Tornado
Quick Reference

Contents

docmosis

1. Fields

Element	Description

`<<name>>`
`<<first-name>>`
`<<{[first-name]}>>`
	Replace this field by the data referenced by "name" or "first-name". Hyphenated names are supported and need to be enclosed in [and] when used in expressions.

`<<## 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.
`<<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.
`<<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"
`<<html:name>>`	Lookup "name" in the data and inject the data as HTML content into the document at this location.
`<<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 in a repeating section and at the last iteration of the repeat.
Image 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).
Image stretched 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 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.
`<<barcode:name:…>>`	Provide information for a barcode image in the template. eg. <<barcode:barcode1:code128>> defines image "barcode1" as a code 128 barcode.
`<<ref:sub1.docx>>`	Insert the template named "sub1.docx" at this location.
`<<refLookup:name>>` `<<refLookupOp:name>>`	Lookup "name" in the data to get the name of the template to insert at this location. refLookup is strict and if the template reference is not provided an error is raised. refLookupOp allows the name to evaluate to blank without being treated as an error.
`<<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.

`<<fc:colour>>`
`<<fc:$colour>>`
`<<fc:#ff0000>>`
	Set the font-colour for all substituted-fields going forward from this point in the template. A null value means the colour is not changed.
	e.g. the tag <<fc:#00ff00>> would make all fields render in green from that point onwards.

`<<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.

Element	Description
`<<coordinator:padToEvenPage>>` `<<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. Currently this only applies to Docmosis Cloud and Tornado when it combines PDF output into a single PDF document.
`<<coordinator:newFile>>` `<<coordinator:$newFile>>`	At this point in processing the coordinator template, 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. Currently this only applies to Docmosis Cloud and Tornado when it combines PDF output into a single PDF document or creating a zip of documents.

2. Expressions

Element	Description
`<<{expr}>>`	Curly {} brackets trigger expression processing. This field is replaced with the results of the given expression.
`<<{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

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

Operator	Description
||	boolean or
!	boolean not

3. Repeating

Element	Description	Closing Element
`<<rs_items>>` `<<rs_$abc>>`	Content between the opening element and closing element is repeated whilst there is data associated with "items" or the variable "abc".	`<<es_items>>` `<<es_$abc>>` or `<<es_>>` for any match
`<<list:reset>>`	To be used in a repeating section to force a numbered list to restart numbering (rather than continue numbering from previous repeat).	
`<<rr_items>>` `<<rr_$abc>>`	In a table, the rows between the opening element row and the closing element row are repeated whilst there is data associated with "items" or the variable "abc".	`<<er_items>>` `<<er_$abc>>` or `<<er_>>` for any match
`<<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.	

3.1. Repeating With Stepping

Element	Description
`<<rs_items:step2>>` `<<rs_items:step2down>>` `<<rr_items:step2>>` `<<rr_items:step2down>>`	For the "items" data, repeat the content following until the matching closing element. "rs_" applies to general content, "rr_" applies to table rows. "stepN" indicates that the data ("items") should be iterated in steps of N size. When stepping is used, the variables $i1, $i2,…$iN are created automatically so items can be referenced in each step. This allows an array of data to be presented across a rows of N columns. "stepNdown" indicates that the data ("items") should be iterated in steps of N size and data should be presented in a "down"-ward (columnar) manner. Variables $i1, $i2,… $iN are created automatically. This allows an array of data to be presented in rows of N across, and values are placed filling column 1 first, then filling column 2 etc.

3.2. 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

Element	Description
`<<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

3.3. Repeating With Filters

Element	Description

`<<rs_persons:filter(ID<10)>>`
`<<rs_product:filter(startsWith(Code, 'AWA'))>>`
 	
`<<rr_persons:filter(ID<10)>>`
`<<rr_product:filter(startsWith(Code, 'AWA'))>>`
	Repeat over the "persons" data after filtering each item by the specified expression.

