

Deltek Maconomy 2.3 GA

MDML and Expression Language
Standard Functions

December 2, 2016

While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This publication contains proprietary information that is protected by copyright. All rights are reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, or translated into another language, without the prior written consent of Deltek, Inc.

This edition published December 2016.

© Deltek, Inc.

Deltek's software is also protected by copyright law and constitutes valuable confidential and proprietary information of Deltek, Inc. and its licensors. The Deltek software, and all related documentation, is provided for use only in accordance with the terms of the license agreement. Unauthorized reproduction or distribution of the program or any portion thereof could result in severe civil or criminal penalties.

All trademarks are the property of their respective owners.

Contents

Introduction 1

 MDML and the Expression Language..... 1

 Format 1

Functions 2

 Environment Variables 2

 System Parameters..... 5

 System Information 8

 Misc System Functions 11

 Date and Time Functions 12

 Utility Functions 20

 Pop-Up Functions..... 21

 User Information 22

 User Derived Dimensions 23

 MDML-Specific Functions 27

 String Functions 31

 Mathematical Functions 37

Introduction

This manual documents the set of primitive standard functions that are available in the Expression Language that is used in MDML.

MDML and the Expression Language

A function call in the Expression Language has the following form:

```
maconomy:userEmployeeNumber()
```



maconomy is the namespace of the function and “userEmployeeNumber” is the function name.

Case is not significant, “currentdate()” means the same thing as “CURRENTDATE().”



The **maconomy** namespace is the default namespace and can be omitted from function calls.

A function may take arguments. Arguments are supplied comma-separated within the parenthesis:

```
hasRole('ProjectManager')
```



For a full reference of the expression language and its use in MDML, see the MDML Language Reference.

Format

Functions in this document are written in the following format:

```
maconomy:functionName( mandatoryParameter, [optionalParameter] )
```

Description of the function

Parameters

mandatoryParameter – Description of accepted parameter values

optionalParameter – Description of accepted parameter values

Returns

Description of return value including data type.



Optional parameters are presented in square brackets.

Functions

Environment Variables

The **envVar** family of functions is used to access the environment directly and fetch values from it using the full environment path.



Most standard functions are shortcuts to frequently used individual values or parts of the environment. The **envVar** family of function provides access to the entire environment.

Each function takes one parameter that designates the path to fetch from the environment. The environment may contain more than one value at a given path. In this case an appropriate type-specific **envVar** function must be used to fetch the value.

Sample use of an envVar function:

```
envVar('user.info.username') = 'Administrator'
```

In this example the path user.info.username is fetched and compared to the string 'Administrator'. Assuming that the value that is contained in the path user.info.username is a string data value, the following expression is equivalent:

```
stringEnvVar('user.info.username') = 'Administrator'
```

The string specific **envVar** function fails if the value that is contained at the specified path in the environment does not have the expected data type.

envVar

maconomy:envVar(path)

Fetch a value from the environment.

Parameters

path – String value that designates the full path in the environment.

Returns

If a single value is located at the specified path and that value has a maconomy data type, this function returns that value. Otherwise, the function fails.

isEnvVarDefined

maconomy:isEnvVarDefined(path)

Determine whether a value exists in the environment for a given path.

Parameters

path – String value that designates the full path in the environment.

Returns

True if the environment contains a value at the path. Otherwise, False.

amountEnvVar

```
maconomy:amountEnvVar( path )
```

Fetch an amount value from the environment.

Parameters

path – String value that designates the full path in the environment.

Returns

If a single amount value is located at the specified path, this function returns that value. Otherwise, the function fails.

booleanEnvVar

```
maconomy:booleanEnvVar( path )
```

Fetch a boolean value from the environment.

Parameters

path – String value that designates the full path in the environment

Returns

If a single boolean value is located at the specified path, this function returns that value. Otherwise, the function fails.

dateEnvVar

```
maconomy:dateEnvVar( path )
```

Fetch a date value from the environment.

Parameters

path – String value that designates the full path in the environment.

Returns

If a single date value is located at the specified path, this function returns that value. Otherwise, the function fails.

decimalEnvVar

```
maconomy:decimalEnvVar( path )
```

Fetch a decimal value from the environment.

Parameters

path – String value that designates the full path in the environment.

Returns

If a single decimal value is located at the specified path, this function returns that value. Otherwise, the function fails.

integerEnvVar

```
maconomy:integerEnvVar( path )
```

Fetch an integer value from the environment.

Parameters

path – String value that designates the full path in the environment.

