String Operations

The Bridge supports string manipulation operations. These operations are offered by the **String** class:

- capture() Operation
- concat() Operation
- convertBase64ToBlob() Operation
- convertDurationToDateTime() Operation
- convertHexToBlob() Operation
- convertToBoolean Operation()
- convertToDateTime() Operation
- convertToFloat() Operation for Strings
- convertToInteger() Operation for Strings
- endsWith() Operation
- escapeURI() Operation
- extendedJSONToClass() Operation
- findPattern() Operation
- findPatterns() Operation
- findString() Operation
- jsonToClass() Operation
- match() Operation
- normalizeSpaces() Operation
- padLeft() Operation
- padRight() Operation
- parseDateTimeExpression() Operation
- parseFloatExpression() Operation
- parseIntegerExpression() Operation
- parseLocalDateTimeExpression() Operation
- removeAccents() Operation
- replace() Operation
- split() Operation
- startsWith() Operation
- stringLength() Operation
- substring() Operation
- substringAfter() Operation
- substringBefore() Operation
- toASCII() Operation
- toLower() Operation
- toUpper() Operation
- transcodeToBlob() Operation
- transliterate() Operation
- unescapeURI() Operation
- xmlToClass() Operation for Strings

Internally, strings are represented as UTF-8 strings.

Example File (Builder project E2E Action Language/BaseTypes):

<your example path>\E2E Action Language\BaseTypes\uml\string.xml

On this Page:

- Escaping Characters
- UML Examples
  - Example of Basic String Operations
  - Type Converter Operations

Related Pages:

- Base Types
- Text Macro
- Blob Operations

Escaping Characters

String literals are created by putting character sequences into single (**'...'**) or double quotes (**"..."**). If the string literal contains double quotes, they can be used without escaping if the literal is enclosed by single quotes and vice versa:

`"myString'StringWithinQuotes'end"` or `'myString"StringWithinDoubleQuotes"end'`

An alternative to have quotes within a string is escaping the quotes with a backslash (`\"` or `\'`):

`"myString\"StringWithinDoubleQuotes\"end"` or `'myString\'StringWithinDoubleQuotes\'end'`

Another backslash character is quoting the backslash character itself (`\\`).

The following sections explain the various **String** operations.

If you want to use white spaces like tabs, new lines, or carriage returns in a string, use the text() macro described in Text Macro.

UML Examples

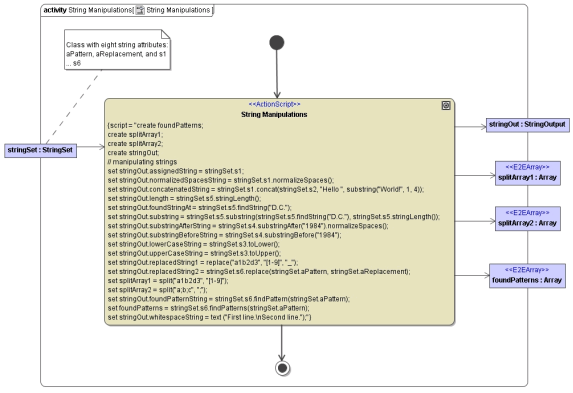
Example of Basic String Operations

Example File (Builder project E2E Action Language/BaseTypes):

<your example path>\E2E Action Language\BaseTypes\uml\string.xml

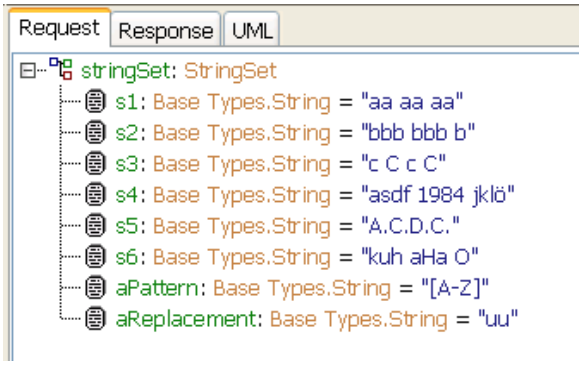
Figure *String Manipulation Operations* shows several examples of string manipulations with the Bridge.

Figure: String Manipulation Operations



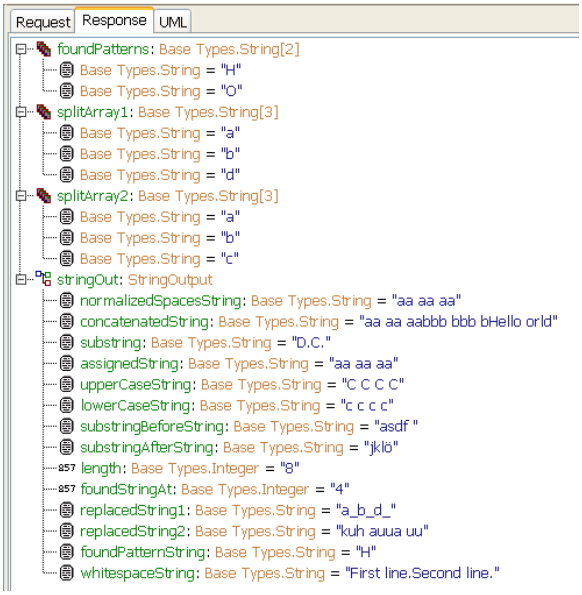
If you used the following input in Figure Input Values of the String Example for the example above, you would receive the output shown in Figure Response of the String Example.

Figure: Input Values of the String Example



Running a test case with these input values would result in the following output.

Figure: Response of the String Example



The value of attribute whitespaceString in figure *Response of the String Example* is displayed in one line on the Response tab of the E2E SOAP Test Tool. Actually, there is a line break before the text second line, which would be visible when browsing the runtime values on the UML tab of the SOAP Test Tool.

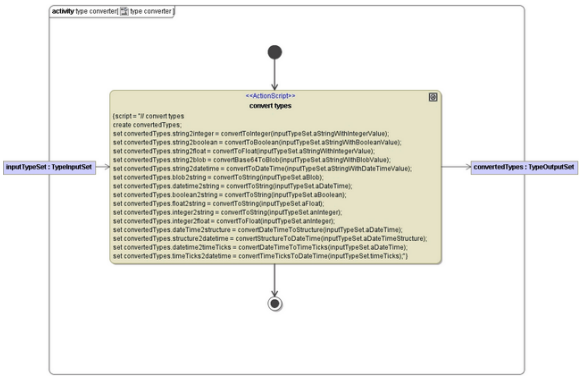
Type Converter Operations

Example File (Builder project E2E Action Language/BaseTypes):

<your example path>\E2E Action Language\BaseTypes\uml\converter.xml

The example in figure Type Converter Operations shows the use of converter operations.

Figure:Type Converter Operations



A blob as used by the Bridge is internally encoded as **base64Binary**. This means that in order to read anything out of a blob from a frontend (client) or backend, you first need to decode it (see Blob Operations). To use the information in the blob, you need to know what format it is (e.g. html, xml, gif, string, etc.). There is only one exception: when the backend understands **base64Binary** (for instance SOAP services), you do not have to decode it.