**Integration Framework**

**Mapping Tool**

**Last Modified: Januar 4, 2020**

Table of Contents

[1 Getting Started with the Mapping Tool 2](#_Toc29038179)

[1.1 Mapping Tool Introduction 2](#_Toc29038180)

[1.2 Mapping Tool Overview 3](#_Toc29038181)

[1.3 XSL Generation 4](#_Toc29038182)

[1.4 Generating XML Documents Based on XSDs 5](#_Toc29038183)

[1.4.1 Creating XSD Definition Documents Based on XSDs 5](#_Toc29038184)

[1.4.2 Generating Full XML Documents 5](#_Toc29038185)

[1.4.3 Generating a View for XML Documents 6](#_Toc29038186)

[1.4.4 Displaying Generation Protocols 6](#_Toc29038187)

[1.5 Creating Mapping Documentation 7](#_Toc29038188)

[2 Using the Mapping Tool 7](#_Toc29038189)

[2.1 Changing Mapping Tool Settings 7](#_Toc29038190)

[2.2 Selecting Groups and Source and Target XML Documents 8](#_Toc29038191)

[2.3 Adding Groups to the Source and Target XML Selection 8](#_Toc29038192)

[2.4 Context Menu Functions of the Source XML Structure 9](#_Toc29038193)

[3 Mapping Definitions 9](#_Toc29038194)

[3.1 Getting Started with Mapping Definitions 9](#_Toc29038195)

[3.2 Using XPath Functions 10](#_Toc29038196)

[3.3 Using Integration Framework Functions 10](#_Toc29038197)

[3.4 Defining and Using Variables 13](#_Toc29038198)

[3.5 Defining and Using JavaScript Functions 14](#_Toc29038199)

[3.6 Working with Conditions 14](#_Toc29038200)

[3.7 Working with Iterations 15](#_Toc29038201)

[3.8 Working with Internal Iterations 15](#_Toc29038202)

[3.9 Collaborating in Mapping Projects 15](#_Toc29038203)

[4 XSLT Document Generation and Testing 16](#_Toc29038204)

[4.1 Developing Mappings and Generating the XSLT 16](#_Toc29038205)

[4.2 Testing Mappings 16](#_Toc29038206)

[4.3 Namespace Handling 17](#_Toc29038207)

[4.4 Namespaces in the Created XML Outbound Document 17](#_Toc29038208)

[4.5 Expressions in Generated XSLT Coding and XPath Functions 18](#_Toc29038209)

[4.6 Mapping Tool Architecture Details 19](#_Toc29038210)

[Copyrights, Trademarks, and Disclaimers 20](#_Toc29038211)

# 1 Getting Started with the Mapping Tool

## 1.1 Mapping Tool Introduction

In scenario steps, you use the transformation atom of the processing phase to map document structures either by using XSLT, JavaScript or a combination of XSLT and JavaScript.

The mapping tool enables you to convert the structure of a source into a target XML document structure with the support of a user interface. To develop a mapping, do the following:

1. Create a mapping project.
2. Load the source and the target XML documents into the tool.
3. For each target element, define the mapped element or elements of the source structure and use functions that transform the source element or elements in the required way.

You can use XPath and SAP functions for mapping definitions.

The tool supports defining conditions and iterations.

1. To define mappings, use predefined variables or define and use individual variables.
2. After saving your mapping definitions, generate and test the XSL document.
3. Assign the XSL document to the transformation atom in your scenario step.

**Access to Mapping Tool**

The integration framework offers access to the mapping tool in the following places:

* In the Scenarios menu of integration framework version 2 as part of package design in the navigation tree

The generated XSL mapping document is available in the sources section of the scenario package. You can directly reference the document in an XSL transformation atom.