Returns

If a single integer value is located at the specified path, this function returns that value. Otherwise, the function fails.

popupEnvVar

```
maconomy:popupEnvVar( path )
```

Fetch a popup value from the environment.

Parameters

path – String value that designates the full path in the environment.

Returns

If a single popup value is located at the specified path, this function returns that value. Otherwise, the function fails.

stringEnvVar

```
maconomy:stringEnvVar( path )
```

Fetch a string value from the environment.

Parameters

path – String value that designates the full path in the environment.

Returns

If a single string value is located at the specified path, this function returns that value. Otherwise, the function fails.

timeEnvVar

```
maconomy:timeEnvVar( path )
```

Fetch a time value from the environment.

Parameters

path – String value that designates the full path in the environment.

Returns

If a single time value is located at the specified path, this function returns that value. Otherwise, the function fails.

System Parameters

The **sysPar** family of functions contains helper functions to fetch the value of system parameters.

Each function takes one mandatory parameter that designates the name of the system parameter to fetch, and one optional parameter that designates a company number.

If a **sysPar** function is called with a company number argument and no company-specific system parameter is defined, the general value for the system parameter is returned.



Some system parameters have a combined decimal/amount format. To fetch the value of such a parameter, one of the type specific variants of the sysPar function must be used.

sysPar

```
maconomy:sysPar( parameterName, [companyNumber] )
```

Fetch the value of a named system parameter.

Parameters

parameterName – String value that designates the name of the parameter to fetch

companyNumber – String value, the company number of the company to fetch the parameter for.

Returns

If the system parameter exists and contains a single value, this function returns that value. Otherwise, the function fails.

amountSysPar

```
maconomy:amountSysPar( parameterName, [companyNumber] )
```

Fetch the amount value of a named system parameter.

Parameters

parameterName – String value that designates the name of the parameter to fetch.

companyNumber – String value, the company number of the company to fetch the parameter for.

Returns

If the system parameter exists and contains an amount value, this function returns that value. Otherwise, the function fails.

booleanSysPar

```
maconomy:booleanSysPar( parameterName, [companyNumber] )
```

Fetch the boolean value of a named system parameter.

Parameters

parameterName – String value that designates the name of the parameter to fetch.

companyNumber – String value, the company number of the company to fetch the parameter for.

Returns

If the system parameter exists and contains a boolean value, this function returns that value. Otherwise, the function fails.

dateSysPar

```
maconomy:dateSysPar( parameterName, [companyNumber] )
```

Fetch the date value of a named system parameter.

Parameters

parameterName – String value that designates the name of the parameter to fetch.

companyNumber – String value, the company number of the company to fetch the parameter for.

Returns

If the system parameter exists and contains a date value this function returns that value. Otherwise, the function fails.

integerSysPar

```
maconomy:integerSysPar( parameterName, [companyNumber] )
```

Fetch the integer value of a named system parameter.

Parameters

parameterName – String value that designates the name of the parameter to fetch.

companyNumber – String value, the company number of the company to fetch the parameter for.

Returns

If the system parameter exists and contains an integer value, this function returns that value. Otherwise, the function fails.

popupSysPar

```
maconomy:popupSysPar( parameterName, [companyNumber] )
```

Fetch the popup value of a named system parameter.

Parameters

parameterName – String value that designates the name of the parameter to fetch.

companyNumber – String value, the company number of the company to fetch the parameter for.

Returns

If the system parameter exists and contains a popup value, this function returns that value. Otherwise, the function fails.

stringSysPar

```
maconomy:stringSysPar( parameterName, [companyNumber] )
```

Fetch the string value of a named system parameter.

Parameters

parameterName – String value that designates the name of the parameter to fetch.

companyNumber – String value, the company number of the company to fetch the parameter for.

Returns

If the system parameter exists and contains a string value, this function returns that value. Otherwise, the function fails.

timeSysPar

```
maconomy:timeSysPar( parameterName, [companyNumber] )
```

Fetch the time value of a named system parameter.

Parameters

parameterName – String value that designates the name of the parameter to fetch.

companyNumber – String value, the company number of the company to fetch the parameter for.

Returns

If the system parameter exists and contains a time value, this function returns that value. Otherwise, the function fails.

System Information

The **sysInfo** function family fetches system information values. Each function takes one parameter that designates the name of the system information value to fetch.

sysInfo

```
maconomy:sysInfo( name )
```

Fetch a named system information value.

Parameters

name – String value that designates the name of system information value to fetch.

