SCROLLBAR_ALWAYS
enumeration HORIZONTAL_SCROLLBAR_NEVER

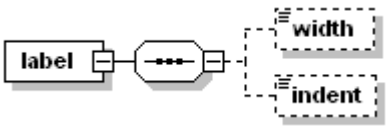
element **XUI/global/style/verticalScrollBarPolicy**

diagram 

type restriction of **xsd:string**

facets
enumeration VERTICAL_SCROLLBAR_AS_NEEDED
enumeration VERTICAL_SCROLLBAR_ALWAYS
enumeration VERTICAL_SCROLLBAR_NEVER

element **XUI/global/style/label**

diagram 

children **width indent**

element **XUI/global/style/label/width**

diagram 

type **xsd:float**

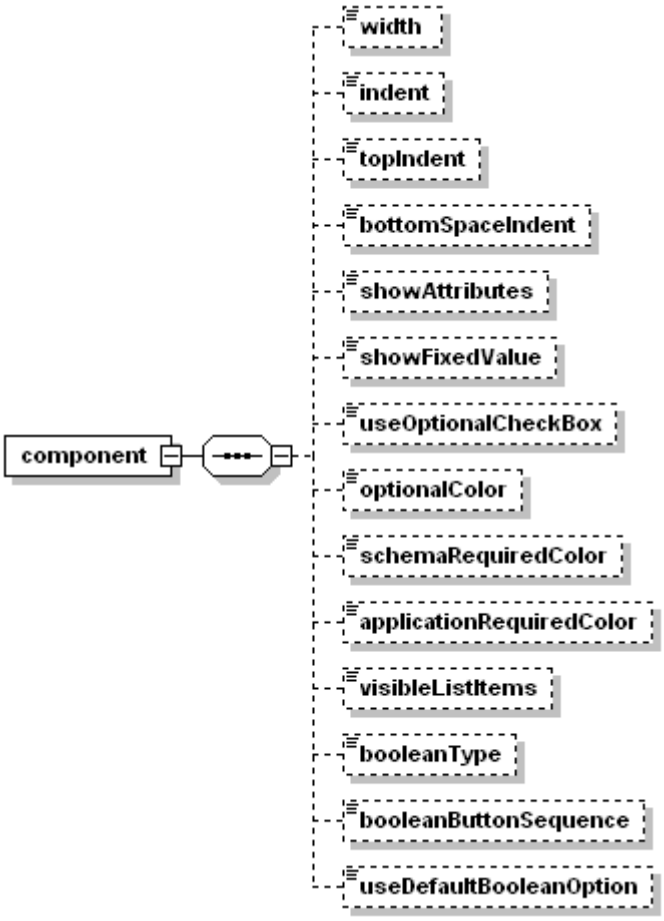
element **XUI/global/style/label/indent**

diagram 

type **xsd:float**

element **XUI/global/style/component**

diagram



children width indent topIndent bottomSpaceIndent showAttributes showFixedValue useOptionalCheckBox optionalColor schemaRequiredColor applicationRequiredColor visibleListItems booleanType booleanButtonSequence useDefaultBooleanOption

element **XUI/global/style/component/width**

diagram

type **xsd:float**

element **XUI/global/style/component/indent**

diagram

type **xsd:float**

element **XUI/global/style/component/topIndent**

diagram

type **xsd:int**

element **XUI/global/style/component/bottomSpaceIndent**

diagram 

type **xsd:int**

element **XUI/global/style/component/showAttributes**

diagram 

type **xsd:boolean**

element **XUI/global/style/component/showFixedValue**

diagram 

type **xsd:boolean**

element **XUI/global/style/component/useOptionalCheckBox**

diagram 

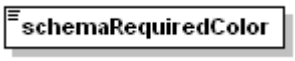
type **xsd:boolean**

element **XUI/global/style/component/optionalColor**

diagram 

type **colorType**

element **XUI/global/style/component/schemaRequiredColor**

diagram 

type **colorType**

element **XUI/global/style/component/applicationRequiredColor**

diagram 

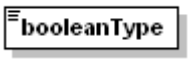
type **colorType**

element **XUI/global/style/component/visibleListItems**

diagram 

type **xsd:int**

element **XUI/global/style/component/booleanType**

diagram 

type restriction of **xsd:string**

facets
enumeration radioButton
enumeration radioButtonUnselected
enumeration checkBox

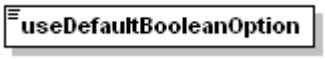
element **XUI/global/style/component/booleanButtonSequence**

diagram 

type restriction of **xsd:string**

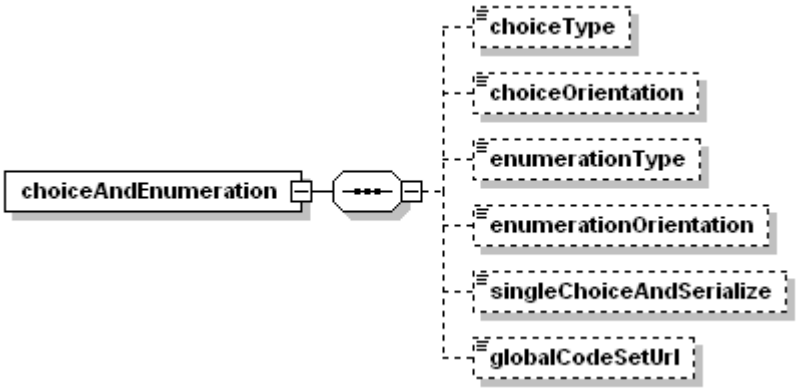
facets
enumeration yes,no
enumeration no,yes

element **XUI/global/style/component/useDefaultBooleanOption**

diagram 

type **xsd:boolean**

element **XUI/global/style/choiceAndEnumeration**

diagram 

children choiceType choiceOrientation enumerationType enumerationOrientation singleChoiceAndSerialize globalCodeSetUrl

element **XUI/global/style/choiceAndEnumeration/choiceType**

diagram 

type restriction of **xsd:string**

facets
enumeration comboBox
enumeration radioButton

element **XUI/global/style/choiceAndEnumeration/choiceOrientation**

diagram 

type restriction of **xsd:string**

facets
enumeration horizontal
enumeration vertical

element **XUI/global/style/choiceAndEnumeration/enumerationType**

diagram 

type restriction of **xsd:string**

facets enumeration comboBox
enumeration radioButton

element **XUI/global/style/choiceAndEnumeration/enumerationOrientation**

diagram 

type restriction of **xsd:string**

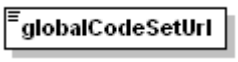
facets enumeration horizontal
enumeration vertical