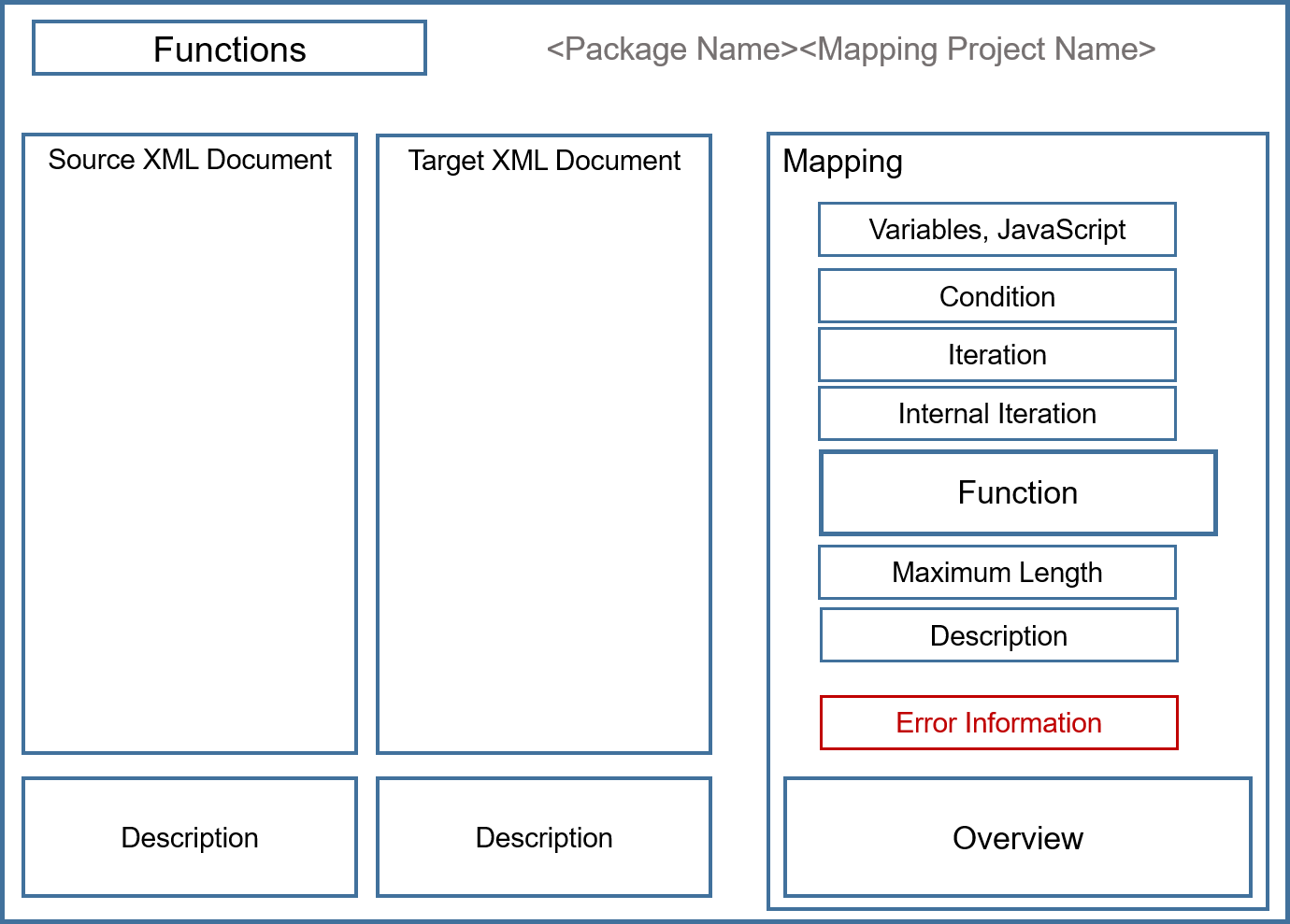
* In the Tools menu of integration framework version 1 and in the Tools menu of integration framework version 2

The mapping tool saves mapping XSL documents to the following BizStore location:

com.sap.b1i.dev.directory.mappings

To link an XSL document in a transformation atom, enter the XPath to the document. For framework version 1 and when running scenarios in version 2 in the B1if compatibility mode, define the mapping root. For more information, see section 2.1 Changing Mapping Tool Settings

## 1.2 Mapping Tool Overview





**Functions**

| **Function** | **Description** |
| --- | --- |
| Refresh | Refresh the user interface |
| New | Create a mapping project |
| Open | Open a mapping project |
| Settings | Open and change mapping tool settings |
| Save | Save the mapping project |
| XSD to XML Generator | Generates XML document based on XSDs including the option to define a view on the XML document |
| Generation of Documentation | Generates mapping documentation in CSV or PDF format |
| Generate XSLT | Generates the XSL document based on the defined mapping functions |
| Test | Test the mapping definitions using an inbound test message and validate against XSD |
| Generated XSLT | Displays the generated XSL document |
| Display Documentation | Display mapping tool documentation |

**Package Name and Mapping Project Name**

The tool displays the package name and the mapping project name. The package name is added, if you use the mapping tool in the package explorer in scenario design of framework version 2. The project name is followed by an asterisk, if there are unsaved definitions.

**Source XML**

Load the source XML document. The tool displays the structure with all nodes and elements. If the structure contains namespace definitions, the tool displays namespace prefixes. To display element attributes, expand the element. The tool displays attributes in blue font with @ in front. Expand or collapse nodes. To reduce the display of nodes and elements, define a view on the structure. For an element, you can copy the XPath to an element. In the structure, you can search for elements and attributes.

**Target XML**

To define mappings, work with the target XML structure. Open the context menu for the root element and you have the following functions available:

* Limit the view of the structure to elements with open mapping definitions (open fields), with mapping definitions (mapped fields) or elements without definitions (unmapped fields).
* You also have the option to limit the view of displayed elements (view definition), work on them and then add more nodes and elements to the view.
* For generated functions, you can display the XSL fragment for each element.

Attributes are displayed in the Mapping section below the description field together with input fields for mapping definitions. In the structure, you can search for elements and attributes.