Returns

If the system information value exists this function returns that value. Otherwise, the function fails.

amountSysInfo

```
maconomy:amountSysInfo( name )
```

Fetch a named system information amount value.

Parameters

name – String value that designates the name of system information value to fetch.

Returns

If the system information value exists and is of type amount, this function returns that value. Otherwise, the function fails.

booleanSysInfo

```
maconomy:booleanSysInfo( name )
```

Fetch a named system information boolean value.

Parameters

name – String value that designates the name of system information value to fetch.

Returns

If the system information value exists and is of type boolean, this function returns that value. Otherwise, the function fails.

dateSysInfo

```
maconomy:dateSysInfo( name )
```

Fetch a named system information date value.

Parameters

name – String value that designates the name of system information value to fetch.

Returns

If the system information value exists and is of type date, this function returns that value. Otherwise, the function fails.

decimalSysInfo

```
maconomy:decimalSysInfo( name )
```

Fetch a named system information decimal value.

Parameters

name – String value that designates the name of system information value to fetch.

Returns

If the system information value exists and is of type decimal, this function returns that value. Otherwise, the function fails.

integerSysInfo

```
maconomy:integerSysInfo( name )
```

Fetch a named system information integer value.

Parameters

name – String value that designates the name of system information value to fetch.

Returns

If the system information value exists and is of type integer, this function returns that value. Otherwise, the function fails.

popupSysInfo

```
maconomy:popupSysInfo( name )
```

Fetch a named system information popup value.

Parameters

name – String value that designates the name of system information value to fetch.

Returns

If the system information value exists and is of type popup, this function returns that value. Otherwise, the function fails.

stringSysInfo

```
maconomy:stringSysInfo( name )
```

Fetch a named system information string value.

Parameters

name – String value that designates the name of system information value to fetch.

Returns

If the system information value exists and is of type string, this function returns that value. Otherwise, the function fails.

timeSysInfo

```
maconomy:timeSysInfo( name )
```

Fetch a named system information time value.

Parameters

name – String value that designates the name of system information value to fetch.

Returns

If the system information value exists and is of type time this function returns that value.
Otherwise, the function fails.

Misc System Functions

In addition to accessing system parameters and system information, there are functions for checking the presence of add-ons and short name of the current system.

hasAddOn

```
maconomy:hasAddOn ( addOnNumber )
```

Determine if a numbered addon is present in the system.

Parameters

addOnNumber – An integer value that identifies the addon which presence is to be determined.

Returns

This function returns boolean data value indicating the presence of the addon.

shortname

```
maconomy:shortname ()
```

Find the shortname of the system.

Returns

This function returns a string value indicating the shortname of the system.

getToken

```
kona:getToken ()
```

Fetch the user's Kona token. The server retrieves this token from www.kona.com after a successful Kona login. The token is used to preserve the user's Kona session during one session with the Workspace client.

Returns

This function returns a string value indicating the token or an empty string if the user is not logged in to Kona.

Date and Time Functions

The family of date and time functions includes functions for querying the current date and time and extracting day, month, year, and others from date values.

date

maconomy:date(year, month, day)

Construct a date value from a year, month and a day. If the arguments are not a valid date this function fails.

Parameters

year – An integer that specifies the year of the resulting date value.

month – An integer that specifies the month of the resulting date value (1 is January, 2 February, etc.).

day – An integer that specifies the day of the resulting date value (1 is the 1st, 2 is the 2nd, etc.).

Returns

This function returns a date value with the specified values.

time

maconomy:time(hours, minutes, [seconds])

Construct a time value from hours, minutes and seconds. If the arguments are not a valid time this function fails.

Parameters

hours – An integer that specifies the hours of the time value (valid range is 0 through 23).

minutes – An integer that specifies the minutes of the time value (valid range is 0 through 59).

seconds – An optional integer that specifies the seconds of the time value (valid range is 0 through 59).

Returns

This function returns a time value with the specified values.

currentDate

maconomy:currentDate()

Find the current date and convert it into the time-zone of the server. This function can be used to get an approximate of the current server date.

Returns

This function returns a date value, the current date converted to the time-zone of the server.

currentTime

maconomy:currentTime()

Find the current time and convert it into the time-zone of the server. This function can be used to get an approximate of the current server time.

Returns

This function returns a time value, the current time converted to the time-zone of the server.

userDate

maconomy:userDate()

Find the current date *without* converting it into the time-zone of the server. When run on the client, this function returns the date of the client machine.

Returns

This function returns a date value, the current date.

userTime

maconomy:userTime()

Find the current time *without* converting it into the time-zone of the server. When run on the client, this function returns the time of the client machine.

