netgenium

Server functions

Framework NET Genium

double ABS(double number)

Use: *HTML, Script*

Description: *The function returns the absolute value from the value specified by the "number" parameter.*

Example:

```
ABS(-1.23)
// Returns "1,23"
ABS(#A#)
```

DateTime ADDDAYS(DateTime date, double days)

Use: *HTML, JavaScript*

Description: *The function adds the number of days defined by the "days" parameter to the date specified by the "date" parameter.*

Example:

```
ADDDAYS(1.1.2000, -1.2)
// Returns "30.12.1999 19:12"
ADDDAYS(1.1.2000, 0)
// Returns "01.01.2000"
ADDDAYS(1.1.2000, 2)
// Returns "03.01.2000"
ADDDAYS(#today#, 0)
```

DateTime ADDDAYS(DateTime date, double days, int skipholidays)

Use: *HTML, JavaScript, Script*

Description: *The function adds the number of days defined by the "days" parameter to the date specified by the "date" parameter. The "skipholidays" parameter, which can be any integer, indicates that holidays will be skipped. Therefore, if the resulting date is a public holiday, Saturday or Sunday, it adds additional "skipholidays" days.*

Example:

```
ADDDAYS(1.1.2000, -9.2, -2)
// Returns "22.12.1999 19:12"
ADDDAYS(1.1.2000, 0, 1)
// Returns "03.01.2000"
ADDDAYS(1.1.2000, 12, 0)
// Returns "13.01.2000"
ADDDAYS(#today#, 0, 3)
```

DateTime ADDDAYS(DateTime date, double days, int skipholidays, string satsun)

Use: *HTML, JavaScript, Script*

Description: *The function adds the number of days defined by the "days" parameter to the date specified by the "date" parameter. The "skipholidays" parameter, which can be any integer, indicates that holidays will be skipped. Therefore, if the resulting date is a public holiday, Saturday or Sunday, it adds additional "skipholidays" days. By default, Saturdays and Sundays are skipped; to prevent Saturday or Sunday from being skipped, the text "sat" or "sun" must be specified as the "satsun" parameter.*

Example:

```
ADDDAYS(1.1.2000, -9.2, -2, sat)
// Returns "22.12.1999 19:12"
ADDDAYS(1.1.2000, 0, 1, sun)
// Returns "02.01.2000"
ADDDAYS(1.1.2000, 12, 0, sat)
// Returns "13.01.2000"
ADDDAYS(#today#, 0, 3, sun)
```

DateTime ADDDAYS(DateTime date, double days, int skipholidays, string sat, string sun)

Use: *HTML, JavaScript, Script*

Description: *The function adds the number of days defined by the "days" parameter to the date specified by the "date" parameter. The "skipholidays" parameter, which can be any integer, indicates that holidays will be skipped. Therefore, if the resulting date is a public holiday, Saturday or Sunday, it adds additional "skipholidays" days. By default, Saturdays and Sundays are skipped; to prevent both of these days from being skipped, the text "sat" and "sun" must be specified as the "sat" and "sun" parameters.*

Example:

```
ADDDAYS(1.1.2000, -9.2, -2, sat, sun)
// Returns "22.12.1999 19:12"
ADDDAYS(1.1.2000, 0, 1, sat, sun)
// Returns "02.01.2000"
ADDDAYS(1.1.2000, 12, 0, sat, sun)
// Returns "13.01.2000"
ADDDAYS(#today#, 0, 3, sat, sun)
```

DateTime ADDHOURS(DateTime date, double hours)

Use: *HTML, JavaScript, Script*

Description: *The function adds the number of hours defined by the "hours" parameter to the date specified by the "date" parameter.*

Example:

```
ADDHOURS(1.1.2000, -1.5)
// Returns "31.12.1999 22:30"
ADDHOURS(1.1.2000, 0)
// Returns "01.01.2000"
ADDHOURS(1.1.2000, 5)
// Returns "01.01.2000 05:00"
ADDHOURS(#now#, 10)
```

DateTime ADDMINUTES(DateTime date, double minutes)

Use: *HTML, JavaScript, Script*

Description: *The function adds the number of minutes defined by the "minutes" parameter to the date specified by the "date" parameter.*

Example:

```
ADDMINUTES(1.1.2000, -2.5)
// Returns "31.12.1999 23:57"
ADDMINUTES(1.1.2000, 0)
// Returns "01.01.2000"
ADDMINUTES(1.1.2000, 2)
// Returns "01.01.2000 00:02"
ADDMINUTES(#now#, 0)
```

DateTime ADDMONTHS(DateTime date, int months)

Use: *HTML, JavaScript, Script*

Description: *The function adds the number of months defined by the "months" parameter to the date specified by the "date" parameter. Entering a decimal number in the "months" parameter causes a conversion to the "int" data type, it does not round, but the numbers after the decimal point are truncated.*

Example:

```
ADDMONTHS(1.1.2000, -2.5)
// Returns "01.11.1999"
ADDMONTHS(1.1.2000, 0)
// Returns "01.01.2000"
ADDMONTHS(1.1.2000, 2)
// Returns "01.03.2000"
ADDMONTHS(#now#, -2)
```

DateTime ADDTIMESPANINWORDS(DateTime date, string expression1, string expression2,…)

Use: *HTML, JavaScript, Script*

Description: *The function adds to the date specified by the "date" parameter the time data expressed by the verbal description defined by the "expression1", "expression2", etc. parameter. The function can contain any number of parameters that is greater than or equal to 2. The verbal description of the time data can be defined in any language that is part of NET Genium.*

Example:

```
ADDTIMESPANINWORDS(1.1.2000, 2 months)
// Returns "01.03.2000"
ADDTIMESPANINWORDS(1.1.2000, 1 year)
// Returns "01.01.2001"
ADDTIMESPANINWORDS(1.1.2000, 1 year, 2 months, 1 day)
// Returns "02.03.2001"
ADDTIMESPANINWORDS(1.1.2000, 1 den)
// Returns "02.01.2000"
ADDTIMESPANINWORDS(1.1.2000 12:05, 1 year, 2 months, 1 day, 5 minutes, 28 seconds)
// Returns "02.03. 2001 12:10"
```

DateTime ADDWORKDAYS(DateTime date, int days)

Use: *HTML, JavaScript, Script*

Description: *The function adds the number of working days defined by the "days" parameter to the date specified by the "date" parameter. Entering a decimal number in the "days" parameter causes a conversion to the "int" data type, it does not round, but the numbers after the decimal point are truncated.*

Example:

```
ADDWORKDAYS(1.1.2000, -2.5)
// Returns "30.12.1999"
ADDWORKDAYS(1.1.2000, 0)
// Returns "01.01.2000"
ADDWORKDAYS(1.1.2000, 2)
// Returns "04.01.2000"
ADDWORKDAYS(#now#, -2)
```

DateTime ADDYEARS(DateTime date, int years)

Use: *HTML, JavaScript, Script*

Description: *The function adds the number of years defined by the "years" parameter to the date specified by the "date" parameter. Entering a decimal number in the "years" parameter causes a conversion to the "int" data type, it does not round, but the numbers after the decimal point are truncated.*

Example:

```
ADDYEARS(1.1.2000, -2.5)
// Returns "01.01.1998"
ADDYEARS(1.1.2000, 0)
// Returns "01.01.2000"
ADDYEARS(1.1.2000, 2)
// Returns "01.01.2002"
ADDYEARS(#today#, 2)
```

double ARCCOS(double number)

Use: *HTML, Script*

Description: *The function returns the arc cosine of the value specified by the "number" parameter.*

Example:

```
ARCCOS(0)
// Returns "1,5707963267949"
ARCCOS(2)
// Returns "Not a number"
ARCCOS(#A#)
```

double ARCSIN(double number)

Use: *HTML, Script*

Description: *The function returns the arc sine from the value specified by the "number" parameter.*

Example:

```
ARCSIN(0)
// Returns "1"
ARCSIN(-1)
// Returns "-1,5707963267949"
ARCSIN(#A#)
```

double ARCTAN(double number)

Use: *HTML, Script*

Description: *The function returns the arc tangent from the value specified by the "number" parameter.*

Example:

```
ARCTAN(-0.5)
// Returns "-0,463647609000806"
ARCTAN(3)
// Returns "1,24904577239825"
ARCTAN(#A#)
```

string ATTACHMAILSIGNATURE(string body, string signature)

Use: *HTML, JavaScript, Script*

Description: *The function inserts the e-mail signature defined by the "signature" parameter into the e-mail message specified by the "body" parameter. If the function evaluates that there are any e-mail replies in the sent message, it inserts an e-mail signature before these replies. Otherwise, the e-mail signature is inserted at the end of the e-mail message.*

Example:

```
ATTACHMAILSIGNATURE(abc, Best regards)
```

int ATTACHMENT2PDF(int file)

Use: *Script*

Description: *The function converts the file attachment specified by the "file" parameter to the "pdf" format. The function accepts the file attachment ID in "doc", "docx", "xls" or "xlsx" format, and returns the ID of the newly created file attachment in "pdf" format.*

Example:

```
ATTACHMENT2PDF(#ng_file#)
```

int ATTACHMENT2PDF(string file)

Use: *Script*

Description: *The function converts the file attachment specified by the "file" parameter to the "pdf" format. The function accepts the full path of the file attachment in "doc", "docx", "xls" or "xlsx" format, and returns the ID of the newly created file attachment in "pdf" format.*

Example:

```
ATTACHMENT2PDF(#rootpath#Temp\file.doc)
ATTACHMENT2PDF(#rootpath#Temp\file.xls)
```

string ATTACHMENTCONTENT(int file)

Use: *HTML, JavaScript, Script*

Description: *The function returns the contents of the file attachment specified by the "file" parameter in the form of a text string.*

Example:

```
ATTACHMENTCONTENT(#ng_file#)
```

string ATTACHMENTDIMENSIONS(int image)

Use: *HTML, JavaScript, Script*

Description: *The function returns the dimensions of the file attachment – image – specified by the "image" parameter in the format "width; height".*

Example:

```
ATTACHMENTDIMENSIONS(#ng_image#)
```

string ATTACHMENTGALLERY(string attachments, int maxwidth, int maxheight)

Use: *HTML*

Description: *The function displays thumbnails or links to file attachments in a list below each other. File attachments are specified by the "attachments" parameter as a semicolon-separated list of file attachments. The "maxwidth" and "maxheight" parameters specify the maximum width and height of the thumbnail in pixels.*

Example:

```
ATTACHMENTGALLERY(#ng_file1:value#;#ng_file2:value#;#ng_file3:value#, 120, 120)
```

string ATTACHMENTIMG(int file)

Use: *HTML, JavaScript, Script*

Description: *The function returns the HTML code of the image with the attachment specified by the "file" parameter.*

Example:

```
ATTACHMENTIMG(#ng_file:value#)
// Returns "<img src="Download.aspx?hWiPNy0VInyQJbufLA1pTA==" alt="Name of the attachment with image.png">"
```

string ATTACHMENTNAME(int file, bool extension)