3.4. Repeating With Sort

Element	Description

`<<rs_persons:sort(name)>>`
`<<rs_persons:sort(DESC, name)>>`
`<<rs_persons:sort(DESC, CASE_SENSITIVE, NULLS_LAST, name)>>`
 	
`<<rs_persons:sortStr(name)>>`
`<<rs_persons:sortNum(ID)>>`
`<<rs_persons:sortDate(DOB)>>`
`<<rs_persons:sortDate(DOB, 'dd/MM/yyyy', 'German', false)>>`
 	
`<<rr_persons:sort(name)>>`
`<<rr_persons:sort(DESC, name)>>`
	Repeat over the given data after sorting by the given data-field or expression.
	
	Sort by "name" alphanumerically
	Sort by "name" descending
	Sort by "name" descending, case-sensitive, nulls last
	
	Sort by "name" as a string (not numerically smart)
	Sort by "ID" as a number
	Sort by "DOB" as a date using defaults
	Sort by "DOB" using with format specifiers
	
	Repeat over table rows using sort directives.
	
	Sort types:
	sort – alphanumeric sort
	sortStr – simple string-based sort
	sortNum – sort as numeric data
	sortDate – sort as data data
	
	Sort directives:
	ASC | DESC – ascending or descending (default ASC)
	CASE_SENSITIVE | CASE_INSENSITIVE (default sensitive)
	NULLS_FIRST | NULLS_LAST

3.5. Repeating With Grouping

Element	Description
`<<rs_animals:group(species)>>` `<<rs_persons:group(dateFormat(DOB,'MM','dd/MM/yyyy'))>>`	Repeat over "animals" grouped by "species". Repeat over "persons" grouped by "DOB".
`<<$groupKey>>`	Inside the repeat, $groupKey is the current grouped term.
`<<rs_$groupItems>>`	Inside the group repeat, repeat over the items in the current group.
`<<rs_animals:group(species)>>` `Species: <<$groupKey>>` `<<rs_$groupItems>>` `  Animal is <<name>>` `<<es_>>` `<<es_>>`	Typical use pattern for grouped data.
`<<rr_animals:group(species)>>` `Species: <<$groupKey>>` `<<rr_$groupItems>>` `  Animal is <<name>>` `<<er_>>` `<<er_>>`	Typical use pattern across rows of a table.

4. Conditional

Element	Description	Closing Element
`<<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_>>`
`<<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_>>`
`<<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.	
`<<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".	

5. Variables

5.1. Creating and Assigning

Variable	Description
`<<$abc=name>>`	Lookup the data associated with "name" and assign it to the variable "abc".

Variable	Description

`<<$abc=10.2>>`
`<<$abc='Fred'>>`
`<<$abc=true>>`
`<<$abc=null>>`
	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

`<<$abc>>`	Lookup the variable "abc" and render its value. The variable must have been defined before use otherwise an error is reported.
`<<$?abc>>`	"Forgiving" lookup of the variable "abc". If found its value is rendered, if not, nothing is rendered. This allows variables to be referenced in a less strict fashion for cases where the variable might not have been defined.

5.2. 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.
`<<$nl>>`	A simple newline character
`<<$nowMS>>`	Current UTC time in milliseconds since 1/1/1970
`<<$nowUTC>>`	Current UTC time as in ISO 8601 format
`<<$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)

5.3. Variables available when Repeating

Variable	Description
`<<$idx>>` Index into data	The current index into the source data, starting from zero.
`<<$itemidx>>` Index in 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.

5.4. Variables available when Stepping

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

Variable	Description
	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
`<<$itemidx1>>, ...` `<<$itemidxN>>`	Shorthand for $i1.$itemidx, ... $iN.$itemidx
`<<$itemnum1>>, ...` `<<$itemnumN>>`	Shorthand for $i1.$itemnum, ... $iN.$itemnum

5.5. Variables available when Grouping

Variable	Description
`<<$groupKey>>`	The key used for the current group of data. Eg if grouping by ProductCode, the key might be "123".
`<<rs_$groupItems>>` `<<rr_$groupItems>>`	A repeating section or row containing the data within the current group. Eg if grouping by ProductCode and the $groupKey is "123" then this would be a list of every product with the ProductCode "123".

