## Elements

| Element | Description | Closing Element |
|---|---|---|
| <<name>><br><<first-name>><br><<{[first-name]}>> | Replace this field by the data referenced by "name" or "first-name".<br>Hyphenated names are supported and need to be enclosed in [ and ] when used in expressions. | |
| <<op:name>> | Replace this field by the data referenced by "name". If name is blank, the entire paragraph is stripped (including any other content). This makes the entire paragraph optional. | |
| <<{expr}>> | Replace this field with the results of the given expression. | |
| <<link:name>><br><<link_name>> | Insert a hyperlink at this location, using the URL from the data referenced by "name". The data can optionally specify display text by using the form: <text>\|<url> eg: "docmosis\|https://www.docmosis.com" | |
| <<$abc=name>><br><br><<$abc=10.2>><br><<$abc='Fred'>><br><<$abc=true>><br><<$abc=null>> | Lookup the data associated with "name" and assign it to the variable "abc".<br>Assign the number 10.2 to variable $abc<br>Assign the string "Fred" to variable $abc<br>Assign the boolean true to variable $abc<br>Assign the value null to variable $abc | |
| <<$abc>> | Lookup the variable "abc" and render its value | |
| <<cs_name>><br><<cs_{expr}>><br><<cs_$abc>> | Content between the opening element and the closing element is included or excluded depending on the value associated with "name" or the expression "expr" or the variable "abc". The end tag must match exactly, or may be anonymous: <<es_>>. | <<es_name>><br><<es_{expr}>><br><<es_$abc>><br><<es_>> |
| <<else_name>><br><<else_{expr}>><br><<else>> | This is the "else" tag related to a <<cs_>> tag to provide the "else" and "else if" options to a condition. | |
| <<rs_name>><br><<rs_$abc>><br><br><<rs_name:step2>><br><br><<rs_name:step2down>> | Content between the opening element and closing element is repeated whilst there is data associated with "name" or the variable "abc".<br>"stepN" indicates that the data ("name") should be iterated in steps of N size. When stepping is used, the variables $i1, $i2,...$iN are created automatically so you can reference the items available in each step.<br>"stepNdown" indicates that the data ("name") should be iterated in steps of N size and data should be presented in a "down"-ward manner. Variables $i1, $i2,... $iN are created automatically. | <<es_name>><br><<es_$abc>><br><<es_name:step2>><br><<es_>> |
| <<cr_name>><br><<cr_{expr}>><br><<cr_$abc>> | Include the following table rows depending on the value associated with "name" or expression "expr" or the variable "abc". | <<er_name>><br><<er_{expr}>><br><<er_$abc>><br><<er_>> |
| <<rr_name>><br><<rr_$abc>><br><br><<rr_name:step2>><br><br><<rr_name:step2down>> | The rows between the opening element row and the closing element row are repeated whilst there is data associated with "name" or the variable "abc".<br>"stepN" indicates that the data ("name") should be iterated in steps of N size. When stepping is used, the variables $i1, $i2,...$iN are created automatically so you can reference the items available in each step.<br>"stepNdown" indicates that the data ("name") should be iterated in steps of N size and data should be presented in a "down"-ward manner. Variables $i1, $i2,... $iN are created automatically. | <<er_name>><br><<er_$abc>><br><<er_>> |
| <<noTableRowAlternate>> | Disable automatic alternate-colouring of table rows. This can appear in a table to disable for the table or appear in the document body to disable for all following tables. | |