Use: *HTML, JavaScript, Script*

Description: *The function returns the file name, including the file attachment extension specified by the "file" parameter. The optional "extension" parameter specifies whether the function should return the file name, including the file attachment extension, or just the file name without the extension.*

Example:

```
ATTACHMENTNAME(#ng_file#)
ATTACHMENTNAME(#ng_file#, true)
```

long ATTACHMENTSIZE(int file)

Use: *HTML, JavaScript, Script*

Description: *The function returns the size in bytes of the file attachment specified by the "file" parameter.*

Example:

```
ATTACHMENTSIZE(#ng_file#)
```

string ATTACHMENTURL(int file)

Use: *HTML, JavaScript, Script*

Description: *The function returns the private Internet address of the file attachment specified by the "file" parameter, functional only for the currently logged in user, with a time validity limited to 24 hours.*

Example:

```
ATTACHMENTURL(1)
// Returns "Download.aspx?hWiPNy0VInyQJbufLA1pTA=="
ATTACHMENTURL(#ng_file#)
```

string ATTACHMENTURL(int file, string url)

Use: *HTML, JavaScript, Script*

Description: *The function returns the public Internet address of the file attachment specified by the "file" parameter with unlimited validity.*

Example:

```
ATTACHMENTURL(1, https://www.netgenium.com)
// Returns "https://www.netgenium.com/Download.aspx?hWiPNy0VInyQJbufLA1pTA=="
ATTACHMENTURL(#ng_file#, https://www.netgenium.com)
```

string ATTACHMENTVIDEO(int file, int maxwidth, int maxheight)

Use: *HTML, JavaScript, Script*

Description: *The function returns the HTML code of the video with the attachment specified by the "file" parameter. The "maxwidth" and "maxheight" parameters specify the maximum width and height of the video in pixels.*

Example:

```
ATTACHMENTVIDEO(#ng_file:value#, 1000, 500)
// Returns "<video><source src="Download.aspx?hWiPNy0VInyQJbufLA1pTA==""></video>"
```

double AVERAGE(object[] array)

Use: *Script*

Description: *The function returns the average of the values of the script variable specified by the "array" parameter, if the variable is an array of values. If the variable is not an array of values, the function returns the value of the variable.*

Example:

```
AVERAGE(#A#)
```

string BASE64DECODE(string base64)

Use: *HTML, JavaScript, Script*

Description: *The function returns a decoded string from the text string specified by the "base64" parameter.*

Example:

```
BASE64DECODE(YWJj)
// Returns "abc"
```

string BASE64ENCODE(string value)

Use: *HTML, JavaScript, Script*

Description: *The function returns an encoded string in the format "base64" from the text string specified by the "value" parameter.*

Example:

```
BASE64ENCODE(abc)
// Returns "YWJj"
```

string BUTTON(string title, string question, string button, string key)

Use: *View table*

Description: *The function displays a user-defined button in the view table (DataGrid). The title of the button is defined by the "title" parameter. The "question" parameter specifies the question that will be displayed in the "confirm" type confirmation window when the button is pressed. The "button" parameter represents the name of the hidden control button, which must be on the same view page or edit form as the view table with the displayed button. After pressing a button in the view table and then confirming the question, pressing the control button is programmatically invoked, which starts the script on the server side. The "key" parameter defines the name of the auxiliary form element that will be automatically created when the button is clicked, and the ID of the processed record will be stored in its value. This ID can subsequently be found in the script of the control button using the server function "REQUESTFORM(key)". To display the button, the "BUTTON" function call must be stored in the value of the "TextBox" control, set to "Read Only" and as "Hidden Field".*

Example:

```
// Default value of the "TextBox" control:
BUTTON(Delete, Are you sure you want to delete the record?, DeleteItem, abc)
// "Assign Value" command in the service button script:
A = "Integer(REQUESTFORM(abc))"
```

string CAPTCHA()

Use: *HTML in the edit form*

Description: *The function inserts the captcha verification code into the HTML code of the edit form, which is used to distinguish real users from robots. The function is linked to the "EVALCAPTCHA" function, without which the "CAPTCHA" function would not make sense.*

Example:

```
CAPTCHA()
```

double CEILING(double number)

Use: *HTML, Script*

Description: *The function returns the smallest integer that is greater than or equal to the value specified by the "number" parameter.*

Example:

```
CEILING(-1.23)
// Returns "-1"
CEILING(1.23)
```

```
// Returns "2"
```

string COLORLEGEND()

Use: *HTML*

Description: *The function returns the translation of the word "LEGEND" based on the language setting of the currently logged in user.*

Example:

```
COLORLEGEND()
```

string COLORLEGEND(string text)

Use: *HTML, JavaScript, Script*

Description: *The function returns a translation of the word "LEGEND (text)" based on the language setting of the currently logged in user, together with the text that defines the "text" parameter. This text can be multilingual.*

Example:

```
COLORLEGEND(Unpaid invoices#en:Unpaid)
```

string COLORLEGEND(string color, string text)

Use: *HTML, JavaScript, Script*

Description: *The function returns a rectangle filled with the color specified by the "color" parameter in the HEX format "#ffffff" with a text label specified by the "text" parameter. This text can be multilingual.*

Example:

```
COLORLEGEND(#44FF0A, Unpaid invoices#en:Unpaid)
```

string COLORLEGEND(int R, int G, int B, string text)

Use: *HTML, JavaScript, Script*

Description: *The function returns a rectangle filled with the color specified by the RGB parameters in decimal with a text label specified by the "text" parameter. This text can be multilingual.*

Example:

```
COLORLEGEND(255, 100, 100, Unpaid invoices#en:Unpaid)
```

string COLORLINK(string text, string url)

Use: *HTML, JavaScript, Script*

Description: *The function returns a colored arrow and the text specified by the "text" parameter with a link specified by the "url" parameter. If the link is clicked, the destination webpage will open in the current page frame.*

Example:

```
COLORLINK(abc, ViewPage.aspx?viewpage=878)
// Opens the view page with ID 878 in the current page frame
COLORLINK(abc, Form.aspx?form=396)
// Opens the edit form with ID 396 in the current page frame
```

string COLORLINK(string text, string url, bool center)

Use: *HTML, JavaScript, Script*

Description: *The function returns a colored arrow and the text specified by the "text" parameter with a link specified by the "url" parameter. The "center" parameter says that the link placed in the portlets will be center-aligned. If the link is clicked, the destination webpage will open in the current page frame.*

Example:

```
COLORLINK(abc, ViewPage.aspx?viewpage=878, true)
// Opens the view page with ID 878 in the current page frame
COLORLINK(abc, Form.aspx?form=396, true)
// Opens the edit form with ID 396 in the current page frame
```

string COLORLINK2(string text, string url)

Use: *HTML, JavaScript, Script*

Description: *The function returns a colored arrow and the text specified by the "text" parameter with a link specified by the "url" parameter. If the link is clicked, the destination webpage will open in a new window.*

Example:

```
COLORLINK2(abc, ViewPage.aspx?viewpage=878)
// Opens the view page with ID 878 in the current page frame
COLORLINK2(abc, Form.aspx?form=396)
// Opens the edit form with ID 396 in the current page frame
```

string COLORLINK2(string text, string url, bool center)

Use: *HTML, JavaScript, Script*

Description: *The function returns a colored arrow and the text specified by the "text" parameter with a link specified by the "url" parameter. The "center" parameter says that the link placed in the portlets will be center-aligned. If the link is clicked, the destination webpage will open in a new window.*

Example:

```
COLORLINK2(abc, ViewPage.aspx?viewpage=878, true)
// Opens the view page with ID 878 in the current page frame
COLORLINK2(abc, Form.aspx?form=396, true)
// Opens the edit form with ID 396 in the current page frame
```

void COLUMN(int columnWidth, int labelWidth, int defaultControlWidth)

Use: *HTML control name – The body of the control must be left blank*

Description: *The function starts a block of controls that are placed in columns and use a reverse design – the controls do not have a frame, have a colored background, and display the color bar below the control when focused. The total column width is defined by the "columnWidth" parameter. The width of the column with the control name is defined by the "labelWidth" parameter. When the value of the "labelWidth" parameter is set to "0", the column with the name of the control is not displayed, and the controls start from the left. The default size of the controls is defined by the "defaultControlWidth" parameter, and is important for the CheckBox and Radio controls, which are located in the reverse design in the same way as the other controls in the color box.*

Example:

```
COLUMN(500, 100, 150)
```

void /COLUMN

Use: *HTML control name – The body of the control must be left blank*

Description: *The function ends a block of controls that are placed in columns and use a reverse design.*

Example:

```
/COLUMN
```

string COMPANYID(string company)

Use: *HTML, JavaScript, Script*

Description: *The function returns the company name specified by the "company" parameter without the company type (eg "s.r.o", "a.s.", etc.), spaces, hyphens, commas and periods, and then converted to uppercase.*

Example:

```
COMPANYID(NetGenium s.r.o.)
// Returns "NETGENIUM"
```

string COMPAREDATE(DateTime value1, string operator, DateTime value2, string yes, string no)

Use: *HTML, JavaScript, Script*

Description: *The function returns the value of the "yes" parameter if the comparison of the value of the "value1" parameter with the value of the "value2" parameter is satisfied using the operator specified by the "operator" parameter. Otherwise, it returns the value of the "no" parameter.*

Example:

```
COMPAREDATE(#today#, =, #now#, Yes, No) // Returns No
COMPAREDATE(#today#, <, #now#, Yes, No) // Returns Yes
COMPAREDATE(#today#, <=, #now#, Yes, No) // Returns Yes
COMPAREDATE(#today#, >, #now#, Yes, No) // Returns No
COMPAREDATE(#today#, >=, #now#, Yes, No) // Returns No
```

string COMPAREDOUBLE(double value1, string operator, double value2, string yes, string no)

Use: *HTML, JavaScript, Script*

Description: *The function returns the value of the "yes" parameter if the comparison of the value of the "value1" parameter with the value of the "value2" parameter is satisfied using the operator specified by the "operator" parameter. Otherwise, it returns the value of the "no" parameter.*

Example:

```
COMPAREDOUBLE(1, =, 2, Yes, No) // Returns No
COMPAREDOUBLE(1, <, 2, Yes, No) // Returns Yes
COMPAREDOUBLE(1, <=, 2, Yes, No) // Returns Yes
COMPAREDOUBLE(1, >, 2, Yes, No) // Returns No
COMPAREDOUBLE(1, >=, 2, Yes, No) // Returns No
```

string CONTACT(int id)

Use: *HTML, JavaScript, Script*

Description: *The function returns the full name and e-mail address of the user specified by the "id" parameter.*

Example:

```
CONTACT(1)
// Returns "NetGenium s.r.o." <info@netgenium.com>"
CONTACT(#userid#)
```

string CONTACT(string loginname)

Use: *HTML, JavaScript, Script*

Description: *The function returns the full name and e-mail address of the user specified by the "loginname" parameter.*

Example:

```
CONTACT(NetGenium)
// Returns ""NetGenium s.r.o." <info@netgenium.com>"
CONTACT(#loginname#)
```

string CONTACTEMAIL(string contact)

