xmlToClass Strings

Syntax		`set anObject = aString.xmlToClass()`
		`set anObject = xmlToClass(LITERAL)`
		`set anObject = aString.xmlToClass(xmlOptions)`

Semantics		The operation takes an XML string (`aString`) and tries to map the XML document to `anObject`. If this is not possible, an error is raised (e.g. XML parser errors, invalid mappings, etc.). By default, the following mapping rules apply: • XML attributes are mapped to class attributes. • XML elements are mapped to class associations ends. These default rules can be overridden by using <<XMLElement>>, <<XMLAttribute>>, and <<XMLCharacters>> stereotypes on UML class attributes and UML association ends. More about these mapping rules, stereotypes and tagged values (e.g. for number and date & time formatting) can be found in chapter XML / UML Class Mapping. Frequently, an XML document is given as blob instead of a string (for instance after receiving it from the file system adapter). In such cases, it is possible to apply xmlToClass() to a **Blob** as well. For details see xmlToClass() for Blobs.
Substitutables	`aString`	An XML document as **String**.
	`anObject`	Target object, can be any object.
	`LITERAL`	String literal.
	`xmlOptions`	This is an optional parameter of type **Base Components::Basic Behavior::XML::XMLOptions**. This parameter controls the parsing behavior. For example, it defines whether the XML document is validated against an XML schema. The available options are explained below.

On this Page:

Related Pages:

- xmlToClass() Operation for Blobs
- Deploying and Managing Resources

Examples

```
set myAddress = addressAsXMLDocument.xmlToClass();
```

The action script below creates an object of type **Address**. An output object flow state named **myAddress** of type **Address** needs to be defined in the activity diagram.

```
create myAddress;
set myAddress = addressAsXMLDocument.xmlToClass();
```

Beneath, a sample XML document is shown to illustrate the mapping executed by the `xmlToClass()` operation. The XML document is mapped to an instance of **Address** as shown in the class diagram.

```
        <myAddress
id="myAddressID">

<street>Lautengarte
nstr. 12</street>

<city>Basel</city>
        <
/myAddress>
```

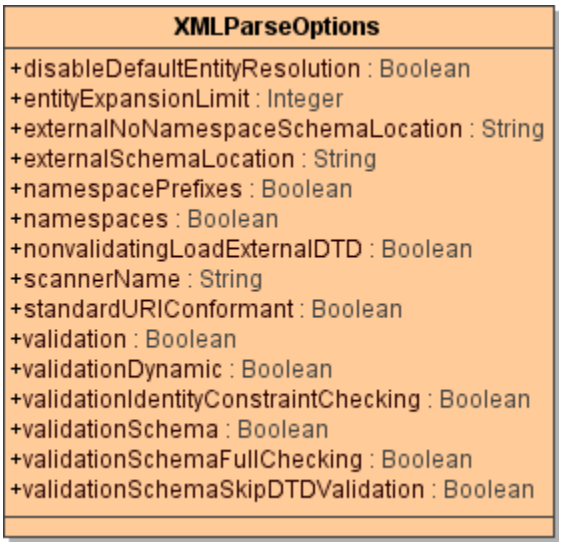
Figure: Class Diagram for XML Mapping

Note, that the XML element myAddress is mapped to the object **myAddress**, which is of type **Address**. This type has the UML attribute **id** which corresponds to the XML attribute id. Additionally, the XML elements street and city are mapped to the assocition ends **city** respectivly **street**. Both are having the type **String**.

XML Parsing Options (Validation against a Schema)

The optional parameter of `xmlToClass()` of type **XMLOptions** offers various parameters to control schema and DTD location and validation:

Figure: XML Parse Options Class

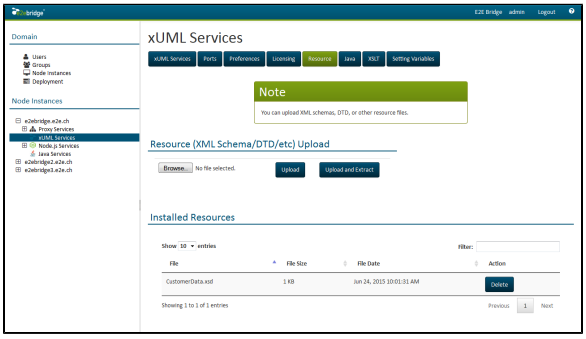


Be aware that by default, schemas are parsed, but documents are not validated against them. Set the **val idation** attribute to **true** if you want to enforce validation beyond well-formedness. For example, assume that you want to validate your document against a schema called **CustomerData.xsd** having the namespace **http://example.com/customer** . Then, the following **xmlOption** attributes have to be set:

```
externalSchemaLocation ="http://example.com/customer CustomerData.xsd";
validation = true;
```

If the XML document refers a schema file (.xsd) with filename (and optional path), it is sufficient to put the schema file into the folder **<e2e data>/resource**. The E2E Bridge will automatically load this one instead of the specified. Using the administration interface of the Bridge, it is possible to load XML schemas into this folder remotely by using the following dialog:

Figure: Uploading XML Schemas



Refer to Deploying and Managing Resources in the Bridge User's Guide for more information on uploading of resources to the E2E Bridge.