element **XUI/global/style/choiceAndEnumeration/singleChoiceAndSerialize**

diagram 

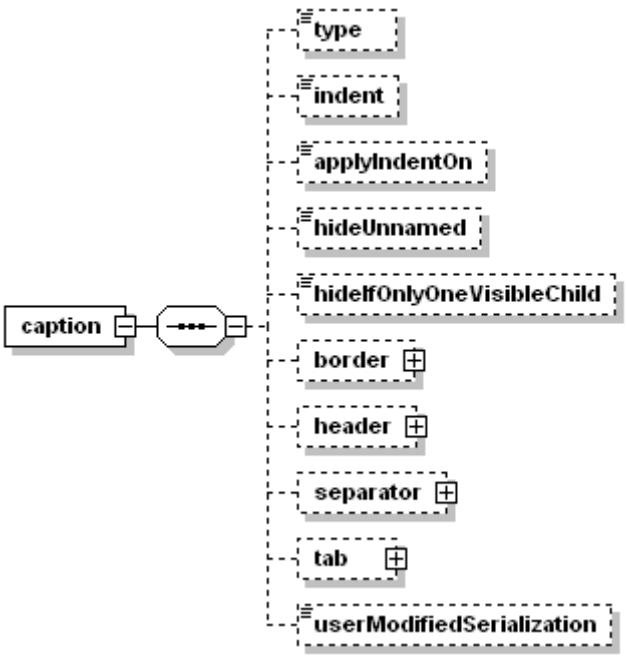
type **xsd:boolean**

element **XUI/global/style/choiceAndEnumeration/globalCodeSetUrl**

diagram 

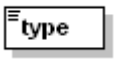
type **xsd:string**

element **XUI/global/style/caption**

diagram 

children type indent applyIndentOn hideUnnamed hideIfOnlyOneVisibleChild border header separator tab userModifiedSerialization

element **XUI/global/style/caption/type**

diagram 

type restriction of **xsd:string**

facets
- enumeration border
- enumeration header
- enumeration separator_classic
- enumeration separator_modern

element **XUI/global/style/caption/indent**

diagram 

type **xsd:float**

element **XUI/global/style/caption/applyIndentOn**

diagram 

type restriction of **xsd:string**

facets
- enumeration label
- enumeration component
- enumeration both

element **XUI/global/style/caption/hideUnnamed**

diagram 

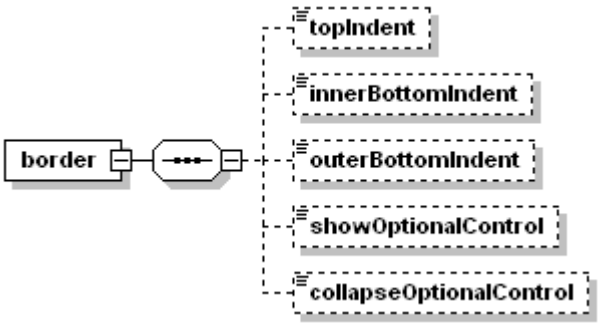
type **xsd:boolean**

element **XUI/global/style/caption/hideIfOnlyOneVisibleChild**

diagram 

type **xsd:boolean**

element **XUI/global/style/caption/border**

diagram 

children topIndent innerBottomIndent outerBottomIndent showOptionalControl collapseOptionalControl

element **XUI/global/style/caption/border/topIndent**

diagram 

type **xsd:int**

element **XUI/global/style/caption/border/innerBottomIndent**

diagram 

type **xsd:int**

element **XUI/global/style/caption/border/outerBottomIndent**

diagram 

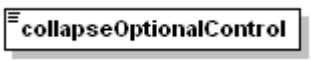
type **xsd:int**

element **XUI/global/style/caption/border/showOptionalControl**

diagram 

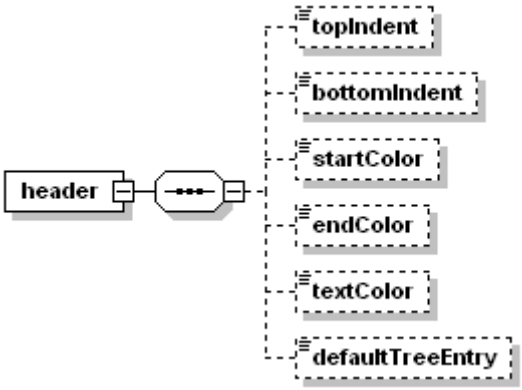
type **xsd:boolean**

element **XUI/global/style/caption/border/collapseOptionalControl**

diagram 

type **xsd:boolean**

element **XUI/global/style/caption/header**

diagram 

children topIndent bottomIndent startColor endColor textColor defaultTreeEntry

element **XUI/global/style/caption/header/topIndent**

diagram 

type **xsd:int**

element **XUI/global/style/caption/header/bottomIndent**

diagram 

type **xsd:int**

element **XUI/global/style/caption/header/startColor**

diagram 

type **colorType**

element **XUI/global/style/caption/header/endColor**

diagram 

type **colorType**

element **XUI/global/style/caption/header/textColor**

diagram 

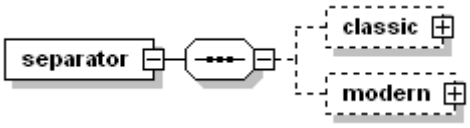
type **colorType**

element **XUI/global/style/caption/header/defaultTreeEntry**

diagram 

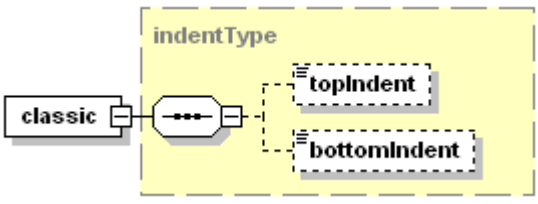
type **xsd:boolean**

element **XUI/global/style/caption/separator**

diagram 

children **classic** **modern**

element **XUI/global/style/caption/separator/classic**

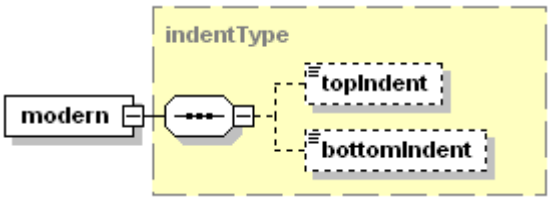
diagram 

type **indentType**

children **topIndent** **bottomIndent**

element **XUI/global/style/caption/separator/modern**

diagram

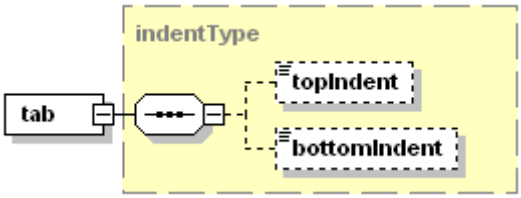


type **indentType**

children **topIndent bottomIndent**

element **XUI/global/style/caption/tab**

diagram



type **indentType**

children **topIndent bottomIndent**

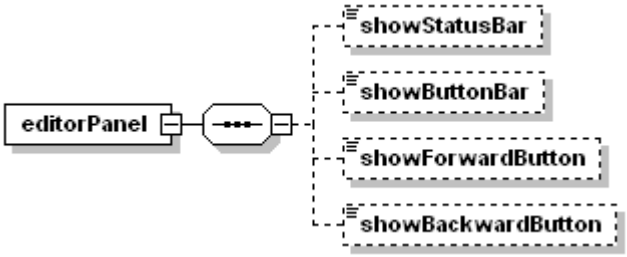
element **XUI/global/style/caption/userModifiedSerialization**