| Element | Description | Closing Element |
|---|---|---|
| `<<cc_name>>`<br>`<<cc_{expr}>>`<br>`<<cc_$abc>>` | Include or exclude the table column containing this field depending on the value associated with "name" or the expression "expr" or the variable "abc". | |
| `Image`<br>*MS Word:* bookmarked with label "img_name"<br>*OpenOffice or LibreOffice Writer:* image named "img_name"<br><br>(deprecated "bm_name") | Replace an image in the template with the image data associated with "name" using the default scaling settings (which is stretch).<br>The default setting can be changed by setting the docmosis property: `docmosis.analyzer.image.scaling.default` to `fit` or `stretch`.<br>See the Docmosis Developer's Reference for information about setting properties. | |
| `Image stretched`<br>bookmarked with label or named "imgstretch_name" | Replace an image in the template with the image data associated with "name" and stretch the new image to match the template image placeholder. | |
| `Image scaled to fit`<br>bookmarked with label or named "imgfit_name" | Replace an image in the template with the image data associated with "name" and fit the new image into the template image placeholder preserving the new image aspect ratio. | |
| `<<ref:sub1.doc>>` | Insert the template named "sub1.doc" at this location. | |
| `<<refLookup:name>>` | Lookup "name" in the data to get the name of the template to insert at this location. | |
| `<<list:continue>>` | To be used inside a sub-template numbered list. Specifies that numbering should be continued on from an existing numbered list when inserted. | |
| `<<list:reset>>` | To be used in a repeating section to force a numbered list to restart numbering (rather than continue numbering from previous repeat). | |
| `<<html:name>>` | Lookup "name" in the data and inject the data as HTML content into the document at this location. | |
| `<<barcode:name:…>>` | Provide information for a barcode image in the template.<br>eg. <<barcode:barcode1:code128>> defines image "barcode1" as a code 128 barcode. | |
| `<<## and ##>>`<br>`<</* and */>>` | Template-comments are delimited by the matching open and closing sequences. Content inside comments is not processed and is removed when creating documents. | |
| `<<coordinator:>>` | Mark this template as a coordinator of other templates. This template will not be part of the output, but instead specifies other templates to be rendered. | |
| `<<coordinator:padToEvenPage>>`<br>`<<coordinator:padToOddPage>>` | Before then next referenced template, if necessary, insert a blank page to make the output document an even or odd number of pages.<br>Currently this only applies to Docmosis Cloud and Tornado when it combines PDF output into a single PDF document. | |
| `<<coordinator:newFile>>` | At this point in processing, create a new file for the following content. This means that, when creating PDF output multiple templates may be combined into a collection of different PDF documents.<br>Currently this only applies to Docmosis Cloud and Tornado when it combines PDF output into a single PDF document. | |
| `<<pageBreak>>` | Insert a page break at this point. This is an alternative to inserting an actual page break in the template. | |
| `<<pageBreakNotLast>>` | Insert a page break at this point unless we are in a repeating section and at the last iteration of the repeat. | |

## Expression Operators

| Operator | Description |
|---|---|
| ( | open parentheses |
| ) | close parentheses |
| + | addition (for numbers and strings) |
| - | subtraction |
| * | multiplication |
| / | division |
| % | modulus |
| + | unary plus |
| - | unary minus |
| = | equal (for numbers and strings) |
| == | equal (for numbers and strings) |
| != | not equal (for numbers and strings) |
| < | less than (for numbers and strings) |
| <= | less than or equal (for numbers and strings) |
| > | greater than (for numbers and strings) |
| >= | greater than or equal (for numbers and strings) |
| && | boolean and |
| \|\| | boolean or |
| ! | boolean not |

## Example Expressions

| Element | Description |
|---|---|
| `<<{10 * 3.0}>>` | Calculate 10 multiplied by 3.0 |
| `<<{amount * qty}>>` | Lookup data elements "amount" and "qty" and multiply them together. |
| `<<{round(item/10)}>>` | Lookup data element "item", divide it by 10 then round the result. |
| `<<cs_{a<10}>>` | Lookup data element "a" and see if it is **less than** 10 numerically. If "a" is not numeric, a string comparison is performed automatically. |
| `<<cs_{a='fred'}>>` | Lookup data element "a" and see if it is **equal to** the String literal "fred". |
| `<<cs_{$a!=10}>>` | Lookup the variable "a" and see if it is **not equal to** the numeric value 10. If variable "a" does not resolve to a numeric value, a String comparison is performed. |
| `<<cs_{a=null}>>` | Lookup the data element "a" and determine if it's value is null |
| `<<cs_{$a}>>` | Determine if the value of the template variable $a is true |

## Logic and Transform Functions

| Function | Synopsis / Example |
|---|---|
| `ifBlank` | <<{ifBlank(name, 'none')}>> returns "none" if there is no data for name |
| `isBlank` | <<{isBlank(name)}>> returns true if the name is null or empty |
| `map` | <<map(data, in1, out1, in2, out2,….,default)}>><br>Eg. <<{map(gender, 'M', 'Male', 'F', 'Female', 'Other')}>> |
| `mapi` | <<mapi(data, in1, out1, in2, out2,….,default)}>> (case-insensitive)<br>Eg. <<{mapi(gender, 'M', 'Male', 'F', 'Female', 'Other')}>> |