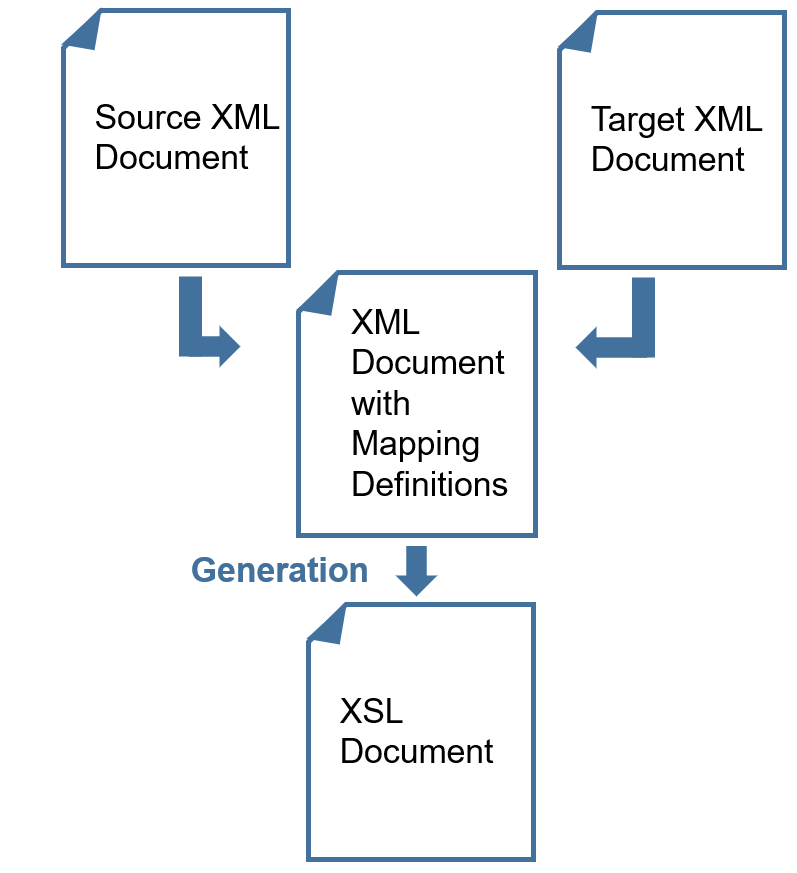
**Description**

If enabled in the mapping tool settings, the tool displays descriptions that are part of the XML structures in the description fields, if you select an element of the structure.

**Mapping**

In the Function field of the Mapping section, enter the definition to map the source element or elements to the target element. To define the mapping, you can use variables, JavaScript functions, conditions, iterations, and you can define internal iterations. Define the maximum length of the target element and add a description of the mapping.

## 1.3 XSL Generation



In the mapping tool, you map a source XML document to a target XML document. The tool writes the mapping definitions to an XML document with the same name as the mapping project. The XSL generation is based on the XML definition document.

During XSL generation, the tool reports errors of the XML processor.

## 1.4 Generating XML Documents Based on XSDs

The function supplements the mapping tool by allowing the generation of an XML document based on XML schema definition (XSD) documents. You can choose to either generate a full XML document or to define a view for the XML document. After generation, you can use the XML document as a source or target in the mapping tool.

### 1.4.1 Creating XSD Definition Documents Based on XSDs

Based on a selected XSD document, the function generates an XSD definition document. The XSD definition document has the following structure:

* The <namespaces> tag defines all namespaces. Default namespaces get namespace prefixes. Relative namespace identifiers are converted to absolute namespace identifiers.
* In the <xsd.documents> tag, you find the XSD and additional imported or included XSDs, if required.
* Each element has a b1i.id identifier.

The XSD definition document is the basis for the generation of XML outbound documents.

**Prerequisites**

* You loaded the XSD document or documents to the BizStore, using the ZIP Import function of the Maintenance menu or by dragging and dropping folders and documents to the BizStore in the BizStore menu.
* If an XSD includes or imports other XSDs, you also loaded the documents to the BizStore.
* The BizStore has a two-level hierarchy. You can currently only load XSDs that do not require more levels of hierarchy.

**Procedure**

1. To generate an XML document that is based on an XSD, choose Scenarios, open a scenario package, click Mapping and click the  (XSD to XML Generator) button.

The tool displays the initialization status (Idle) and the scenario package identifier.

1. To load the XSD document from the BizStore, click Select in the XSD Document field, and select the BizStore dataset, group and the document.
2. In the XML Outbound Document field, select or enter the BizStore location for the generated XML document. The default location is the xsdxml group in the …dev.directory dataset.
3. Click Initialization and check the process completion by clicking Refresh.

**Results**

The process loads the XSD document and creates the XSD definition document.

To display the document, click  (Open XSD Definition Document)

### 1.4.2 Generating Full XML Documents

To generate a complete XML document based on an XSD, click Generate Full.