Use: *HTML, JavaScript, Script*

Description: *The function returns the e-mail address of the contact specified by the "contact" parameter of the "e-mail address" type according to the RFC2822 specification.*

Example:

```
CONTACTEMAIL("NetGenium s.r.o." <info@netgenium.com>)
// Returns "info@netgenium.com"
```

string CONTACTNAME(string contact)

Use: *HTML, JavaScript, Script*

Description: *The function returns the name of the contact specified by the "contact" parameter of the "e-mail address" type according to the RFC2822 specification.*

Example:

```
CONTACTNAME("NetGenium s.r.o." <info@netgenium.com>)
// Returns "NetGenium s.r.o."
```

string CONTROL(string name, string html, string help)

Use: *HTML in the edit form*

Description: *The function generates the HTML code of the custom control in the edit form.*

Example:

```
CONTROL(Name, <INPUT type=text class=tb>, Enter your name)
```

void COMMIT()

Use: *Script (where a comment is expected)*

Description: *The function commits the currently executed transaction of the server script, and performs database operations of the following script commands, which are by default executed only after the successful execution of the entire script:*

- Save the variable back to the database
- Delete entry
- New record – immediately after starting the "COMMIT" function, the content of the variable "#nid#" will be filled
- Save the value to the database
- Store multiple values in a database

Example:

```
COMMIT()
```

int COPYATTACHMENT(int file)

Use: *Script*

Description: *The function creates a copy of the file attachment specified by the "file" parameter, and returns its ID.*

Example:

```
COPYATTACHMENT(#ng_file#)
```

int COPYATTACHMENT(int file, string directory)

Use: *Script*

Description: *The function creates a copy of the file attachment specified by the "file" parameter and saves it in the directory specified by the "directory" parameter. The function returns the value "0".*

Example:

```
COPYATTACHMENT(#ng_file#, C:\Directory)
```

double COS(double number)

Use: *HTML, Script*

Description: *The function returns the cosine of the value specified by the "number" parameter.*

Example:

```
COS(-1.23)
// Returns "0,334237727124503"
COS(1)
// Returns "0,540302305868139"
COS(#A#)
```

int COUNT(object[] array)

Use: *Script*

Description: *The function returns the number of values of the script variable specified by the "array" parameter, if the variable is an array of values. If the variable is not an array of values, the function returns the value "1".*

Example:

```
COUNT(#A#)
```

string CRC16(string value1 [, string value2])

Use: *HTML, JavaScript, Script*

Description: *The function returns "CRC16" the sum of all values specified in the function parameters. When determining the sum, the values are first separated by a hyphen.*

Example:

```
CRC16(a, b, c)
// Returns "27651"
```

int CREATEATTACHMENT(string base64)

Use: *Script*

Description: *The function creates a file attachment from the image specified by the "base64" parameter. The function accepts the encoded string specified by the "base64" parameter, and returns the file attachment number of the created image.*

Example:

```
CREATEATTACHMENT(#ng_kod#)
CREATEATTACHMENT(
4nOzd+1tSWRs38PePW7oVJVEUUQwhD5hKkhTlidIOFmlROqRFqWVZmuWkWZZmWWRalmWZlmVDWc4YZY8TM...)
```

int CREATEATTACHMENT(string code, string type, int width, int height, bool label)

Use: *Script*

Description: *The function creates a file attachment from the barcode specified by the "code" parameter. The barcode type specified by the "type" parameter can take one of the following values:*

- CODABAR
- CODE128
- CODE128A
- CODE128B
- CODE128C
- CODE39
- CODE93
- EAN13
- EAN8
- UPCA
- UPCE
- ITF14
- QRCODE

The width of the image is defined by the "width" parameter, height by the "height" parameter. The "label" parameter specifies whether a barcode should be displayed in Latin at the bottom of the image. An error may occur during image generation due to a barcode type mismatch with its content. These are mainly illegal characters or the length of the barcode. In this case, there is no interruption, but the error is listed in the final image itself in red on a white background. The function returns the file attachment number of the created image.

Example:

```
CREATEATTACHMENT(abc, CODE128A, 200, 100, false)
```

void CREATEATTACHMENTTEST(string base64)

Use: *Script*

Description: *The function verifies the source data of the image specified by the "base64" parameter. The function accepts the encoded string specified by the "base64" parameter, and returns "OK" if the encoded string has the correct form. Otherwise, the function throws an exception.*

Example:

```
CREATEATTACHMENTTEST(#ng_kod#)
CREATEATTACHMENTTEST(
EQVR4nOzd+1tSWRs38PePoVJVEUUQwhD5hKkhTlidIOFmlROqRFqWVZmuWkWZZmWWRalmWZlmVDWc4YZY8TM...)
```

void CREATEATTACHMENTTEST(string code, string type, int width, int height, bool label)

Use: *Script*

Description: *The function verifies the input parameters of the barcode specified by the "code" parameter. The barcode type specified by the "type" parameter can take one of the following values:*

- CODABAR
- CODE128
- CODE128A
- CODE128B
- CODE128C
- CODE39
- CODE93
- EAN13
- EAN8
- UPCA
- UPCE
- ITF14
- QRCODE

The width of the image is defined by the "width" parameter, height by the "height" parameter. The "label" parameter specifies whether a barcode should be printed in Latin at the bottom of the image. The function returns the value "OK" if the input parameters have valid values. Otherwise, the function throws an exception.

Example:

```
CREATEATTACHMENTTEST(abc, CODE128A, 200, 100, false)
```

int CREATEPDFATTACHMENT(string attachments, string filename)

Use: *Script*

Description: *The function creates a PDF file by combining multiple file attachments with the file name specified by the "filename" parameter. File attachments are specified by the "attachments" parameter as a semicolon-separated list of file attachment IDs. The function returns the file attachment number of the created PDF file.*

Example:

```
CREATEPDFATTACHMENT(1;2;3, abc.pdf)
```

void CREATETEXTFILE(string text, string path)

Use: *Script (where a comment is expected)*

Description: *The function creates a text file specified by the "path" parameter on the disk and writes the text defined by the "text" parameter in the default server encoding.*

Example:

```
CREATETEXTFILE(text, C:\Directory\file.txt)
```

int CREATEZIPATTACHMENT(string attachments, string filename)

Use: *Script*

Description: *The function creates a ZIP archive from file attachments with the archive name specified by the "filename" parameter. File attachments are specified by the "attachments" parameter as a semicolon-separated list of file attachments. The function returns the file attachment number of the created ZIP archive.*

Example:

```
CREATEZIPATTACHMENT(1;2;3, abc.zip)
```

DateTime DATE

Use: *HTML, JavaScript, Script*

Description: *The function returns the date (without hours, minutes, etc.) from the date specified by the "date" parameter.*

Example:

```
DATE(15.2.2013 12:34)
// Returns "15.2.2013"
DATE(#now#)
```

long DATE2LONG(DateTime date)

Use: *HTML, JavaScript, Skript*

Description: *The function returns the date value specified by the "date" parameter converted to a numeric value of type "long".*

Example:

```
DATE2LONG(1.1.2000)
// Returns "630822816000000000"
```

DateTime[] DateArray(DateTime date1; DateTime date2; …)

Use: *Script*

Description: *The function declares a script variable of the value field type "DateTime" from individual values specified by the parameter "date1", "date2", etc. The function parameters are separated by a semicolon or a tab. The "DateArray" function call must be case sensitive in the function name.*

Example:

```
DateArray(#now#;#today#)
```

string DATECONDITION(string ng_date, DateTime date1, DateTime date2)

Use: *HTML, Script*

Description: *The function returns the SQL code of the condition that is used to compare whether the column named "ng_date" (the content of the column is the date) lies between the data specified "date1" and "date2".*

Example:

```
To use in a script designer query condition:
[(ID)] [equals] [OK#crlf#DATECONDITION(ng_from, ng_to, #ng_date#, #ng_to#)]
[(ID)] [equals] [OK#crlf#DATECONDITION(ng_from, ng_to, #ng_date:history#, #ng_to:history#)
There must be no space before OK#crlf#

DATECONDITION(ng_date, 10.6.2013, 10.6.2013)
// Returns "(ng_date >= '2013-06-10' AND ng_date < '2013-06-11')"
DATECONDITION(ng_date, 10.6.2013, 10.6.2013 18:30)
// Returns "(ng_date >= '2013-06-10' AND ng_date < '2013-06-10 18:30')"
DATECONDITION(ng_date, 10.6.2013 15:00, 10.6.2013)
// Returns "(ng_date = '2013-06-10' OR (ng_date >= '2013-06-10 15:00' AND ng_date < '2013-06-
11'))"
DATECONDITION(ng_date, 10.6.2013 15:00, 10.6.2013 18:30)
```

```
// Returns "(ng_date = '2013-06-10' OR (ng_date >= '2013-06-10 15:00' AND ng_date < '2013-06-10 18:30'))"
```

string DATECONDITION(string ng_from, string ng_to, DateTime date1, DateTime date2)

Use: *HTML, Script*

Description: *The function returns the SQL code of the condition that is used to compare whether the date interval specified by the parameters (database columns) "ng_from" and "ng_to" overlaps with the interval specified by the parameters "date1" and "date2".*

Example:

```
DATECONDITION(ng_from, ng_to, 10.6.2013, 10.6.2013)
// Returns "((ng_from >= '2013-06-10' AND ng_from < '2013-06-11' AND ng_to IS NULL) OR (ng_from < '2013-06-11' AND ng_to >= '2013-06-10'))"
DATECONDITION(ng_from, ng_to, 10.6.2013, 10.6.2013 18:30)
// Returns "((ng_from >= '2013-06-10' AND ng_from < '2013-06-10 18:30' AND ng_to IS NULL) OR (ng_from < '2013-06-10 18:30' AND ng_to >= '2013-06-10'))"
DATECONDITION(ng_from, ng_to, 10.6.2013 15:00, 10.6.2013)
// Returns "(((ng_from = '2013-06-10' OR (ng_from >= '2013-06-10 15:00' AND ng_from < '2013-06-11')) AND ng_to IS NULL) OR (ng_from < '2013-06-11' AND (ng_to = '2013-06-10' OR ng_to >= '2013-06-10 15:00')))"
DATECONDITION(ng_from, ng_to, 10.6.2013 15:00, 10.6.2013 18:30)
// Returns "(((ng_from = '2013-06-10' OR (ng_from >= '2013-06-10 15:00' AND ng_from < '2013-06-10 18:30')) AND ng_to IS NULL) OR (ng_from < '2013-06-10 18:30' AND (ng_to = '2013-06-10' OR ng_to >= '2013-06-10 15:00')))"
```

string DATECONDITION2(string ng_from, string ng_to, DateTime date1, DateTime date2)

Use: *HTML, Script*

Description: *The function returns the SQL code of the condition that is used to compare whether the date interval specified by the parameters (database columns) "ng_from" and "ng_to" overlaps with the interval specified by parameters "date1" and "date2", including the interval before the date "ng_from". The function finds use when comparing, for example, the validity of prices – the price is valid from and until a certain date, but the validity can also be defined only by date from ("ng_to" is NULL), so with an unlimited validity period (price is valid from a certain date until present).*