Returns

This function returns a time value, the current time.

second

maconomy:second(time)

Find the second a time value. E.g. second(time(14:32:46)) evaluates to 46.

Parameters

time – The time value to convert.

Returns

This function returns an integer value, the second of the time value.

minute

maconomy:minute(time)

Find the minute a time value. E.g. minute(time(14:32:46)) evaluates to 32.

Parameters

time – The time value to convert.

Returns

This function returns an integer value, the minute of the time value.

hour

maconomy:hour(time)

Find the hour a time value. E.g. hour (time(14:32:46)) evaluates to 14.

Parameters

time – The time value to convert.

Returns

This function returns an integer value, the hour of the time value.

day

maconomy:day(date)

Find the day of month of a date value. E.g. day(date(2010/03/15)) evaluates to 15.

Parameters

date – The date value to convert.

Returns

This function returns an integer value, the day of month of the date value.

month

maconomy:month(date)

Find the month of a date value. E.g. month(date(2010/03/15)) evaluates to 3.

Parameters

date – The date value to convert.

Returns

This function returns an integer value, the month of the date value.

year

maconomy:year(date)

Find the year of a date value. E.g. year(date(2010/03/15)) evaluates to 2010.

Parameters

date – The date value to convert.

Returns

This function returns an integer value, the year of the date value.

week

maconomy:week (date)

Find the week number of a date value. Week numbers are computed using the ISO 8601 week numbering scheme with Monday as the first day of the week.
E.g. day(date(2010/03/15)) evaluates to 11.

Parameters

date – The date value to convert.

Returns

This function returns an integer value, the week number of the date value.

intWeekday

maconomy:intWeekday (date)

Find the week day of a date value. The week starts with Monday (1) and ends on Sunday (7).

Parameters

date – The date value to convert.

Returns

This function returns an integer value, the weekday of the date value.

stringWeekday

maconomy:stringWeekday (date)

Find the week day of a date value. The week day is returned as a localized string value.

Parameters

date – The date value to convert.

Returns

This function returns a string value, the name of the weekday of the date value.

addDays

maconomy:addDays (date , days)

Construct a date value by adding a number of days to a date.

Parameters

date – A date value.

days – An integer that specifies a number of days to add to the given date (may be negative).

Returns

This function returns a date value that is the argument date plus the number of days.

addMonths

```
maconomy:addMonths( date, months )
```

Construct a date value by adding a number of months to a date.

Parameters

date – A date value.

months – An integer that specifies a number of months to add to the given date (may be negative).

Returns

This function returns a date value that is the argument date plus the number of months.

addYears

```
maconomy:addYears( date, years )
```

Construct a date value by adding a number of years to a date.

Parameters

date – A date value.

years – An integer that specifies a number of years to add to the given date (may be negative).

Returns

This function returns a date value that is the argument date plus the number of years.

addPeriod

```
maconomy:addPeriod( date, years, months, days )
```

Construct a date value by adding a number of years, months and days to a date.

Parameters

date – A date value.

years – An integer that specifies a number of years to add to the given date (may be negative).

months – An integer that specifies a number of months to add to the given date (may be negative).

days – An integer that specifies a number of days to add to the given date (may be negative).

Returns

This function returns a date value that is the argument date plus the number of years, months and days.

addHours

```
maconomy:addHours( time, hours )
```

Construct a date time by adding a number of hours to a time.

Parameters

time – A time value.

hours – An integer that specifies a number of hours to add to the given time (may be negative).

Returns

This function returns a time value that is the argument time plus the number of hours.

addMinutes

```
maconomy:addMinutes( time, hours )
```

Construct a date time by adding a number of minutes to a time.

Parameters

time – A time value.

minutes – An integer that specifies a number of minutes to add to the given time (may be negative).

Returns

This function returns a time value that is the argument time plus the number of minutes.

addSeconds

```
maconomy:addSeconds( time, hours )
```

Construct a date time by adding a number of seconds to a time.

Parameters

time – A time value.

seconds – An integer that specifies a number of seconds to add to the given time (may be negative).

Returns

This function returns a time value that is the argument time plus the number of seconds.

daysBetween

```
maconomy:daysBetween( date1, date2 )
```

Find the number of days in the interval between two dates.

Parameters

date1 – The start date of the interval.

date2 – The end date of the interval.

Returns

This function returns an integer value, the number of days between the two dates.

daysBetween

```
maconomy:daysBetween( date1, date2 )
```

Find the number of days in the interval between two dates.