**Procedure**

1. For the XML generation, select the following options:

| **Name** | **Select to** |
| --- | --- |
| XML Element Handling | Select to either generate all XML or only mandatory XML elements |
| XML Element Occurrence | Select to either generate the element only at the first occurrence or for all occurrences |
| XML Attribute Handling | Select to either generate mandatory or all attributes |
| Choice Handling | Select to either generate the first choice definition or all choice definitions. |
| XML Element Documentation | Select to generate or leave out documentation for XML elements |

1. To generate the XML outbound document, click Generate XML.

The generation status changes from Idle to Generation in progress.

1. To check the completion of the generation, click Refresh.

The generation status changes back to Idle.

1. To display the document, click  (Open XML Document).

### 1.4.3 Generating a View for XML Documents

The generated XML document can be large with many nodes and elements that you probably do not need to consider in a mapping project. Therefore, you can define a view for the document and generate a reduced XML document for mapping.

**Procedure**

1. To generate a view on the XML document, click Generate View.

The function displays the collapsed structure of the XML document.

1. Expand the structure and select the elements and attributes that you require for mapping.

Selected elements and attributes have a blue background.

1. Click  (Generate XML).
2. To display the generation result, click  (Display Generated XML Document).

### 1.4.4 Displaying Generation Protocols

The generation function protocols the generation history.

**Procedure**

1. To display generation protocols, click Protocol.

The function displays the generation history of the document with the generation result, start timestamp, the generation type and the document name.

1. To display all generation information, click Details.
2. Display the extended XML document based on the XSD, the view definition document and the generated XML document.

## 1.5 Creating Mapping Documentation

You can generate documentation for a mapping project that contains all definitions. You can either generate a delimiter-separated file for upload, for example, to Microsoft Excel or a PDF document.

**Procedure**

1. To generate documentation for a mapping project, click  (Generation of Mapping Documentation).
2. In the Document Format field, select either Excel CSV or PDF.
3. In the Content field, select to either generate documentation for all fields or only for mapped fields.
4. For the CSV document format, select the separator.
5. Click Generate.

**Results**

* Click the print icon to print or download the documentation in PDF format.
* For Excel CSV, the function creates a <mapping project name>.csv document for download.

# 2 Using the Mapping Tool

## 2.1 Changing Mapping Tool Settings

The function allows you to change settings of the mapping tool, such as the display. Define the mapping root and update your SAP Business One DI API XML documents based on a reference system.

**Procedure**

1. In the mapping tool, click the Settings button.
2. In the Mapping Root field, enter an XPath that defines the place where mapping definitions start in a larger document.

**Example**

In scenario definition configuration, you selected the Smart Message Model checkbox. The integration framework message is available like in version 1.

The source XML document starts with:

<?bpc.pltype.out bpm.pltype=xml?>

<vpf:Msg xmlns:vpf="urn:com.sap.b1i.vplatform:entity">

<vpf:Header/>

<vpf:Body>

<vpf:Payload id="atom1">

<TestMessage type="Invoice">

<Invoice ident="123">

You want to start the mapping definition with the <TestMessage> tag.

In the Mapping Root field, enter: /vpf:Msg/vpf:Body/vpf:Payload[./@id='atom1']

1. In the View Settings field, you can choose to display the mapping area either on the right side (default) or between the source and target XML documents.
2. In the *Node Space* field, adjust the space between nodes of the source and target XML documents.
3. To let the mapping tool only generate tags and transformations for fields with mapping definitions, select Generate Mapped Fields Only.
4. Select *Display Description* to display field descriptions in source and target XML documents in the Description field for the selected element.
5. The mapping tool offers the SAP Business One DI API documents for mappings. To update the structures with fields from an SAP Business One reference system, click Update, select a reference system from SLD and click Generate Now.

## 2.2 Selecting Groups and Source and Target XML Documents

The mapping tool requires source and target messages in XML format. The integration framework delivers documents in the following groups:

* SAP Business One DI API XML documents

To update the documents with fields from an SAP Business One reference system, choose Settings and for the Business One DI-API XML Documents field, click update.

* SAP Business One Service Layer XML documents
* Test Messages

In this section, find documents that were saved in debugging when clicking the context menu of the red arrow.

* XSD to XML Generator

In this section, find documents generated by the XSD to XML generator.

**Procedure**

1. To select available source or target XML documents, click the  (Select Source or Target Document) button in the Source XML or Target XML section, select a group and then a document of the group.

The tool displays the document structure.

1. To edit an XML document of the Test Messages group, select the message and click Edit.

The XML editor opens, and you can reduce, for example, the test message that is becomes suitable for mapping definitions.

1. Apply changes, save and close the editor.

## 2.3 Adding Groups to the Source and Target XML Selection

Add groups and documents that you want to use for mappings.

**Procedure**

1. In the file system, create a directory with the BizStore dataset name. In the BizStore dataset directory, create a directory with the BizStore group name. Choose appropriate names for the dataset and group.
2. Add the XML documents to the BizStore group directory in the file system.
3. Archive the dataset directory and use the ZIP import function of the Maintenance menu to load the documents to the BizStore.

The ZIP import adds the dataset, group and documents to the BizStore.

