DocuWare

Workflow Expression Parser

Documentation

March 2025

Disclaimer

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by DocuWare GmbH. DocuWare GmbH assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

DocuWare GmbH

Planegger Str. 1

D-82110 Germering

http://www.docuware.com

DocuWare

Table of contents

1 Introduction

For configuring workflows with DocuWare Workflow Designer, you can use the arithmetic expressions in this documentation. These are expressions from Visual Basic for Applications (VBA) and .NET as well as specific DocuWare customizations of expressions.

Unless otherwise specified, the functions are available in all currently supported DocuWare versions. Each DocuWare version is supported for 3 years from its release date.
For more information visit: DocuWare Support Lifecycle Policy

Both workflow variable names and document index field names are case sensitive.

When a function returns a different data type e.g. DateTime, but you want to assign it to a text variable, you must convert it to a string (CStr).

Example:

```
DateSerial(2022, 9, 16)
Returns: 16.09.2022 //DateTime

CStr(DateSerial(2022, 9, 16))
Returns: 16.09.2022 //String
```

Optional parameters are defined at the end of the parameter list, after the required parameters. They are described in brackets [ ]. If you specify an argument for one of several optional parameters, you must also specify arguments for all preceding optional parameters. Comma-separated gaps in the argument list are not supported.

1.1 Supported types

Various types can be used inside the expressions but when it comes to the result, the return type has to be one of the supported variable types in DocuWare. As in some situations incompatibility might be observed, caution is advised. For example, the *GetChar* VBA function returns a result of type Char. Char is not directly assignable to String, for example, so the whole expression should be converted to string, using the *CStr* function, in this specific case, or other related.

Date and Numeric workflow variables usually are of nullable type (DateTime?, Decimal? etc.). So, when you want to use a method of those types, for example *ToUniversalTime()* of a date variable you should use the <*variable*>.***Value*** and then the method: WF_CURRENT_DATE.Value.ToUniversalTime()

DocuWare variables of type keyword, index fields of type keyword and index table columns of type text or date can be used in methods that accept String[] or Object[] parameters. Index table columns of type decimal can be used in methods that accept String[] Int[] or Object[] parameters.

To provide some date handling, a special custom type **VbaDate** is introduced.

Properties

Name	Description
Now	Returns the current date and time
Today	Returns the current date (DATE datatype) according the UTC time zone.
Today(TimeZoneOffset)	Returns the current date (DATE datatype) according the specified time zone offset to UTC.

1.2 Supported constants

The following constants are supported:

- True
- False
- Null
- Nothing

1.3 Supported operators

The table below shows the operators supported by the expression language in order of precedence from the highest to the lowest. Operators in the same category have equal precedence.

In the table, x, y, and z denote expressions, T denotes a type, and m denotes a member.

Category	Expression	Description
Primary	x.m	Instance field or instance property access. Any public field or property can be accessed.
Primary	x.m(...)	Instance method invocation. The method must be public and must be declared in an accessible type.
Primary	x[...]	Array or indexer access. Multi-dimensional arrays are not supported.
Primary	T.m	Static field or static property access. Any public field or property can be accessed.
Primary	T.m(...)	Static method invocation. The method must be public and must be declared in an accessible type.
Primary	T(...)	Explicit conversion or constructor invocation. Note that new is not required in front of a constructor invocation.
Primary	iif(x, y, z)	Conditional expression. Alternate syntax for x ? y : z.
Unary	-x	Negation. Supported types are Int32, Int64, Decimal, Single, and Double.
Unary	!x not x	Logical negation. Operand must be of type Boolean.
Multiplicative	x * y	Multiplication. Supported types are Int32, UInt32, Int64, UInt64, Decimal, Single, and Double.
Multiplicative	x / y	Division. Supported types are Int32, UInt32, Int64, UInt64, Decimal, Single, and Double.
Multiplicative	x % y x mod y	Remainder. Supported types are Int32, UInt32, Int64, UInt64, Decimal, Single, and Double.

Category	Expression	Description
Additive	x + y	Addition or string concatenation. Performs string concatenation if either operand is of type String. Otherwise, performs addition for the supported types Int32, UInt32, Int64, UInt64, Decimal, Single, Double , DateTime, and TimeSpan.
Additive	x – y	Subtraction. Supported types are Int32, UInt32, Int64, UInt64, Decimal, Single, Double, DateTime, and TimeSpan.
Additive	x & y	String concatenation. Operands may be of any type.
Relational	x = y x == y	Equal. Supported for reference types and the primitive types. Assignment is not supported. Comparison with null:x == null is supported, but x is null is not supported.
Relational	x != y x <> y	Not equal. Supported for reference types and the primitive types.
Relational	x < y	Less than. Supported for all primitive types except Boolean, Object and Guid.
Relational	x > y	Greater than. Supported for all primitive types except Boolean, Object and Guid.
Relational	x <= y	Less than or equal. Supported for all primitive types except Boolean, Object and Guid.
Relational	x >= y	Greater than or equal. Supported for all primitive types except Boolean, Object and Guid.
Logical / Bitwise	x && y x AND y	Logical AND. Operands must be of type Boolean. Bitwise AND. Operands must be of type Numeric.
Logical / Bitwise	x || y y OR y	Logical OR. Operands must be Boolean. Bitwise OR. Operands must be numeric.
Bitwise	X >> Y	Right-shift operator.
Bitwise	X << Y	Left-shift operator.
Conditional	x ? y : z	Evaluates y if x is true, evaluates z if x is false.

1.4 Method and constructor invocations

The expression language limits the invocation of methods and constructors to those declared public in the accessible types. This restriction exists for protection against unintended side effects from the invocation of arbitrary methods.

The expression language permits getting (but not setting) the value of any reachable public field, property, or indexer.

The overload resolution for methods, constructors, and indexers uses rules similar to C#. In informal terms, the overload resolution will pick the best matching method, constructor, or indexer, or report an ambiguity error if no single best match can be identified.

Note that the constructor invocations are not prefixed by new. The following example creates a **DateTime** instance for a specific year, month, and day using a constructor invocation:

```
OrderDate >= DateTime(2007, 1, 1)
```

1.5 Enumerations

The following enumerations have been imported from Microsoft.VisualBasic namespace.

Enumeration	Description
CompareMethod	Indicates how to compare strings when calling comparison functions.
DateFormat	Indicates how to display dates when calling the **FormatDateTime** function. 0- GeneralDate 1- LongDate 2- ShortDate 3- LongTime 4- ShortTime
DateInterval	Indicates how to determine and format date intervals when calling date-related functions.
DueDate	Indicates when payments are due when calling financial methods.
FirstDayOfWeek	Indicates the first day of the week to use when calling date-related functions.
FirstWeekOfYear	Indicates the first week of the year to use when calling date-related functions.
TriState	Indicates a Boolean value or whether the default should be used when calling number-formatting functions.
VbStrConv	Indicates which type of conversion to perform when calling the **StrConv** function.

2 Supported VBA features

2.1 Control Chars

Name	Description
Back	Represents a backspace character (**vbBack**).
Cr	Represents a carriage return character (**vbCr**).
CrLf	Represents a carriage return/linefeed character combination (**vbCrLf**).
FormFeed	Represents a form feed character for print functions (**vbFormFeed**).
Lf	Represents a line feed character (**vbLf**).
NewLine	Represents a new line character (**vbNewLine**).
NullChar	Represents a null character (**vbNullChar**).
Quote	Represents a double-quote character.
Tab	Represents a tab character (**vbTab**).
VerticalTab	Represents a vertical tab character (**vbVerticalTab**).

2.2 Conversion

Name	Description
Format(Expression, [Format], [FirstDayOfWeek] , [FirstWeekOfYear]) Expression Required. Any valid expression. Format Optional. A valid named or user-defined format expression. FirstDayOfWeek Optional. A constant that specifies the first day of the week. FirstWeekOfYear Optional. A constant that specifies the first week of the year.	The VBA Format function applies a specified format to an expression and returns the result as a **String**. Examples: Format(50000) Returns: 50000 Format(50000, "Currency") Returns: $50,000.00 Format(0.88, "Percent") Returns: 88.00% Format(50000, "#,##0.0") Returns: 50,000.0 Format(0.88, "0.0") Returns: 0.9
Hex(Number)	Returns a **String** representing the hexadecimal value of a number.

Name	Description
Oct(Number)	Returns a **String** representing the octal value of a number.
Str(Number, [Culture])	Returns a **String** representation of a number. If *Culture* is provided, the output is formatted according to the specified Culture.
Val(String, [Culture])	Returns the numbers contained in a string as a **Numeric** value of appropriate type. If Culture is provided, the input string is interpreted according to the specified Culture.

2.3 DateAndTime

2.3.1 General considerations

Workflow expressions expect all date/time variables to be specified as UTC values.

When you enter date/time values in an expression, you can use the ToUniversalTime() method to convert the value to Coordinated Universal Time (UTC).

In DocuWare, date/time values are always automatically stored in UTC. They are adjusted at runtime, e.g. in the browser in the user's time zone.
When you convert a date/time value to a string, its value is fixed to the exact value, which is the UTC value. So, it is necessary to specify a culture and time zone for the conversion. Therefore, you can specify a **Culture** and/or **TimeZone** parameter for some methods. For more details see chapters Localization and Time zone specifics.

2.3.2 Properties

Name	Description
DateString([TimeZone])	Returns or sets a **String** value representing the current date according to your system. If TimeZone is supplied, the date in the specified time zone is taken (see Time zone specifics). Examples: Current date and time = 26.04.2022 11:30 DateString() Returns: 26.04.2022 To add or subtract X hours, you can use the TimeSpan.FromHours() function. DateString(TimeSpan.FromHours(-12)) Returns: 25.04.2022 DateString(TimeSpan.FromDays(-12)) Returns: 14.04.2022
Now	Returns a **DateTime** value containing the current date and time according to your system.

Name	Description
Timer	Returns a **Double** value representing the number of seconds elapsed since midnight.
TimeOfDay	Returns or sets a **Date** value containing the current time of day according to your system.

TimeString([TimeZone])	Returns a **String** value representing the current time of day according to your system. If TimeZone is supplied, the date in the specified time zone is taken (see Time zone specifics). Examples:
	
	Current date and time = 26.04.2022 11:30:24 UTC+2
	TimeString()
	Returns: 11:30:24
	
	To add or subtract X hours, you can use the TimeSpan.FromHours() function, which will denote the time zone which is before UTC.
	
	TimeString(TimeSpan.FromHours(-4)) //UTC-4
	Returns: 05.30:24

Today	Returns a **Date** value containing the current date according to your system.

