Controlling the XML Serialization With Stereotypes

Using stereotypes, you can control how classes from your data model are serialized to an XML document, and build a class structure that implements an XML Schema. If no stereotype is assigned, the following default rules apply:

Level	Rule
general	The XML root element is named after the name of the object that contains the data to be mapped.
class	Each class who's type is not derived from a built-in simple type (e.g. **String**) is serialized to an XML element.
attribute	Each attribute of simple type (or derived from a simple type) is serialized to an XML attribute of the XML element defined by the containing class.
	Each attribute of complex type is serialized to an XML element.

On this Page:

Stereotypes

To control the serialization, the following stereotypes are available:

Stereotype	Description	Additional Attributes
Class		
XML	Specify this stereotype to set one of the attributes listed on the right on class level. You need to set **isMixed** and **isOrdered**, if you want to build mixed content (see example Serializing Mixed Content XML to a Class Structure below) or order XML elements.	**xmlNamespace** **xmlElementName** **isMixed** **isOrdered**
Property		
XMLAttribute	When this stereotype is applied to a class property, it will be serialized to an XML attribute. The property must be of simple type. If the property type is complex, the xUML Compiler will report an error.	**xmlNamespace** **xmlForm** **xmlFormat** **externalName** **xmlType**
XMLCharacters	When this stereotype is applied to a class property, its content will be serialized to a character stream. The property must be of simple type.	**xmlFormat** **order**
XMLElement	When this stereotype is applied to a class property, it will be serialized to an XML element.	**xmlNamespace** **xmlForm** **xmlFormat** **isNillable** **xmlArrayElement** **order** **externalName** **xmlType**
XMLNamespace	When this stereotype is applied to a class property, it will be serialized to a XML namespace. The property must be of base type **String**. • The prefix of the namespace is given by the property name. • The URI of the namespace is given in the property value. • If the same namespace is declared more than once, the runtime will suppress namespaces further down the XML hierarchy, if prefix and namespace are identical. If not, the xUML Runtime will throw an exception.	

Attributes

Tagged Value	Stereotype	Description	Allowed Values	
Classes				
xmlNamespace	XML	Specify the XML namespace.	a valid namespace	
xmlElementName	XML	Specify the name of the XML root element.	a valid element name	
isMixed	XML	Specify whether the XML contains mixed content, means attributes that are serialized as character stream (attributes with stereotype XMLCharacters, see also Stereotypes above).	true	XML contains serialized attributes.
			false	XML does not contain serialized attributes (default).
isOrdered	XML	Specify whether the class properties should be serialized to XML using the **order** tag that has been specified on the properties.	true	Serialize in order of **order** tags from the properties.
			false	Serialize in order of property definition on class (default).
Property				
xmlNamespace	XMLElement XMLAttribute	Each XML attribute and element may have its own namespace. If this attribute contains an URI, the Runtime will automatically generate a unique prefix. Examples: • An **xmlNamespace** = " http://scheer-acme.com " on a property **anElement** will result in the XML document `<ns0:anElement xmlns:ns0=" http://scheer-acme.com ">`. • Also, it is possible to define the prefix by using the following syntax: `'xmlns:'<prefix name>'="<namespace uri>"`. For example, tagged value **xmlNamespace** = 'xmlns:typens=" http://scheer-acme.com '" of a property **anElement** will lead to the following XML fragment: `<typens:anElement xmlns:typens=" http://scheer-acme.com ">`.	an URI	
			a valid xlmns syntax	
xmlForm	XMLElement XMLAttribute	Depending on this tagged value, XML elements or attributes may not be qualified by a namespace prefix even if they have one.	qualified	The element or attribute must always be qualified by a namespace prefix. **Default for XML elements.**
			unqualified	No namespace prefixes are allowed (for details see http://www.w3.org/TR/xmlschema-0/#NS). **Default for XML attributes.**

xmlFormat	XMLElement
	XMLAttribute
	XMLCharacters
		If numbers and date/time types are parsed or composed, the XML parser respectively composer expects simple date types following the XML schema specification. However, legacy XML documents may contain different number and date/time formats. In this case, the tagged value **xmlFormat** may hold a format string.
		