1. To add a group to the source or target XML section of the mapping tool, click Select Document and then click Add.
2. Select the BizStore dataset and group, enter a name for the group and click Add.

**Results**

The group and documents are available for selection in the source and target XML selections.

## 2.4 Context Menu Functions of the Source XML Structure

The context menu of the source XML document offers the following functions:

* At the root node, select View Definition.

To reduce the number of nodes and elements that you display, deselect the nodes and elements that you do not want display and click Apply.

* Expand or collapse the structure
* Select or deselect an element. Selected elements have a blue background.
* Copy the XPath to the element

# 3 Mapping Definitions

## 3.1 Getting Started with Mapping Definitions

For simple mapping definitions, assign a fixed value, a source or several source elements and define the field length.

**Procedure**

1. If the target XML structure is large, reduce the displayed elements.
2. On the top node, open the context menu and click View Definition.

The tool displays all nodes and elements of the structure.

1. Deselect the nodes and elements that you want to work on later and click Apply.

The tool only displays the selected nodes.

1. To assign a mapping definition to a target element, select the element in the *Target XML* section.

A blue background indicates the selection.

1. To define a mapping definition, do the following:

* Assign a fixed value to the target element. Enter the value in single quotes, for example, ’2’ in the Function field.
* To map a source element to the target element, click the element in the source XML.

The tool adds the XPath to the source element to the Function field. The tool displays the mapping in the overview section and indicates the selection with a blue background in the source XML.

* To map several source elements to the target element, press Ctrl and click the source elements. The tool adds the concat function with the XPaths to the source elements to the Function field.

1. To define the maximum length for a target element, enter the number in the Maximum Field Length field. The definition uses the XPath substring function in the background.
2. Save your settings.

**Results**

Mapped elements have a green background. In the target XML section, you can define a view to only display mapped elements.

## 3.2 Using XPath Functions

The mapping tool does not use a proprietary language but supports using XPath functions.

XPath includes over 200 functions for string and numeric values, Booleans, date and time comparison, node and sequence manipulation, and much more.

To display available functions, start typing in the Function field and the tool provides information about available functions. You can enhance predefined functions by valid XPath statements, addressing, for example, attributes.

## 3.3 Using Integration Framework Functions

The integration framework provides additional functions for element mapping. In the tables below, the Generated Function Call line displays the JavaScript function that the framework processes for the function. In the mapping tool, you can display the function after generation when using the Show Fragment XSL context menu function:

|  |  |
| --- | --- |
| **align(string1,string2,alignleft)** | |
| Description | Returns a string that aligns string1 within string2 |
| Example | align('abc','dddd','true') |
| Result | abcd |
| Generated Function Call | js:invoke($sg,'algn',string(string1),string(string2), string(alignleft)) |

| **delLeadingBlanks(string1)** | |
| --- | --- |
| Description | Returns string1 without leading blanks |
| Example | delLeadingBlanks(' 'abc') |
| Result | abc |
| Generated Function Call | js:invoke($sg,'delLeadingB',string(string1)) |

|  |  |
| --- | --- |
| **delLeadingZeros(string1)** | |
| Description | Returns string1 without leading zeros |
| Example | delLeadingZeros ('000abc') |
| Result | abc |
| Generated Function Call | js:invoke($sg,'delLeading0',string(string1)) |

| **delTrailingBlanks(string1)** | |
| --- | --- |
| Description | Returns string1 without trailing blanks |
| Example | delTrailingBlanks('abc ') |
| Result | abc |
| Generated Function Call | js:invoke($sg,'delTrailingB',string(string1)) |

|  |  |
| --- | --- |
| **delTrailingZeros(string1)** | |
| Description | Returns string1 without trailing zeros |
| Example | delTrailingZeros('abc000') |
| Result | abc |
| Generated Function Call | js:invoke($sg,'delTrailing0',string(string1)) |

| **ends-with(string1,string2)** | |
| --- | --- |
| Description | Returns true, if string1 ends with string2, otherwise function returns false |
| Example | ends-with('XML','X') |
| Result | false |
| Generated Function Call | js:invoke($sg,'endswith',string(string1),string(string2)) |

|  |  |
| --- | --- |
| **format-date(date,sfrm,tfrm)** | |
| Description | Converts a date string in sfrm format to a string in tfrm target |
| Example | format-date('2019-01-23', 'yyyy-MM-dd', 'MM/dd/yyyy') |
| Result | 01/23/2019 |
| Generated Function Call | fljs:invoke($sg,'frmtDate',string(date),string(sfrm),string(tfrm)) |

| **format-timestamp(ts,sfrm,tfrm)** | |
| --- | --- |
| Description | Converts a timestamp string in sfrm format to a string in tfrm target format |
| Example | format-timestamp('2019-01-23 234512','yyyy-MM-dd HHmmss','MM/dd/yyyyTHH:mm:ss') |
| Result | 01/23/2019T23:45:12 |
| Generated Function Call | fljs:invoke($sg,'frmtTS',string(date),string(sfrm),string(tfrm)) |