If the XML document refers no schema or you would like to provide another than the referred one, use the **XMLParseOptions** structure and set **externalSchemaLocation** (or **externalNoNamespaceSchemaLocation**) accordingly. As above, missing or relative paths will be redirected to the folder **<e2e data>/resource**. The same applies also for validating against DTDs. Be aware that validation is turned off by default (see option **validation** below).

The following table lists all available XML options. Default values used when an option is not explicitly set are written in bold. Since the Bridge is using the Xerces parser internally, more information for all options can be found on the Xerces home page by following the **read more** link in the **Description** column.

Option	Description	Values	
validation	If this feature is set to **true**, the document must specify a grammar. If this option is set to **false** and document specifies a grammar, that grammar might be parsed but no validation of the document contents will be performed. **read more**	true	Report all validation errors.
		false	Do not report validation errors.
validationDynamic	**read more**	true	The parser will validate the document only if a grammar is specified. (**validation** must be **true**).
		false	Validation is determined by the state of the **validation** option.
validationSchema	If set to **true**, the option **namespaces** must also be turned on. **read more**	**true**	Enable the parser's schema support.
		false	Disable the parser's schema support.
validationSchemaFullChecking	This option checks the schema grammar itself for additional errors that are time-consuming or memory intensive. It does **not** affect the level of checking performed on document instances that use schema grammars. **read more**	true	Enable full schema constraint checking, including checking which may be time-consuming or memory intensive. Currently, particle unique attribution constraint checking and particle derivation restriction checking are controlled by this option.
		false	Disable full schema constraint checking.

nonvalidatingLoadExternalDTD	This feature is ignored and DTD is always loaded when the option **validation** is **true.** **read more**	true	Load external DTD.
		false	Ignore external DTD completely.
standardURIConformant	If set to **true**, malformed URI will be rejected and fatal error will be issued. **read more**	true	Force standard URI conformance.
		false	Do not force standard URI conformance.
validationIdentityConstraintChecking	**read more**	**true**	Enable identity constraint checking.
		false	Disable identity constraint checking.
validationSchemaSkip-DTDValidation	**read more**	true	When **validationSchema** is **true** the parser will ignore the DTD, except for entities.
		false	The parser will not ignore DTDs when validating.
disableDefaultEntityResolution	**read more**	true	The parser will **not** attempt to resolve the entity if the E2E Bridge can't find it.
		false	The parser will attempt to resolve the entity on its own if the E2E Bridge can't find it.
namespaces	If the **validation** option is set to **true**, then the document must contain a grammar that supports the use of namespaces. **read more**	**true**	Perform Namespace processing.
		false	Do not perform Namespace processing.
namespacePrefixes	**read more**	true	Report the original prefixed names and attributes used for Namespace declarations.
		false	Do not report attributes used for Namespace declarations, and optionally do not report original prefixed names.
externalSchemaLocation	The XML Schema Recommendation explicitly states that the inclusion of schemaLocation/noNamespaceSchemaLocation attributes in the instance document is only a hint; it does not mandate that these attributes must be used to locate schemas. Similar situation happens to <import> element in schema documents. This property allows the user to specify a list of schemas to use. If the targetNamespace of a schema specified using this method matches the targetNamespace of a schema occurring in the instance document in schemaLocation attribute, or if the targetNamespace matches the namespace attribute of <import> element, the schema specified by the user using this property will be used (i.e., the schemaLocation attribute in the instance document or on the <import> element will be effectively ignored). **read more**	The syntax is the same as for schemaLocation attributes in instance documents: e.g, " http://www.example.com file_name.xsd". The user can specify more than one XML Schema in the list.	
externalNoNamespaceSchemaLocation	The XML Schema Recommendation explicitly states that the inclusion of schemaLocation/noNamespaceSchemaLocation attributes in the instance document is only a hint; it does not mandate that these attributes must be used to locate schemas. This property allows the user to specify the no target namespace XML Schema Location externally. If specified, the instance document's noNamespaceSchemaLocation attribute will be effectively ignored. **read more**	The syntax is the same as for the noNamespaceSchemaLocation attribute that may occur in an instance document: e.g." file_name.xsd".	
scannerName	This property allows the user to specify the name of the XMLScanner to use for scanning XML documents. **read more**	The recognized scanner names are:	

		WFXMLScanner	A scanner that performs well-formedness checking only.
		DGXMLScanner	A scanner that handles XML documents with DTD grammar information.
		SGXMLScanner	A scanner that handles XML documents with XML schema grammar information.
		IGXMLScanner	A scanner that handles XML documents with DTD or/and XML schema grammar information.
entityExpansionLimit	To mitigate an entity expansion attack (aka "XML bomb" or "the billion laughs" attack) you use this tagged value to limit entity expansion to the specified level.	Any integer, no default. If using this tagged value, provide at least value **1**. Otherwise the standard XML entities will not be parsed.	