diagram

type **xsd:boolean**

element **XUI/global/style/editorPanel**

diagram



children **showStatusBar showButtonBar showForwardButton showBackwardButton**

element **XUI/global/style/editorPanel/showStatusBar**

diagram

type **xsd:boolean**

element **XUI/global/style/editorPanel/showButtonBar**

diagram

type **xsd:boolean**

element **XUI/global/style/editorPanel/showForwardButton**

diagram 

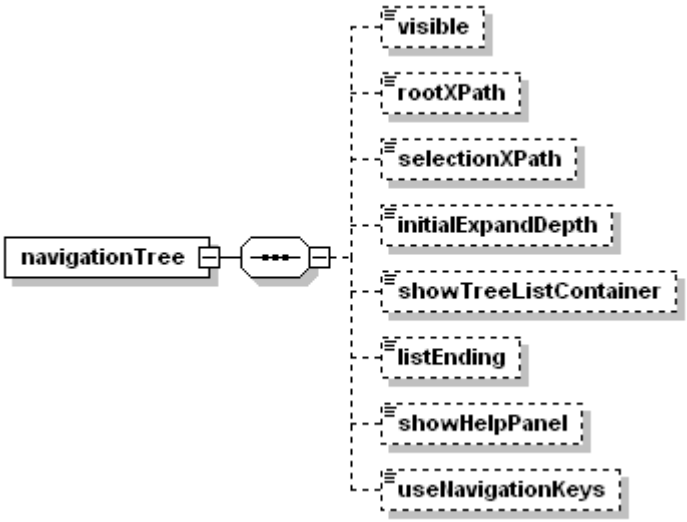
type **xsd:boolean**

element **XUI/global/style/editorPanel/showBackwardButton**

diagram 

type **xsd:boolean**

element **XUI/global/style/navigationTree**

diagram 

children visible rootXPath selectionXPath initialExpandDepth showTreeListContainer listEnding showHelpPanel useNavigationKeys

element **XUI/global/style/navigationTree/visible**

diagram 

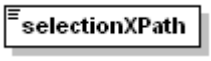
type **xsd:boolean**

element **XUI/global/style/navigationTree/rootXPath**

diagram 

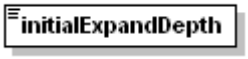
type **xsd:string**

element **XUI/global/style/navigationTree/selectionXPath**

diagram 

type **xsd:string**

element **XUI/global/style/navigationTree/initialExpandDepth**

diagram 

type **xsd:int**

element **XUI/global/style/navigationTree/showTreeListContainer**

diagram 

type **xsd:boolean**

element **XUI/global/style/navigationTree/listEnding**

diagram 

type **xsd:string**

element **XUI/global/style/navigationTree/showHelpPanel**

diagram 

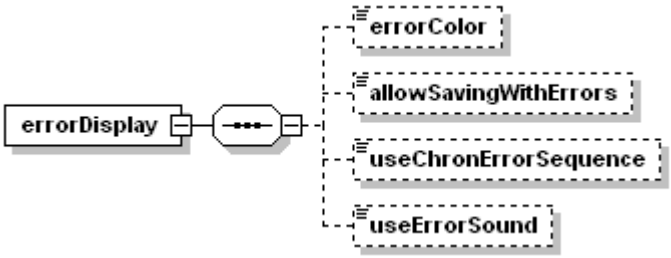
type **xsd:boolean**

element **XUI/global/style/navigationTree/useNavigationKeys**

diagram 

type **xsd:boolean**

element **XUI/global/errorDisplay**

diagram 

children errorColor allowSavingWithErrors useChronErrorSequence useErrorSound

element **XUI/global/errorDisplay/errorColor**

diagram 

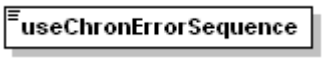
type colorType

element **XUI/global/errorDisplay/allowSavingWithErrors**

diagram 

type **xsd:boolean**

element **XUI/global/errorDisplay/useChronErrorSequence**

diagram 

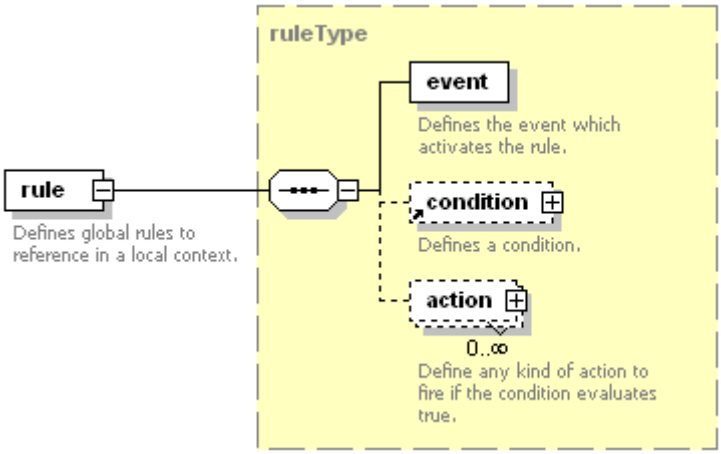
type **xsd:boolean**

element **XUI/global/errorDisplay/useErrorSound**

diagram 

type **xsd:boolean**

element **XUI/rule**

diagram 

type extension of **ruleType**

children **event condition action**

attributes

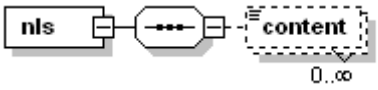
Name	Type	Use	Default	Fixed	Annotation
priority	xsd:int	optional			
id	xsd:string	required			
context	xsd:string	optional			

identity constraints

Name	Refer	Selector	Field(s)
key	ruleId	.	@id

annotation documentation Defines global rules to reference in a local context.

element **XUI/nls**

diagram 

children **content**

element **XUI/nls/content**

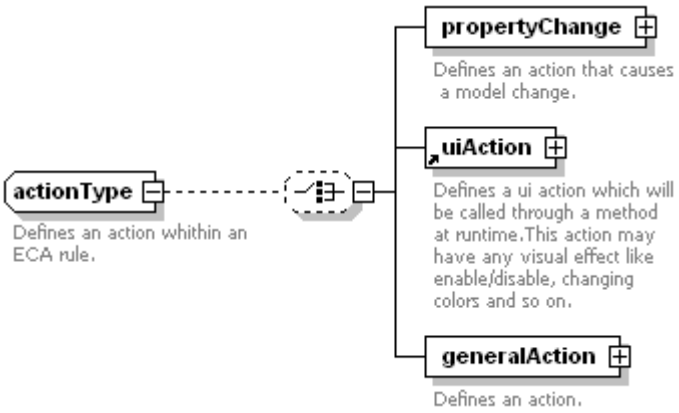
diagram 

attributes

Name	Type	Use	Default	Fixed	Annotation
language	xsd:string	required			

complexType **actionType**

diagram



children **propertyChange** **uiAction** **generalAction**

used by elements **ruleType/action/inverseAction** **ruleType/action/mainAction**

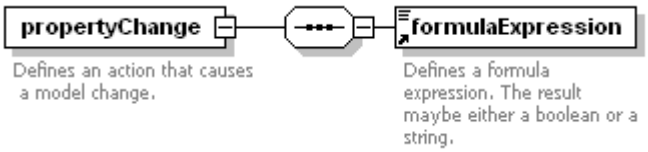
attributes

Name	Type	Use	Default	Fixed	Annotation
target	xsd:string	optional			
includeOnlySerializableNodes	xsd:boolean	optional	false		

annotation documentation Defines an action whithin an ECA rule.

element **actionType/propertyChange**

diagram

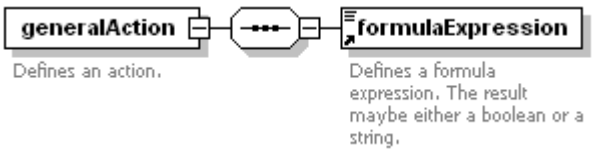


children **formulaExpression**

annotation documentation Defines an action that causes a model change.