| **left(string1,number1)** | |
| --- | --- |
| Description | Returns first number1 characters of string1 |
| Example | left('abcdefg',5) |
| Result | abcde |
| Generated Function Call | substring(string1,1,number1) |

| **lower(string1)** | |
| --- | --- |
| Description | Converts string1 to lower case |
| Example | lower('Abc5DEfG') |
| Result | abc5defg |
| Generated Function Call | js:invoke($sg,'toLC',string(string1)) |

|  |  |
| --- | --- |
| **now()** | |
| Description | Returns current timestamp in yyyy-mm-ddThh:mm:ss format |
| Example | now() |
| Result | 2019-01-31T16:05:15, for example |
| Generated Function Call | js:invoke($sg,'getTS) |

|  |  |
| --- | --- |
| **padding(len,string1)** | |
| Description | Returns a string filled with string1 with len length |
| Example | padding(10,'ABC') |
| Result | ABCABCABCA |
| Generated Function Call | js:invoke($sg,'pstr',number(len),string(string1)) |

| **padding-align(target,len,padchr,alignleft,truncate)** | |
| --- | --- |
| Description | Padding the target string at the defined len length with the defined padchar padding character, including left alignment (alignleft) and truncating (truncate). The default value of truncate is true. |
| Example | padding-align('567',6,'0','false','true') |
| Result | 000567 |
| Generated Function Call | js:invoke($sg,'padAlg',string(target),number(len),string(padchr), string(alignleft),string(truncate)) |

|  |  |
| --- | --- |
| **replace(string1,string2,string3)** | |
| Description | Replaces in string1 up to 5 occurrences of string2 by string3 |
| Example | replace('AB\*#DE\*#\*#FF','\*#','+++') |
| Result | AB+++DE++++++FF |
| Generated Function Call | js:invoke($sg,'strgrepl',string(string1),string(string2), string(string3)) |

| **right(string1,number1)** | |
| --- | --- |
| Description | Returns last number1 characters of string1 |
| Example | right('abcdefg',5) |
| Result | cdefg |
| Generated Function Call | js:invoke($sg,'substrgR',string(string1),number(number1)) |

|  |  |
| --- | --- |
| **roundup(number1,number2)** | |
| Description | Rounds up number1 to number2 decimal places |
| Example | roundup('1054.32179', '2') |
| Result | 1054.32 |
| Generated Function Call | js:invoke($sg,'roundUp',number(number1),number(number2)) |

| **today()** | |
| --- | --- |
| Description | Returns current date in yyyy-mm-dd format |
| Example | today() |
| Result | 2019-01-23, for example |
| Generated Function Call | js:invoke($sg,'getDate) |

|  |  |
| --- | --- |
| **trim(string1)** | |
| Description | Returns string1 without leading and trailing blanks |
| Example | trim(' abc ') |
| Result | abc |
| Generated Function Call | js:invoke($sg,'delLeadingTrailingB',string(string1)) |

| **upper(string1)** | |
| --- | --- |
| Description | Returns string1 in uppercase |
| Example | upper('Abc5DEfG') |
| Result | ABC5DEFG |
| Generated Function Call | js:invoke($sg,'toUC',string(string1)) |

## 3.4 Defining and Using Variables

The tool supports using the following global variables:

* SysType properties are properties delivered with SAP Business One system types. As soon as you select a source or target XML document with the <BOM> <BO> structure, the Systype properties are available for mapping definitions.
* Deployment properties of the scenario package that you defined selecting the package and choosing Definitions → Deployment Properties from the context menu
* Individual variables defined in the mapping project

**Procedure**

1. To define an individual variable, click Variables in the Global field and click Add.
2. In the red input field on the right side, enter the variable name and in the text field below, enter the variable definition. Enter a fixed value or add XSLT coding and click Save.
3. Use variables in all definitions for the mapping, in functions, conditions, iterations, internal iterations.
4. Start typing $ and the help displays available variables. Use the Arrow Down key to browse the variables.

**Results**

During generation, all variables that you use in mapping definitions are created at the beginning of the XSL file as public variables. If you create variables but you do not them, they are not created in the XSL document.

## 3.5 Defining and Using JavaScript Functions

**Procedure**

1. To define a JavaScript function, click Functions in the Global field and click Add.
2. In the red input field on the right side, enter the function name and in the text field below, enter the function definition and click Save.
3. Use JavaScript functions in all definitions for the mapping, in functions, conditions, iterations, internal iterations.
4. Start typing the function name and the help displays available functions. Use the Arrow Down key to browse the functions.

## 3.6 Working with Conditions

Use the condition field, if you want to map an element under certain conditions.

**Procedure**

1. To perform a mapping definition under certain conditions, enter a function in the Condition field.

You can use, for example, the following functions:

* empty
* notempty
* choose(cond1=’1’;cond2=’2’;cond1=’3’)

Separate conditions by semicolon.

1. In the Function field, enter the mapping definition. When using the choose() function, add the same number of options as you defined in the Condition field separated by semicolon.