6. Functions

6.1. Numeric

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"
`ordinal`	<<{ordinal(value [,'short'|'suffix'|'long'|'longNoAnds'|'longAllAnds'])}>> <<{ordinal(123)}>> returns "123rd" <<{ordinal(123, 'suffix')}>> returns "rd" <<{ordinal(123, 'long')}>> returns "one hundred and twenty third" <<{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])}>> <<{round(153.73455)}>> returns "154" <<{round(153.73455,2)}>> returns "153.73"
`sqrt`	<<{sqrt(81.0)}>> returns "9.0"

6.2. Text

Function	Synopsis / Example
`char`	<<{char(169)}>> returns the copyright character © The value can be specified in decimal, hex, html-style etc.
`charAt`	<<{charAt('abcdefg',3)}>> returns the character "d"

Function	Synopsis / Example
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])}>> <<{replaceStr('11 plus 11', '11', 'Eleven')}>> returns "Eleven plus Eleven"
replaceFirst	<<{replaceFirst(data, find, replaceWith [,ignoreCase])}>> <<{replaceFirst('11 plus 11', '11', 'Eleven')}>> returns "Eleven plus 11"
countStr	<<{countStr (data, find [, ignoreCase]})>> <<{countStr('Bob', 'b', true)}>> returns "2"
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"
left	<<{left('bob mathews', 3)}>> returns "bob"
right	<<{right('bob mathews', 7)}>> returns "mathews"
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"

6.3. Logic and Transform

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)}>> Eg. <<{map(gender, 'M', 'Male', 'F', 'Female', 'Other')}>>
mapi	<<{mapi(data, in1, out1, in2, out2,....,default)}>> (case-insensitive) Eg. <<{mapi(gender, 'M', 'Male', 'F', 'Female', 'Other')}>>

6.4. Locale

Function	Synopsis / Example
locale	<<{locale([spec])}>> where: spec = the language or country name or code to lookup <<{locale('GERMANY')}>> returns "de_DE"
localeInfo	<<{localeInfo([spec])}>>

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

7. Date and Number Formatting

Function	Synopsis / Example
dateAdd	<<{dateAdd(value, n, units [,outputFormat [,inputFormat]])}>> where units = millis | seconds | minutes | hours | days | weeks | months | years (plural optional) <<{dateAdd(date1, 1, 'day')}>> adds one day to date1 <<{dateAdd(date1, -2, 'months')}>> subtracts two months from date1 <<{dateAdd(date1, 1, 'year', 'yyyy')}>> add one year to the given date1 and output as 4 digit year <<{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
dateDiff	<<{dateDiff(date1, date2, units [,inputFormat])}>> where units = millis | seconds | minutes | hours | days | weeks | months | years (plural optional) <<{dateDiff(date1, date2, 'day')}>> calculate date2 minus date1 in days.
dateFormat	<<{dateFormat(value [,outputFormat [, inputFormat [, outputLocale [, inputLocale[, outputFormatLocalized [, inputFormatLocalized]]]]]])}>> <<{dateFormat('2015-12-15' , 'EEEE, dd MMMM yyyy' , 'yyyy-MM-dd')}>> returns "Tuesday, 15 December 2015"
numFormat	<<{numFormat(data, format [, locale [, applyLocaleToInput [, formatIsLocalized]]])}>> <<{numFormat('1457.1', '#,###.00')}>> returns "1,457.10" <<{numFormat('0,0257', '#.##0,00', 'nl')}>> returns "0,03" <<{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'])}>> <<{numToText(123)}>> returns "one hundred and twenty three" <<{numToText(123, 'andNone')}>> returns "one hundred twenty three"

8. Diagnostics

Element	Description / Example
<<dump:name>> <<dump:$builtInVariable>>	Dump the data as it is understood into the document for diagnostics purposes. Lookup "name" in the data and dump the data object into the document at this

---

Element	Description / Example
	location. Built in variables can also be used, eg: <dump:$top>> <dump:$parent>> <dump:$this>>