element **actionType/generalAction**

diagram



children **formulaExpression**

annotation documentation Defines an action.

complexType **captionIndentType**

diagram

used by elements **style/caption/indent/applyOnDescendants/allDescendants** **style/caption/indent/applyOnDescendants/directDescendants**

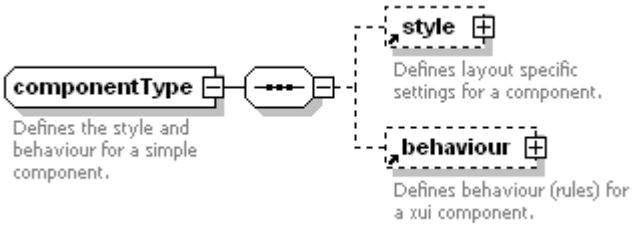
attributes

Name	Type	Use	Default	Fixed	Annotation
applyCaptionIndentOn	xsd:string	optional	label		
nestedIndent	xsd:float	optional			

annotation documentation Defines the caption indent of a component.

complexType **componentType**

diagram



children **style** **behaviour**

used by element **component**

attributes

Name	Type	Use	Default	Fixed	Annotation
xpath	xsd:string	required			

annotation documentation Defines the style and behaviour for a simple component.

complexType **fontType**

diagram

used by element **tableLayoutType/cell/type/label/font**

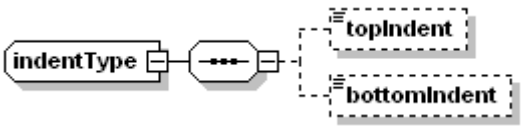
attributes

Name	Type	Use	Default	Fixed	Annotation
name	xsd:string	optional			
size	xsd:int	optional			
style	xsd:string	optional			
color	colorType	optional			

annotation documentation Defines a specific font.

complexType **indentType**

diagram



children **topIndent** **bottomIndent**

used by elements **XUI/global/style/caption/separator/classic** **XUI/global/style/caption/separator/modern** **XUI/global/style/caption/tab**

element **indentType/topIndent**

diagram

type **xsd:int**

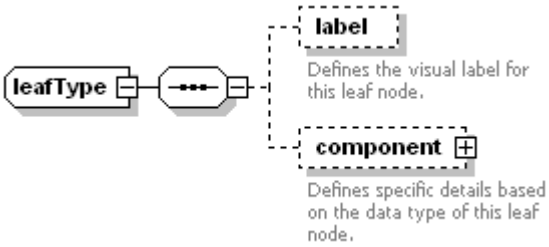
element **indentType/bottomIndent**

diagram

type **xsd:int**

complexType **leafType**

diagram

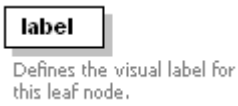


children **label** **component**

used by elements **style/layout/labelComponent** **listType/labelComponent** **style/leaf**

element **leafType/label**

diagram



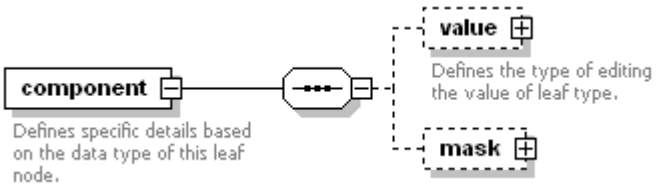
attributes

Name	Type	Use	Default	Fixed	Annotation
visible	xsd:boolean	optional	true		
columnWidth	xsd:float	optional			
indent	xsd:float	optional			
multiLine	xsd:boolean	optional	false		
useUnderline	xsd:boolean	optional	false		
icon	xsd:string	optional			
iconPosition	xsd:string	optional	left		
href	xsd:string	optional			

annotation documentation Defines the visual label for this leaf node.

element **leafType/component**

diagram



children **value** **mask**

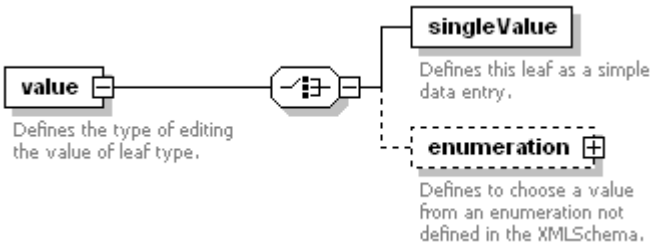
attributes

Name	Type	Use	Default	Fixed	Annotation
visible	xsd:boolean	optional	true		
columnWidth	xsd:float	optional			
indent	xsd:float	optional			
size	xsd:float	optional	-1.0		
booleanVisualizer	xsd:string	optional			
multiLine	xsd:int	optional	1		documentation -1 means use preferred size
showNewButton	xsd:boolean	optional	true		
showDetailsButton	xsd:boolean	optional	true		
showBackLinkButton	xsd:boolean	optional	true		
timeFormat	xsd:string	optional	hh:mm:ss		
href	xsd:string	optional			
useGroupingDelimiters	xsd:boolean	optional	false		

annotation documentation Defines specific details based on the data type of this leaf node.

element **leafType/component/value**

diagram



children **singleValue** **enumeration**

attributes

Name	Type	Use	Default	Fixed	Annotation
serializeContentAsCDATA	xsd:boolean		false		
dataType	xsd:string	optional			

annotation documentation Defines the type of editing the value of leaf type.

element **leafType/component/value/singleValue**

diagram

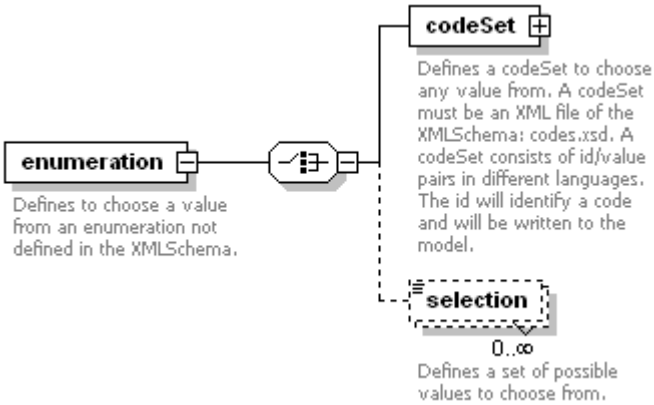
attributes

Name	Type	Use	Default	Fixed	Annotation
defaultValue	xsd:string	optional			

annotation documentation Defines this leaf as a simple data entry.

element **leafType/component/value/enumeration**

diagram



children **codeSet** **selection**

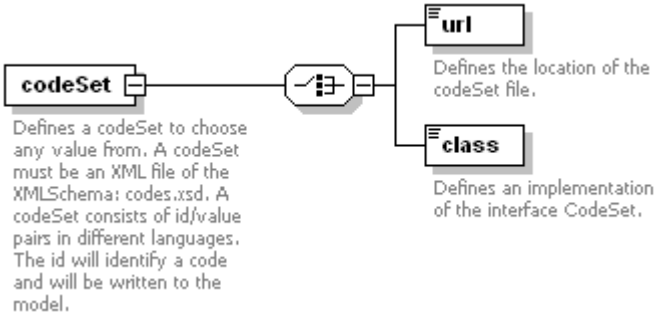
attributes

Name	Type	Use	Default	Fixed	Annotation
type	xsd:string	optional			
orientation	xsd:string	optional			
sort	xsd:string	optional			
defaultSelection	xsd:string	optional			

annotation documentation Defines to choose a value from an enumeration not defined in the XMLSchema.