Parameters

date1 – The start date of the interval.

date2 – The end date of the interval.

Returns

This function returns an integer value, the number of days between the two dates.

monthsBetween

```
maconomy:monthsBetween( date1, date2 )
```

Find the number of months in the interval between two dates.

Parameters

date1 – The start date of the interval.

date2 – The end date of the interval.

Returns

This function returns an integer value, the number of months between the two dates.

yearsBetween

```
maconomy:yearsBetween( date1, date2 )
```

Find the number of years in the interval between two dates.

Parameters

date1 – The start date of the interval.

date2 – The end date of the interval.

Returns

This function returns an integer value, the number of years between the two dates.

secondsBetween

```
maconomy:secondsBetween( time1, time2 )
```

Find the number of seconds in the interval between two times.

Parameters

time1 – The start time of the interval.

time2 – The end time of the interval.

Returns

This function returns an integer value, the number of seconds between the two times.

minutesBetween

```
maconomy:minutesBetween( time1, time2 )
```

Find the number of minutes in the interval between two times.

Parameters

time1 – The start time of the interval.

time2 – The end time of the interval.

Returns

This function returns an integer value, the number of minutes between the two times.

hoursBetween

```
maconomy:hoursBetween( time1, time2 )
```

Find the number of hours in the interval between two times.

Parameters

time1 – The start time of the interval.

time2 – The end time of the interval.

Returns

This function returns an integer value, the number of hours between the two times.

Utility Functions

A number of utility functions can be used to modify the output of other functions.

format

maconomy:format(value, format)

Format a value to a string of the given format. Depending on the type of the value different format string are allowed.

Integer, Real, Amount are formatted using a DecimalFormat. For more information, see <http://docs.oracle.com/javase/6/docs/api/java/text/DecimalFormat.html>

Time and Dates are formatted using a DateFormatter. For more information, see <http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html>

String are not formatted but returned as is.

PopupValue returns a string representation of the literal.

Booleans are formatted as "true" and "false."

Returns

This function converts the value to a string by using the given format. The function returns a string.

urlEncode

maconomy:urlEncode(value)

Creates a canonical string representation of a value of any type and then the URL encodes this representation. The URL encoding is done using the URLEncoder. For more information, see <http://download.oracle.com/javase/6/docs/api/>

The encoding scheme is always UTF-8.

Returns

This function converts the value to a URL encoded string. The function returns a string.

Pop-Up Functions

popupTitle

```
maconomy:popupTitle( popupValue )
```

Extract the display title of a pop-up value.

Parameters

popupValue – The pop-up value to convert.

Returns

This function returns a string value containing the display title of the pop-up value.

popupLiteral

```
maconomy:popupLiteral( popupValue )
```

Extract the literal value of a pop-up value.

Parameters

popupValue – The pop-up value to convert.

Returns

This function returns a string value containing the literal value of the pop-up value.

popupOrdinal

```
maconomy:popupOrdinal( popupValue )
```

Extract the ordinal value of a pop-up value.

Parameters

popupValue – The pop-up value to convert.

Returns

This function returns the ordinal of the pop-up value as an integer value.

popup

```
maconomy:popup( type, literal, ordinal )
```

Construct a pop-up value.

Parameters

type – The type of the pop-up value.

literal – The literal of the pop-up value.

Ordinal – The ordinal of the pop-up value.

Returns

This function returns a pop-up value with the specified type, literal, and ordinal.

User Information

There are a few shortcut functions for accessing common user information, such as username and roles, despite the fact user information is stored in the environment under the path “user.info.*”

username

```
maconomy:username()
```

Find the username of the user.

Returns

This function returns a string value indicating the username of the user.

userEmployeeNumber

```
maconomy:userEmployeeNumber()
```

Find the employee number of the user.

Returns

This function returns a string value indicating the employee number of the user. If the user does not have an associated employee number, the empty string is returned.

isAdministrator

```
maconomy:isAdministrator()
```

Determine if the user is an administrator.

Returns

This function will return a boolean value indicating if the user is an administrator.

hasRole

```
maconomy:hasRole( role )
```

Determine if the user has a named role.

Parameter

role – Either (a) a string value indicated the name of the role, or (b) a value of the popup type GroupNameType.

Returns

This function returns a boolean value indicating whether the user has the named role.

User Derived Dimensions

The derived dimensions functions fetch the derived dimensions values of the user.



The functions first attempt to fetch the information stored on the employee associated with the user. If there is no employee associated with the user, then the value are acquired from the user information.

