Action

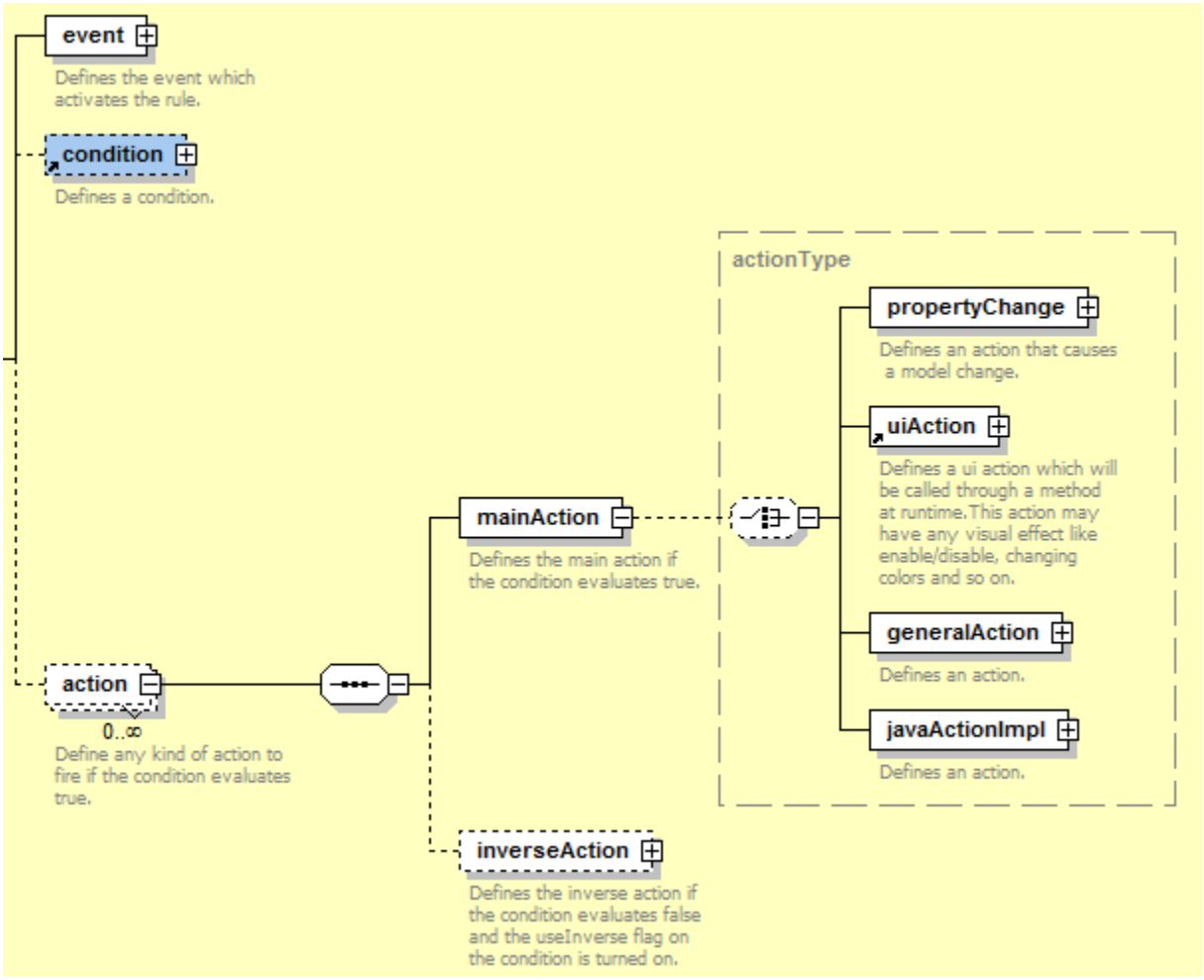
An *action* defines an act implemented when an event occurred and a relevant condition was examined. This action is meant for a firmly determined target, which concerns a single node or multiple nodes. To indicate the target, you need to formulate the XPath expression.

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).



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..

Types of actions

There are three different types of actions:

Type of action	Description
Property change action (propertyChange)	Sets the formula result to every action target element. The model changes are applied on simple types only. If the XPath addresses a complex node, the action is ignored.
UI action	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*
Formula action (generalAction)	Triggers any kind of action.
Java action (javaActionImpl)	Calls a java action implementation (must implement Action interface) through java reflection api. Be sure this class implmenents the jaxfront action interface (com.jaxfront.core.rule.Action)

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 change of model

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

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.

The following example shows an action with a visual event

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

Use general action if you which to fire a formula (JEP expression).

The following example shows an action with a visual event

```
<generalAction>
<formulaExpression>any alid formula comes here<formulaExpression/>
</generalAction>
```

Use a javaActionImpl in case you want to call your own action implementation.

The following example shows an action with a visual event

```
<javaActionImpl className="com.mypackage.myClass">
<param name="name1" type="String">my String<param/>
</javaActionImpl>
```

The parameters are used to find an according constructor in your action implementation class. In the case above the must be a constructor with on string parameter value:

```
package com.mypackage
import com.jaxfront.core.rule.Action;

public class MyClass implements Action
        public MyClass(String anyString) {

        }
}
```