Supported math operators, formulas and functions

List of math operators and functions supported in ConfiForms Field macro (calculated/formula fields and TableView Merger macro) + ConfiForms ValueView macro

Please note that formulas are calculated **PER ROW**.

And if used within TableViewMerger or ValueView macros then calculated per row values might also be aggregated across the matched rows

> Function names are CASE SENSITIVE

Supported Operators

Mathematical Operators	
Operator	**Description**
+	Additive operator
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Remainder operator (Modulo)
^	Power operator

Boolean Operators*	
Operator	**Description**
=	Equals
==	Equals
!=	Not equals
<>	Not equals
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
&&	Boolean and
||	Boolean or

*Boolean operators result always in a BigDecimal value of 1 or 0 (zero). Any non-zero value is treated as a *true* value. Boolean *not* is implemented by a function.

Supported Functions

Function*	**Description**
NOT(*expression*)	Boolean negation, 1 (means true) if the expression is not zero

IF(*condition*,*value_if_true*,*value_if_false*)	Returns one value if the condition evaluates to true or the other if it evaluates to false
	
	Condition should be a mathematic expression that results in true or false or a function, such as EQUALS/EMPTY which has a true or false as a result
	
	`IF(1<2, "1", "2")`
	`IF(1==2, "one", "two")`
	`IF(EQUALS("1", "2"), "one", "two")`
	`IF([entry.field1]<[entry.field2], "Field1 is smaller", "Field2 is bigger")`
	
	`IF(EQUALS("[entry.field1]", "[entry.field2]"), "Values for field1 and field2 equally match", "Values for field1 and field2 are not equal")`

RANDOM()	Produces a random number between 0 and 1
MIN(*e1*,*e2*)	Returns the smaller of both expressions
MAX(*e1*,*e2*)	Returns the bigger of both expressions
ABS(*expression*)	Returns the absolute (non-negative) value of the expression
ROUND(*expression*, precision)	Rounds a value to a certain number of digits, uses the current rounding mode
FLOOR(*expression*)	Rounds the value down to the nearest integer
CEILING(*expression*)	Rounds the value up to the nearest integer
LOG(*expression*)	Returns the natural logarithm (base e) of an expression
SQRT(*expression*)	Returns the square root of an expression
SIN(*expression*)	Returns the trigonometric sine of an angle (in degrees)
ASIN(expression)	Returns the trigonometric ASIN of an angle
COS(*expression*)	Returns the trigonometric cosine of an angle (in degrees)
ACOS(expression)	Returns the trigonometric ACOS of an angle
TAN(*expression*)	Returns the trigonometric tangens of an angle (in degrees)
ATAN(expression)	Returns the trigonometric ATAN of an angle
SINH(*expression*)	Returns the hyperbolic sine of a value
COSH(*expression*)	Returns the hyperbolic cosine of a value
TANH(*expression*)	Returns the hyperbolic tangens of a value
RAD(*expression*)	Converts an angle measured in degrees to an approximately equivalent angle measured in radians
DEG(*expression*)	Converts an angle measured in radians to an approximately equivalent angle measured in degrees
FORMATDATE(*expression*)	Formats date (timestamp) using date format configured in Confluence
FORMATDATETIME(*expression*)	Formats datetime (timestamp) using datetime format configured in Confluence
NOW()	Useful for tracking last updated timestamps (could be used together with FORMATDATE or FORMATDATETIME) see below
FORMATFILESIZE(*value*)	Shows file size in MB and KB, instead of long value in bytes
FORMATMINSECAGO(*value*)	Shows minutes and seconds ago since the given timestamp
FORMATHOURMINSECAGO(*value*)	Shows hours, minutes and seconds ago since the given timestamp
FORMATHOURMINAGO(value)	Formats given timestamp value as a string with hours and minutes

FORMATDAYSAGO(*value*)	Shows days ago since the given timestamp
FORMATDATEAS (value, format)	Formats date in given format (format pattern should be https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html)
USER()	Returns current user full name
USERNAME()	Returns current user username
EMPTY(*value*)	Checks if given value is empty
NOTEMPTY(*value*)	Checks if given value is not empty
LEN(*value*)	Calculates length for given value (length = number of characters)
LENGTH(*value*)	Same as LEN(*value*)
FORMATNUMBER (value, format)	Where format is a pattern as described here https://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html. Example: FORMATNUMBER([entry.f1], "###,###.00")
ZEROIFEMPTY ("value")	If value is empty, then it will be passed further as 0. Useful when you might have an empty value for a field but would like to format it with FORMATNUMBER function for example
EQUALS(value1, value2) EQUALS("value1", "value2")	Compares two values. Return true if values are equal and false otherwise First example works when values are numeric (EQUALS(value1, value2)), while the 2nd example works for "text" values (EQUALS("value1", "value2"))
CONCAT("value1", "value2") SINCE V. 3.5.3 CONCAT("value1", "value2", <any_number of arguments>)	Will concatenate values together into one Since version 3.5.3 you can supply **any** number of arguments to CONCAT function For versions before the mentioned please use nesting `CONCAT(CONCAT("[entry.value1]", " [entry.value2]"), "[entry.value3]")`
MATCHES("value", regExpPattern) SINCE V. 3.4.5	Returns true if a given value matches the regular expression given

*Functions names are case insensitive.

For functions which work with text values and text values could potentially be empty then you must use quotes or double quotes for your parameters. For example:

- IF(EMPTY("[entry.somefield]"), "ERROR", "SUCCESS")
- IF(LEN("[entry.someotherfield]")>1, "Good", "Not good at all")

Supported Constants

Constant	Description
PI	The value of *PI*, exact to 100 digits
TRUE	The value one
FALSE	The value zero

Examples:

[entry.f1] + ([entry.f2] * [entry.f3])	Simple math expression, assuming f3 = 2, f2 = 1 and f1 = 5 the calculated value will be 7

IF(0, hi, bye)	bye
IF([entry.somefield], hi, bye)	depending on the field value: if 0 then "bye" will be outputted and "hi" otherwise
IF([entry.field1]+31, IF([entry.field2], 4, 12)*10, NA)	also, depends on a values for fields field1 and field2
FORMATDATE(NOW())	will print current date using Confluence date format
IF(EMPTY("[entry.somefield]", "ERROR", "SUCCESS")	will print ERROR if the value for field "somefield" is empty and SUCCESS if not empty
IF(LEN("[entry.someotherfield]")>1, "Good", "Not good at all")	will print *Good* if someotherfield's value is longer than 1 character (and if not then *Not good at all* is printed)
FORMATNUMBER([entry.f1], "###,###.00")	when entry.f1 = 100 the output will be: 100.00 when entry.f1 = 1100.01 the output will be: 1,100.01
FORMATNUMBER([entry.f1], "###,###.##")	when entry.f1 = 100 the output will be: 100 when entry.f1 = 1100.01 the output will be: 1,100.01
FORMATNUMBER(ZEROIFEMPTY("[entry.f1]"), "###,###.##")	when entry.f1 is empty (nothing set), then 0 will be given to FORMATNUMBER function and the result would be: 0
IF(EMPTY("[entry.somefield]", "ERROR", IF(EMPTY("[entry.anotherfield]", "ERROR", "SUCCESS"))))	to check if both values for fields "somefield" and "anotherfield" do present and set the label to "SUCCESS" (and to "ERROR" otherwise)

As always, using [entry.field_name] notations you can access other field properties (depending on a field type) and apply functions whenever needed

Accessing field values and properties

Virtual functions