2.3.3 Functions

Name	Description
DateAdd(Interval, Number, DateTime) Interval DateInterval. Required. A DateInterval enumeration value or a string expression representing the time interval you want to add. Number Double. Required. Floating-point expression representing the number of intervals you want to add. It can be positive (to get date/time values in the future) or negative (to get date/time values in the past). DateTime DateTime. Required. An expression representing the date and time to which the interval is to be added.	Returns a **DateTime** value containing a date and time value to which a specified time interval has been added. Example: Adds 2 days GV_MyDateTime = 26.04.2022 11:30 DateAdd(DateInterval.Day, 2, GV_MyDateTime) Returns: 28.04.2022

Name	Description
DateDiff(Interval, Date1, Date2, [FirstDayOfWeek], [FirstWeekOfYear]) Interval DateInterval. Required. A DateInterval enumeration value or a string expression representing the time interval you want to use as the unit of difference between Date1 and Date2. Date1 DateTime. Required. The first date/time value you want to use in the calculation. Date2 DateTime. Required. The second date/time value you want to use in the calculation. FirstDayOfWeek FirstDayOfWeek. Optional. A value chosen from the FirstDayOfWeek enumeration that specifies the first day of the week. If not specified, Sunday is used. FirstWeekOfYear FirstWeekOfYear. Optional. A value chosen from the FirstWeekOfYear enumeration that specifies the first week of the year. If not specified, Jan1 is used.	Returns a **Long** value specifying the number of time intervals between two **DateTime** values. Example: GV_MyDateTime = 26.04.2022 11:30 GV_MyDateTime2 = 30.04.2022 11:30 DateDiff("d", GV_MyDateTime, GV_MyDateTime2) Returns: 4
DatePart(Interval, DateTime, [FirstDayOfWeek], [FirstWeekOfYear]) Interval DateInterval. Required. A DateInterval enumeration value or a string expression representing the part of the date/time value you want to return. DateTime DateTime. Required. The Date value that you want to evaluate. FirstDayOfWeek FirstDayOfWeek. Optional. A value chosen from the FirstDayOfWeek enumeration that specifies the first day of the week. If not specified, Sunday is used.	Returns an **Integer** value containing the specified component of a given **DateTime** value. Example: GV_MyDateTime = 26.04.2022 11:30 DatePart("ww", GV_MyDateTime) Returns: 18

Name	Description
FirstWeekOfYear FirstWeekOfYear. Optional. A value chosen from the FirstWeekOfYear enumeration that specifies the first week of the year. If not specified, Jan1 is used.	
DateSerial(Year, Month, Day) Year Int32. Required. Integer expression from 1 through 9999. If Year is less than 1, it is subtracted from the current year. Month Int32. Required. Integer expression from 1 through 12. Day Int32. Required. Integer expression from 1 through 31.	Returns a **DateTime** value representing a specified year, month, and day, with the time information set to UTC midnight (00:00:00). Example: DateSerial(2022, 9, 16) Returns: 16.09.2022 //DateTime If you want to convert the value to string without showing any time portion, you can use the CStr function around this. CStr(DateSerial(2022, 9, 16)) Returns: 16.09.2022 //String
DateValue(String, [Culture], [TimeZone]) String String. Required. String expression representing a date/time value from 00:00:00 on January 1 of the year 1 through 23:59:59 on December 31, 9999.	Returns a **DateTime** value containing the date information represented by a string, with the time information set to UTC midnight (00:00:00). The string is interpreted according to the Culture parameter, if provided. If TimeZone is provided - Input time string is compensated first with the time-zone offset and then the Date-only portion is returned.
Year(DateTime) DateTime DateTime. Required. A Date value from which you want to extract the year.	Returns an **Integer** value from 1 through 9999 representing the year. Example: GV_MyDateTime = 26.04.2022 11:30 Year(GV_MyDateTime) Returns: 2022
Month(DateTime) DateTime DateTime. Required. A Date value from which you want to extract the year.	Returns an **Integer** value from 1 through 12 representing the month of the year. Example: GV_MyDateTime = 26.04.2022 11:30 Month(GV_MyDateTime) Returns: April OR 4
MonthName(Month, [Abbreviate]) Month Int32. Required. The numeric designation of the month, from 1 through 13; 1 indicates January and 12 indicates December. You can use the value 13 with a 13-month calendar. If your system is using a 12-month calendar and Month is 13, MonthName returns an empty string.	Returns a **String** value containing the name of the specified month. Set "true" to abbreviate the month name. Example: GV_MyDateTime = 26.04.2022 11:30 MonthName(Month(GV_MyDateTime, true)) Returns: Apr

Name	Description
Abbreviate Boolean Optional. True to abbreviate the month name; otherwise, False. The default is False.	
Weekday(DateTime, [FirstDayOfWeek]) DateTime DateTime. Required. A Date value for which you want to determine the day of the week. FirstDayOfWeek FirstDayOfWeek. Optional. A value chosen from the FirstDayOfWeek enumeration that specifies the first day of the week. If not specified, Sunday is used.	Returns an **Integer** value chosen from the FirstDayOfWeek enumeration that specifies the first day of the week. If not specified, Sunday is used. Example: GV_MyDateTime = 26.04.2022 11:30 Weekday(GV_MyDateTime) Returns: 3
WeekdayName(Weekday, [Abbreviate], [FirstDayOfWeek], [Culture]) Weekday Int32. Required. The numeric designation for the weekday, from 1 through 7; 1 indicates the first day of the week and 7 indicates the last day of the week. The identities of the first and last days depend on the setting of FirstDayOfWeekValue. Abbreviate Boolean. Optional. Boolean value that indicates if the weekday name is to be abbreviated. If omitted, the default is False. FirstDayOfWeek FirstDayOfWeek. Optional. A value chosen from the FirstDayOfWeek enumeration that specifies the first day of the week. If not specified, FirstDayOfWeek.System is used.	Returns a **String** value containing the name of the specified weekday. If Culture is provided, the weekday name is according to the specified Culture. Example: GV_MyDateTime = 26.04.2022 11:30 WeekdayName(Weekday(GV_MyDateTime)) Returns: Tuesday
Day(DateTime) DateTime DateTime. Required. A Date value from which you want to extract the day.	Returns an **Integer** value from 1 through 31 representing the day of the month of the input. Example: GV_MyDateTime = 26.04.2022 Day(GV_MyDate) Returns: 26
Hour(DateTime)	Returns an **Integer** value from 0 through 23 representing the hour of the day. Important: The hour is calculated from time in UTC (UTC+0 time!). If you need the time in your local time

Name	Description
DateTime DateTime. Required. A Date value from which you want to extract the hour.	zone, you have to add/subtract the corresponding offset. Example: GV_MyDateTime = 26.04.2022 11:30 (UTC+2) Hour(GV_MyDateTime) Returns: 9
Minute(DateTime) DateTime DateTime. Required. A Date value from which you want to extract the minute.	Returns an **Integer** value from 0 through 59 representing the minute of the hour. Example: GV_MyDateTime = 26.04.2022 11:30 Minute(GV_MyDateTime) Returns: 30
Second(DateTime) DateTime DateTime. Required. A Date value from which you want to extract the second.	Returns an **Integer** value from 0 through 59 representing the second of the minute. Example: GV_MyDateTime = 26.04.2022 11:30:36 Second(GV_MyDateTime) Returns: 36
TimeSerial(Hour, Minutes, Second) Hour Int32. Required. Integer expression from 0 through 23. However, values outside this range are also accepted. Minute Int32. Required. Integer expression from 0 through 59. However, values outside this range are also accepted. Second Int32. Required. Integer expression from 0 through 59. However, values outside this range are also accepted.	Returns a **DateTime** value representing a specified hour, minute, and second, with the date information set relative to January 1 of the year 1.
TimeValue(StringTime, [Culture], [TimeZone]) StringTime String. Required. A string expression representing a date/time value from 00:00:00 on January 1 of the year 1 through 23:59:59 on December 31, 9999.	Returns a **DateTime** value containing the time information represented by a string, with the date information set to January 1 of the year 1. The string is interpreted according to the Culture parameter, if provided. If TimeZone is provided - Input time string is compensated with the time-zone offset.

2.4 Financial

Name	Description
DDB(Cost, Salvage, Life, Period, Factor) Get more information here	Returns a **Double** specifying the depreciation of an asset for a specific time period using the double-declining balance method or some other method you specify.
FV(Rate, NPer, Pmt, PV, DueDate) Get more information here	Returns a **Double** specifying the future value of an annuity based on periodic, fixed payments and a fixed interest rate.
IPmt(Rate, Per, NPer, PV, FV, DueDate) Get more information here	Returns a **Double** specifying the interest payment for a given period of an annuity based on periodic, fixed payments and a fixed interest rate.
NPer(Rate, Pmt, PV, FV, DueDate) Get more information here	Returns a **Double** specifying the number of periods for an annuity based on periodic fixed payments and a fixed interest rate.
Pmt(Rate, NPer, PV, FV, DueDate) Get more information here	Returns a **Double** specifying the payment for an annuity based on periodic, fixed payments and a fixed interest rate.
PPmt(Rate, Per, NPer, PV, FV, DueDate) Get more information here	Returns a **Double** specifying the principal payment for a given period of an annuity based on periodic fixed payments and a fixed interest rate.
PV(Rate, NPer, Pmt, FV, DueDate) Get more information here	Returns a **Double** specifying the present value of an annuity based on periodic, fixed payments to be paid in the future and a fixed interest rate.
Rate(NPer, Pmt, PV, FV, DueDate, Guess) Get more information here	Returns a **Double** specifying the interest rate per period for an annuity.
SLN(Cost, Salvage, Life) Get more information here	Returns a **Double** specifying the straight-line depreciation of an asset for a single period.
SYD(Cost, Salvage, Life, Period) Get more information here	Returns a **Double** specifying the sum-of-years digits depreciation of an asset for a specified period.

2.5 Information

Name	Description
IsArray(Object) Get more information here	Returns a **Boolean** value indicating whether a variable points to an array.
IsDate(Object, [Culture]) Get more information here	Returns a **Boolean** value indicating whether an expression represents a valid **Date** value. If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture.

Name	Description
IsNothing(Object) Get more information here	Returns a **Boolean** value indicating whether an expression has no object assigned to it. Example: IsNothing(GV_ MyVariable) Returns: 0 or 1 //false or true
IsNumeric(Object, [Culture]) Get more information here	Returns a **Boolean** value indicating whether an expression can be evaluated as a number. Example: IsNumeric(GV_MyVariable) Returns: 0 or 1 //false or true If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture.
IsReference(Object) Get more information here	Returns a **Boolean** value indicating whether an expression evaluates to a reference type.
LBound(Array, [Rank]) Array Array. Required. Array of any data type. The array in which you want to find the lowest possible subscript of a dimension. Rank Int32. Optional. Integer. The dimension for which the lowest possible subscript is to be returned. Use 1 for the first dimension, 2 for the second, and so on. If Rank is omitted, 1 is assumed.	Returns the lowest available subscript for the indicated dimension of an array.
QBColor(Color) Color Int32. Required. A whole number in the range 0-15.	Returns an **Integer** value representing the RGB color code corresponding to the specified color number.
RGB(Red, Green, Blue) Red Int32. Required. Integer in the range 0-255, inclusive, that represents the intensity of the red component of the color. Green Int32. Required. Integer in the range 0-255, inclusive, that represents the intensity of the green component of the color. Blue Int32. Required. Integer in the range 0-255, inclusive, that represents the intensity of the blue component of the color.	Returns an **Integer** value representing an RGB color value from a set of red, green and blue color components.

Name	Description
UBound(Array, [Rank]) Array Array. Required. Array of any data type. The array in which you want to find the lowest possible subscript of a dimension. Rank Int32. Optional. The dimension for which the lowest possible subscript is to be returned. Use 1 for the first dimension, 2 for the second, and so on. If Rank is omitted, 1 is assumed.	Returns the highest available subscript for the indicated dimension of an array.

2.6 Strings

Name	Description
Asc(String) String String/Char. Required. Any valid Char or String expression. If String is a String expression, only the first character of the string is used for input. If String is Nothing or contains no characters, an ArgumentException error occurs.	Returns an **Integer** value representing the character code corresponding to a character.
AscW(String) String String/Char. Required. Any valid Char or String expression. If String is a String expression, only the first character of the string is used for input. If String is Nothing or contains no characters, an ArgumentException error occurs.	Returns an **Integer** value representing the character code corresponding to a character.
Chr(CharCode) CharCode Int32. Required. An Integer expression representing the code point, or character code, for the character.	Returns the character associated with the specified character code.
ChrW(CharCode) CharCode Int32. Required. An Integer expression representing the	Returns the character associated with the specified character code.