Example:

```
DATECONDITION2(ng_from, ng_to, 10.6.2013, 10.6.2013)
// Returns "((ng_from < '2013-06-12' AND ng_to IS NULL) OR (ng_from < '2013-06-12' AND ng_to >= '2013-06-10'))"
DATECONDITION2(ng_from, ng_to, 10.6.2013 15:00, 10.6.2013)
```

```
// Returns "((ng_from < '2013-06-12' AND ng_to IS NULL) OR (ng_from < '2013-06-12' AND (ng_to
= '2013-06-11' OR ng_to >= '2013-06-10 15:00')))"
DATECONDITION2(ng_from, ng_to, 10.6.2013, 10.6.2013 18:30)
// Returns "((ng_from < '2013-06-11 18:30' AND ng_to IS NULL) OR (ng_from < '2013-06-11 18:30'
AND ng_to >= '2013-06-10'))"
DATECONDITION2(ng_from, ng_to, 10.6.2013 15:00, 10.6.2013 18:30)
// Returns "((ng_from < '2013-06-11 18:30' AND ng_to IS NULL) OR (ng_from < '2013-06-11 18:30'
AND (ng_to = '2013-06-11' OR ng_to >= '2013-06-10 15:00')))"
```

int DAYOFWEEK(DateTime date)

Use: *HTML, JavaScript, Skript*

Description: *The function returns the number of the current day of the week (1-7).*

Example:

```
DAYOFWEEK(15.2.2024)
// Returns „4"
DAYOFWEEK(#today#)
```

int DAYOFYEAR(DateTime date)

Use: *HTML, JavaScript, Skript*

Description: *The function returns the number of the current day of the year (1-365).*

Example:

```
DAYOFYEAR(15.2.2024)
// Returns „44"
DAYOFYEAR(#today#)
```

void DELETEDATA(string dbname, string condition)

Use: *Script (where a comment is expected)*

Description: *The function permanently deletes data from the database table specified by the "dbname" parameter under the conditions specified by the "condition" parameter. This function does not use history writing, and is optimized for mass data deletion that would otherwise fail at the timeout if run directly on the database, for example using "Microsoft SQL Server Management Studio".*

Example:

```
DELETEDATA(ng_form, id < 123)
DELETEDATA(ng_form, date < FORMATDATESQL(#today#))
```

void DELETEHISTORY()

Use: *Script (where a comment is expected)*

Description: *The function deletes the history of all edit forms from the database*

- older than 6 months,
- older than X months if X is specified in the edit form settings.

Example:

```
DELETEHISTORY()
```

void DELETEHISTORY(string dbname)

Use: *Script (where a comment is expected)*

Description: *The function deletes the history of the edit form specified by the "dbname" parameter from the database*

- older than 6 months,
- older than X months if X is specified in the edit form settings.

Example:

```
DELETEHISTORY(susers)
```

void DELETEHISTORY(string dbname, string condition)

Use: *Script (where a comment is expected)*

Description: *The function deletes from the database the history of the editing form specified by the "dbname" parameter under the conditions specified by the "condition" parameter. This function is optimized for mass data deletion that would otherwise fail at the timeout if run directly on the database, for example using "Microsoft SQL Server Management Studio".*

Example:

```
DELETEHISTORY(susers, changedid = #id#)
```

void DISABLEAUTOFILL()

Use: *Note of the edit form*

Description: *The function disables the web browser from automatically filling out forms for TextBox controls.*

Example:

```
DISABLEAUTOFILL()
```

void DISABLEFORM()

Use: *Script*

Description: *The function disables all controls located in the edit form, including the "Save" and "Delete" buttons. Using the "DISABLEFORM" function has the same effect from the users' point of view as using the "controls_Disable" javascript function, but it is safer for "File" and "Image" controls. By default, these controls contain buttons for uploading file attachments from the client station to the server – the server function "DISABLEFORM" safely removes these buttons, while the javascript function "controls_Disable" only hides them.*

Example:

```
DISABLEFORM()
```

void DISABLEPAGETOOLS()

Use: *Note of the view page or edit form*

Description: *The function disables the floating icons on the view page or in the edit form that appear in the upper right corner.*

Example:

```
DISABLEPAGETOOLS()
```

double DISTANCE(double lat1, double lon1, double lat2, double lon2)

Use: *Script*

Description: *The function returns the distance in kilometers between two geographical places specified by the parameters "lat1", "lon1", "lat2" and "lon2". The individual parameters represent the geographical coordinates of the places.*

Example:

```
DISTANCE(14.108998537, 50.0723726721, 14.4476009458, 50.0794492033)
// Returns "2.859"
```

void DIV(string id)

Use: *HTML control name – The body of the control must be left blank*

Description: *The function generates an HTML tag " ", Resp. launch a new wonder. The tag ID defines the "id" parameter.*

Example:

```
DIV(D1)
```

void DIV(string id, bool visible)

Use: *HTML control name – The body of the control must be left blank*

Description: *The function generates an HTML tag " ", Resp. launch a new wonder. The tag ID defines the "id" parameter, its visibility the "visible" parameter.*

Example:

```
DIV(D1, false)
```

void /DIV

Use: *HTML control name – The body of the control must be left blank*

Description: *The function generates an HTML tag " ", Resp. the end of the previous wonder generated by the "DIV" function.*

Example:

```
/DIV
```

double[] DoubleArray(double number1; double number2;…)

Use: *Script*

Description: *The function declares a script variable of the value field type "Double" from individual values specified by the parameter "number1", "number2", etc. The function parameters are separated by a semicolon or a tab. The "DoubleArray" function call must be case sensitive in the function name.*

Example:

```
DoubleArray(1.1;2.2;3.3)
```

string EMAIL(int id)

Use: *HTML, JavaScript, Script*

Description: *The function returns the e-mail address of the user specified by the "id" parameter.*

Example:

```
EMAIL(1)
```

```
EMAIL(#userid#)
```

string EMAIL(string loginname)

Use: *HTML, JavaScript, Script*

Description: *The function returns the e-mail address of the user specified by the "loginname" parameter.*

Example:

```
EMAIL(Administrator)
EMAIL(#loginname#)
```

string EMAILS(int id1 [, int id2])

Use: *HTML, JavaScript, Script*

Description: *The function returns a semicolon-separated list of user email addresses whose ID is specified in the function parameters "id1", "id2", and so on.*

Example:

```
EMAILS(1, 2, 3)
```

string EMAILS(string loginname1 [, string loginname2])

Use: *HTML, JavaScript, Script*

Description: *The function returns a semicolon-separated list of user email addresses whose login name is specified in the function parameters "loginname1", "loginname2", and so on.*

Example:

```
EMAILS(loginname1, loginname2, loginname3)
```

string[] EmailArray(string email1; string email2; …)

Use: *Script*

Description: *The function declares a script variable of the field type of values "String" from individual values specified by parameters "email1", "email2", etc. The function parameters are separated by a semicolon or a tab. The "EmailArray" function call must be case sensitive in the function name. The "EmailArray" function is used for the e-mail list because the contact specified in the e-mail address may contain a semicolon.*

Example:

```
EmailArray("NetGenium; s.r.o." <info@netgenium.com>;"NetGenium; s.r.o." <info@netgenium.com>)
```

string EMAILLINK(string dbname, int id)

Use: *HTML, JavaScript, Script*

Description: *The function returns an HTML reference to the database record specified by the "dbname" and "id" parameters. The "dbname" parameter specifies the identifier of the edit form, the "id" parameter is the primary key of the database record. The function is used in the text of the e-mail in which the reference to the database record needs to be placed.*

Example:

```
EMAILLINK(susers, 1)
```

string EQUALS(string compare1, string compare2, string yes, string no)

Use: *HTML, JavaScript, Script*

Description: *The function returns the value of the "yes" parameter if the value of the "compare1" parameter is equal to the value of the "compare2" parameter. Otherwise, it returns the value of the "no" parameter.*

Example:

```
EQUALS(#loginname#, Administrator, 0, -1)
```

void EVALATTACHMENTSTATUS(int file, string status, string message)

Use: *Script*

Description: *The function determines whether the file attachment specified by the "file" parameter acquires the state defined by the "status" parameter. If the attachment is in one of the defined states, the function invokes a script interrupt with the message defined in the "message" parameter. A typical use is when scanning an attachment with an antivirus system.*

Example:

```
EVALATTACHMENTSTATUS(ng_attachment, 0, Antivirus scan of attached documents is currently in progress. Please resubmit the form in a few seconds)
EVALATTACHMENTSTATUS(ng_ attachment, 1, A viral infection was found in the attached XYZ document. Delete this attachment or use another one)
```

void EVALCAPTCHA(string captcha, string message)

Use: *Script*

Description: *The function tests whether the text entered by the "captcha" parameter corresponds to the displayed captcha verification code, which was displayed in the edit form using the "CAPTCHA" function. If the texts do not match, the function interrupts the script with the error message specified by the "message" parameter. The function is linked to the "CAPTCHA" function, without which the "EVALCAPTCHA" function would not make sense.*

Example:

```
EVALCAPTCHA(#ng_verificationcode#, Please type the verification code from the image.)
```

void EVALGOOGLECAPTCHA(string secretKey, string message)

Use: *Script*

Description: *The function tests whether the control that was displayed in the edit form using the "GOOGLECAPTCHA" or "GOOGLECAPTCHAINVISIBLE" function is checked. The first parameter is the unique key "secretKey", which can be obtained together with the key "siteKey" when registering the Google reCAPTCHA element with Google. If the control is not checked, the function triggers a script with an error message specified by the "message" parameter. The function is linked to the "GOOGLECAPTCHA" function, without which the "EVALGOOGLECAPTCHA" function would not make sense.*

Example:

```
EVALGOOGLECAPTCHA(d5f545g5ccd5f2c5df1, Please check the CAPTCHA box.)
```

double EXP(double number)

Use: *HTML, Script*

Description: *The function returns the value of the Euler number raised to the power specified by the "number" parameter.*

Example:

```
EXP(1.23)
// Returns "3,42122953628967"
EXP(#A#)
```

string EXPORTIMAGES(string html)

Use: *Script*

Description: *The function exports inline images from the HTML code specified by the "html" parameter, and replaces all image URL links ("Download.aspx? Abc") from the HTML code with a string of image data encoded with "base64".*

Example:

```
A = String
A = EXPORTIMAGES(#ng_html#)
```

string EXPORTIMAGES(string html, int maxWidth, int maxHeight)

Use: *Script*

Description: *The function exports inline images from the HTML code specified by the "html" parameter, and replaces all image URL links ("Download.aspx? Abc") from the HTML code with a string of image data encoded with "base64". For all encoded images, the "width" and "height" parameters will be set proportionally depending on the "maxWidth" and "maxHeight" parameters.*

Example:

```
A = String
A = EXPORTIMAGES(#ng_html#, 20, 15)
```

void EXPORTIMAGES(string html, string dir, string url)