## Numeric Functions

| Function | Synopsis / Example |
|---|---|
| abs | <<{abs(-153.57)}>> returns "153.57" |
| ceil | <<{ceil(153.57)}>> returns "154.0" |
| floor | <<{floor(153.57)}>> returns "153.0" |
| isNumber | <<{isNumber(123)}>> returns "true" |
| max | <<{max(53.5,23.1)}>> returns "53.5" |
| min | <<{min(53.5,23.1)}>> returns "23.1" |
| numFormat | <<{numFormat(data, format [, locale [, applyLocaleToInput [, formatIsLocalized]]])}>><br><<{numFormat('1457.1', '#,###.00' )}>> returns "1,457.10"<br><<{numFormat('0,0257', '#.##0,00', 'nl')}>> returns "0,03"<br><<{numFormat(0.0257, '#.##0,00', 'nl', false)}>> returns "0,03" (the false parameter indicates the locale 'nl' should not be used to interpret the input value) |
| numToDollars | <<{numToDollars(12.33)}>> returns "twelve dollars and thirty three cents" |
| numToText | <<{numToText(value [,'andLast'\|'andAlways'\|'andNone'])}>><br><<{numToText(123)}>> returns "one hundred and twenty three"<br><<{numToText(123, 'andNone')}>> returns "one hundred twenty three" |
| ordinal | <<{ordinal(value [,'short'\|'suffix'\|'long'\|'longNoAnds'\|'longAllAnds'])}>><br><<{ordinal(123)}>> returns "123rd"<br><<{ordinal(123, 'suffix')}>> returns "rd"<br><<{ordinal(123, 'long')}>> returns "one hundred and twenty third"<br><<{ordinal(123, 'longNoAnds')}>> returns "one hundred twenty third" |
| pow | <<{pow(7,2)}>> returns "49.0" |
| random | <<{round(random()*100)}>> returns a random number between 0 and 100. |
| round | <<{round(value [,decimal places])}>><br><<{round(153.73455)}>> returns "154"<br><<{round(153.73455,2)}>> returns "153.73" |
| sqrt | <<{sqrt(81.0)}>> returns "9.0" |

## Date Functions

| Function | Synopsis / Example |
|---|---|
| dateAdd | <<{dateAdd(value, n, units [,outputFormat [,inputFormat]])}>><br>where units = millis \| seconds \| minutes \| hours \| days \| weeks \| months \| years (plural optional)<br><<{dateAdd(date1, 1, 'day')}>> adds one day to date1<br><<{dateAdd(date1, -2, 'months')}>> subtracts two months from date1<br><<{dateAdd(date1, 1, 'year', 'yyyy')}>> add one year to the given date1 and output as 4 digit year<br><<{dateAdd(date1, 1, 'year', 'yyyy', 'dd-MM-yyyy')}>> add one year to the given date1 which is format dd-MM-yyyy and output as 4 digit year |

| Function | Synopsis / Example |
|---|---|
| dateDiff | <<{dateDiff(date1, date2, units [,inputFormat])}>> where units = millis \| seconds \| minutes \| hours \| days \| weeks \| months \| years (plural optional)<br><<{dateDiff(date1, date2, 'day')}>> calculate date2 minus date1 in days. |
| dateFormat | <<{dateFormat(value [,outputFormat [, inputFormat [, outputLocale [, inputLocale[, outputFormatLocalized [, inputFormatLocalized]]]]]]}>><br><<{dateFormat('2015-12-15' , 'EEEE, dd MMMM yyyy' , 'yyyy-MM-dd')}>><br>returns "Tuesday, 15 December 2015" |

## Text Functions

| Function | Synopsis / Example |
|---|---|
| char | <<{char(169}>> returns the copyright character ©<br>The value can be specified in decimal, hex, html-style etc. |
| charAt | <<{charAt('abcdefg',3)}>> returns the character "d" |
| endsWith | <<{endsWith('The first string', 'ing')}>> returns the value "true" |
| equalsIgnoreCase | <<{equalsIgnoreCase ('Bob', 'bob')}>> returns the value "true" |
| indexOf | <<{indexOf('abcde', 'bc')}>> returns "1.0" |
| length | <<{length('Bob')}>> returns the number "3.0" |
| replace | <<{replace('JHMAB52EC8oo65o','o','0')}>> returns "JHMAB52EC800650" |
| replaceStr | <<{replaceStr(data, find, replaceWith [,ignoreCase])}>><br><<{replaceStr('11 plus 11', '11', 'Eleven')}>> returns "Eleven plus Eleven" |
| replaceFirst | <<{replaceFirst(data, find, replaceWith [,ignoreCase])}>><br><<{replaceFirst('11 plus 11', '11', 'Eleven')}>> returns "Eleven plus 11" |
| split | <<{split('John\|Mathews\|47\|Approved' , '\|' , 1)}>> returns "Mathews" |
| squote | <<{squote('This is Amy"s')}>> returns "This is Amy's". |
| startsWith | <<{startsWith('The first string', 'The')}>> returns the value "true" |
| substring | <<{substring('0123456' , 2 , 5)}>> returns "234" |
| titleCase | <<{ titleCase ('bob mathews')}>> returns "Bob Mathews" |
| toLowerCase | <<{toLowerCase('Bob Mathews')}>> returns "bob mathews" |
| toUpperCase | <<{toUpperCase('Bob Mathews')}>> returns "BOB MATHEWS" |
| toAlpha | <<{toAlpha(1)}>> returns "a". <<{toAlpha(28)}>> returns "bb" |
| toAlpha2 | <<{toAlpha2(1)}>> returns "a". <<{toAlpha2(28)}>> returns "ab" |
| toRoman | <<{toRoman(1)}>> returns "i". <<{toRoman(28)}>> returns "xxviii" |
| toSentence | <<{toSentence('about 2. that is for Paul.')}>> returns "About 2. That is for Paul." |
| trim | <<{trim(' 12CVCV123-454 ')}>> returns "12CVCV123-454" |