Name	Description
code point, or character code, for the character.	
Filter(Array, Match, [Include], [CompareMethod]) Array Array. Required. One-dimensional array of strings to be searched. Match String. Required. String to search for. Include Boolean. Optional. Boolean value indicating whether to return substrings that include or exclude Match. If Include is True, the Filter function returns the subset of the array that contains Match as a substring. If Include is False, the Filter function returns the subset of the array that does not contain Match as a substring. CompareMethod CompareMethod. Optional. Numeric value indicating the comparison to use when evaluating substrings. 0 performs a binary comparison 1 performs a textual comparison	Returns a zero-based array containing a subset of a **String** array based on specified filter criteria. Example: GV_MyKeyword: { "A", "B", "C" } Filter(GV_MyKeyword, "A") Returns: { "A" }
Format(Expression, [Style | Culture]) Expression Object. Required. Any valid expression. Style String. Optional. A valid named or user-defined format String expression.	Returns a string formatted according to instructions contained in a format **Style** expression. If **Culture** is provided, the output is formatted according to the specified Culture.
FormatCurrency(Expression, [NumDigitsAfterDecimal], [IncludeLeadingDigit], [UseParensForNegativeNumbers], [GroupDigits], [Culture]) Expression Object. Required. Expression to be formatted. NumDigitsAfterDecimal Int32. Optional. Numeric value indicating how many	Returns an expression formatted as a currency value using the currency symbol defined in the system control panel. If **Culture** is provided, the output is formatted according to the specified Culture.

Name	Description
places are displayed to the right of the decimal. Default value is -1, which indicates that the computer's regional settings are used. IncludeLeadingDigit TriState. Optional. TriState enumeration that indicates whether or not a leading zero is displayed for fractional values. UseParensForNegativeNumbers TriState. Optional. TriState enumeration that indicates whether or not to place negative values within parentheses. GroupDigits TriState. Optional. TriState enumeration that indicates whether or not numbers are grouped using the group delimiter specified in the computer's regional settings.	
FormatDateTime(DateTime, [DateFormat], [Culture]) DateTime DateTime. Required. Date expression to be formatted. DateFormat DateFormat. Optional. Numeric value that indicates the date/time format used. If omitted, DateFormat.GeneralDate is used.	Returns a string expression representing a date/time value. DateFormat indicates how to display dates: 0- GeneralDate 1- LongDate 2- ShortDate 3- LongTime 4- ShortTime If **Culture** is provided, the output is formatted according to the specified Culture.
FormatNumber(Expression, [NumDigitsAfterDecimal], [IncludeLeadingDigit], [UseParensForNegativeNumbers], [GroupDigits], [Culture]) Expression Object. Required. Expression to be formatted. NumDigitsAfterDecimal Int32. Optional. Numeric value indicating how many places are displayed to the right of the decimal. The default value is -1, which indicates that the computer's regional settings are used. IncludeLeadingDigit TriState. Optional. TriState constant that indicates whether	Returns an expression formatted as a number. If **Culture** is provided, the output is formatted according to the specified Culture.

Name	Description
a leading 0 is displayed for fractional values. UseParensForNegativeNumbers TriState. Optional. TriState constant that indicates whether to place negative values within parentheses. GroupDigits TriState. Optional. TriState constant that indicates whether or not numbers are grouped using the group delimiter specified in the locale settings.	
FormatPercent(Expression, [NumDigitsAfterDecimal], [IncludeLeadingDigit], [UseParensForNegativeNumbers], [GroupDigits], [Culture]) Expression Object. Required. Expression to be formatted. NumDigitsAfterDecimal Int32. Optional. Numeric value indicating how many places are displayed to the right of the decimal. The default value is -1, which indicates that the computer's regional settings are used. IncludeLeadingDigit TriState. Optional. TriState constant that indicates whether a leading 0 is displayed for fractional values. UseParensForNegativeNumbers TriState. Optional. TriState constant that indicates whether to place negative values within parentheses. GroupDigits TriState. Optional. TriState constant that indicates whether or not numbers are grouped using the group delimiter specified in the locale settings.	Returns an expression formatted as a percentage (that is, multiplied by 100) with a trailing % character. If **Culture** is provided, the output is formatted according to the specified Culture.
GetChar(String, Index) String String. Required. Any valid String expression.	Returns a **Char** value representing the character from the specified index in the supplied string.

Name	Description
Index Int32. Required. Integer expression. The (1-based) index of the character in str to be returned.	
InStr([Start], String1, String2, [CompareMethod]) Start Int32. Optional. Numeric expression that sets the starting position for each search. If omitted, search begins at the first character position. The start index is 1-based. String1 String. Required. String expression being searched. String2 String. Required. String expression sought. CompareMethod CompareMethod. Optional. Numeric value indicating the comparison to use when evaluating substrings. 0 performs a binary comparison 1 performs a textual comparison	Returns an integer specifying the start position of the first occurrence of one string within another. Example: GV_MyVariable = ThisIs1Variable InStr(GV_MyVariable, "1") Returns: 7
InStrRev(StringCheck, StringMatch, [Start], [CompareMethod]) StringCheck String. Required. String expression being searched. StringMatch String. Required. String expression being searched for. Start Int32. Optional. Numeric expression setting the one-based starting position for each search, starting from the left side of the string. If Start is omitted then -1 is used, meaning the search begins at the last character position. Search then proceeds from right to left. CompareMethod CompareMethod. Optional. Numeric value indicating the comparison to use when evaluating substrings.	Returns the position of the first occurrence of one string within another, starting from the right side of the string. Example: GV_MyVariable = ThisIs1Variable InStrRev(GV_MyVariable, "1") Returns: 9

Name	Description
0 performs a binary comparison 1 performs a textual comparison.	
Join(Array, [Delimiter]) Array Array. Required. One-dimensional array containing substrings to be joined. Delimiter String. Optional. Any string, used to separate the substrings in the returned string. If omitted, the space character (" ") is used. If Delimiter is a zero-length string ("") or Nothing, all items in the list are concatenated with no delimiters.	Returns a string created by joining a number of substrings contained in an array.
LCase(String) String String or Char. Required. Any valid String or Char expression.	Returns a string or character converted to lowercase. Example: GV_MyVariable = ThisIsAVariable LCase(GV_MyVariable) Returns: thisisavariable
Left(String, Length) String String. Required. String expression from which the leftmost characters are returned. Length Int32. Required. Integer expression. Numeric expression indicating how many characters to return. If 0, a zero-length string ("") is returned. If greater than or equal to the number of characters in string, the entire string is returned.	Returns a string containing a specified number of characters from the left side of a string. Example: GV_MyVariable: ThisIsAVariable Left(GV_MyVariable, 4) Returns: This
Len(Boolean)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
Len(Char)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
Len(DateTime)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
Len(Decimal)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.

Name	Description
Len(Int)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
Len(String)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
LSet(String, Length) String String. Required. String expression. Name of string variable. Length Int32. Required. Integer expression. Length of returned string.	Returns a left-aligned string containing the specified string adjusted to the specified length.
LTrim(String) String String. Required. Any valid String expression.	Returns a string containing a copy of a specified string with no leading spaces (**LTrim**), no trailing spaces (**RTrim**), or no leading or trailing spaces (**Trim**). Example: Example: GV_MyVariable = This is a variable LTrim(GV_MyVariable) Returns: This is a variable
Mid(String, Start) String String. Required. String expression from which characters are returned. Start Int32. Required. Integer expression. Starting position of the characters to return. If Start is greater than the number of characters in string, the Mid function returns a zero-length string (""). Start is one-based.	Returns a string that contains all the characters starting from a specified position in a string. Example: GV_MyVariable = My test string Mid(GV_MyVariable, 4) Returns: test string
Replace(String, Find, Replacement, [Start], [Count], [CompareMethod]) String String. Required. String expression containing substring to replace. Find String. Required. Substring being searched for.	Returns a string in which a specified substring has been replaced with another substring a specified number of times. Examples: GV_MyVariable = This*Is*A*Variable Replace(GV_MyVariable, "*","#") Returns: This#Is#A#Variable REPLACE ("go GO Go", "go", "went", 1, -1, 0) Returns: went GO Go //case sensitive REPLACE ("go GO Go", "go", "went", 1, -1, 1) Returns: went went went //case insensitive

Name	Description
Replacement String. Required. Replacement substring. Start Int32. Optional. Position within Expression that starts a substring used for replacement. The return value of Replace is a string that begins at Start, with appropriate substitutions. If omitted, 1 is assumed. Count Int32. Optional. Number of substring substitutions to perform. If omitted, the default value is -1, which means "make all possible substitutions." CompareMethod CompareMethod. Optional. Numeric value indicating the comparison to use when evaluating substrings. 0 performs a binary comparison 1 performs a textual comparison	When you specify a starting position, Replace truncates the characters from the 1st character to the Nth character you specify: REPLACE ("go GO Go", "go", "went", 3, -1, 1) Returns: went went To replace ALL letters use: DW_VARIABLE = ThisIsAVariable123 Replace(Replace(Replace(Replace(Replace(Replace(Replace(Replace (Replace(Replace(Replace(Replace(Replace(Replace(Replace(Replace (Replace(Replace(Replace(Replace(Replace(Replace(Replace(Replace (Replace(Replace(Replace(Replace(Replace(Replace(UCase(DW_VAR IABLE),"A",""),"B",""),"C",""),"D",""),"E",""),"F",""),"G",""),"H", ""),"I",""),"J", ""),"K",""),"L",""),"M",""),"N",""),"O",""),"P",""),"Q",""),"R","")," S",""),"T",""), "U",""),"V",""),"W",""),"X",""),"Y",""),"Z",""),"Ü",""),"Ä",""),"Ö", ""),"ß","") Returns: 123 Note that you can also replace strings using the String.Replace() or the Regex.Replace() method.
Right(String, Length) String String. Required. String expression from which the leftmost characters are returned. Length Int32. Required. Integer expression. Numeric expression indicating how many characters to return. If 0, a zero-length string ("") is returned. If greater than or equal to the number of characters in string, the entire string is returned.	Returns a string containing a specified number of characters from the right side of a string. Example: GV_MyVariable = This is a variable Right(GV_MyVariable, 4) Returns: able
RSet(String, Length) String String. Required. String expression. Name of string variable. Length Int32. Required. Integer expression. Length of returned string.	Returns a right-aligned string containing the specified string adjusted to the specified length.

Name	Description
RTrim(String) String String. Required. Any valid String expression.	Returns a string containing a copy of a specified string with no leading spaces (**LTrim**), no trailing spaces (**RTrim**), or no leading or trailing spaces (**Trim**). Example: GV_MyVariable = This is a variable RTrim(GV_MyVariable) Returns: This is a variable
Space(Number) Number Int32. Required. Integer expression. The number of spaces you want in the string.	Returns a string consisting of the specified number of spaces.

Split(String, [Delimiter], [Limit], [CompareMethod])
 	
String
String. Required.
String expression containing substrings and delimiters.
 	
Delimiter
String. Optional.
Any single character used to identify substring limits. If Delimiter is omitted, the space character (" ") is assumed to be the delimiter.
 	
Limit
Int32. Optional.
Maximum number of substrings into which the input string should be split. The default, -1, indicates that the input string should be split at every occurrence of the Delimiter string.
 	
CompareMethod
CompareMethod. Optional.
Numeric value indicating the comparison to use when evaluating substrings.
0 performs a binary comparison
1 performs a textual comparison
	Returns a zero-based, one-dimensional array containing a specified number of substrings. Examples:
	
	Split("a:b:c", ":", 2)
	Returns: {"a", "b:c"}
	
	Split("abc", "B", -1, 1) //case insensitive
	Returns: {"a", "c"}
	
	GV_MyVariable = My*test*string
	Split(GV_MyVariable, "*")
	Returns: ["My", "test", "string"]
	
	To return the first element of the array use (0), then you can assign it to a text variable:
	Split(GV_MyVariable, "*")(0)
	Returns: My
	
	Note that you can also split strings using the String.Split() or the Regex.Split() method.