element **leafType/component/value/enumeration/codeSet**

diagram



children **url** **class**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	xsd:string	required			
allowInvalidCodes	xsd:boolean	optional	false		

annotation documentation Defines a codeSet to choose any value from. A codeSet must be an XML file of the XMLSchema: codes.xsd. A codeSet consists of id/value pairs in different languages. The id will identify a code and will be written to the model.

element **leafType/component/value/enumeration/codeSet/url**

diagram

type **xsd:string**

annotation documentation Defines the location of the codeSet file.

element **leafType/component/value/enumeration/codeSet/class**

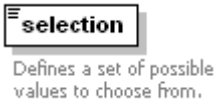
diagram

type **xsd:string**

annotation documentation Defines an implementation of the interface CodeSet.

element **leafType/component/value/enumeration/selection**

diagram

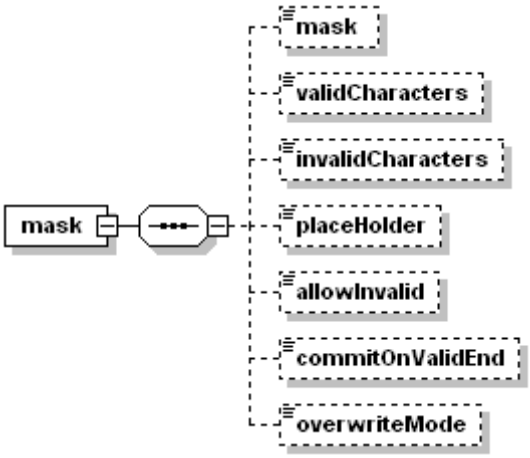


type **xsd:string**

annotation documentation Defines a set of possible values to choose from.

element **leafType/component/mask**

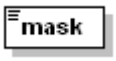
diagram



children mask validCharacters invalidCharacters placeHolder allowInvalid commitOnValidEnd overwriteMode

element **leafType/component/mask/mask**

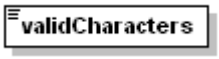
diagram



type **xsd:string**

element **leafType/component/mask/validCharacters**

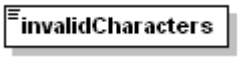
diagram



type **xsd:string**

element **leafType/component/mask/invalidCharacters**

diagram



type **xsd:string**

element **leafType/component/mask/placeHolder**

diagram

type **xsd:string**

element **leafType/component/mask/allowInvalid**

diagram

type **xsd:boolean**

element **leafType/component/mask/commitOnValidEnd**

diagram

type **xsd:boolean**

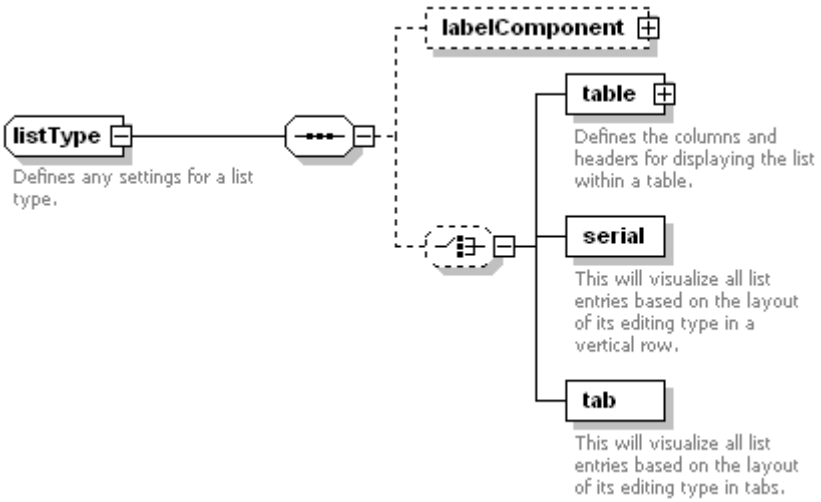
element **leafType/component/mask/overwriteMode**

diagram

type **xsd:boolean**

complexType **listType**

diagram



children **labelComponent table serial tab**

used by element **style/list**

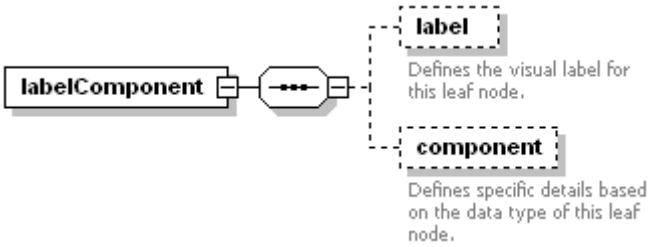
attributes

Name	Type	Use	Default	Fixed	Annotation
visible	xsd:boolean	optional	true		
autoCreateMinOccurListItems	xsd:boolean	optional			
showSequenceNumbering	xsd:boolean	optional	true		
columnWidth	xsd:int	optional	100		
visibleRowCount	xsd:int	optional	5		
allowReOrdering	xsd:boolean	optional	false		
allowSorting	xsd:boolean	optional	false		
selectionMode	xsd:string	optional	multi		
startupSelection	xsd:string	optional	first		
useZebraLook	xsd:boolean	optional	true		
selectionOnly	xsd:boolean	optional	false		
uniqueListItemIDPath	xsd:string	optional			
showButtons	xsd:boolean	optional	true		
showNewButton	xsd:boolean	optional	true		
showCopyButton	xsd:boolean	optional	true		
showDeleteButton	xsd:boolean	optional	true		
showEditButton	xsd:boolean	optional	true		
useDeletionConfirmation	xsd:boolean	optional	true		
deletionConfirmationMessage	xsd:string	optional			

annotation documentation Defines any settings for a list type.

element **listType/labelComponent**

diagram

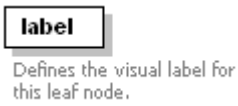


type restriction of **leafType**

children **label** **component**

element **listType/labelComponent/label**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
visible	xsd:boolean	optional	true		
columnWidth	xsd:float	optional			
indent	xsd:float	optional			
multiLine	xsd:boolean	optional	false		
useUnderline	xsd:boolean	optional	false		
icon	xsd:string	optional			
iconPosition	xsd:string	optional	left		

annotation
documentation Defines the visual label for this leaf node.

element **listType/labelComponent/component**

diagram

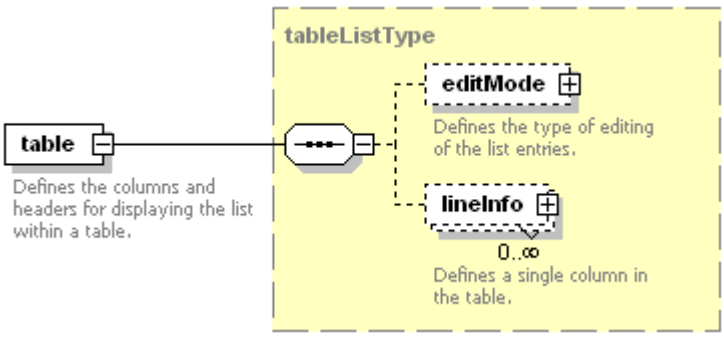
attributes

Name	Type	Use	Default	Fixed	Annotation
visible	xsd:boolean	optional	true		
columnWidth	xsd:float	optional			

annotation
documentation Defines specific details based on the data type of this leaf node.