Each of the derived dimensions functions takes an optional date argument. It returns the value that applies at that date. If no date argument is supplied, the value that applies to the current date is returned.

companyNumber

```
maconomy:companyNumber ( [date] )
```

Fetch the value of the derived dimension 'company number' for the user.

Parameter

date – An optional date value. If a value is supplied, the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

accountNumber

```
maconomy:accountNumber ( [date] )
```

Fetch the value of the derived dimension 'account number' for the user.

Parameter

date – An optional date value. If a value is supplied, the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

locationName

```
maconomy:locationName ( [date] )
```

Fetch the value of the derived dimension 'location name' for the user.

Parameter

date – An optional date value. If a value is supplied, then the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

entityName

```
maconomy:entityName( [date] )
```

Fetch the value of the derived dimension 'entity name' for the user.

Parameter

date – An optional date value. If a value is supplied, the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

projectName

```
maconomy:projectName( [date] )
```

Fetch the value of the derived dimension 'project name' for the user.

Parameter

date – An optional date value. If a value is supplied, the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

purposeName

```
maconomy:purposeName( [date] )
```

Fetch the value of the derived dimension 'purpose name' for the user.

Parameter

date – An optional date value. If a value is supplied, the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

specification1Name

```
maconomy:specification1Name( [date] )
```

Fetch the value of the derived dimension 'specification 1 name' for the user.

Parameter

date – An optional date value. If a value is supplied, the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

specification2Name

```
maconomy:specification2Name( [date] )
```

Fetch the value of the derived dimension 'specification 2 name' for the user.

Parameter

date – An optional date value. If a value is supplied, the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

specification3Name

```
maconomy:specification3Name( [date] )
```

Fetch the value of the derived dimension 'specification 3 name' for the user.

Parameter

date – An optional date value. If a value is supplied, the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

localSpec1Name

```
maconomy:localSpec1Name( [date] )
```

Fetch the value of the derived dimension 'local spec 1 name' for the user.

Parameter

date – An optional date value. If a value is supplied, the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

localSpec2Name

```
maconomy:localSpec2Name( [date] )
```

Fetch the value of the derived dimension 'local spec 2 name' for the user.

Parameter

date – An optional date value. If a value is supplied, the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

localSpec3Name

```
maconomy:localSpec3Name ( [date] )
```

Fetch the value of the derived dimension 'local spec 3 name' for the user.

Parameter

date – An optional date value. If a value is supplied, the value valid at the specified date is returned. Otherwise, the value valid at the current date is returned.

Returns

This function returns a string value indicating the value of the derived dimension for the user.

MDML-Specific Functions

MDML offers a collection of functions that are only available in the context of MDML.

inEmpty

maconomy: inEmpty ()

Determine if a pane is in an empty state.

Returns

This function returns a boolean value indicating whether the pane is in an empty state.

inCreate

maconomy: inCreate ()

Determine if a pane is in a 'create' state.

Returns

This function returns a boolean value indicating whether the pane is in a create state.

inUpdate

maconomy: inUpdate ()

Determine if a pane is in an 'update' state.

Returns

This function returns a boolean value indicating whether the pane is in an update state.

hasWorkspace

maconomy: hasWorkspace (workspaceName)

Determine if a named workspace is an ancestor of the current pane.

Parameter

workspaceName – The name of the workspace whose presence in the tree context is to be determined.

Returns

This function returns a boolean value indicating whether the workspace is an ancestor of the pane.

workspace

maconomy: workspace ()

Fetch the name of the main workspace in the current context.

Returns

This function returns a string value containing the name of the workspace in the current context.

container

maconomy:container()

Fetch the name of the container in the current context.

Returns

This function returns a string value containing the name of the container in the current context.

pane

maconomy:pane()

Fetch the name of the pane in the current context.

Returns

This function returns a string value containing the name of the pane in the current context.

entity

maconomy:entity()

Fetch the name of the entity in the current context.

Returns

This function returns a string value containing the name of the entity in the current context.

field

maconomy:field()

Fetch the name of the field in the current context.

Returns

This function returns a string value containing the name of the field in the current context.
If there is no field in the current context, this function fails.

fieldValue

maconomy:fieldValue()

Fetch the value of the field in the current context.

Returns

This function returns a value containing the value of the field in the current context.
If there is no field in the current context, this function fails.

isSelectedOption

```
maconomy:isSelectedOption( [optionName] )
```

Determine if a given filter option is selected. This only makes sense in filter panes; in other panes this function will always return false.

Parameters

optionName – The name of the option. This is a string value that matches the name attribute on the option element in the MDML specification.