StrComp(String1, String2, [CompareMethod]) String1 String. Required. Any valid String expression. String2 String. Required. Any valid String expression.	Returns -1, 0, or 1, based on the result of a string comparison. String1 sorts ahead of String2 -1 String1 is equal to String2 0 String1 sorts after String2 1

Name	Description
CompareMethod CompareMethod. Optional. Numeric value indicating the comparison to use when evaluating substrings. 0 performs a binary comparison 1 performs a textual comparison	
StrConv(String, Conversion, LocaleID) String String. Required. String expression to be converted. Conversion VbStrConv. Required. VbStrConv member. The enumeration value specifying the type of conversion to perform. LocaleID Int32 Optional. The LocaleID value, if different from the system LocaleID value. (The system LocaleID value is the default.)	Returns a string converted as specified. Example: StrConv("A String", VbStrConv.Lowercase) Returns: a string Only following constants are supported: • 0 VbStrConv.None • 1 VbStrConv.Uppercase • 2 VbStrConv.Lowercase • 3 VbStrConv.ProperCase • 4 VbStrConv.Wide (applies to Katakana only) • 8 VbStrConv.Narrow (applies to Katakana only)
StrDup(Number, Character) Number Int32. Required. Integer expression. The length to the string to be returned. Character Char. Required. Any valid Char, String, or Object expression. Only the first character of the expression will be used. If Character is of type Object, it must contain either a Char or a String value.	Returns a string or object consisting of the specified character repeated the specified number of times. Example: StrDup(5, "D") Returns: DDDDD
StrReverse(String) String String. Required. String expression whose characters are to be reversed.	Returns a string in which the character order of a specified string is reversed. Example: StrReverse("DocuWare") Returns: eraWucoD
Trim(String) String String. Required. Any valid String expression.	Returns a string containing a copy of a specified string with no leading spaces (**LTrim**), no trailing spaces (**RTrim**), or no leading or trailing spaces (**Trim**).Example: GV_MyVariable = This is a variable Trim(GV_MyVariable) Returns: This is a variable
UCase(Char)	Returns a string or character containing the specified string converted to uppercase.

Name	Description
UCase(String)	Returns a string or character containing the specified string converted to uppercase. Example: GV_MyVariable: This*is*a*variable UCase(GV_MyVariable) Returns: THIS*IS*A*VARIABLE

2.7 Math

Name	Description
Fix(Number)	Returns the integer portion of a number. Returns 0 if parameter is null.
Int(Number)	Returns the integer portion of a number. Returns 0 if parameter is null.
Rnd()	Returns a random number of type **Single**.
Rnd(Single)	Returns a random number of type **Single**. If the parameter number is less than zero, Rnd generates the same number every time, using *Number* as the seed. If number is greater than zero, Rnd generates the next random number in the sequence. If number is equal to zero, Rnd generates the most recently generated number. If number is not supplied, Rnd generates the next random number in the sequence.

3 Supported .NET features

To further enhance the functional range of the Workflow Expression Parser, we have added support for the functions contained in classes within the System namespace of Microsoft .NET Framework. For example:

- DateTime(GV_Year, GV_Month, GV_Day) – will yield a DateTime value
- GV_MyVariable.Split('*') – will split the provided text variable by *
- GV_MyVariable.Replace('*', '#') – will replace * with # for provided text variable

Here is a list of the supported C# types and classes:

Boolean	UInt64
Char	Single
String	Double
SByte	Decimal
Byte	DateTime
Int16	DateTimeOffset
UInt16	TimeSpan
Int32	Math
UInt32	Convert
Int64	Regex

These types/classes can be used with their constructors and/or to invoke their members. More information and examples of the 4 classes **String**, **Math**, **Convert** and **Regex** can be found in the next subchapters.

Members with the OUT keyword in front of a parameter are **not** supported.

Members with the PARAMS keyword in front of a parameter are **not** supported.
In some cases, this limitation can be worked around by converting the parameter to array. For example:

- GV_MyVariable.Split(" ".ToCharArray()) – split by space
- GV_MyVariable.Split(", ".ToCharArray()) – split by space and comma
- GV_MyVariable.Split(" ,;".ToCharArray()) – split by space, comma, and semi-colon.

Members with CLSCompliant attribute set to false are also **not** supported.

3.1 String Class (System.String)

The String class provides many methods for safely creating, manipulating, and comparing strings.

A reference to the class functionality can be found on Microsoft's online documentation site: https://docs.microsoft.com/en-us/dotnet/api/system.string

Only parameters of types inherent to the DocuWare infrastructure are supported (like int, decimal, string, keyword (string[]), DateTime).

Here are some examples of the String Class that can be used in workflow expressions:

Name	Description
String.Concat(Array) Array Array. Required. One-dimensional array containing substrings to be concatenated	Concatenates all the elements of a string array. DW_COST_CENTER: {10000, 20000} String.Concat(DW_COST_CENTER) Return: 1000020000
String.IsNullOrEmpty(String) String String. Required. The string to test.	Indicates whether the specified string is null or an empty string ("").Returns **Boolean** GV_MyVariable: Test String.IsNullOrEmpty(GV_MyVariable) Returns: false
String.IsNullOrWhiteSpace(String) String String. Required. The string to test.	Indicates whether a specified string is null, empty, or consists only of white-space characters. Returns **Boolean** GV_MyVariable: Test String.IsNullOrWhiteSpace(GV_MyVariable) Returns: false
String.Join(Delimiter, Array) Array Array. Required. One-dimensional array containing substrings to be joined. Delimiter String. Optional. Any string, used to separate the substrings in the returned string. If omitted, the space character (" ") is used. If Delimiter is a zero-length string ("") or Nothing, all items in the list are concatenated with no delimiters.	Concatenates all the elements of a string array, using the specified delimiter between each element. DW_COST_CENTER: {10000, 20000} String.Join("#",DW_COST_CENTER) Return: 10000#20000 Note that you can also split join using the Join() method.
<MyString>.Contains(Value) Value String. Required. The string to compare	Returns a value indicating whether a specified character occurs within this string.

Name	Description
<MyString>.EndsWith(Value) Value String. Required. The string to compare	Determines whether the end of this string instance matches a specified string.
<MyString>.Replace(Find, Replacement, IgnoreCase, Culture) String String. Required. String expression containing substring to replace. Find String. Required. Substring being searched for. Replacement String. Required. Replacement substring. IgnoreCase Bool. Optional. true: to ignore casing when comparing; false: otherwise. Culture CultureInfo. Optional. The culture to use when comparing.	Returns a new string in which all occurrences of Find String are replaced with Replacement String. GV_MyVariable = This*Is*A*Variable GV_MyVariable.Replace("*","#") Returns: This#Is#A#Variable GV_MyVariable = Find the first small letter GV_MyVariable.Replace("F","M") Returns: Mind the first small letter GV_MyVariable = Find the first small letter GV_MyVariable.Replace("F","M", true, CultureInfo.InvariantCulture) Returns: Mind the mirst small letter Note that you can also split strings using Replace() or the Regex.Replace() method.
<MyString>.StartsWith(Value) Value String. Required. The string to compare	Determines whether the beginning of the string matches the specified string.

<MyString>.Split(Delimiter, [Count], [Options])
 	
Delimiter
String/char/char[]/string[]. Required.
A String, char or an array of separators (char or string) to split to.
 	
Count
Int. Optional
The maximum number of elements expected in the array.
 	
Options
StringSplitOptions. Optional
RemoveEmptyEntries: Omit array elements that contain an empty string from the result
TrimEntries: Trim white-space characters from each substring in the result
 	
For more options read here
	Returns a string array as a result of split a string into substrings that are based on the provided string separator.
	
	GV_MyVariable = My*test*string
	GV_MyVariable.Split('*') //delimiter is in single quote
	Returns: {"My", "test", "string"}
	
	If you want to use more than one separator, use a list with separators in double quotes ",;." and apply ToCharArray():
	GV_MyVariable.Split(",;.*".ToCharArray())
	Returns: {"My", "test", "string"}
	
	If you want to remove empty entries in the result use the split option "RemoveEmptyEntries":
	DW_STATUS = ONE,,TWO,,THREE,
	DW_STATUS.Split(',', StringSplitOptions.RemoveEmptyEntries)
	Returns: {"ONE", "TWO", "THREE"}
	
	If you want to trim the results use the split option "TrimEntries":
	DW_STATUS = ONE ,,TWO ,,THREE ,
	DW_STATUS.Split(',', StringSplitOptions.TrimEntries)
	Returns: {"ONE", "", "TWO", "", "THREE", ""}

Name	Description
	Or use both: DW_ STATUS.Split(',',StringSplitOptions.TrimEntries | StringSplitOptions.RemoveEmptyEntries) Returns: {"ONE", "TWO", "THREE"} Note that you can also split strings using the Split() or the Regex.Split() method.
<MyString>.Substring(Start) Start Int32. Required. The zero-based starting character position of a substring in this instance..	Retrieves a substring from this instance. The substring starts at a specified character position and continues to the end of the string. GV_MyVariable = My*test*string GV_MyVariable.Substring(3) Return: test*string
<MyString>.ToLower()	Returns a copy of this string converted to lowercase.
<MyString>.ToUpper()	Returns a copy of this string converted to uppercase.

3.2 Math Class (System.Math)

Provides constants and static methods for trigonometric, logarithmic, and other common mathematical functions.

A reference to the class functionality can be found on Microsoft's online documentation site: https://docs.microsoft.com/en-us/dotnet/api/system.math

Only parameters of types inherent to the DocuWare infrastructure are supported (like int, decimal, string, keyword (string[]), DateTime).

Here are some examples of the Math Class that can be used in workflow expressions:

Name	Description
Math.Max(Num, Num) Num Double, Integer. Required. A decimal or integer to be compared.	Returns the larger of two specified numbers. DW_AMOUNT1: 100,00 DW_AMOUNT2: 50,00 Math.Max(DW_AMOUNT1,DW_AMOUNT2) Returns: 100,00
Math.Min(Num, Num) Num Double, Integer. Required. A decimal or integer to be compared.	Returns the smaller of two specified numbers. DW_AMOUNT1: 100,00 DW_AMOUNT2: 50,00 Math.Min(DW_AMOUNT1,DW_AMOUNT2) Returns: 50,00

Name	Description
Math.Round(Decimal, Decimals, [MidpointRounding]) Decimal A decimal number to be rounded. Decimals Int32 The number of decimal places in the return value MidpointRounding MidpointRoundingMode. Optional. One of the enumeration values that specifies which rounding strategy to use	Rounds a value to the nearest integer or to the specified number of fractional digits. The MidpointRounding parameter is optional to specify different rounding modes: • MidpointRounding.ToEven (Round to even) • MidpointRounding.AwayFromZero (Round "up") • MidpointRounding.ToZero (Round "down") Examples: GV_MyDecimal: 8,755 Math.Round(GV_MyDecimal, 2, MidpointRounding.AwayFromZero) Returns: 8,76 Math.Round(GV_MyDecimal, 2, MidpointRounding.ToZero) Returns: 8,75 For amounts we recommend using *MidpointRounding.AwayFromZero*.

3.3 Convert Class (System.Convert)

The static methods of the Convert class are primarily used to support conversion to and from the base data types in the .NET Framework.

A reference to the class functionality can be found on Microsoft's online documentation site: https://docs.microsoft.com/en-us/dotnet/api/system.convert

Only parameters of types inherent to the DocuWare infrastructure are supported (like int, decimal, string, keyword (string[]), DateTime).

Currently, the following functions from the Convert Class are supported:

Name	Description
ToBoolean(Value) Value – A value to convert.	Converts a specified value to an equivalent Boolean value.
ToByte(Value) Value – A value to convert.	Converts a specified value to an 8-bit unsigned integer.
ToChar(Value) Value – A value to convert.	Converts a specified value to a Unicode character.
ToDateTime(Value) Value – A value to convert.	Converts a specified value to a DateTime value.

Name	Description
	Should be used with ToUniversalTime, as to compensate for time-zone specifics, DocuWare expressions return and use result in UTC only
ToDecimal(Value) Value – A value to convert.	Converts a specified value to a decimal number.
ToDouble(Value) Value – A value to convert.	Converts a specified value to a double-precision floating-point number.
ToInt16(Value) Value – A value to convert.	Converts a specified value to a 16-bit signed integer.
ToInt32(Value) Value – A value to convert.	Converts a specified value to a 32-bit signed integer.
ToInt64(Value) Value – A value to convert.	Converts a specified value to a 64-bit signed integer.
ToSByte(Value) Value – A value to convert.	Converts a specified value to an 8-bit signed integer.
ToSingle(Value) Value – A value to convert.	Converts a specified value to a single-precision floating-point number.
ToString(Value) Value – A value to convert.	Converts the specified value to its equivalent string representation.
ToUInt16(Value) Value – A value to convert.	Converts a specified value to a 16-bit unsigned integer.
ToUInt32(Value) Value – A value to convert.	Converts a specified value to a 32-bit unsigned integer.
ToUInt64(Value) Value – A value to convert.	Converts a specified value to a 64-bit unsigned integer.