element **listType/table**

diagram



type **tableListType**

children **editMode** **lineInfo**

attributes

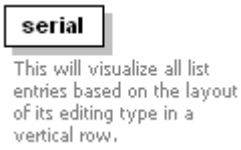
Name	Type	Use	Default	Fixed	Annotation
direction	xsd:string	optional	AS_ROW		

allowTableFlippingOnTheFly	xsd:boolean	optional	true
tablePosition	xsd:string	optional	north
columnHeaderLabellingStrategy	xsd:string	optional	com.jaxfront.core.ui.PrimitiveRowHeaderLabellingStrategy
tableModelClass	xsd:string	optional	com.jaxfront.core.ui.FlipOrientationTableModel
cellRendererClass	xsd:string	optional	com.jaxfront.swing.ui.beans.ZebraCellRenderer

annotation documentation Defines the columns and headers for displaying the list within a table.

element **listType/serial**

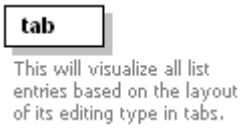
diagram



annotation documentation This will visualize all list entries based on the layout of its editing type in a vertical row.

element **listType/tab**

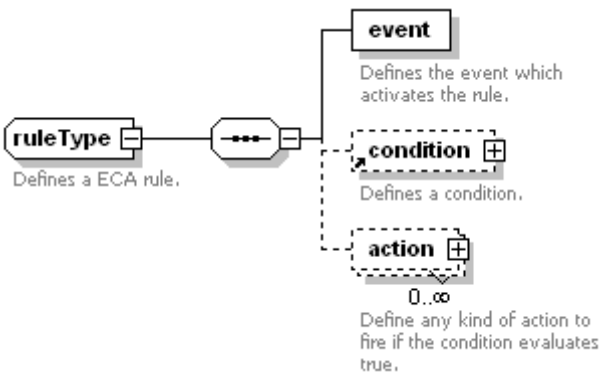
diagram



annotation documentation This will visualize all list entries based on the layout of its editing type in tabs.

complexType **ruleType**

diagram



children **event** **condition** **action**

used by elements **XUI/rule** **behaviour/rule**

attributes

Name	Type	Use	Default	Fixed	Annotation
priority	xsd:int	optional			

annotation documentation Defines a ECA rule.

element **ruleType/event**

diagram

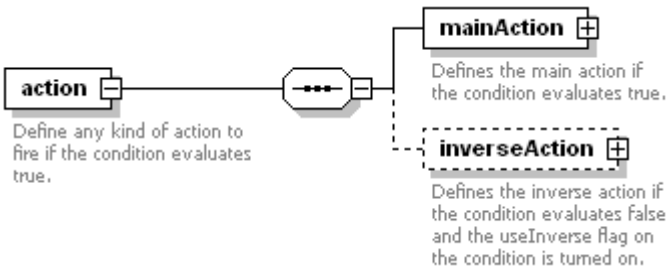
attributes

Name	Type	Use	Default	Fixed	Annotation
type	xsd:string	required			
executionMode	xsd:string	optional	clientAndServer		
scope	xsd:string	optional	sourceAndReferencedNodes		
doesConcernNoneSerializableSource	xsd:boolean	optional	true		
doesConcernOnlyVisibleSource	xsd:boolean	optional	false		

annotation documentation Defines the event which activates the rule.

element **ruleType/action**

diagram

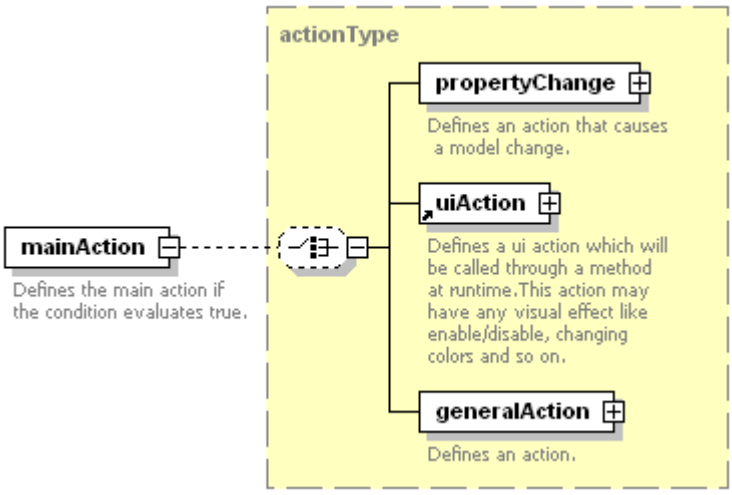


children **mainAction** **inverseAction**

annotation documentation Define any kind of action to fire if the condition evaluates true.

element **ruleType/action/mainAction**

diagram



type extension of **actionType**

children **propertyChange** **uiAction** **generalAction**

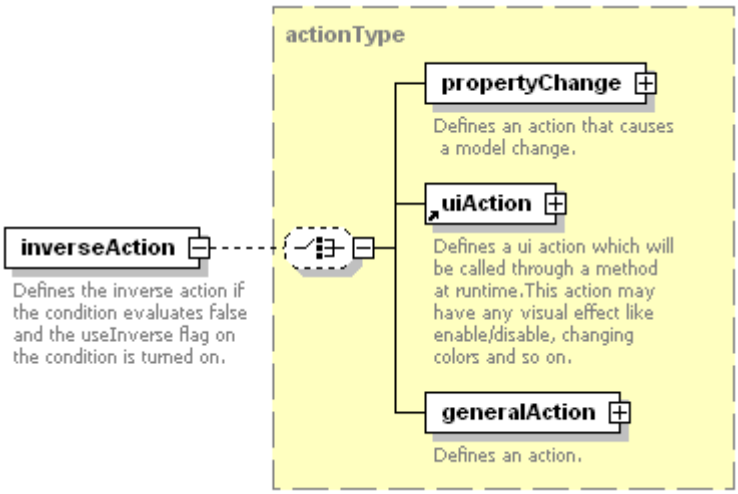
attributes

Name	Type	Use	Default	Fixed	Annotation
target	xsd:string	optional			
includeOnlySerializableNodes	xsd:boolean	optional	false		
className	xsd:string	optional			

annotation documentation Defines the main action if the condition evaluates true.

element **ruleType/action/inverseAction**

diagram



type extension of **actionType**

children **propertyChange** **uiAction** **generalAction**

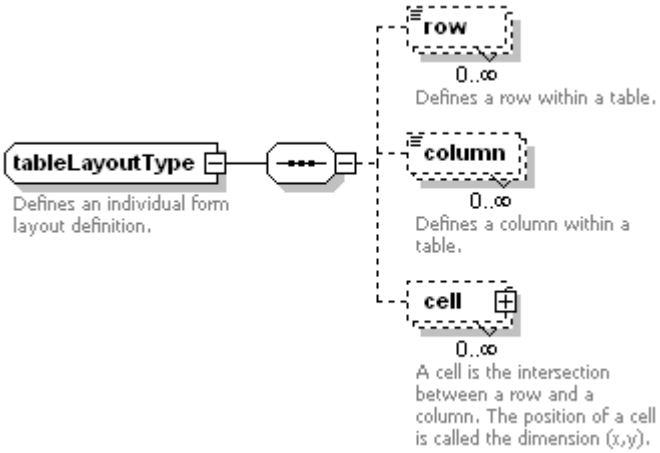
attributes

Name	Type	Use	Default	Fixed	Annotation
target	xsd:string	optional			
includeOnlySerializableNodes	xsd:boolean	optional	false		
className	xsd:string	optional			

annotation
documentation Defines the inverse action if the condition evaluates false and the useInverse flag on the condition is turned on.