Use: *Script*

Description: *The function exports inline images from the HTML code specified by the "html" parameter to the directory on the disk specified by the "dir" parameter. Images will be named in the order in which they are used sequentially in the HTML code. The first image will be named eg "img01.jpg", the second "img02.jpg" etc. The "url" parameter is a publicly available URL from which to download these images.*

Example:

```
A = String(#guid#)
EXPORTIMAGES(#ng_html#, #rootpath#Temp\#A#, https://www.netgenium.com/temp/#A#)
```

object FIRST(object[] array)

Use: *Script*

Description: *The function returns the first value of a script variable specified by the "array" parameter if the variable is an array of values. If the variable is not an array of values, then the function returns the value of the variable. If the value field does not contain any element, the function returns an empty string.*

Example:

```
FIRST(#A#)
```

DateTime FIRSTDAYINMONTH(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns the date of the first day of the month from the date specified by the "date" parameter.*

Example:

```
FIRSTDAYINMONTH(15.2.2013)
// Returns "01.02.2013"
FIRSTDAYINMONTH(#today#)
```

DateTime FIRSTDAYINQUARTAL(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns the date of the first day of the quarter from the date specified by the "date" parameter.*

Example:

```
FIRSTDAYINQUARTAL(15.2.2013)
// Returns "01.01.2013"
FIRSTDAYINQUARTAL(#today#)
```

DateTime FIRSTDAYINWEEK(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns the date of the first day of the week from the date specified by the "date" parameter.*

Example:

```
FIRSTDAYINWEEK(15.2.2013)
// Returns "11.02.2013"
FIRSTDAYINWEEK(#today#)
```

DateTime FIRSTDAYINYEAR(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns the date of the first day of the year from the date specified by the "date" parameter.*

Example:

```
FIRSTDAYINYEAR(15.2.2013)
// Returns "01.01.2013"
FIRSTDAYINYEAR(#today#)
```

double FLOOR(double number)

Use: *HTML, Script*

Description: *The function returns the highest integer that is less than or equal to the value specified by the "number" parameter.*

Example:

```
FLOOR(-1.23)
// Returns "-2"
FLOOR(1.23)
// Returns "1"
```

string FORMATCURRENCY(double number)

Use: *HTML, JavaScript, Script*

Description: *The function formats the number specified by the "number" parameter into text – with a measure as a thousands separator, with a decimal point or a dot, depending on the language setting of the currently logged in user, and rounded to two decimal places.*

Example:

```
FORMATCURRENCY(1100)
// Returns "1 100,00"
FORMATCURRENCY(1100.5)
// Returns "1 100,50"
FORMATCURRENCY(1100.5412)
// Returns "1 100,54"
```

string FORMATCURRENCY(double number, string symbol)

Use: *HTML, JavaScript, Script*

Description: *The function formats the number specified by the "number" parameter into text – with a measure as a thousands separator, with a decimal point or a dot, depending on the language setting of the currently logged in user, rounded to two decimal places, and with a symbol specified by the "symbol" parameter.*

Example:

```
FORMATCURRENCY(1100, Kč)
// Returns "1 100,00 Kč"
FORMATCURRENCY(1100.5, Kč)
// Returns "1 100,50 Kč"
FORMATCURRENCY(1100.5412, Kč)
// Returns "1 100,54 Kč"
```

string FORMATDATATABLE(string query)

Use: *HTML, JavaScript, Script*

Description: *The function returns the result of a query specified by the "query" parameter formatted in a table.*

Example:

```
FORMATDATATABLE(SELECT * FROM susers ORDER BY id)
FORMATDATATABLE(SELECT * FROM susers WHERE loginname = FORMATSTRINGSQL(#loginname#) ORDER BY
id)
```

string FORMATDATATABLE(string query, string title)

Use: *HTML, JavaScript, Script*

Description: *The function returns the result of a query specified by the "query" parameter formatted in a table, and with a title specified by the "title" parameter.*

Example:

```
FORMATDATATABLE(SELECT * FROM susers ORDER BY id, Users)
FORMATDATATABLE(SELECT * FROM susers WHERE loginname = FORMATSTRINGSQL(#loginname#) ORDER BY
id, Current user)
```

string FORMATDATAVIEW(string query)

Use: *HTML, JavaScript, Script*

Description: *The function returns the result of a query specified by the "query" parameter formatted in a table.*

Example:

```
FORMATDATAVIEW(SELECT * FROM susers ORDER BY id)
FORMATDATAVIEW(SELECT * FROM susers WHERE loginname = FORMATSTRINGSQL(#loginname#) ORDER BY
id)
```

string FORMATDATAVIEW(string query, string title)

Use: *HTML, JavaScript, Script*

Description: *The function returns the result of a query specified by the "query" parameter formatted in a table, and with a title specified by the "title" parameter.*

Example:

```
FORMATDATAVIEW(SELECT * FROM susers ORDER BY id, Users)
FORMATDATAVIEW(SELECT * FROM susers WHERE loginname = FORMATSTRINGSQL(#loginname#) ORDER BY
id, Current user)
```

string FORMATDATE(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function formats the date specified by the "date" parameter into text based on the default template "D" and the language setting of the currently logged in user.*

Example:

```
FORMATDATE(#today#)
// Returns for example "Tuesday 2. April 2019"
```

string FORMATDATE(DateTime date, string template)

Use: *HTML, JavaScript, Script*

Description: *The function formats the date specified by the "date" parameter into text based on the template defined by the "template" parameter and the language setting of the currently logged in user. Templates include:*

- "d": 29.2.2000
- "D": 29. February 2000
- "f": 29. February 2000 12:34
- "F": 29. February 2000 12:34:00 PM
- "g": 29.2.2000 12:34
- "G": 29.2.2000 12:34:00
- "r": Tue, 29 Feb 2000 12:34:00 GMT
- "s": 2000-02-29T12:34:00
- "t": 12:34
- "T": 12:34:00
- "y": February 2000
- "dddd, MMMM dd yyyy": Tuesday, February 29 2000
- "dd-MM-yy": 29-02-00
- "w": 9
- "ww": 09
- "www": 2000 / 09

Example:

```
FORMATDATE(#today#, dd.MM.yyyy)
FORMATDATE(#today#, s)
```

string FORMATDATES(string value, bool dayPrefix, boll weekNumber)

Use: *HTML, JavaScript, Script*

Description: *The function replaces all occurrences of dates in the text string specified by the "value" parameter with a formatted value into text based on the default template "dd.MM.yyyy" and the language setting of the currently logged in user. The "dayPrefix" parameter indicates whether an abbreviated name of the day is inserted before each date. The "weekNumber" parameter indicates whether a week number is inserted after each date.*

Example:

```
FORMATDATES(abc #today# def, true, true)
// Returns "abc fri 14. 11. 2014 (46th week) def"
FORMATDATES(abc #today# def, true, false)
// Returns "abc fri 14. 11. 2014 def"
FORMATDATES(abc #today# def, false, true)
```

```
// Returns "abc 14. 11. 2014 (46th week) def"
```

string FORMATDATESQL(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function formats the date specified by the "date" parameter into a text form intended for use in a database query condition, for example in conjunction with the "SQL" function, and based on the habits of the current database.*

Example:

```
FORMATDATESQL(#today#)
SQL(SELECT name FROM sholiday WHERE date_ = FORMATDATESQL(1.1.2019))
```

string FORMATDIFFERENCES(string compare1, string compare2)

Use: *HTML, JavaScript, Script*

Description: *The function returns a table with two columns, where in the first column the values separated by a semicolon from the parameter "compare1" are displayed on individual rows, and in the second column the values from the parameter "compare2" are displayed in the same way. Each row of this table is compared, and if the values on each row are not the same, they are highlighted in red. The first column is named "A", the second "B".*

Example:

```
FORMATDIFFERENCES(Text1;Secondtext2, Text1;Text2)
```

string FORMATDIFFERENCES(string compare1, string compare2, string header1, string header2)

Use: *HTML, JavaScript, Script*

Description: *The function returns a table with two columns, where in the first column the values separated by a semicolon from the parameter "compare1" are displayed on individual rows, and in the second column the values from the parameter "compare2" are displayed in the same way. Each row of this table is compared, and if the values on each row are not the same, they are highlighted in red. The first column has a name defined by the "header1" parameter, the second by the "header2" parameter.*

Example:

```
FORMATDIFFERENCES(Text1;Secondtext2, Text1;Text2, First, Second)
```

string FORMATDIFFERENCES(string compare1, string compare2, string header1, string header2, string title)

Use: *HTML, JavaScript, Script*

Description: *The function returns a table with two columns, where in the first column the values separated by a semicolon from the parameter "compare1" are displayed on individual rows, and in the second column the values from the parameter "compare2" are displayed in the same way. Each row of this table is compared, and if the values on each row are not the same, they are highlighted in red. The first column has a name defined by the "header1" parameter, the second by the "header2" parameter. The entire table has a title defined by the "title" parameter.*

Example:

```
FORMATDIFFERENCES(Text1;Secondtext2, Text1;Text2, First, Second, Differences)
```

string FORMATDOUBLE(double number)

Use: *HTML, JavaScript, Script*

Description: *The function formats the number specified by the "number" parameter into text – with a scale as a thousands separator, and with a decimal point or a dot, depending on the language setting of the currently logged in user.*

Example:

```
FORMATDOUBLE(1100)
// Returns "1 100"
FORMATDOUBLE(1100.5)
// Returns "1 100,5"
FORMATDOUBLE(1100.5412)
// Returns "1 100,5412"
```

string FORMATDOUBLE(double number, string symbol)

Use: *HTML, JavaScript, Script*

Description: *The function formats the number specified by the "number" parameter into text – with a scale as a thousands separator, with a decimal point or a dot, depending on the language setting of the currently logged in user, and with a symbol specified by the "symbol" parameter.*

Example:

```
FORMATDOUBLE(1100, kg)
// Returns "1 100 kg"
FORMATDOUBLE(1100.5, kg)
// Returns "1 100,5 kg"
FORMATDOUBLE(1100.5412, kg)
// Returns "1 100,5412 kg"
```

string FORMATDOUBLESQL(double number)

Use: *HTML, JavaScript, Script*

Description: *The function formats the number specified by the "number" parameter into a text form intended for use in a database query condition, for example in conjunction with the "SQL" function, and based on the habits of the current database.*

Example:

```
SQL(SELECT COUNT(*) FROM sstatistics2 WHERE session > FORMATDOUBLESQL(1000.1234))
```

string FORMATINTARRAYSQL(string array)

Use: *HTML, JavaScript, Script*

Description: *The function formats the semicolon-separated list of non-zero numbers specified by the "number" parameter into a text format intended for use in a database query condition, for example in conjunction with the "SQL" function, and based on current database conventions. If the list of numbers does not contain any non-zero numeric value, the value "0" is returned.*

Example:

```
SQL(SELECT loginname FROM susers WHERE id IN (FORMATINTARRAYSQL(1;2)))
```

string FORMATINTSQL(int number)

Use: *HTML, JavaScript, Script*

Description: *The function formats the number specified by the "number" parameter into a text form intended for use in a database query condition, for example in conjunction with the "SQL" function, and based on the habits of the current database.*