For the methods that have IFormatProvider as a parameter, we can provide the proper CultureInfo as in the following example:

GV_MyDate.ToString(DateSerial(2018,5,17), CultureInfo.CreateSpecificCulture("en-US")) Or

GV_MyDate.ToString(DateSerial(2018,5,17), CultureInfo.InvariantCulture)

3.4 Regex Class (System.Text.RegularExpressions.Regex)

Available starting with DocuWare version 7.9.

The Regex class can be used to quickly parse large amounts of text to find specific character patterns; to extract, edit, replace, or delete text substrings, using regular expression patterns.

A reference to the class functionality can be found on Microsoft's online documentation site: https://learn.microsoft.com/en-us/dotnet/api/system.text.regularexpressions.regex

Currently, the following functions from the Regex Class are supported; the RegexOptions parameter is not supported:

Name	Description
Regex.IsMatch(String, Pattern, IgnoreCase, [Multiline]) String String. Required. String expression to search for a match in it. Pattern String. Required. The regular expression pattern to match. IgnoreCase Bool. Required. true: Specifies case-insensitive matching false: Specifies case-sensitive matching. Multiline Bool. Optional. true: Use multiline mode, where ^ and $ match the beginning and end of each line (instead of the beginning and end of the input string). Default is "false".	Returns Boolean indicating whether the regular expression finds a match in the input string GV_TestValue: "a12-123" Regex.IsMatch(GV_TestValue, "[A-Z]\d{2}-\d{3}", false) Returns: true GV_TestValue: "a12-123" Regex.IsMatch(GV_TestValue, "[A-Z]\d{2}-\d{3}", true) Returns: false GV_TestValue: "Referring to invoice A0123456 here" Regex.IsMatch(GV_TestValue,"[A]\d*", false) Returns: true GV_TestValue: "Regarding project alpha22" Regex.IsMatch(GV_TestValue,"Alpha22", true) Returns: true
Regex.Match(String, Pattern, IgnoreCase, [Multiline],[RightToLeft]) String String. Required. String expression to search a match in. Pattern String. Required. The regular expression pattern to match.	Returns a string or null if no match found. Searches an input string for a substring that matches a regular expression pattern and returns the first occurrence. GV_TestValue: "12-123-1234-12345" Regex.Match(GV_TestValue, "\d{4}", false) Returns: 1234 GV_TestValue: "Referring to invoice A0123456 here" Regex.Match(GV_TestValue,"[A]\d*", false) Returns: A0123456 GV_TestValue: "Regarding project alpha22" Regex.Match(GV_TestValue,"Alpha22", true) Returns: alpha22

Name	Description
IgnoreCase Bool. Required. true: Specifies case-insensitive matching false: Specifies case-sensitive matching. Multiline Bool. Optional. true: Use multiline mode, where ^ and $ match the beginning and end of each line (instead of the beginning and end of the input string). Default is "false" RightToLeft Bool. Optional. true: returns the first match from right to left false: returns the first match left to right. default is "false".	
Regex.Matches(String, Pattern, IgnoreCase, [Multiline]) String String. Required. String expression to search a match in. Pattern String. Required. The regular expression pattern to match. IgnoreCase Bool. Required. true: Specifies case-insensitive matching false: Specifies case-sensitive matching. Multiline Bool. Optional. true: Use multiline mode, where ^ and $ match the beginning and end of each line (instead of the beginning and end of the input string). Default is "false".	Returns an array of strings that match the regular expression pattern in the input string GV_TestValue: "Mahesh Chand, Raj Kumar, Mike Gold" Regex.Matches(GV_TestValue, "\b[M]\w+", false) Returns: {"Mahesh", "Mike"}
Regex.Replace (String, Pattern, Replacement, IgnoreCase, [Multiline]) String String. Required. String expression containing substring to replace.	Returns the specified input string, in which strings that match a regular expression pattern are replaced with a specified replacement string. GV_TestValue: "Something123Anything45" Regex.Replace(GV_TestValue, "\d","", false) Returns: SomethingAnything

Name	Description
Pattern String. Required. The regular expression pattern to match. Replacement String. Required. Replacement substring. IgnoreCase Bool. Required. true: Specifies case-insensitive matching false: Specifies case-sensitive matching. Multiline Bool. Optional. true: Use multiline mode, where ^ and $ match the beginning and end of each line (instead of the beginning and end of the input string). Default is "false".	GV_TestValue: "Something123Anything45" Regex.Replace(GV_TestValue, "[a-zäöüßA-ZÄÖÜ]","", false) Returns: 12345
Regex.Split(String, Pattern, IgnoreCase, [Multiline]) String String. Required. A String to split containing substrings and delimiters. Pattern String. Required. The regular expression pattern to match. IgnoreCase Bool. Required. true: Specifies case-insensitive matching false: Specifies case-sensitive matching. Multiline bool. Optional. true: Use multiline mode, where ^ and $ match the beginning and end of each line (instead of the beginning and end of the input string). Default is "false".	Returns an array of strings. Splits an input string into an array of substrings at the positions defined by a regular expression pattern. GV_TestValue: "07/01/2023-11:45-AM" Regex.Split(GV_TestValue, "\.|V|\-|\:", false) Returns: {"07", "01", "2023", "11", "45", "AM"}

3.5 LINQ Enumerable Class (System.Linq)

Available starting with DocuWare version 7.9.

LINQ (Language Integrated Query) is a powerful feature that allows you to perform various operations on collections/arrays, such as filtering, sorting, grouping, and aggregating data.

A reference to the class functionality can be found on Microsoft's online documentation site: https://learn.microsoft.com/en-us/dotnet/api/system.linq.enumerable

Currently, the following functions from the LINQ Enumerable Class are supported:

Name	Description
<MyArray>.Concat(Array) Array Array. Required. Array with which the elements of the other Array are to be concatenated. <MyArray>.Concat(Value) Value Required. The value to be concatenated to the array.	Concatenates two sequences or a sequence and a single value. Both arrays or value must be of same type. GV_MyNames1: { "John", "Sam" } GV_MyNames2: { "Jane", "Sarah" } GV_MyNames1.Concat(GV_MyNames2) Returns: { "John", "Sam", "Jane", "Sarah" } GV_MyNames1: { "John", "Sam" } GV_MyName: "Sarah" GV_MyNames1.Concat(GV_MyName) Returns: { "John", "Sam", "Sarah" }
<MyArray>.Contains(Value) Value Required. The value(s) to locate in the sequence. Contains method is case insensitive.	Determines whether a sequence contains a specified element. The comparison is case insensitive. GV_MyNames: { "Sandy", "Thomas" } GV_MyNames.Contains("Thomas") Returns: true If multiple values are specified, all values must be present for true to be returned. GV_MyNames1: { "Sandy", "Thomas", "Andy" } GV_MyNames2: { "Sandy", "Andy" } GV_MyNames1.Contains(GV_MyNames2) Returns: true
<MyArray>.Distinct() Distinct method is case sensitive.	Returns distinct elements from a sequence. GV_MyNames: { "Sandy", "Thomas", "Sandy" } GV_MyNames.Distinct() Returns: { "Sandy", "Thomas" }
<MyArray>.Except(Array) Array Array. Required. Array with which the elements of the other Array are to be compared. Except method is case sensitive.	Produces the set difference of two sequences or a sequence and a single value by **case-sensitive** matching of values. Both arrays or value must be of same type. GV_MyNames1: { "Sandy", "Thomas", "Andy" } GV_MyNames2: { "Sandy", "Andy" } GV_MyNames1.Except(GV_MyNames2) Returns: { "Thomas" }

Name	Description
<MyArray>.Except(Value) Value Required. The value to locate in the sequence. Except method is case sensitive.	GV_MyNames1: { "Sandy", "Thomas", "Andy" } GV_MyName = "Andy" GV_MyNames1.Except(GV_MyName) Returns: { "Sandy", "Thomas" }
<MyArray>.Intersect(Array) Array Array. Required. Array with which the elements of the other Array are to be compared. Intersect method is case sensitive.	Produces the set intersection of two sequences by **case-sensitive** matching of values Both arrays must be of same type. GV_MyNames1: { "Sandy", "Thomas", "Andy" } GV_MyNames2: { "Sandy", "Andy" } GV_MyNames1.Intersect(GV_MyNames2) Returns: { "Sandy", "Andy" }
<MyArray>.Order()	Sorts the elements of a sequence in ascending order. GV_MyNumbers: { 2, 3, 1, 4, 5 } GV_MyNumbers.Order() Returns: { 1, 2, 3, 4, 5 } GV_MyNames: { "Sandy", "Thomas", "Andy" } GV_MyNames.Order() Returns: { "Andy", "Sandy", "Thomas" }
<MyArray>.OrderDescending()	Sorts the elements of a sequence in descending order. GV_MyNumbers: { 2, 3, 1, 4, 5 } GV_MyNumbers.OrderDescending () Returns: { 5, 4, 3, 2, 1 } GV_MyNames: { "Sandy", "Thomas", "Andy" } GV_MyNames.OrderDescending() Returns: { "Thomas", "Sandy", "Andy" }
<MyArray>.Union(Array) Array Array. Required. Array with which the elements of the other Array are to be compared. Union method is case sensitive. <MyArray>.Union(Value) Value Required. The value with which the elements of the other Array are to be compared. Union method is case sensitive.	Produces the set union of two sequences or a sequence and a single value by **case-sensitive** matching of values. Both arrays or value must be of same type. GV_MyNames1: { "Sandy", "Thomas", "Andy" } GV_MyNames2: { "Sandy", "Andy", "John", "Sam" } GV_MyNames1.Union(GV_MyNames2) Returns: { "Sandy", "Thomas", "Andy", "John", "Sam" } GV_MyNames1: { "Sandy", "Thomas", "Andy" } GV_MyName: "Andy" GV_My Names1.Union(GV_MyName) Returns: { "Sandy", "Thomas", "Andy" }
<MyArray>.AllStartsWith(String) Array Array. Required. One-dimensional array of strings.	Checks if all elements in the string array start with the specified string. Returns **Boolean** Example: GV_List: {"abc", "abd", "abe"} GV_List.AllStartsWith("ab")

Name	Description
String String. Required.	Returns: true
<MyArray>.AllEndsWith(String) Array Array. Required. One-dimensional array of strings. String String. Required.	Checks if all elements in the string array end with the specified string. Returns **Boolean** Example: GV_List: {"cat", "bat", "hat"} GV_List.AllEndsWith("at") Returns: true
<MyArray>.AllContainsText(String) Array Array. Required. One-dimensional array of strings. String String. Required.	Checks if all elements in the string array contain the specified text. Returns **Boolean** Example: GV_List: {"hello", "help", "helm"} GV_List.AllContainsText("he") Returns: true
<MyArray>.AnyStartsWith(String) Array Array. Required. One-dimensional array of strings. String String. Required.	Checks if any element in the string array starts with the specified string. Returns **Boolean** Example: GV_List: {"hello", "world", "test"} GV_List.AnyStartsWith("wo") Returns: true
<MyArray>.AnyEndsWith(String) Array Array. Required. One-dimensional array of strings. String String. Required.	Checks if any element in the string array ends with the specified string. Returns **Boolean** Example: GV_List: {"apple", "banana", "orange"} GV_List.AnyEndsWith("e") Returns: true
<MyArray>.AnyContainsText(String) Array Array. Required. One-dimensional array of strings. String String. Required.	Checks if any element in the string array contains the specified text. Returns **Boolean** Example: GV_List: {"red", "blue", "green"} GV_List.AnyContainsText("lu") Returns: true
<MyArray>.DefaultIfEmpty(String) Array Array. Required. One-dimensional array of strings. String String. Optional.	Returns the original array if not empty, otherwise returns an array with the default value. Returns **Array** or **String** Example: GV_List: {} GV_List.DefaultIfEmpty("default") Returns: default
<MyArray>.FirstOrDefault(Value) Array	Returns the first element of the array or a default value if the array is empty.