## Locale Functions

| Function | Synopsis / Example |
|---|---|
| locale | <<{locale([spec])}>><br>where:<br>spec = the language or country name or code to lookup<br><<{locale('GERMANY')}>> returns "de_DE" |
| localeInfo | <<{localeInfo([spec])}>><br>where:<br>spec = the language or country name or code tolookup<br><<{locale('GERMANY')}>> returns<br>"Locale:[GERMANY] country=Germany, lang=German, variant=,id=de_DE" |
| localeDatePattern | <<{localeInfo(pattern [, locale])}>><br>where:<br>pattern = the non-localized pattern: eg dd-MMMM-yyyy<br>locale = the locale for which to display thelocalized pattern<br><<{localeDatePattern('dd-MMMM-yyyy', 'GERMANY')}>><br>returns tt-MMMM-uuuu |

## Ranges

| Element | Description |
|---|---|
| `<<hotel[0]>>` | The first hotel (indexing starts at zero) |
| `<<hotel[F]>>` | The first hotel (equivalent to index zero) |
| `<<hotel[L]>>` | The last hotel |
| `<<hotel[*]>>` | All hotels |
| `<<hotel[F3]>>` | The first 3 hotels |
| `<<hotel[L3]>>` | The Last 3 hotels |
| `<<hotel[1,2,4]>>` | The hotels at indexes 1,2 and 4 |
| `<<hotel[1-3,L2]>>` | The hotels at indexes 1 to 3 inclusive and the last 2 |
| `<<hotel[0-L2]>>` | All but the last 2 hotels |
| `<<hotel[3].floor[L].room[0].name>>` | The name of the first room of the last floor of the hotel at index 3 |

## Built-in Variables

| Variable | Description |
|---|---|
| `<<$top>>`<br>`or <<$root>>` | The root of the data regardless of the current position or context in the template |
| `<<$this>>`<br>`or <<$current>>` | The current source of data in the current position in the template. This allows for anonymous data lookups from arrays or collections such as <<$current[0]>>. |
| `<<$parent>>` | The parent or container of data in the current context of the template. Allows data lookup in the current "hotel" when the current context is a "floor" for example. |
| `<<$nl>>` | A simple newline character |
| `<<$nowMS>>` | Current UTC time in milliseconds since 1/1/1970 |

| Variable | Description |
|---|---|
| <<$nowUTC>> | Current UTC time as in ISO 8601 format |
| <<$nowUTCFormat>> | The ISO 8601 format used for $nowUTC |
| <<$quot>> | The single-quote character |
| <<$templateName>> | The name of the current template being rendered |
| <<$templateFolder>> | The name of the folder containing the template being rendered |
| <<$templatePath>> | The full path to the template being rendered (the combination of the name and folder) |

Variables available in repeating sections

| Variable | Description |
|---|---|
| <<$idx>><br>Index into data | The current index into the source data, starting from zero. |
| <<$itemidx>><br>Index in our iteration | The current index into an iteration, starting from zero |
| <<$num>> | Like $idx but starting from one. |
| <<$itemnum>> | Like $itemidx but starting from one. |
| <<$size>> | The size of the current repeating data set. |
| <<$rownum>> | The current row number (starting at 1) when repeating (either repeating rows or repeating sections). This is most useful when using the "stepping" directives and the $itemnum is not suitable. |
| <<$rowidx>> | The current row number (starting at 0) when repeating (either repeating rows or repeating sections). This is most useful when using the "stepping" directives and the $itemidx is not suitable. |

Variables available when in "stepping" repeating sections

| Variable | Description |
|---|---|
| <<$i1>>,..<<$iN>> | References to the Nth item when repeating data in "steps of N". For example <<rs_people:step3>> steps through the people in "steps of 3" and Docmosis automatically creates variables $i1, $i2 and $i3 to access each element in the step. |
| <<$idx1>>,..<<$idxN>> | Shorthand for $i1.$idx, … $iN.$idx |
| <<$num1>>,..<<$numN>> | Shorthand for $i1.$num, … $iN.$num |
| <<$itemidx1>>, ...<br><<$itemidxN>> | Shorthand for $i1.$itemidx, … $iN.$itemidx |
| <<$itemnum1>>, ...<br><<$itemnumN>> | Shorthand for $i1.$itemnum, … $iN.$itemnum |