Example:

```
SQL(SELECT loginname FROM susers WHERE id = FORMATINTSQL(1))
```

string FORMATLONGARRAYSQL(string array)

Use: *HTML, JavaScript, Script*

Description: *The function formats the semicolon-separated list of nonzero large numbers specified by the "number" parameter into a text format intended for use in a database query condition, for example, in conjunction with the "SQL" function, and based on current database conventions. If the list of numbers does not contain any non-zero numeric value, the value "0" is returned.*

Example:

```
SQL(SELECT loginname FROM susers WHERE id IN (FORMATLONGARRAYSQL(1;2)))
```

string FORMATLONGSQL(long number)

Use: *HTML, JavaScript, Script*

Description: *The function formats the number specified by the "number" parameter into a text form intended for use in a database query condition, for example in conjunction with the "SQL" function, and based on the habits of the current database.*

Example:

```
SQL(SELECT loginname FROM susers WHERE id = FORMATLONGSQL(1))
```

string FORMATNUMBERINWORDS(double number)

Use: *HTML, JavaScript, Script*

Description: *The function formats the number specified by the "number" parameter into the text form indicated by the words, mathematically rounded, and in the language of the currently logged-in user.*

Example:

```
FORMATNUMBERINWORDS(123456)
// Returns "one hundred and twenty three thousand four hundred and fifty six"
FORMATNUMBERINWORDS(123456.5)
// Returns "one hundred and twenty three thousand four hundred and fifty six"
```

string FORMATSTRING(string template, string variable0,…[string variableXY])

Use: *HTML, JavaScript, Script*

Description: *The function formats the text specified by the "template" parameter according to compound formatting conventions in C#. Variables whose occurrence will be replaced by variable values are declared in braces and are numbered from zero.*

Example:

```
FORMATSTRING({0} + {1} = {2}, 1, 2, 3)
// Returns "1 + 2 = 3"
```

string FORMATSTRINGSQL(string text)

Use: *HTML, JavaScript, Script*

Description: *The function formats the text string specified by the "text" parameter into a text form intended for use in a database query condition, for example in conjunction with the "SQL" function, and based on current database conventions – starting with "N" in the case of MSSQL database. escaped apostrophes that occur within the text after double apostrophes.*

Example:

```
SQL(SELECT id FROM susers WHERE loginname = FORMATSTRINGSQL(Administrator))
```

string FORMATTABLE(string column1, string values1, [string column2, string values2 ...])

Use:

Description: *The function returns the HTML code of a table formatted in a universal appearance intended for e-mail messages, for example. The "column1" parameter defines the name of the first column of the table, the "values1" parameter defines the semicolon-separated values specified for this table column. The number of columns in the table is not limited, you only need to specify other parameters of the function in pairs – the column name together with the values in this column. A semicolon is used as a value separator, so individual values in a column cannot contain a semicolon as a substring.*

Example:

```
FORMATTABLE(A, 1;2;3, B, 4;5;6)
FORMATTABLE(A, #A#, B, #B#)
```

A	B
1	4
2	5
3	6

string FORMATTIME(int minutes)

Use: *HTML, JavaScript, Script*

Description: *The function formats the number of minutes specified by the "minutes" parameter into text in the format "HH:mm".*

Example:

```
FORMATTIME(1156)
// Returns "19:16"
FORMATTIME(-150)
// Returns "-02:30"
```

string FORMATTIME2(double seconds)

Use: *HTML, JavaScript, Script*

Description: *The function formats the number of seconds specified by the "seconds" parameter into a text format in the format "HH:mm:ss".*

Example:

```
FORMATTIME2(1156)
// Returns "00:16:40"
FORMATTIME2(-150)
// Returns "-00:02:30"
```

string FORMNAME(int form, string separator)

Use: *HTML, JavaScript, Script*

Description: *The function returns the full name of the edit form specified by the "form" parameter, including the name of the application group and application. The text string specified by the "separator" parameter is used as a separator between these three names.*

Example:

```
FORMNAME(#form#, " | ")
```

string GANTT(DateTime from, DateTime to, int R, int G, int B)

Use: *View table*

Description: *The function displays the Gantt chart on the row of the view table (DataGrid). The start date of the diagram is determined by the "from" parameter, the end date is determined by the "to" parameter. The color of the diagram is determined by the color values of the parameters "R", "G" and "B". To display the diagram, the call to the "GANTT" function must be stored in the value of the "TextBox" control, set to "Read Only" and as "Hidden Field".*

Example:

```
GANTT(01.01.2013, 01.02.2013, 25, 25, 112)
```

string GETCOOKIE(string key)

Use: *HTML, JavaScript, Script*

Description: *The function returns the cookie value defined by the "key" parameter.*

Example:

```
GETCOOKIE(abc)
```

string GOOGLECAPTCHA(string siteKey)

Use: *HTML in the edit form*

Description: *The function inserts the Google reCAPTCHA verification code into the HTML code of the edit form, which is used to distinguish real users from robots. The embedded verification code contains a unique domain key specified by the "siteKey" parameter, which can be obtained by registering Google reCAPTCHA with Google. The function is linked to the "EVALGOOGLECAPTCHA" function, without which the "GOOGLECAPTCHA" function would not make sense.*

Example:

```
GOOGLECAPTCHA(5ghjhddfg54cddry5tvggfffg)
```

string GOOGLECAPTCHAINVISIBLE(string siteKey, string button, [string url, string position])

Use: *HTML in the edit form*

Description: *The function inserts the Google reCAPTCHA verification code into the HTML code of the edit form, which is used to distinguish real users from robots. The embedded verification code contains a unique domain key specified by the "siteKey" parameter, which can be obtained by registering Google reCAPTCHA with Google. The "button" parameter is the identifier of the hidden button that contains the "EVALGOOGLECAPTCHA" function. The button runs on the server side when processing a response from Google. The optional "url" parameter is the address of the Google endpoint that returns the responses. The optional parameter "position" can take the values bottomright, bottomleft and inline. The function is tied to the "EVALGOOGLECAPTCHA" function, without which the "GOOGLECAPTCHAINVISIBLE" function would not make sense.*

Example:

```
GOOGLECAPTCHAINVISIBLE(5ghjhddfg54cddry5tvggfffg, BT123)
```

object[] GROUP(object[] array)

Use: *Script*

Description: *The function declares a new value field type script variable from the value field type script variable in which duplicates are removed. If the variable is not an array of values, then the function returns the value of the variable.*

Example:

```
GROUP(#A#)
```

void HIDEFORM(string message)

Use: *Script (where a comment is expected)*

Description: *The function hides the content of the edit form or view page, and prints the message specified by the "message" parameter.*

Example:

```
HIDEFORM("Thank you for sending us your reply.")
```

string HOURCONDITION(string ng_date, int hour1, int hour2)

Use: *HTML, JavaScript, Script*

Description: *The function returns the SQL code of the condition that is used to compare whether the column named "ng_date" (the content of the column is the date) lies in the time interval between the hours "hour1" and "hour2".*

Example:

```
Use in query condition:
[(ID)] [equals] [OK#crlf#HOURCONDITION(ng_date, 8, 20)]
There must be no space before OK#crlf#
```

string HTMLDECODE(string text)

Use: *Script*

Description: *The function replaces all character sequences contained in the "text" parameter with the corresponding HTML characters.*

Example:

```
HTMLDECODE(&lt;h1&gt;Administrator&lt;/h1&gt;)
// Returns "<h1>Administrator</h1>"
HTMLDECODE(&lt;p&gt;Hi, my name is&lt;br&gt;NET Genium&lt;/p&gt;)
// Returns "<p>Hi, my name is <br>NET Genium</p>"
```

string HTMLENCODE(string text)

Use: *Script*

Description: *The function replaces all HTML characters contained in the "text" parameter with the corresponding character sequence.*

Example:

```
HTMLENCODE(<h1>Administrator<h1>)
// Returns "&lt;h1&gt;Administrator&lt;/h1&gt;"
HTMLENCODE(<p>Hi, my name is <br>NET Genium<p>)
// Returns "&lt;p&gt;Hi, my name is &lt;br&gt;NET Genium&lt;p&gt;"
```

string HTML2TEXT(string html, bool simplehtml)

Use: *HTML, JavaScript, Script*

Description: *The function converts a text string in HTML format to plain text, which uses "\r\n" as a line separator. The optional "simplehtml" parameter specifies whether to maintain the basic text formatting using the "b" and "i" HTML tags.*

Example:

```
HTML2TEXT(<div><b>abc</b></div>, true)
// Returns "<b>abc</b>"
```

string HTML2TEXT(abc)

Use: *HTML, JavaScript, Script*

Description: *The function converts a text string in HTML format to plain text, which uses "\r\n" as a line separator.*

Example:

```
HTML2TEXT(<div>abc</div>)
// Returns "abc"
```

string CHARS(string text)

Use: *HTML, JavaScript, Script*

Description: *The function returns individual characters separated by a semicolon from the text string specified by the "text" parameter. The function further converts the escape characters "\r" and "\n" in the parameter to "\\r" and "\\n".*

Example:

```
CHARS(Administrator)
// Returns "A;d;m;i;n;i;s;t;r;a;t;o;r"
CHARS(Administrator#crlf#Anonymous)
// Returns "A;d;m;i;n;i;s;t;r;a;t;o;r;\r;\n;A;n;o;n;y;m;o;u;s"
```

string IFF(string expression, string yes, string no)

Use: *HTML, JavaScript, Script*

Description: *The function evaluates whether the mathematical expression specified by the "expression" parameter is valid. If the expression is valid, the value specified by the "yes" parameter is returned, otherwise the value specified by the "no" parameter is returned. The function expects number comparisons, and allows the use of operators "<", "<=", ">= ", ">" and " = ".*

Example:

```
IFF(-1.5 <= 2, Valid, Non valid)
// Returns "Valid"
```

int INDEXOF(string text, string subtext)

Use: *HTML, JavaScript, Script*

Description: *The function returns the position of the substring specified by the "subtext" parameter in the string specified by the "text" parameter, starting with the index "0". If no such occurrence is found, it returns the value "-1".*

Example:

```
INDEXOF(Good morning, morning)
// Returns "5"
```

string INT2TIME(int minutes)

Use: *HTML, JavaScript, Script*

Description: *The function returns a text string in the format "HH:mm" expressed from the number of minutes specified by the "minutes" parameter.*

Example:

```
INT2TIME(480)
// Returns "08:00"
INT2TIME(0)
// Returns "00:00"
```

int[] IntegerArray(int count)

Use: *Script*

Description: *The function declares a script variable of the "Integer" value field type, and creates an array of as many elements as specified by the "count" parameter. It fills these elements with values gradually from one, each element has a stored value one higher than the previous element. The "IntegerArray" function call must be case sensitive in the function name.*

Example:

```
IntegerArray(5)
```

string JTSK2WGS(double x, double y)

Use: *HTML, JavaScript, Script*

Description: *The function returns the conversion of the JTSK coordinate system values specified by the "x" and "y" parameters to the WGS system. The resulting values are separated by a semicolon.*