Name	Description
Array. Required. One-dimensional array of strings. Value Default value. Optional.	Example: GV_List: {"x", "y", "z"} GV_List.FirstOrDefault("default") Returns: x
<MyArray>.LastOrDefault(Value) Array Array. Required. One-dimensional array of strings. Value Default value. Optional.	Returns the last element of the array or a default value if the array is empty. Example: GV_List: {"a", "b", "c"} GV_List.LastOrDefault("default") Returns: c
<MyArray>.Distinct() Array Array. Required. One-dimensional array.	Returns an array with unique elements from the input array. Returns **Array** Example: GV_List: {"a", "b", "a", "c", "b"} GV_List.Distinct() Returns: {"a", "b", "c"}

4 Data type "Date”

This data type was introduced to provide consistent support for DocuWare **Date** variables and index fields. It holds information for just date information (Year, Mont and Day) without any time component. That means the functions are attributes of the date variable. Examples: GV_MyDate.AddYears(5) or GV_MyDate.IsLeapYear

Date variables can be compared to other Date and DateTime variables with basic comparison operators (>, >=, <, <=, ==, !=). When compared to DateTime variables, only the date portion is taken into consideration. Note that DateTime index fields are internally stored on the server in their UTC equivalent, which may be misleading compared to the value you see in the DocuWare client where the value is converted to the local browser time. This is important when comparing Date and DateTime fields.

Here are the properties and methods of the Date data type that can be used in workflow expressions:

Name	Description
<MyDate>.Year	Gets the year component of the date represented by this instance. The Year property returns **Integer** value. The year, between 1 and 9999.
<MyDate>.Month	Gets the month component of the date represented by this instance. The Month property returns **Integer** value. The month component, expressed as a value between 1 and 12.
< MyDate>.Day	Gets the day component of the date represented by this instance. The Day property returns **Integer** value. The day component, expressed as a value between 1 and 31.
<MyDate>.DayOfYear	Gets the day of the year represented by this instance. The DayOfYear property returns **Integer** value. The day of the year, expressed as a value between 1 and 366
<MyDate>.DayOfWeek	Gets the day of the week represented by this instance. The DayOfWeek property returns an enumerated constant. The value of the constants in the DayOfWeek enumeration ranges from DayOfWeek.Sunday to DayOfWeek.Saturday. If cast to an integer, its value ranges from 0, which indicates DayOfWeek.Sunday to 6, which indicates DayOfWeek.Saturday. Example: Cast to string to return the localized name of the day of the week CStr(GV_MyDate.DayOfWeek) Cast to Integer to return the number of the day of the week CInt(GV_MyDate.DayOfWeek)
<MyDate>.IsLeapYear	Returns **Boolean** true if the year is leap year; otherwise, false.
<MyDate>.AddYears(Number)	Gets a **Date** object whose value is ahead or behind the value of this instance by the specified number of years.

Name	Description

Number
Integer. Required.
Number of years which is used for calculation. The value can be negative or positive
	Positive values will move the date forward; negative values will move the date backwards.
	The value parameter can be negative or positive.
	Returns new Date Object whose value is the result of this operation. Examples:
	GV_MyDate.AddYears(5) //Date
	GV_MyDate.AddYears(5).ToString //String
	CStr(GV_MyDate.AddYears(5)) //String
	
	With a specified culture
	GV_MyDate.AddYears(5).ToString("ddd d MMM ", CultureInfo.CreateSpecificCulture("en-EN"))

<MyDate>.AddMonths(Number) Number Integer. Required. Number of months which is used for calculation. The value can be negative or positive.	Gets a Date object whose value is ahead or behind the value of this instance by the specified number of months. Positive values will move the date forward; negative values will move the date backwards. Returns new **Date** object whose value is the result of this operation.
<MyDate>.AddDays(Number) Number Integer. Required. Number of days which is used for calculation. The value can be negative or positive.	Gets a **Date** object whose value is ahead or behind the value of this instance by the specified number of days. Positive values will move the date forward; negative values will move the date backwards. Returns new Date object whose value is the result of this operation.

<MyDate>.DaysUntil(Date)
 	
Date
Date. Required.
Date for which the number of days remaining is calculated.
	Returns the number of days remaining from one date to another date specified. If the date has already passed, the result will be negative. Returns **Integer** value. Example:
	GV_MyDate.DaysUntil(GV_MyDate2) //Integer
	GV_MyDate.DaysUntil(GV_MyDate2).ToString //String
	CStr(GV_MyDate.DaysUntil(Date)) //String

<MyDate>.DaysSince(Date)
 	
Date
Date. Required.
Date for which the number of days elapsed is calculated
	Returns the number of days elapsed from the date specified to this date. If the date has not yet passed, the result will be negative. Returns **Integer** value. Example:
	GV_MyDate.DaysSince(GV_MyDate2) //Integer
	GV_MyDate.DaysSince(GV_MyDate2).ToString //String
	CStr(GV_MyDate.DaysSince(GV_MyDate2)) //String

<MyDate>.MonthsUntil(Date)
 	
Date
Date. Required.
Date for which number of whole months remaining is calculated
	Returns the number of whole months remaining from this date to the date specified. If the date has already passed, the result will be negative. Returns **Integer** value.
	Example:
	GV_MyDate.MonthsUntil(GV_MyDate2) //Integer
	GV_MyDate.MonthsUntil(GV_MyDate2).ToString //String
	CStr(GV_MyDate.MonthsUntil(GV_MyDate2)) //String

<MyDate>.MonthsSince (Date)
 	
Date
Date. Required
Date for which the number of whole months elapsed is calculated
	Returns the number of whole months elapsed from the date specified to this date. If the date has not yet passed, the result will be negative. Returns **Integer** value.
	Example:
	GV_MyDate.MonthsSince(GV_MyDate2) //Integer
	GV_MyDate.MonthsSince(GV_MyDate2).ToString //String
	CStr(GV_MyDate.MonthsSince(GV_MyDate2)) //String

Name	Description
<MyDate>.YearsUntil (Date) Date Date. Required Date for which the number of whole years remaining is calculated	Returns the number of whole years remaining from this date to the date specified. If the date has already passed, the result will be negative. Returns **Integer** value. Example: GV_MyDate.YearsUntil(Date).ToString CStr(Date.YearsUntil(GV_MyDate2))
<MyDate>.YearsSince (Date) Date Date. Required Date for which the number of whole years elapsed is calculated	Returns the number of whole years elapsed from the date specified to this date. If the date has not yet passed, the result will be negative. Returns **Integer** value. Example: GV_MyDate.YearsSince(GV_MyDate2).ToString CStr(GV_MyDate.YearsSince(GV_MyDate2))
<MyDate>.ToString	Converts the value of the provided Date object to its equivalent string representation.
<MyDate>.ToString (format)	Converts the value of the provided Date object to its equivalent string representation using the specified format. Example: GV_MyDate.ToString("d") Returns: 2/28/2013 GV_MyDate.ToString("MM/dd/yyyy") Returns: 02/28/2013 GV_MyDate.ToString("dddd, dd MMMM yyyy") Returns: Thursday, 28 February 2013 GV_MyDate.ToString("MMMM dd") Returns: February 28 GV_MyDate.ToString("MMMM dd", CultureInfo.CreateSpecificCulture("de-DE")) Returns: February 28 GV_MyDate.ToString("yyyy MMMM") Returns: 2013 February GV_MyDate.ToString("yyyy-MM-dd") Returns: 2008-02-28
<MyDate>.ToLongDateString([Culture])	Converts the value of the provided Date object to its equivalent long date string representation. Returns **String** that contains the long date string representation of the current Date object. If **Culture** is provided, the output is formatted according to the specified Culture.
<MyDate>.ToLongDateStringInvariant	Converts the value of the provided Date object to its equivalent long date string representation. Returns **String** that contains the long date string representation of the current Date object. The results from formatting the current instance by using the conventions of the invariant Culture.
<MyDate>.ToShortDateString([Culture])	Converts the value of the provided Date object to its equivalent short date string representation.

Name	Description
	Returns **String** that contains the short date string representation of the current Date object. If **Culture** is provided, the output is formatted according to the specified Culture.
<MyDate>.ToShortDateStringInvariant	Converts the value of the provided Date object to its equivalent short date string representation. Returns **String** that contains the short date string representation of the current Date object. The results from formatting the current instance by using the conventions of the invariant Culture.
<MyDate>.ToIsoString	Converts the value of the provided Date object to its equivalent ISO standard string representation (ISO-8601), which has the format: yyyy-MM-dd. Returns **String**.
<MyDate>.ToDateTimeAtMidnight([TimeZone])	Creates a UTC DateTime object from the provided Date, with the time set to midnight. Returns **DateTime** object. If TimeZone is provided the result is corrected/shifted with the specified timezone, so that the result appears at midnight for the specified time-zone Examples: GV_MyDate.ToDateTimeAtMidnight.ToString Returns: 30.1.2019 0:00:00 //String CStr(GV_MyDate.ToDateTimeAtMidnight) Returns: 30.01.2019 //String GV_MyDate.ToDateTimeAtMidnight Returns: 30.01.2019 02:00 //DateTime GV_MyDate.ToDateTimeAtMidnight Returns: 30.01.2019 //Date

5 Custom functions

5.1 Conversion

Name	Description
CBool(Input)	Converts input to bool Text or Numeric values of 1 and -1 will return true. 0, "0" or null will return false. Valid input are also literals "true" and "false". Any other input will produce error.
CByte(input)	Converts input to byte.
CChar(Input)	Converts input to char.
CDate(Input, [Culture], [TimeZone])	Converts input to DateTime. If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture. If **TimeZone** is provided the result is corrected/shifted with the specified time-zone.
CDbl(Input, [Culture])	Converts input to double. If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture.
CDec(Input, [Culture])	Converts input to decimal. If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture.
CInt(Input, [Culture])	Converts input to int. If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture.
CLng(Input, [Culture])	Converts input to long. If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture.
CSByte(Input)	Converts input to signed byte.
CShort(Input, [Culture])	Converts input to short. If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture.
CSng(Input, [Culture])	Converts input to single. If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture.
CStr(Input, [Culture], [TimeZone])	Converts input to string. When the input DateTime is of type Unspecified or the time portion is midnight, then only the Date portion is returned. If **Culture** is provided the output is formatted according to the specified Culture. If **TimeZone** is provided the result is corrected/shifted with the specified time-zone.
CUInt(Input, [Culture])	Converts input to unsigned int. If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture.
CULng(Input, [Culture])	Converts input to unsigned long. If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture.
CUShort(Input, [Culture])	Converts input to unsigned short. If **Culture** is provided, and the input is string – it is interpreted according to the specified Culture.
ConvertFromBase64String(String)	Decodes the specified base-64 encoded string to its equivalent binary data as a string..