complexType **tableLayoutType**

diagram



children **row** **column** **cell**

used by
elements **style/layout/table** **tableLayout**

attributes

Name	Type	Use	Default	Fixed	Annotation
optional	xsd:boolean	optional	true		
id	xsd:string	optional	main		

annotation
documentation Defines an individual form layout definition.

element **tableLayoutType/row**

diagram

type **xsd:string**

annotation documentation Defines a row within a table.

element **tableLayoutType/column**

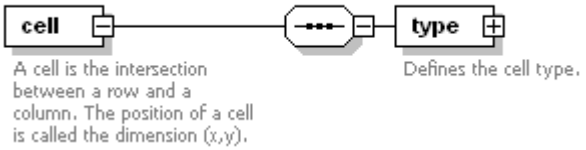
diagram

type **xsd:string**

annotation documentation Defines a column within a table.

element **tableLayoutType/cell**

diagram



children **type**

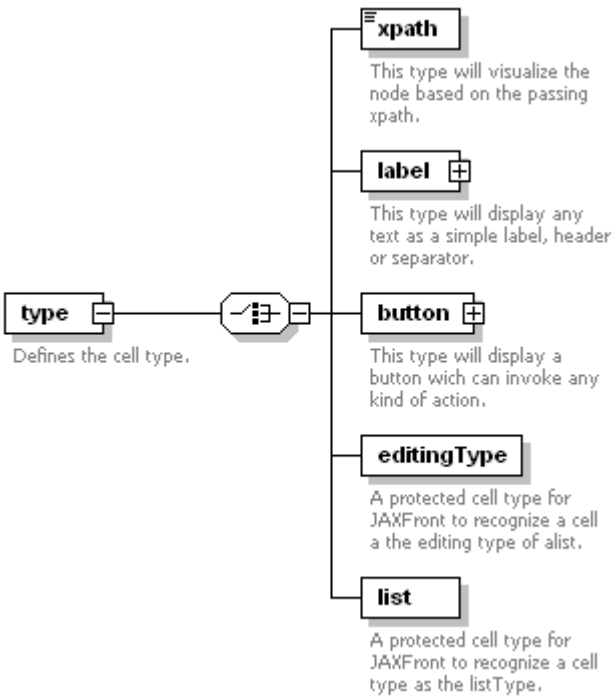
attributes

Name	Type	Use	Default	Fixed	Annotation
dimension	xsd:string	required			
focusIndex	xsd:integer	optional			

annotation documentation A cell is the intersection between a row and a column. The position of a cell is called the dimension (x,y).

element **tableLayoutType/cell/type**

diagram



children **xpath label button editingType list**

annotation documentation Defines the cell type.

element **tableLayoutType/cell/type/xpath**

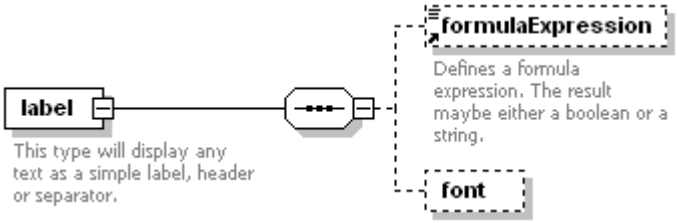
diagram

type **xsd:string**

annotation documentation This type will visualize the node based on the passing xpath.

element **tableLayoutType/cell/type/label**

diagram



children **formulaExpression** **font**

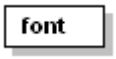
attributes

Name	Type	Use	Default	Fixed	Annotation
labelType	xsd:string	optional	label		
xpath	xsd:string	optional			
text	xsd:string	optional			
url	xsd:string	optional			

annotation documentation This type will display any text as a simple label, header or separator.

element **tableLayoutType/cell/type/label/font**

diagram



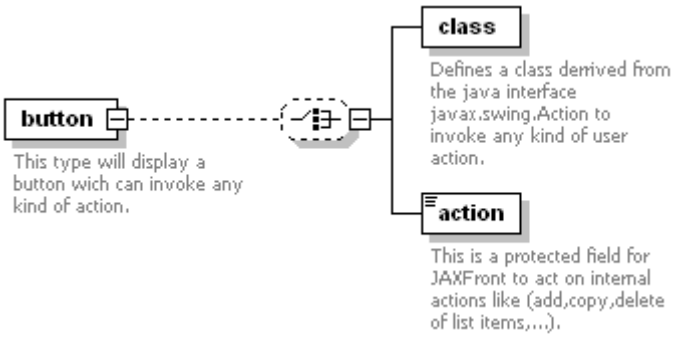
type **fontType**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	xsd:string	optional			
size	xsd:int	optional			
style	xsd:string	optional			
color	colorType	optional			

element **tableLayoutType/cell/type/button**

diagram



children **class** **action**

attributes

Name	Type	Use	Default	Fixed	Annotation
isSelector	xsd:boolean	optional	false		
ruleIndex	xsd:string	optional			

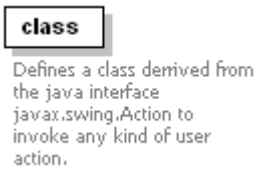
identity constraints

	Name	Refer	Selector	Field(s)
keyref	ruleIndexRef	ruleId	.	@ruleIndex

annotation documentation This type will display a button wich can invoke any kind of action.

element **tableLayoutType/cell/type/button/class**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
className	xsd:string	required			
buttonName	xsd:string	required			

annotation documentation Defines a class derrived from the java interface javax.swing.Action to invoke any kind of user action.

element **tableLayoutType/cell/type/button/action**

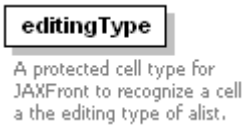
diagram

type **xsd:string**

annotation documentation This is a protected field for JAXFront to act on internal actions like (add,copy,delete of list items,...).

element **tableLayoutType/cell/type/editingType**

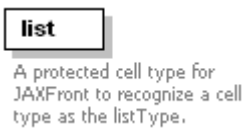
diagram



annotation documentation A protected cell type for JAXFront to recognize a cell a the editing type of alist.

element **tableLayoutType/cell/type/list**

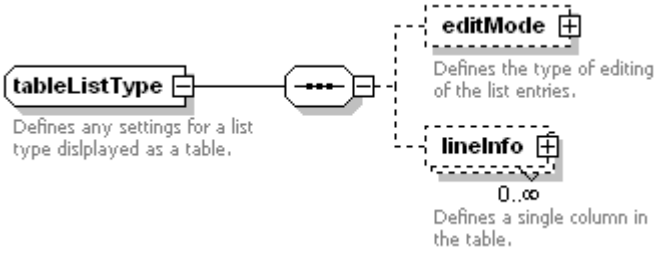
diagram



annotation documentation A protected cell type for JAXFront to recognize a cell type as the listType.