Example:

```
JTSK2WGS(x, y)
```

string LANGUAGE(string loginname)

Use: *HTML, JavaScript, Script*

Description: *The function returns the language setting of the user specified by the "loginname" parameter. The function takes the values "cs", "de", "en", "fr" or "sk".*

Example:

```
LANGUAGE(Administrator)
LANGUAGE(#loginname#)
```

object LAST(object[] array)

Use: *Script*

Description: *The function returns the last value of the script variable specified by the "array" parameter if the variable is an array of values. If the variable is not an array of values, then the function returns the value of the variable. If the value field does not contain any element, the function returns an empty string.*

Example:

```
LAST(#A#)
```

DateTime LASTDAYINMONTH(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns the date of the last day of the month from the date specified by the "date" parameter.*

Example:

```
LASTDAYINMONTH(15.2.2013)
// Returns "29.02.2013"
LASTDAYINMONTH(#today#)
```

DateTime LASTDAYINQUARTAL(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns the date of the last day of the quarter (quarter) from the date specified by the "date" parameter.*

Example:

```
LASTDAYINQUARTAL(15.2.2013)
// Returns "31.03.2013"
LASTDAYINQUARTAL(#today#)
```

DateTime LASTDAYINWEEK(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns the date of the last day of the week from the date specified by the "date" parameter.*

Example:

```
LASTDAYINWEEK(15.2.2013)
// Returns "17.02.2013"
LASTDAYINWEEK(#today#)
```

DateTime LASTDAYINYEAR(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns the date of the last day of the year from the date specified by the "date" parameter.*

Example:

```
LASTDAYINYEAR(15.2.2013)
// Returns "31.12.2013"
LASTDAYINYEAR(#today#)
```

void LEFTALIGNMENT

Use: *Note of the edit form or HTML control name (the body of the control must be left blank)*

Description: *The function starts a block of controls that have the control name left-aligned in the left column.*

Example:

```
LEFTALIGNMENT()
```

void /LEFTALIGNMENT

Use: *HTML control name – The body of the control must be left blank*

Description: *The function ends the block of controls that have the control name left-aligned in the left column.*

Example:

```
/LEFTALIGNMENT
```

int LENGTH(string text)

Use: *HTML, JavaScript, Script*

Description: *The function returns the length (number of characters) of the text string specified by the "text" parameter.*

Example:

```
LENGTH(Administrator)
// Returns "13"
LENGTH(#loginname#)
LENGTH(#A#)
```

double LOG10(double number)

Use: *HTML, Script*

Description: *The function returns the logarithm of base 10 from the value specified by the "number" parameter. The values specified by the "number" parameter can be any non-negative numbers (even decimal).*

Example:

```
LOG10(1.123)
// Returns "0,0503797562614578"
LOG10(0)
// Returns "-infinity"
```

void LOGSERVICE(string service)

Use: *Script*

Description: *The function logs the successful execution of a service named "service" to the database table "ng_servicewindows". The feature finds use mainly in scheduled script execution using scheduled Windows tasks. The default interval after which the service should start again is set to 1 day (1440 minutes).*

Example:

```
LOGSERVICE(RunScript)
```

void LOGSERVICE(string service, int interval)

Use: *Script*

Description: *The function logs the successful execution of a service named "service" to the database table "ng_servicewindows". The feature finds use mainly in scheduled script execution using scheduled Windows tasks. The interval after which the service should start again is defined in minutes by the "interval" parameter.*

Example:

```
LOGSERVICE(RunScript, 720)
```

void LOGVERBOSE(string message)

Use: *Script*

Description: *The function logs the text string "message" in the file "Logs\yyyy-MM-dd.log". At the beginning of the line is the word VERBOSE, then the current date and time in the format "yyyy-MM-dd HH:mm:ss", and then the text string "message" follows.*

Example:

```
LOGVERBOSE(abc)
```

DateTime LONG2DATE(long value)

Use: *HTML, JavaScript, Script*

Description: *The function returns the numeric value specified by the "value" parameter converted to a value of type "DateTime".*

Example:

```
LONG2DATE(630822816000000000)
// Returns "1.1.2000"
```

string LOWER(string text)

Use: *HTML, JavaScript, Script*

Description: *The function returns the text string specified by the "text" parameter, in which all characters are converted to lowercase.*

Example:

```
LOWER(NET Genium)
// Returns "net genium"
```

string LOWRES(string text1, string text2)

Use: *JavaScript, Script*

Description: *The function returns the text string specified by the "text1" parameter if the user is logged in to NET Genium on a low-resolution mobile device (the "Small display" field is checked when logging in). If the above condition is not met, the function returns the string specified by the "text2" parameter. The function is used for setting the dimensions of controls in the edit form for different types of devices (mobile and non-mobile).*

Example:

```
LOWRES(abc, def)
// Returns "abc" for low resolution mobile devices
// Returns "def" for non-mobile or high-resolution mobile devices
```

void LOWRESOLUTION()

Use: *Note of the edit form*

Description: *The feature sets the mobile device to low resolution so that edit forms appear optimized for low resolution even in incognito mode. By default, there is a choice of low resp. high resolution is located on the main page in the login window, but for anonymous edit forms this method is not relevant.*

Example:

```
LOWRESOLUTION()
```

void MARGINS(string top, string right, string bottom, string left)

- ***Use:*** *Note of the edit form*
- ***Description:*** *The function sets the size of the margins of the edit form. The values of the parameters "top", "right", "bottom" and "left" are given in pixels, or they can contain the value "auto", which indicates the default value.*

Example:

```
MARGINS(auto, auto, auto, 100)
```

double MEASUREWORKDAYS(DateTime startdate, DateTime enddate)

- ***Use:*** *HTML, JavaScript, Script*
- ***Description:*** *The function returns the number of working days from the time period specified by the "startdate" and "enddate" parameters. The function takes into account public holidays registered in the basic application "Settings".*

Example:

```
MEASUREWORKDAYS(1.1.2000, 1.2.2000)
// Returns "22"
```

double MEASUREWORKHOURS(DateTime startdate, DateTime enddate)

- ***Use:*** *HTML, JavaScript, Script*
- ***Description:*** *The function returns the number of working hours from the time period specified by the "startdate" and "enddate" parameters. The function takes into account public holidays registered in the basic application "Settings", working hours are set at 24 hours a day.*

Example:

```
MEASUREWORKHOURS(1.1.2000, 1.2.2000)
// Returns "528"
```

double MEASUREWORKHOURS(DateTime startdate, DateTime enddate, int starthour, int endhour)

Use: *HTML, JavaScript, Script*

Description: *The function returns the number of working hours from the time period specified by the "startdate" and "enddate" parameters. The function takes into account the public holidays registered in the basic application "Settings", and the working hours specified by the parameters "starthour" and "endhour".*

Example:

```
MEASUREWORKHOURS(1.1.2000, 1.2.2000, 8, 18)
// Returns "220"
```

string MINUTES2TIME(int minutes)

Use: *HTML, JavaScript, Script*

Description: *The function returns a text string in the format "HH:mm" expressed from the number of minutes specified by the "minutes" parameter.*

Example:

```
MINUTES2TIME(480)
// Returns "08:00"
MINUTES2TIME(0)
// Returns "00:00"
```

int MONTH(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns the month number from the date specified by the "date" parameter.*

Example:

```
MONTH(15.2.2013)
// Returns "2"
MONTH(#now#)
```

string NGDPH()

Use: *HTML, JavaScript, Script*

Description: *The function returns the current VAT rates separated by a semicolon.*

Example:

```
NGDPH()
```

double NGDPH(int rate)

Use: *HTML, JavaScript, Script*

Description: *The function returns the VAT rate defined by the "rate" parameter. The parameter is defined by number 2 for the increased VAT rate, number 1 for the reduced VAT rate and number 0 for the zero VAT rate.*

Example:

```
NGDPH(1)
// Returns current reduced VAT rate, e.g "15"
```

string NGDPH(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns VAT rates separated by a semicolon on the day specified in the "date" parameter.*

Example:

```
NGDPH(1.1.2000)
// Returns semicolon-separated VAT rates valid on a given day, e.g. "22;5;0"
```

double NGDPH(int rate, DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns the VAT rate defined by the "rate" parameter. The parameter is defined by number 2 for the increased VAT rate, number 1 for the reduced VAT rate and number 0 for the zero VAT rate. The rate is determined on the day specified in the "date" parameter.*

Example:

```
NGDPH(2, 1.1.2000)
// Returns increased VAT rate on a given day, e.g. "22"
```

double NGDPH(double base, string type1, double rate, string type2)

Use: *HTML, JavaScript, Script*

Description: *The function calculates either the amount or the VAT value, depending on the "type2" parameter. The basic amount defined by the "base" parameter and the VAT rate defined by the "rate" parameter are used for the calculation. The "type1" parameter specifies the value of the "base" parameter.*

Possible values for the "type1" parameter:

- bezdph – parameter 1 is the sum without VAT
- sdph – parameter 1 is the amount with VAT

Possible values for the "type2" parameter:

- bezdph – calculates the tax base
- VAT – calculates the value of VAT
- zdph – calculates the VAT value rounded to the nearest whole number
- dph1 – calculates the VAT value of the reduced rate
- zdph1 – calculates the value of the reduced rate VAT rounded to the nearest whole number
- dph2 – calculates the VAT value of the increased rate
- zdph2 – calculates the VAT value of the increased rate rounded to an integer
- sdph – calculates the amount with VAT
- zsdph – calculates the amount with VAT rounded up to a whole number
- zsdph-1 – calculates the amount with VAT rounded mathematically to an integer
- zsdph-2 – calculates the amount with VAT rounded to 50 pennies

Example:

```
NGDPH(-10.55, bezdph, 19, sdph)
// Returns "-12,55"
NGDPH(100, sdph, NGDPH(1), dph)
```

string ngef(string id, string arg0, string arg1, string arg2, …)

Use: *HTML, JavaScript, Script*

Description: *The function starts the external function specified by the "id" parameter and returns its return value. Parameters "arg0", "arg1", "arg2" etc. they are passed to the external function as "string[] args".*

Example:

```
ngef(MyFirstFunction, A, B, C, 1, 2, 3, 1.1.2000, …)
```

string ngef2(string id, string arg0, string arg1, string arg2, …)

Use: *Visual controls*

Description: *The function starts the external function specified by the "id" parameter in the visual control, and returns its return value. Parameters "arg0", "arg1", "arg2" etc. they are passed to the external function as "string[] args". To run the external function itself, the call to the "ngef2" function must be stored in the value of the "TextBox" control, set to "Read Only" and as "Hidden Field". Textboxes that have the external function call "ngef2" set as the default value, and at the same time have the default value checked each time the edit form is opened or before each record is saved in the database, may have an empty or blank value in the database, and still displaying this value everywhere in the application except the edit form evaluates the external function set as the default value. When calling the "ngef2" function, the last parameter with the ID of the displayed record is automatically added.*

Example:

```
ngef2(MyFirstFunction, A, B, C, 1, 2, 3, 1.1.2000, 123, …)
```

string NGVAT()