		• If numbers are parsed or composed use the format strings defined in section Number Formatting.
		• If date/time expressions must be parsed or composed, use the format strings defined in Date and Time Formatting.
		  ◦ Use xmlFormat = "CDATA" together with stereotype XMLElement to compose strings as CDATA with classToXML() Operation.
		  ◦ *Parsing* CDATA elements works out of the box, you do not need to set **xmlFormat**.
			a valid format string (see Number Formatting or Date and Time Formatting)	

			CDATA	Compose string as CDATA.
isNillable	XMLElement	By default, properties that are NULL are not serialized into XML documents. However, if it is necessary to do so, you can set **isNillable** to true. In this case, NULL properties will look like: `<aProperty xsi:nil="true"></aProperty>`. Some client code generators will use this attribute in the WSDL file for type generation. If false, they will generate simple types, if true they will generate complex types.	true	Serialize NULL properties.
			false (default)	Do not serialize NULL properties.
order	XMLElement XMLCharacters	Use **order** to specify the order in which the XML elements will be generated to the XML document.	a valid float	
externalName	XMLElement XMLAttribute	Sometimes names of XML attributes or elements do not comply with the rules specified on Syntax Scheme of the xUML Action Language. Use **externalName** to specify the name of the XML attribute or element, and specify a valid name for the property for internal usage.		

Examples

XML_Serialization_Example

Click the icon to download a simple example model that shows how to control XML mappings with stereotypes in **Scheer PAS** *Designer*.

Serializing Mixed Content XML to a Class Structure

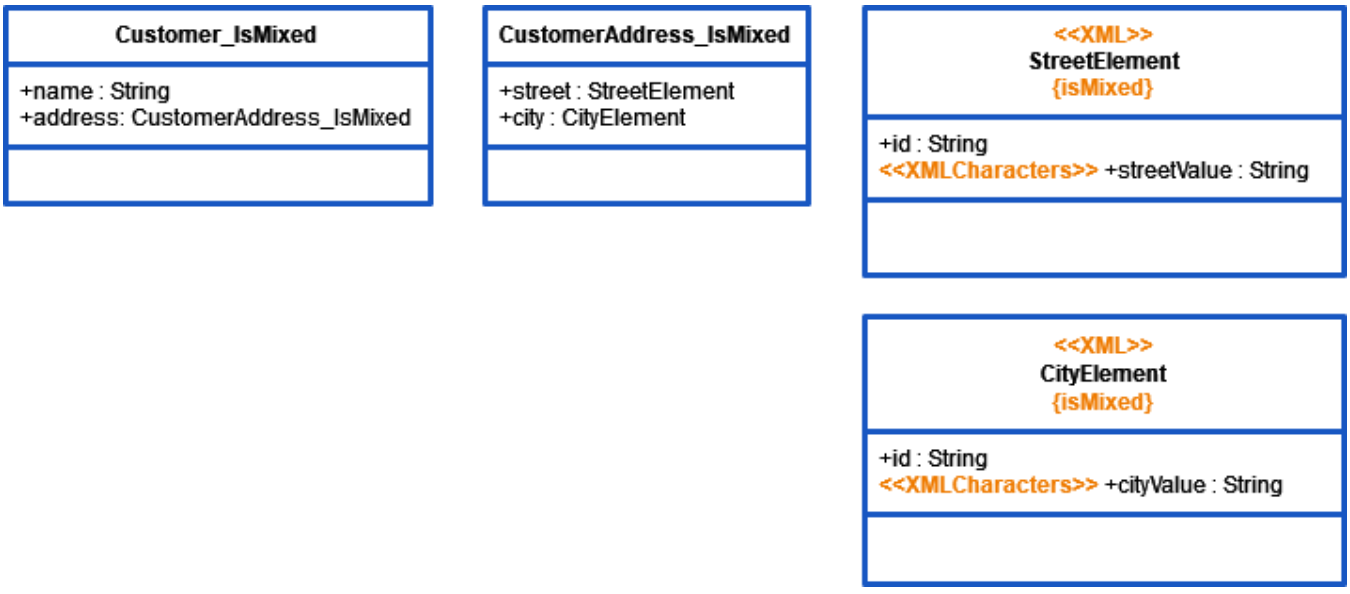
The following example shows how to depict an XML data structure with mixed content with a class in the data model of the Designer.