complexType **tableListType**

diagram



children **editMode** **lineInfo**

used by element **listType/table**

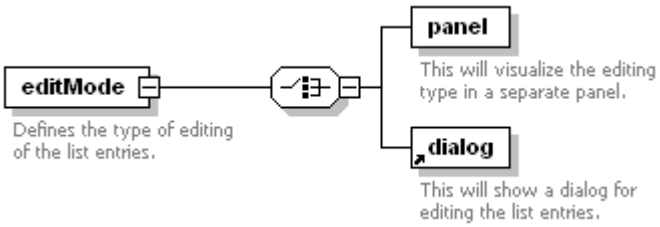
attributes

Name	Type	Use	Default	Fixed	Annotation
direction	xsd:string	optional	AS_ROW		
allowTableFlippingOnTheFly	xsd:boolean	optional	true		
tablePosition	xsd:string	optional	north		
columnHeaderLabellingStrategy	xsd:string	optional	com.jaxfront.core.ui.PrimitiveRowHeaderLabellingStrategy		
tableModelClass	xsd:string	optional	com.jaxfront.core.ui.FlipOrientationTableModel		
cellRendererClass	xsd:string	optional	com.jaxfront.swing.ui.beans.ZebraCellRenderer		

annotation documentation Defines any settings for a list type dislplayed as a table.

element **tableListType/editMode**

diagram



children **panel** **dialog**

annotation documentation Defines the type of editing of the list entries.

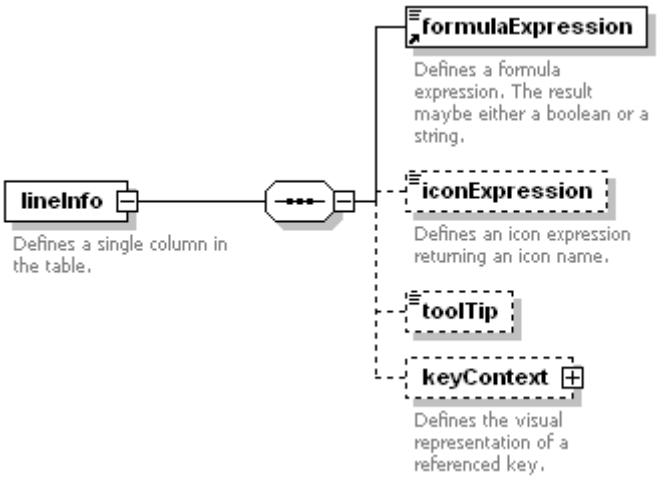
element **tableListType/editMode/panel**

diagram

annotation documentation This will visualize the editing type in a separate panel.

element **tableListType/lineInfo**

diagram



children **formulaExpression** **iconExpression** **toolTip** **keyContext**

attributes

Name	Type	Use	Default	Fixed	Annotation
width	xsd:int	optional	100		

annotation documentation Defines a single column in the table.

element **tableListType/lineInfo/iconExpression**

diagram

type **xsd:string**

annotation documentation Defines an icon expression returning an icon name.

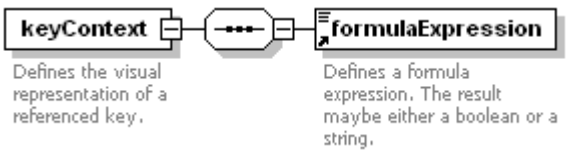
element **tableListType/lineInfo/toolTip**

diagram

type **xsd:string**

element **tableListType/lineInfo/keyContext**

diagram



children formulaExpression

attributes

Name	Type	Use	Default	Fixed	Annotation
allowContextNavigation	xsd:boolean	optional	false		

annotation documentation Defines the visual representation of a referenced key.

attributeGroup **enumerationType**

used by elements style/choice leafType/component/value/enumeration

attributes

Name	Type	Use	Default	Fixed	Annotation
type	xsd:string	optional			
orientation	xsd:string	optional			

annotation documentation Defines the type and the orientation of any enumeration.

attributeGroup **scopeGroup**

used by elements ruleType/event behaviour/log

attributes

Name	Type	Use	Default	Fixed	Annotation
scope	xsd:string	optional	sourceAndReferencedNodes		

annotation documentation Defines the scope within a subtree.

element **dummyRoot**

diagram

complexType **anchorType**

diagram

attributes

Name	Type	Use	Default	Fixed	Annotation
id	xsd:string	required			
displayName	xsd:string	required			

	href	xsd:string	optional

simpleType **colorType**

type **xsd:string**

used by
- elements **XUI/global/style/component/applicationRequiredColor**, **XUI/global/style/caption/header/endColor**, **XUI/global/errorDisplay/errorColor**, **XUI/global/style/component/optionalColor**, **XUI/global/style/component/schemaRequiredColor**, **XUI/global/style/caption/header/startColor**, **XUI/global/style/caption/header/textColor**
- attribute **fontType/@color**

annotation
- documentation Defines the RGB value for a color (eq. 244,212,233).

6 Codes schema specification (codes.xsd)

This section talks about Codes schema specification.

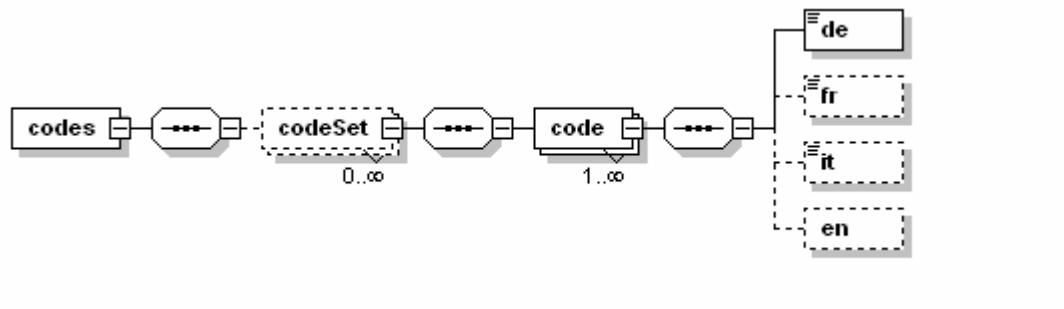


Figure 120: Codes schema specification (codes.xsd)

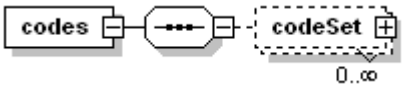
Schema **codes.xsd**

(Generated XML schema documentation)

Elements
codes

element **codes**

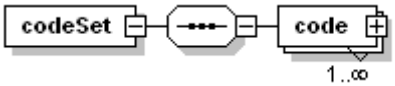
diagram



children codeSet

element **codes/codeSet**

diagram



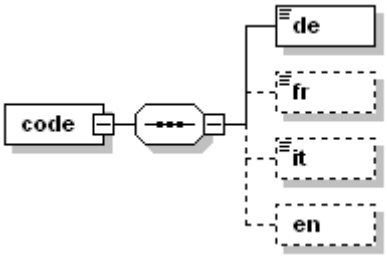
children code

attributes

Name	Type	Use	Default	Fixed
name	xsd:string	required		

element **codes/codeSet/code**

diagram



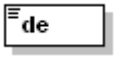
children **de fr it en**

attributes

Name	Type	Use	Default	Fixed
id	xsd:string	required		
status	xsd:string	optional	valid	

element **codes/codeSet/code/de**

diagram



type **xsd:string**

element **codes/codeSet/code/fr**

diagram

type **xsd:string**

element **codes/codeSet/code/it**

diagram

type **xsd:string**

element **codes/codeSet/code/en**

diagram