Returns

This function returns a boolean value. True, if the specified option is the selected filter option. Otherwise, false.

selectedOption

```
maconomy:selectedOption()
```

Fetch the name of the currently selected option. This only makes sense in filter panes. In other panes this function will always return the empty string.

Returns

This function returns a string value containing the name of the currently selected filter option.

visibleFields

```
maconomy:visibleFields()
```

Fetch the list of the currently visible fields in the current pane. The list is sorted in layout order.

Returns

This function returns a list (number indexed data map) of the visible fields in the current pane.

visibleFieldTitles

```
maconomy:visibleFieldTitles()
```

Fetch the field titles of the currently visible fields in the current pane. The returned value is a map from field name to field title.

Returns

This function returns a data map from field name to the field titles for all visible fields in the current pane.

isActionEnabled

```
maconomy:isActionEnabled( actionName )
```

Test if an action is enabled.

Parameters

actionName – The name of the tested action. This is the name specified by the application and not the name specified by the name-attribute in MDML.

Returns

This function returns true if the action with the name actionName is enabled.

String Functions

String functions can be used to manipulate strings and perform validations.

length

```
maconomy:length( string )
```

Find the length of the input string.

Parameters

string – The string value to find the length of.

Returns

This function returns an integer of the number of characters in the argument string.

startsWith

```
maconomy:startsWith( haystack, needle, [fromIndex] )
```

Test if a string begins with a given other string.

Parameters

haystack – The string to check.

needle – The string to check if haystack begins with.

fromIndex – Optional integer. The index in haystack to start looking for needle. Defaults to zero.

Returns

This function returns true, if haystack starts with needle at the index. Otherwise, false.

endsWith

```
maconomy:endsWith( haystack, needle )
```

Test if a string ends with a given other string.

Parameters

haystack – The string to check.

needle – The string to check if haystack ends with.

Returns

This function returns true, if haystack ends with needle at the index. Otherwise, false.

firstIndexOf

```
maconomy:firstIndexOf( haystack, needle )
```

Find the first occurrence of needle in haystack.

Parameters

haystack – The string to search in.

needle – The string to look for in haystack.

Returns

This function returns the index of the first occurrence of needle in haystack, or -1 if needle does not occur in haystack.

indexOf

```
maconomy:indexOf( haystack, needle, [fromIndex] )
```

Find the first occurrence of needle in haystack after a given index.

Parameters

haystack – The string to search in.

needle – The string to look for in haystack.

fromIndex – Optional integer. The index in haystack to start looking for needle. Defaults to zero.

Returns

This function returns the index of the first occurrence of needle in haystack, or -1 if needle does not occur in haystack after the specified index.

lastIndexOf

```
maconomy:lastIndexOf( haystack, needle )
```

Find the last occurrence of needle in haystack.

Parameters

haystack – The string to search in.

needle – The string to look for in haystack.

Returns

This function returns the index of the last occurrence of needle in haystack, or -1 if needle does not occur in haystack.

contains

```
maconomy:contains( haystack, needle )
```

Test if needle occurs in haystack.

Parameters

haystack – The string to search in.

needle – The string to look for in haystack.

Returns

This function returns true if needle occurs in haystack. Otherwise, false.

substring

```
maconomy:substring( string, fromIndex, [toIndex] )
```

Take a substring of a string.

Parameters

string – The string to take a substring of.

fromIndex – Integer value that specified the index where the substring starts.

toIndex – Optional integer. The index where the substring ends. If no index is specified, the substring continues to the end of the argument string.

Returns

This function returns a string value that is the specified substring.

trim

```
maconomy:trim( string )
```

Trim whitespace from the ends of a string.

Parameters

string – The string to trim.

Returns

This function returns a string value that is argument string trimmed for any leading and trailing whitespace.

upperCase

```
maconomy:upperCase( string )
```

Uppercase a string using US locale rules.

Parameters

string – The string to uppercase.

Returns

This function returns the argument string converted to uppercase.

lowerCase

```
maconomy:lowerCase( string )
```

Lowercase a string using US locale rules.

Parameters

string – The string to lowercase.

Returns

This function returns the argument string converted to lowercase.

replaceFirst

```
maconomy:replaceFirst( haystack, needle, replacement )
```

Look for needle in haystack and replace the first occurrence with replacement.

Parameters

haystack – The string to search in.

needle – The string to look for in haystack.

replacement – The string to replace the first occurrence of needle with.