String String. Required. A value to convert from BASE64.	Example: ConvertFromBase64String("W0NPTVBBTlldID0gIkludGVybmF0aW9uYWwgU3RlZWwgQlYiIEFORCBbRE9DVU1FTlRfVFlQRV0gPSAiRGVsaXZlcnkgTm90ZSBJbiI=") Returns: [COMPANY] = "International Steel BV" AND [DOCUMENT_TYPE] = "Delivery Note In"
ConvertToBase64String(String) String String. Required. A value to convert from BASE64. Note: Double quotes (") inside a string must be escaped by doubling them ("").	Encodes the specified binary data string to its equivalent base-64 format. Example: ConvertToBase64String("[COMPANY] = ""International Steel BV"" AND [DOCUMENT_TYPE] = ""Delivery Note In""") Returns: W0NPTVBBTlldID0gIkludGVybmF0aW9uYWwgU3RlZWwgQlYiIEFORCBbRE9DVU1FTlRfVFlQRV0gPSAiRGVsaXZlcnkgTm90ZSBJbiI=
ConvertFromURLString(String) String String. Required. A URL-encoded string to decode..	Converts a URL-encoded string back to its original form, decoding any special characters Example: ConvertFromURLString("Hello%20World%21") Returns: "Hello World!"
ConvertToURLString(String) String String. Required. A URL-encoded string to encode.	Encodes a string into a URL-safe format, replacing special characters with their URL-encoded equivalents Example: ConvertToURLString("Hello World!") Returns: "Hello%20World%21"

5.2 Concatenation

Name	Description
KeywordAsString(Array, [Delimiter]) Array Array. Required. One-dimensional array of strings. Delimiter String. Optional Any string, used to separate the substrings in the returned string. If omitted, the space character (" ") is used. If Delimiter is a zero-length string ("") or Nothing, all items in the list are concatenated with no delimiters.	Concatenates a string array with the specified delimiter. Example: GV_MyKeyword = {1,2,3,4,5,6,7,8,9} KeywordAsString(GV_MyKeyword, ",") Returns: 1,2,3,4,5,6,7,8,9

5.3 Aggregation over index table columns

To support some aggregate functionality over index table columns, we have provided the following functions:

Name	Description
Sum(TableFieldColumn)	Calculates the sum of all rows for the specified decimal index table column.
Count(TableFieldColumn)	Calculates the count of all rows for the specified Numeric, DateTime or Text index table column.
Avg(TableFieldColumn)	Calculates the average value of all rows for the specified decimal index table column.

5.4 LIKE comparison

The LIKE comparison is supported. For this you can use the wildcards "%" or "*".
Both wildcards are supported and can be used individually or in combination.

Examples:

To compare if the index field COMPANY contains "Test Organization Inc." you can use either the wildcard "%" or "*" at different places:

```
DW_COMPANY LIKE "Test%"  Or
DW_COMPANY LIKE "%Organization*" Or
DWCOMPANY LIKE "*Inc."
```

To compare with a global variable, first assign the value you want to compare to the global variable in a separated Assign Data activity, for example "Test":

```
GV_MyCompany = "Test"
```

Then compare using LIKE and adding a wildcard after it.
The character "&" is needed to concatenate the global variable with the wildcard:

```
DW_COMPANY LIKE GV_MyCompany&"%" Or
DW_COMPANY LIKE GV_MyCompany&"*"
```

5.5 Fuzzy string comparison

The fuzzy string comparison functions allow you to compare two strings by measuring the distance ("inverse similarity") between the text strings for an approximate string match or comparison.

Name	Description
Similarity.EditDistance(String1,String2) String1 String. Required. The first value to be compared. String2 String. Required. The second value to be compared.	Evaluates the similarity of two strings by calculating the minimum number of single-character operations (insertions, deletions, or substitutions) required to transform one string into the other. The smaller the edit distance, the more similar the strings are. Casing is ignored. Returns **Double** value. The resulting value falls between 0 and 1 (Double), where a higher score indicates a stronger similarity between the strings. If one of the strings is null or empty, the result will be 0. Suggested "acceptable" similarity is achieved with an output above 0.85-0.90. Example: Similarity.EditDistance("Peter Peterson", "Peter Pitersen") Returns: 0.92 The similarity score between "Peter Peterson" and "Peter Pitersen" is around "0.92".
Similarity.Cosine(String1, String2, NGram) String1 String. Required. The first value to be compared. String2 String. Required. The second value to be compared. NGram Int32. Optional. Must be a positive value. Default value is 2. The value by which the strings are split.	Evaluates the similarity of two strings by breaking them into NGram values and comparing each corresponding token. Casing is ignored. Returns **Double** value. The resulting value falls between 0 and 1 (Double), where a higher score indicates a stronger similarity between the strings. If one of the strings is null or empty, the result will be 0. Suggested "acceptable" similarity is achieved with an output above 0.85-0.90. Example: Similarity.Cosine("Peter Peterson", "Peter Pitersen", 2) Returns: 0.93 When using an NGram of 2, the similarity score between "Peter Peterson" and "Peter Piterson" is very high at around 0.93. However, when the NGram is increased to 5, the score decreases to approximately 0.64. As the NGram size increases, we can anticipate that the two strings will share a longer subsequence. When the NGram is set to 1, the behavior of the algorithm is like the "EditDistance" but not in all cases. If, in your case, the two algorithms produce similar or the same results, consider using the "EditDistance" method as it is more optimized.

You can use these functions in combination with other functions like CBool() to return a boolean value (true|false) and determine if the string matches or not:

Examples:

Similarity.EditDistance("Peter Peterson", "Peter Pitersen")
Returns: 0.92

CBool(Similarity.EditDistance("Peter Peterson", "Peter Pitersen") > 0.85)
Returns: true

The similarity score between "Peter Peterson" and "Peter Pitersen" is around "0.92".
In the CBool function, it is specified that the result must be over 0.85 to be true.

Of course, you can also use global variables of type "text" for the comparison functions:
CBool(Similarity.EditDistance("GV_String1", "GV_String2") > 0.85)

5.6 Array functions

Name	Description
AllStartsWith(Array, String) Array Array. Required. One-dimensional array of strings. String String. Required.	Checks if all elements in the string array start with the specified string. Returns **Boolean** Example: AllStartsWith({"abc", "abd", "abe"}, "ab") Returns: true
AllEndsWith(Array, String) Array Array. Required. One-dimensional array of strings. String String. Required.	Checks if all elements in the string array end with the specified string. Returns **Boolean** Example: AllEndsWith({"cat", "bat", "hat"}, "at") Returns: true
AllContainsText(Array, String) Array Array. Required. One-dimensional array of strings. String String. Required.	Checks if all elements in the string array contain the specified text. Returns **Boolean** Example: AllContainsText({"hello", "help", "helm"}, "he") Returns: true
AnyStartsWith(Array, String) Array Array. Required. One-dimensional array of strings. String String. Required.	Checks if any element in the string array starts with the specified string. Returns **Boolean** Example: AnyStartsWith({"hello", "world", "test"}, "wo") Returns: true

Name	Description
AnyEndsWith(Array, String) Array Array. Required. One-dimensional array of strings. String String. Required.	Checks if any element in the string array ends with the specified string. Returns **Boolean** Example: AnyEndsWith({"apple", "banana", "orange"}, "e") Returns: true
AnyContainsText(Array, String) Array Array. Required. One-dimensional array of strings. String String. Required.	Checks if any element in the string array contains the specified text. Returns **Boolean** Example: AnyContainsText({"red", "blue", "green"}, "lu") Returns: true
DefaultIfEmpty(Array, String) Array Array. Required. One-dimensional array of strings. String String. Optional.	Returns the original array if not empty, otherwise returns an array with the default value. Returns **Array** or **String** Example: DefaultIfEmpty({}, "default") Returns: default
FirstOrDefault(Array, Value) Array Array. Required. One-dimensional array of strings. Value Default value. Optional.	Returns the first element of the array or a default value if the array is empty. Returns **String** Example: FirstOrDefault({"x", "y", "z"}, "default") Returns: x
LastOrDefault(Array, Value) Array Array. Required. One-dimensional array of strings. Value Default value. Optional.	Returns the last element of the array or a default value if the array is empty. Example: LastOrDefault({"a", "b", "c"}, "default") Returns: c
Distinct(Array) Array Array. Required. One-dimensional array.	Returns an array with unique elements from the input array. Returns **Array** Distinct({"a", "b", "a", "c", "b"}) Returns: {"a", "b", "c"}

6 Check for NULL, Nothing and Empty

6.1 Check for empty variables

In comparison operations, you can examine variables or index fields for ***NULL*** or ***NOTHING***.

Name	Description
<Input> = null Input String or Nullable. Required.	Checks if input is null. Returns **Boolean** GV_MyVariable: GV_MyVariable = null Returns: true
IsNull(Input) Input String or Nullable. Required.	Checks if input is null. Returns **Boolean** GV_MyVariable: IsNull(GV_MyVariable) Returns: true
<Input> = nothing Input String or Nullable. Required.	Checks if input is nothing (null). Returns **Boolean** GV_MyVariable: GV_MyVariable = nothing Returns: true
IsNothing(Input) Input String or Nullable. Required.	Checks if input is nothing (null). Returns **Boolean** GV_MyVariable: IsNothing(GV_MyVariable) Returns: true

Use C# methods to check if a variable or field of type String is ***NULL*** or ***EMPTY***.

Name	Description
String.IsNullOrEmpty(String) String String. Required. The string to test.	Indicates whether the specified string is null or an empty string (""). Returns **Boolean** GV_MyContact: Peter String.IsNullOrEmpty(GV_MyContact) Returns: false
String.IsNullOrWhiteSpace(String) String String. Required. The string to test.	Indicates whether a specified string is null, an empty string ("") or consists only of white-space characters (" "). Returns **Boolean** GV_MyContact: Peter String.IsNullOrWhiteSpace(GV_MyContact) Returns: false

To check whether a variable or field of type Decimal, DateTime or Numeric has a value, use can also use ***HasValue***. For variables or fields of type Date, however, use ***!= null***.

Name	Description
<Nullable>.HasValue Nullable<T> Represents a value type that can be assigned null	Gets a value indicating whether the current Nullable<T> object has a valid value of its underlying type. Returns **Boolean** GV_MyAmount: GV_MyAmount.HasValue Returns: false

6.2 Check for empty list variables

To check whether a list variable, an index field of type keyword or an index table column is null or has no elements (length is 0), use ***IsNullOrEmptyList()***.

Name	Description
IsNullOrEmptyList(Array) Array Array. Required. One-dimensional array of strings.	Safely checks a list variable (keyword etc.) for being null (not initialized) or not having any elements (length is 0). Returns **Boolean** GV_MyArray: {} IsNullOrEmptyList(GV_MyArray) Returns: true GV_MyArray: {"",""} IsNullOrEmptyList(GV_MyArray) Returns: false GV_MyArray: {"1","2","3"} IsNullOrEmptyList(GV_MyArray) Returns: false

Be careful with index tables, because a column is not considered null or empty if there are rows. Example:

	ID	Name	Description
1	A1005	Panel	
2	A2541	Girder	

In this index table there are 2 rows, each with no value in the "Description" column. However, checking the "Description" column with IsNullOrEmptyList() returns **false**. This is because the returned array is not empty and does not have zero length, due to the 2 rows.

If you want to check if a table column contains no value, use String.IsNullOrEmpty() or String.IsNullOrWhiteSpace() in combination with KeywordAsString() to merge all values of the column to a single string:

String.IsNullOrWhiteSpace(KeyWordAsString(Array))

7 Localization

Some functions interpret or generate data that might depend on a specific locale. For example, if we have a decimal variable GV_Money = 16325.25 and use it as a parameter to the function **FormatCurrency(GV_Money)**, depending on the locale, we can receive a different output: e.g. 16.325,25 € , 16 325,25 лв. , £16,325.25 etc. To be able to specify the locale for the data we want, we have introduced the ***locale*** parameter for a number of functions. This parameter is optional. If **DWCulture** parameter is not provided, the behavior may vary: for assignments in general/parallel tasks the Culture of the browser is used; for system activities the server Culture is used.

To make providing this parameter easier, we have introduced the ***DWCulture*** enumeration, which has all supported by DocuWare locales/languages predefined:

AR, BG, HR, NL, ENGB, ENUS, FR, DA, DE, DECH, EL, IT, NB, PL, PT, RU, ES, SV, ZHHANS, ZHHANT, JA

So, if we want to invoke the function from the above example and get the result formatted for German Culture, we put it like this:

```
FormatCurrency(GV_Money, -1, TriState.UseDefault, TriState.UseDefault,
TriState.UseDefault, DWCulture.DE),
```

which will produce the value of *16.325,25 €*

Alternatively, we could use the .NET way to create a desired locale parameter - `CultureInfo.GetCultureInfo("DE")`.