**Examples**

Only, if the field is not empty, perform the mapping definition.

* Condition: notempty
* Function: /X/Y/Partners[./@a='b']/row/Phone1

**Condition**

choose($SalesOrganization='123';$SalesOrganization='456';A=B)

**Function**

choose(/X/Y/Addresses/row/Code;'456';'123')

**XSL**

<TargetField>

<xsl:choose>

<xsl:when test="$SalesOrganization='123'">

<xsl:value-of select="/X/Y/Addresses/row/Code"/>

</xsl:when>

<xsl:when test="$SalesOrganization='456'">

<xsl:value-of select="'456'" />

</xsl:when>

<xsl:when test="A=B">

<xsl:value-of select="'123'"/>

</xsl:when>

</xsl:choose>

</TargetField>

## 3.7 Working with Iterations

When selecting a node in the target XML and assigning a node of the source XML, the XPath is not added to the Function, but to the Iteration section.

In the elements underneath the node, you do not need the complete XPath to the element but address the element name directly in the for-each loop.

## 3.8 Working with Internal Iterations

Internal iterations are predefined values for an element, for example, the country element supporting all country codes. The language element supports all language codes.

**Procedure**

1. In the Internal Iteration field, add the XPath to the element in the source document.
2. In the Function field, use the concat() function to add all valid codes.

**Example**

**Internal Iteration**

/X/Y/Addresses/row/County

**Function**

concat(County,' - ')

## 3.9 Collaborating in Mapping Projects

Some element mappings are easy. Assign a source to a target element or enter a fixed value. The concatenation of source elements to a target element is supported by the mapping tool. For some mappings, however, the definition can become complex involving the usage of variables, format conversions, filling values based on certain conditions, and so on. The tool provides some functions to enable the collaboration in mapping projects.

**Procedure**

The mapping tool supports the collaboration in mapping projects in the following way:

1. Provide all simple mapping definitions.
2. For mappings that require further definitions by someone else, enter a description of the mapping task in the Description field. In the target structure, open the context menu for the element and select Mapping Open.

The tool underlines the element in the structure.

1. Another user who continues working on the mapping project opens the context menu at the root element and select the Open Fields option. The option only displays elements that are marked as open.
2. The other user has the description for the mapping task and can continue working on the project.

# 4 XSLT Document Generation and Testing

## 4.1 Developing Mappings and Generating the XSLT

After developing a mapping definition, run the XSLT generation.

**Procedure**

1. For a target element, enter the mapping definition in the Function field.
2. To check whether the definition is correct, click the Generate button.

The tool saves the definition, generates the XSL document and returns the result of the generation.

1. If there is an error in the definition, the tool displays the element where the error occurs with a red background and returns the error information of the XML processor in the Error Information field.
2. To display the generated document, refer to the Sources -> xsl section of your scenario package and open the document.
3. Alternatively, open the context menu for the target element and choose Show Fragment XSL.

## 4.2 Testing Mappings

You can test your mapping definitions. For the test, define a test inbound message. If you want to run a schema validation, select the XSD document.

**Procedure**

1. To test mapping definitions, click the  (Test) button.

The tool displays the status of the test run.

1. To run a test, select a test inbound message from the BizStore, choosing a dataset, group and document in the Inbound Test Message field. If required, click the Edit Inbound Test Message button to adjust the test message.
2. To check the message against the XML schema definition, select the XSD from the BizStore choosing the dataset, group and document in the Schema Validation Document field.
3. The tool displays the XSD definition document and the test result outbound document.
4. Rerun the test and check the status. If the test run is successful, the tool immediately displays the result outbound document.
5. If there is a validation error, the tool displays the affected path and the error text.

## 4.3 Namespace Handling

The mapping tool considers the following namespaces:

* Standard and individual namespaces defined in Definitions → Namespaces for the package
* Namespace declarations in the source and target XML documents
* The generation creates a b1i-fa? prefix for all default namespace definitions in the source and target document

? is the placeholder for a sequential number

The generation sorts and merges all namespace declarations and removes duplicates. Duplicates are namespace declarations with the same prefix and the same namespace definition. Different prefixes linked to the same namespace definition are valid. The generation does not remove them.

**Example**

xmlns:tst="com.sap.b1i.test" xmlns:tst2="com.sap.b1i.test"

Prefix definitions linked to different namespace definitions cause a namespace conflict. The generation detects namespace conflicts and the user gets a notification. The XSLT generator does not generate the XSLT document.

**Example**

xmlns:tst="com.sap.b1i.test" xmlns:tst="com.sap.b1i.test2"

If the source or target XML documents are not valid regarding namespaces, the user gets a notification and the XSLT generator does not generate the XSLT document.

**Example**

A namespace prefix is not mapped to a namespace.

## 4.4 Namespaces in the Created XML Outbound Document

At runtime, when using the XSLT document in a scenario step, the creation of the XML outbound document depends on the definition document for the target XML schema.