Returns

This function returns a string that is haystack with the first occurrence of needle replaced with replacement.

replaceAll

```
maconomy:replaceAll( haystack, needle, replacement )
```

Look for needle in haystack and replace all occurrences with replacement.

Parameters

haystack – The string to search in.

needle – The string to look for in haystack.

replacement – The string to replace all occurrences of needle with.

Returns

This function returns a string that is haystack with all occurrences of needle replaced with replacement.

replaceFirstRegEx

```
maconomy:replaceFirstRegEx( haystack, needlePattern, replacement )
```

Look for needlePattern in haystack and replace the first occurrence with replacement.

NeedlePattern is a regular expression that follows the Java regular expression syntax. For more information, see <http://docs.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html>

The replacement string may contain placeholders that capture part of the regular expression. For more information, see:

[http://docs.oracle.com/javase/1.4.2/docs/api/java/util/regex/Matcher.html#appendReplacement\(java.lang.StringBuffer, java.lang.String\)](http://docs.oracle.com/javase/1.4.2/docs/api/java/util/regex/Matcher.html#appendReplacement(java.lang.StringBuffer, java.lang.String))

Parameters

haystack – The string to search in.

needlePattern – String that contains a regular expression to match in haystack.

replacement – The string to replace the first occurrence of needle with.

Returns

This function returns a string that is haystack with the first occurrence of the needle pattern replaced with replacement.

replaceAllRegEx

```
maconomy:replaceAllRegEx( haystack, needlePattern, replacement )
```

Look for needlePattern in haystack and replace all occurrences with replacement.

NeedlePattern is a regular expression that follows the Java regular expression syntax. For more information, see: <http://docs.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html>

The replacement string may contain placeholders that capture part of the regular expression. For more information, see:

[http://docs.oracle.com/javase/1.4.2/docs/api/java/util/regex/Matcher.html#appendReplacement\(java.lang.StringBuffer, java.lang.String\)](http://docs.oracle.com/javase/1.4.2/docs/api/java/util/regex/Matcher.html#appendReplacement(java.lang.StringBuffer, java.lang.String))

Parameters

haystack – The string to search in.

needlePattern – String that contains a regular expression to match in haystack.

replacement – The string to replace all occurrences of needle with.

Returns

This function returns a string that is haystack with all occurrences of the needle pattern replaced with replacement.

matchRegEx

```
maconomy:matchRegEx( haystack, needlePattern )
```

Match the needle pattern in haystack.

NeedlePattern is a regular expression that follows the Java regular expression syntax. For more information, see: <http://docs.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html>

Parameters

haystack – The string to search in.

needle pattern – String that contains a regular expression to match in haystack.

Returns

This function returns true if haystack matches the needle pattern. Otherwise, false.

Mathematical Functions

These are common mathematical functions:

min

```
maconomy:min( num, nums... )
```

Find the smallest value of one or more numeric arguments.

Parameters

num – The first numeric value.

nums – Any number of additional numeric values.

Returns

This function returns the smallest of its arguments.

max

```
maconomy:max( num, nums... )
```

Find the largest value of one or more numeric arguments.

Parameters

num – The first numeric value.

nums – Any number of additional numeric values.

Returns

This function returns the largest of its arguments.

abs

```
maconomy:abs( num )
```

Find the absolute value of a numeric value.

Parameters

num – The numeric value.

Returns

This function returns the absolute value of its argument.

sign

```
maconomy:sign( num )
```

Find the sign of a numeric value.

Parameters

num – The numeric value.

Returns

This function returns 1 if the value is positive, 0 if the value is zero, and -1 if the value is negative.

round

```
maconomy:round( num )
```

Round the value to the nearest integer using the half away from zero rule.

Parameters

num – The numeric value.

Returns

This function returns its argument rounded to the nearest integer.

floor

```
maconomy:floor( num )
```

Round the value to the nearest integer using, always rounding down.

Parameters

num – The numeric value.

Returns

This function returns its argument rounded to the nearest integer.

ceiling

```
maconomy:ceiling( num )
```


Round the value to the nearest integer using, always rounding up.

Parameters

num – The numeric value.

Returns

This function returns its argument rounded to the nearest integer.



Deltek is the leading global provider of enterprise software and information solutions for professional services firms, government contractors, and government agencies. For decades, we have delivered actionable insight that empowers our customers to unlock their business potential. Over 14,000 organizations and 1.8 million users in approximately 80 countries around the world rely on Deltek to research and identify opportunities, win new business, optimize resource, streamline operations, and deliver more profitable projects. Deltek – Know more. Do more.®

deltek.com