> If an XML element has attributes, this is called **mixed content** in Designer context.

Suppose you want to build the following customer XML data structure, with two elements **street** and **city**:

```
<?xml version="1.0" encoding="utf-8"?>
<customerIsMixed name="Summer Ltd.">
        <address>
                <street id="Summer Ltd.">456, Sunny Beach</street>
                <city id="Summer Ltd.">Miami, FL 33101</city>
        </address>
</customerIsMixed>
```

This is mixed content XML because the **street** and **city** elements have attributes (`id="Summer Ltd."`) and element (`456, Sunny Beach`) values. The class diagram below illustrates how you need to build the class structure to serialize the mixed content. The stereotypes you should specify, and the related attributes you should set are highlighted in orange.

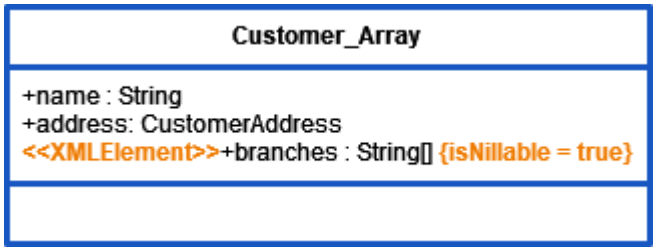


XML Element /Attribute	Representation in Class Structure		
address	To serialize the XML element **address**, you need a dedicated class (`CustomerAddress_IsMixed` in this case).		

street and **city**	**address** contains the sub-element **street**. To serialize XML element **street** and its attribute **id**, you need a dedicated class as a container (class StreetElelement in the example class diagram). XML attribute **id** of element **street** is serialized to class property **id** of class **StreetElement** via matching names. The content of the XML element **street** is serialized to the class property that has stereotype **XMLCharacters**. In the example, this is **streetValue**. Class **StreetElement** must have stereotype **XML** applied with attribute **isMixed** set to true because it contains the serialized attribute **st reetValue** (also see explanation of attribute isMixed above). Container **StreetElement** is associated to the main address class. The name of the association end (**street**) must match the name of the XML element. Associated class **StreetElement** is only a container for the **street** element and its name is not relevant for the serialization. The same applies to **city**.		

Arrays and XML Serialization

The behavior of XML serialization in this case is not always self-explanatory. This is a consequence of the definition of arrays in XML schema. The table below shows the behavior of XML serialization for the following class structure containing an array (branches). The necessary stereotypes and attributes are highlighted in orange.




```
{
        "customerArray" : {
                "name" : "Summer Ltd.",
                "branches" : ["Miami", "Orlando",
"Tampa"]
        }
}
```

Description	XML result

All values are present.	`<customerArray name="Summer Ltd.">`
	`  <branches>Miami</branches>`
	`  <branches>Orlando</branches>`
	`  <branches>Tampa</branches>`
	`</customerArray>`

The second value is NULL and **isNillable** =false.	`<customerArray name="Summer Ltd.">`
	`  <branches>Miami</branches>`
	`  <branches>Tampa</branches>`
	`</customerArray>`

The second value is NULL and **isNillable** =true.	`<customerArray xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="`
	`Summer Ltd.">`
	`  <branches>Miami</branches>`
	`  <branches xsi:nil="true"/>`
	`  <branches>Tampa</branches>`
	`</customerArray>`

The array is NULL or the array is empty or all elements are NULL and **isNillable**=fal se.	`<customerArray name="Summer Ltd.">` `</customerArray>`

All elements are NULL and **isNillable**=true.	<customerArray xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Summer Ltd."> <branches xsi:nil="true"/> <branches xsi:nil="true"/> <branches xsi:nil="true"/> </customerArray>