Finally, we support for a special locale parameter – **DWCulture.ORG**, which is dynamically replaced at runtime with the company locale of the customer.

8 Time zone specifics

As time calculations of expressions are executed on the server, all dates are automatically converted to UTC. Additionally, when DateTime data is generated on the server, like when parsing string as date or getting current time, this data is supplied in UTC. This means that in some scenarios it is necessary to convert the DateTime to a different time zone.

8.1 Time zones

As stated, if you want to adjust the time by the time zone shift, we provide the optional *TimeZone* parameter for relevant functions. The type of this parameter is *TimeSpan* and we supply it using the standard .NET class:

TimeSpan.FromHours(-4)
Denotes the time zone which is 4 hours before UTC (UTC-4).

8.2 Time zone of a DocuWare organization

8.2.1 Parameter DWTimeZone.ORG

We supply ***DWTimeZone.ORG*** parameter of type *TimeSpan*, which is calculated at runtime and corresponds to the customer's organization time zone, set in the organization settings. However, daylight saving time is **not** supported.

Example:

Organization's time zone: UTC+1
Current time: 01 June 2023 08:41:06 (UTC)

In this example to retrieve the current time we use *TimeString()*:

TimeString(DWTimeZone.ORG)
Returns: 09:41

Problem: It is June 1, which means it is daylight saving period. So, the desired time zone is UTC+2. But the result is still UTC+1 because daylight saving time is not supported.

8.2.2 Parameter DWTimeZoneInfo.ORG

Available starting with DocuWare version 7.8.

We supply ***DWTimeZoneInfo.ORG*** parameter of type *TimeZoneInfo, which is calculated at runtime and provides the time zone information set in the organization's settings.*

Example:

- DWTimeZoneInfo.ORG.GetUtcOffset
  Returns a TimeSpan object that indicates the time difference between two time zones.
- DWTimeZoneInfo.ORG.IsDaylightSavingTime
  Returns true if the dateTime parameter is a daylight saving time; otherwise, false.

8.2.3 Function ToOrgDateTime(DWTimeZoneInfo.ORG)

We have introduced a custom function ***ToOrgDateTime(DWTimeZoneInfo.ORG)*** to shift the provided DateTime to the customer's organization time zone, supporting the daylight-saving time. This is the recommended way for such conversions.

<MyDateTime>.ToOrgDateTime(DWTimeZoneInfo.ORG)

Example:

Organization's time zone: UTC+1
Current time: 01 June 2023 08:41:06 (UTC)

In this example to retrieve the current date and time we use *Now*:

Now.ToOrgDateTime(DWTimeZoneInfo.ORG).ToString
Returns: 6/01/2023 10:41:06 AM

More examples how you can use this function:

You can change the format by specifying a custom format:

Now.ToOrgDateTime(DWTimeZoneInfo.ORG).ToString("MMMM-dd-yyyy h:mm tt")
Returns: June-01-2023 10:41 AM

When inserting custom text, ensure that format specifiers are escaped:

Now.ToOrgDateTime(DWTimeZoneInfo.ORG).ToString("T\he MM/dd/yyy a\t h:mm tt")
Returns: The 06/01/2023 at 10:41 AM

Using the *FormatDate()* function instead of *ToString*, to return the date in the format of a specific culture or the organization's culture (e.g., German):

FormatDateTime(Now.ToOrgDateTime(DWTimeZoneInfo.ORG), 0, DWCulture.ENGB)
Returns: 01/06/2023 10:41:06

FormatDateTime(Now.ToOrgDateTime(DWTimeZoneInfo.ORG), 0, DWCulture.ORG)
Returns: 01.06.2023 10:41:06

Also, you can combine it with other functions like *DateAdd()*, for example to add 2 days:

DateAdd(DateInterval.Day, 2, Now).ToOrgDateTime(DWTimeZoneInfo.ORG).ToString
Returns: 6/03/2023 10:41:06 AM

Finally, an example how to use *ToOrgDateTime(DWTimeZoneInfo.ORG)*, *DateAdd()* and *FormatDate()* combined:

FormatDateTime(DateAdd(DateInterval.Day, 2,
Now).ToOrgDateTime(DWTimeZoneInfo.ORG), 0, DWCulture.ORG)
Returns: 03.06.2023 10:41:06

9 More examples

9.1 Check if an index field is not equal to a certain value

In a condition, the index field "Company" should be checked if it is not equal to a certain value:

DW_COMPANY: DocuWare

DW_COMPANY != DocuWare
Returns: false

Or

DW_COMPANY <> DocuWare
Returns: false

9.2 Check if an index field does not contain a certain value

In a condition, the index field "Company" should be checked if it does not contain a certain value:

DW_COMPANY: DocuWare

!(DW_COMPANY like "*F*")
Returns: True

9.3 Return a certain value based on a condition using iif()

Return a value, depending on the evaluation of an expression.
Note that the comparison of strings is case sensitive. Example:

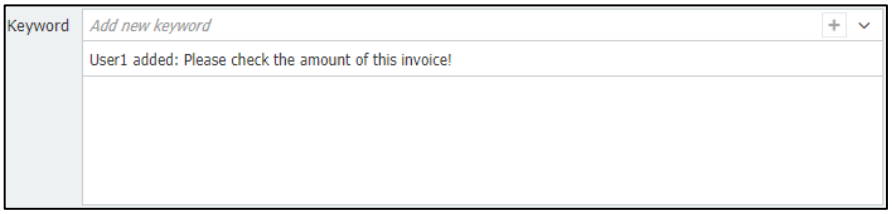
DW_COMPANY: "DocuWare"
iif(DW_COMPANY = "DocuWare", "Correct", "Wrong")
Returns: Correct

DW_COMPANY: "DOCUWARE"
iif(DW_COMPANY = "DocuWare", "Correct", "Wrong")
Returns: Wrong

Use LCase or UCase before the variable to avoid the case sensitive comparison:
iif(UCase(DW_COMPANY) = "DOCUWARE", "Correct", "Wrong")
Returns: Correct

9.4 Fill keyword field with a custom comment

Write a comment the user provided in a workflow task into a keyword field and add more information to it:



Comment by user User1: "Please check the amount of this invoice!"

"" + WF_ASSIGNED_TO + " added: " + Check_Amount_Comment + ""

Returns: User1 added: Please check the amount of this invoice!

9.5 Return the sum of an index table column

Return the sum of all rows of the index table column "Amount":

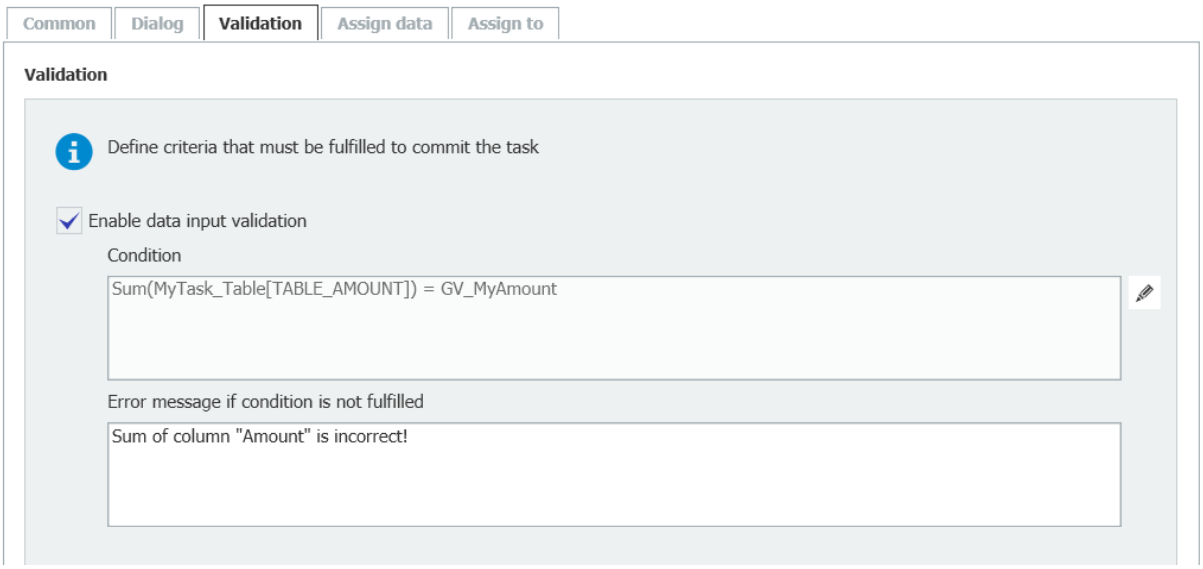
TABLE_1:

Name	Date	Amount
Value1	05.07.2022	5000.00
Value2	10.07.2022	2000.00
Value3	02.08.2022	1000.00
DocuWare	15.08.2022	3000.00

Sum(DW_Table_1[TABLE_AMOUNT])
Returns: 11000.00

9.6 Validate the sum of an index table column

Check if for example the sum of the column "Amount" is equal to the variable "MyAmount", before confirming the workflow task.



GV_MyAmount: 11000.00
MyTask_Table:

Name	Date	Amount
Value1	05.07.2022	5000.00
Value2	10.07.2022	2000.00
Value3	02.08.2022	1000.00
DocuWare	15.08.2022	3000.00

Make sure that the table to be checked is the **workflow form field** and not the index table. If the form field is used, the check will be performed including the entered data. Otherwise, only the values that were previously stored in the index table are checked!

Sum(MyTask_Table[TABLE_AMOUNT]) = GV_MyAmount
Returns: true

9.7 Validate the content of an index table column

Check if column "Name" contains a fixed value "DocuWare".

Table:

Name	Date	Amount
Value1	05.07.2022	5000.00
Value2	10.07.2022	2000.00
Value3	02.08.2022	1000.00
DocuWare	15.08.2022	3000.00

DW_TABLE[TABLE_NAME].Contains("DocuWare")
Returns: true

Check if column "Name" contains any empty (null) value.

GV_NULL: NULL
Table:

Name	Date	Amount
Value1	05.07.2022	5000.00
NULL	10.07.2022	2000.00
Value3	02.08.2022	1000.00

Use the Contains() function in combination with an empty global variable, e.g. GV_NULL

DW_TABLE[TABLE_NAME].Contains(GV_NULL)
Returns: true

Check if column "Name" contains a value stored in index field COMPANY.

COMPANY: DocuWare
Table:

Name	Date	Amount
Value1	05.07.2022	5000.00
Value2	10.07.2022	2000.00
Value3	02.08.2022	1000.00
DocuWare	15.08.2022	3000.00

DW_TABLE[TABLE_NAME].Contains(DW_COMPANY)
Returns: true

Check if column "Name" contains a value stored in index field COMPANY, before confirming the workflow task.

Make sure that the table to be checked is the **workflow form field** and not the index table. If the form field is used, the check will be performed including the entered data. Otherwise, only the values that were previously stored in the index table are checked!

COMPANY: DocuWare
MyTask_Table:

Name	Date	Amount
Value1	05.07.2022	5000.00
Value2	10.07.2022	2000.00
Value3	02.08.2022	1000.00
DocuWare	15.08.2022	3000.00

MyTask_Table[TABLE_NAME].Contains(DW_COMPANY)
Returns: true

9.8 Add certain time to a variable

DateTime variables can be set to a certain time.

Example:

DateAdd("d", 3, GV_Date.ToDateTimeAtMidnight)
Returns: 01.05.2022 00:00:00

DateAdd("d", 3, DateAndTime.today.addhours(16))
Returns: 01.05.2022 16:00:00

In DocuWare Cloud, the returned value is adapted to the client's time zone.

Example: Germany = UTC+2

DateAdd("d", 3, GV_Date.ToDateTimeAtMidnight)
Returns: 01.05.2022 02:00:00

9.9 Return first/last day of Month

The first/last day of a month needs to be returned:

DateAdd("d", 1,DateSerial(Year(Today), Month(Today)+ 0, 0))
Returns: 01.09.2022

DateSerial(Year(Today), Month(Today) + 1, 0)
Returns: 30.09.2022