Namespace declarations of the target XML document that the generated XSTL uses are declared inside the root element of the XML outbound document at runtime.

<ROOT xmlns:tst="com.sap.b1i.test" …

The XSLT generates the outbound XML document without using default namespaces. Instead, default namespaces are declared when being used. They are addressed with an artificial namespace prefix. Namespaces are generated for multiple different elements with different namespace prefix.

Each element of the outbound XML document follows the definition in the target XML document.

* Elements without prefix, or no default namespace and no parent default namespace are created only with the name.

<Element>

* Elements with a leading prefix are created with a leading prefix. The prefix is defined in the root element.

<pre:Element>

* Elements without a leading prefix and an own empty default namespace are only created with the name. it is possible, because there are no default namespaces in the outbound XML document.

<Element>

* Elements without a leading prefix, an own default namespace or a parent default namespace are created with an explicit namespace prefix. The prefix is generated and declared in the same element.

<nsxxxx:Element xmlns:nsxxxx="namespace definition">

## 4.5 Expressions in Generated XSLT Coding and XPath Functions

All element names in defined mapping expressions are changed to consider namespaces. The mapping tool uses the name() function to display elements. This way, the tool copies the source elements to the expressions that define the mapping. The following options are covered:

* Element with namespace prefix
* Element without namespace prefix

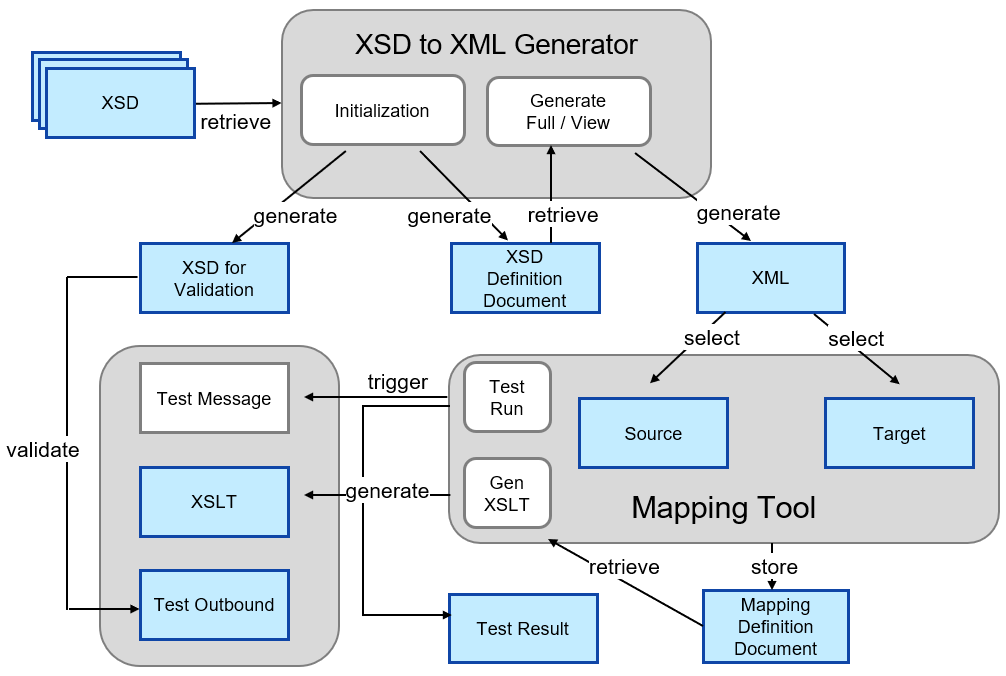
The XSLT generator uses elements with namespace prefix without changes.

Elements without namespaces can be:

* Elements without namespace prefix, but with an individual default namespace or a default namespace of the parent element.
* Elements that are in no namespace

The XSLT generator detects, if the element that is used in an expression has a default namespace. The generator creates an artificial namespace prefix and adds the prefix in front of the element name. The namespace declaration is part of the root tag.

## 4.6 Mapping Tool Architecture Details



If your mapping project starts with one or several related XSDs, work with the **XSD to XML generator**. The **initialization** function of the generator retrieves the XSDs and generates the following documents:

* An **XSD for** an optional later **validation**
* **XSD definition document** that contains all information to generate an XML document that is used for mapping.

In the XSD to XML generator, you can either **generate** the **full** or a reduced XML document. For the reduced XML document, you use the view definition to specify the **view** by selecting the elements that are part of the source or target XML document.

In the **mapping tool**, select a **source** and **target XML document** and define the mappings. All definitions are stored in the **mapping definition document**. Based on the definition document, the tool **generates** the **XSL document**. The test run function returns the **test result**. Additionally, you can add a **test message**, run the XSLT and add the original XSD for validation. The outcome is the **test outbound** document.

# Copyrights, Trademarks, and Disclaimers

© Copyright 2020 SAP SE. All rights reserved.

The current version of the copyrights, trademarks, and disclaimers at <https://www.sap.com/about/legal/copyright.html> is valid for this document.