Elements

Element	Description	Closing Element
<<name>>	Replace this field by the data referenced by "name".	
<<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>> <<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"	
<<$abc=name>> <<$abc=10.2>> <<$abc='Fred'>> <<$abc=true>> <<$abc=null>>	Lookup the data associated with "name" and assign it to the variable "abc". Assign the number 10.2 to variable $abc Assign the string "Fred" to variable $abc Assign the boolean true to variable $abc Assign the value null to variable $abc	
<<$abc>>	Lookup the variable "abc" and render its value	
<<cs_name>> <<cs_{expr}>> <<cs_$abc>>	Content between the opening element and the closing element is included or excluded depending on the value associated with "name" or the expression "expr" or the variable "abc". The end tag must match exactly, or may be anonymous: <<es_>>.	<<es_name>> <<es_{expr}>> <<es_$abc>> <<es_>>
<<else_name>> <<else_{expr}>> <<else>>	This is the "else" tag related to a <<cs_>> tag to provide the "else" and "else if" options to a condition.	
<<rs_name>> <<rs_$abc>> <<rs_name:step2>> <<rs_name:step2down>>	Content between the opening element and closing element is repeated whilst there is data associated with "name" or the variable "abc". "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. "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,... $iN are created automatically.	<<es_name>> <<es_$abc>> <<es_name:step2>> <<es_>>
<<cr_name>> <<cr_{expr}>> <<cr_$abc>>	Include the following table rows depending on the value associated with "name" or expression "expr" or the variable "abc".	<<er_name>> <<er_{expr}>> <<er_$abc>> <<er_>>
<<rr_name>> <<rr_$abc>> <<rr_name:step2>> <<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". "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. "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,... $iN are created automatically.	<<er_name>> <<er_$abc>> <<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.	
<<cc_name>> <<cc_{expr}>> <<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".	

Element	Description	Closing Element
`Image <<img_` *MS Word:* bookmarked with label "img_name" *OpenOffice or LibreOffice Writer:* image named "img_name" (deprecated "bm_name")	Replace an image in the template with the image data associated with "name" using the default scaling settings (which is stretch). The default setting can be changed by setting the docmosis property: `docmosis.analyzer.image.scaling.default` to `fit` or `stretch`. See the Docmosis Developer's Reference for information about setting properties.	
`Image <<imgstrech_` 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 <<imgfit_` 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.	
`<<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. eg. <<barcode:barcode1:code128>> defines image "barcode1" as a code 128 barcode.	
`<<## and ##>>` `<</* and */>>`	Template-comments are delimited by the matching open and closing sequences. Content inside comments is not processed and is removed when creating documents.	

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

General Functions and String Functions

Functions	Example
map	<<{map(gender, 'M', 'Male', 'F', 'Female', 'Other')}>>
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"
length	<<{length('Bob')}>> returns the number "3.0"
replace	<<{replace('JHMAB52EC8oo65o','o','0')}>> returns "JHMAB52EC800650"
split	<<{split('John|Mathews|47|Approved' , '|' , 1)}>> returns "Mathews"
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"
trim	<<{trim(' 12CVCV123-454 ')}>> returns "12CVCV123-454"
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"

Numeric Functions

Functions	Example
abs	<<{abs(-153.57)}>> returns "153.57"
ceil	<<{ceil(153.57)}>> returns "154.0"
floor	<<{floor(153.57)}>> returns "153.0"
max	<<{max(53.5,23.1)}>> returns "53.5"
min	<<{min(53.5,23.1)}>> returns "23.1"
pow	<<{pow(7,2)}>> returns "49.0"
random	<<{round(random()*100)}>> returns a random number between 0 and 100.
round	<<{round(153.73455,2)}>> returns "153.73"
sqrt	<<{sqrt(81.0)}>> returns "9.0"

Formatting Functions

Functions	Example
numFormat	<<{numFormat('1457.1', '#,###.00')}>> returns "1,457.10"
dateFormat	<<{dateFormat('2015-12-15' , 'EEEE, dd MMMM yyyy' , 'yyyy-MM-dd')}>> returns "Tuesday, 15 December 2015"

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
<<$top>> or <<$root>>	The root of the data regardless of the current position or context in the template
<<$this>> or <<$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.
<<$idx>>	The current index when iterating through a data set. For example, if we are repeating over all hotels, $idx would report the index of the hotel we are up to. Note that <<$rowidx>> is the same unless using a step size greater than 1.
<<$itemnum>>	Similar to $idx but is the number of the item which we are currently addressing. Item numbering starts at 1. Note that <<$rownum>> is the same unless using a step size greater than 1 .
<<$size>>	The size of the current repeating data set. For example if we are repeating over all hotels, $size would be the number of hotels.
<<$i1>>,<<$i2>>,..<<$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>>,<<$idx2>>,..<<$idxN>>	The absolute indexes of the items when repeating with "steps of N" (as described above) starting at zero.
<<$itemnum1>>,<<$itemnum2>>, ... <<$itemnumN>>	The absolute indexes of the items when repeating with "steps of N" (as described above) starting at one.
<<$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 $idx is not suitable.