Use: *HTML, JavaScript, Script*

Description: *The function returns a semicolon-separated list of all states for which the "NGVAT" function can be used with parameters.*

Example:

```
NGVAT()
// Returns "Afghanistan;Albania;Algeria;American
Samoa;Andorra;Angola;Anguilla;Antarctica;Antigua and
Barbuda;Argentina;Armenia;Aruba;Australia;Austria;Azerbaijan;Bahamas;Bahrain;Bangladesh;Barbad
os;Belarus;Belgium;Belize;Benin;Bermuda;Bhutan;Bolivia;Bosnia and Herzegovina;Botswana;Bouvet
Island;Brazil;British Indian Ocean Territories;Darussalam;Bulgaria;Burkina
Faso;Burundi;Cambodia;Cameroon;Canada;Cape Verde;Cayman Islands;Central African
Republic;Chad;Chile;China, People's Republic of;Christmas Island;Cocos
Islands;Colombia;Comoros;Congo;Cook Islands;Costa Rica;Cote D'ivoire;Croatia;Cuba;Cyprus;Czech
```

```
Republic;Denmark;Djibouti;Dominica;Dominican Republic;East Timor;Ecuador;Egypt;El
Salvador;Equatorial Guinea;Eritrea;Estonia;Ethiopia;Falkland Islands;Faroe
Islands;Fiji;Finland;France;French Guiana;French Polynesia;French Southern
Territories;FYROM;Gabon;Gambia;Georgia;Germany;Ghana;Gibraltar;Greece;Greenland;Grenada;Guadel
oupe;Guam;Guatemala;Guinea;Guinea-Bissau;Guyana;Haiti;Heard Island And Mcdonald
Islands;Honduras;Hong
Kong;Hungary;Iceland;India;Indonesia;Iran;Iraq;Ireland;Israel;Italy;Jamaica;Japan;Jordan;Kazak
hstan;Kenya;Kiribati;Korea, Democratic People's Republic of;Korea, Republic
of;Kuwait;Kyrgyzstan;Lao Peoples Democratic Republic;Latvia;Lebanon;Lesotho;Liberia;Libyan
Arab
Jamahiriya;Liechtenstein;Lithuania;Luxembourg;Macau;Madagascar;Malawi;Malaysia;Maldives;Mali;M
alta;Marshall
Islands;Martinique;Mauritania;Mauritius;Mayotte;Mexico;Micronesia;Moldova;Monaco;Mongolia;Mont
serrat;Morocco;Mozambique;Myanmar;Namibia;Nauru;Nepal;Netherlands;Netherlands Antilles;New
Caledonia;New Zealand;Nicaragua;Niger;Nigeria;Niue;Norfolk Island;Northern Mariana
Islands;Norway;Oman;Pakistan;Palau;Panama;Papua New
Guinea;Paraguay;Peru;Philippines;Pitcairn;Poland;Portugal;Puerto
Rico;Qatar;Reunion;Romania;Russian Federation;Rwanda;Saint Helena;Saint Kitts and Nevis;Saint
Lucia;Saint Pierre and Miquelon;Saint Vincent and The Grenadines;Samoa;San Marino;Sao Tome and
Principe;Saudi Arabia;Senegal;Seychelles;Sierra Leone;Singapore;Slovakia;Slovenia;Solomon
Islands;Somalia;South Africa;South Georgia and Sandwich Islands;Spain;Sri
Lanka;Sudan;Suriname;Svalbard and Jan Mayen;Swaziland;Sweden;Switzerland;Syrian Arab
Republic;Taiwan;Tajikistan;Tanzania;Thailand;Togo;Tokelau;Tonga;Trinidad and
Tobago;Tunisia;Turkey;Turkmenistan;Turks and Caicos Islands;Tuvalu;Uganda;Ukraine;United Arab
Emirates;United Kingdom;USA;Uruguay;Uzbekistan;Vanuatu;Vatican City
State;Venezuela;Vietnam;Virgin Islands (British);Virgin Islands (U.S.);Wallis And Futuna
Islands;Western Sahara;Yemen;Yugoslavia;Zaire;Zambia;Zimbabwe"
```

double NGVAT(double base, DateTime date, string country, string dic)

Use: *HTML, JavaScript, Script*

Description: *The function returns the calculated amount of VAT that must be paid when delivering the goods to the country specified by the "country" parameter. The "base" parameter defines the monetary base. The "date" parameter defines the date of the taxable supply and the "dic" parameter defines the VAT number. If the supplier does not have an assigned VAT number, the parameter will be empty.*

Example:

```
NGVAT(1500.20, #today#, Germany, CZ27092381)
```

string NGVATRATES(string country, DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns VAT rates separated by a semicolon in the given country (CZ, DE, HU, SK, …) specified by the "country" parameter, and on the given date specified by the "date" parameter.*

Example:

```
NGVATRATES(DE, 5.5.2012)
// Returns "19;7;0"
```

void NOBACKGROUND

Use: *Note of the edit form or HTML control name (the body of the control must be left blank)*

Description: *The function starts a block of controls that do not use the colored background in the left column with the control's name. A started block that is not terminated with the "/NOBACKGROUND" function anchors the "Save", "Delete" and "Back" buttons. These buttons will then not be floating, and will always be displayed at the end of the edit form.*

Example:

```
NOBACKGROUND()
```

void /NOBACKGROUND

Use: *HTML control name – The body of the control must be left blank*

Description: *The function ends a block of controls that do not use the colored background in the left column with the control's name.*

Example:

```
/NOBACKGROUND
```

int OCR2FILE(string path, string type)

Use: *Script*

Description: *The function OCR converts the text from the file specified by the "path" parameter, and creates a new file attachment defined by the "type" parameter. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file is defined by the "path" parameter – the full path to the file on disk. The result of the function is the ID of the newly created file attachment. Supported output types are "doc", "docx", "htm", "pdf", "rtf", "txt", "xlsx", "xml".*

Example:

```
OCR2FILE(#rootpath#Temp\source.xlsx, pdf)
```

int OCR2FILE(string path, string type, string url)

Use: *Script*

Description: *The function OCR converts the text from the file specified by the "path" parameter, and creates a new file attachment defined by the "type" parameter. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file is defined by the "path" parameter – the full path to the file on disk. The result of the function is the ID of the newly created file attachment. Supported output types are "doc", "docx", "htm", "pdf", "rtf", "txt", "xlsx", "xml". The OCR recognition itself is performed by a separate web application at the address defined by the "url" parameter.*

Example:

```
OCR2FILE(#rootpath#Temp\source.xlsx, xls, https://www.netgenium.com/netgenium_with_ocr)
```

int OCR2FILE(int file, string type)

Use: *Script*

Description: *The function OCR converts the text from the file attachment specified by the "file" parameter, and creates a new file attachment defined by the "type" parameter. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file attachment is defined by the "file" parameter – the file attachment ID. The result of the function is the ID of the newly created file attachment. Supported output types are "doc", "docx", "htm", "pdf", "rtf", "txt", "xlsx", "xml".*

Example:

```
OCR2FILE(#ng_attachment#, docx)
```

int OCR2FILE(int file, string type, string url)

Use: *Script*

Description: *The function OCR converts the text from the file attachment specified by the "file" parameter, and creates a new file attachment defined by the "type" parameter. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file attachment is defined by the "file" parameter – the file attachment ID. The result of the function is the ID of the newly created file attachment. Supported output types are "doc", "docx", "htm", "pdf", "rtf", "txt", "xlsx", "xml". The OCR recognition itself is performed by a separate web application at the address defined by the "url" parameter.*

Example:

```
OCR2FILE(#ng_attachment#, pdf, https://www.netgenium.com/netgenium_with_ocr)
```

string OCR2HTML(string path)

Use: *Script*

Description: *The function converts the OCR text from the file specified by the "path" parameter, and returns it as a text string in HTML format. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file is defined by the "path" parameter – the full path to the file on disk.*

Example:

```
OCR2HTML(#rootpath#Temp\source.xlsx)
```

string OCR2HTML(string path, string url)

Use: *Script*

Description: *The function converts the OCR text from the file specified by the "path" parameter, and returns it as a text string in HTML format. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file is defined by the "path" parameter – the full path to the file on disk. The OCR recognition itself is performed by a separate web application at the address defined by the "url" parameter.*

Example:

```
OCR2HTML(#rootpath#Temp\source.xlsx, https://www.netgenium.com/netgenium_with_ocr)
```

string OCR2HTML(int file)

Use: *Script*

Description: *The function converts the text from the file attachment specified by the "file" parameter by the OCR method, and returns it as a text string in HTML format. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file attachment is defined by the "file" parameter – the file attachment ID.*

Example:

```
OCR2HTML(#ng_attachment#)
```

string OCR2HTML(int file, string url)

Use: *Script*

Description: *The function converts the text from the file attachment specified by the "file" parameter by the OCR method, and returns it as a text string in HTML format. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file attachment is defined by the "file" parameter – the file attachment ID. The OCR recognition itself is performed by a separate web application at the address defined by the "url" parameter.*

Example:

```
OCR2HTML(#ng_attachment#, https://www.netgenium.com/netgenium_with_ocr)
```

string OCR2TEXT(string path)

Use: *Script*

Description: *The function converts the OCR text from the file specified by the "path" parameter, and returns it as a plain text string without HTML formatting. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file is defined by the "path" parameter – the full path to the file on disk.*

Example:

```
OCR2TEXT(#rootpath#Temp\source.xlsx)
```

string OCR2TEXT(string path, string url)

Use: *Script*

Description: *The function converts the OCR text from the file specified by the "path" parameter, and returns it as a plain text string without HTML formatting. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file is defined by the "path" parameter – the full path to the file on disk. The OCR recognition itself is performed by a separate web application at the address defined by the "url" parameter.*

Example:

```
OCR2TEXT(#rootpath#Temp\source.xlsx, https://www.netgenium.com/netgenium_with_ocr)
```

string OCR2TEXT(int file)

Use: *Script*

Description: *The function OCR converts the text from the file attachment specified by the "file" parameter, and returns it as a plain text string without HTML formatting. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file attachment is defined by the "file" parameter – the file attachment ID.*

Example:

```
OCR2TEXT(#ng_attachment#)
```

string OCR2TEXT(int file, string url)

Use: *Script*

Description: *The function OCR converts the text from the file attachment specified by the "file" parameter, and returns it as a plain text string without HTML formatting. Conversion from MS Word or Excel files is performed natively, in other cases it uses an OCR server. The source file attachment is defined by the "file" parameter – the file attachment ID. The OCR recognition itself is performed by a separate web application at the address defined by the "url" parameter.*

Example:

```
OCR2TEXT(#ng_attachment#, https://www.netgenium.com/netgenium_with_ocr)
```

string PADLEFT(string text, int totalwidth, string pad)

Use: *HTML, JavaScript, Script*

Description: *The function fills the text string specified by the "text" parameter from the left with the character specified by the "pad" parameter to the length specified by the "totalwidth" parameter.*

Example:

```
PADLEFT(x, 5, 0)
// Returns "0000x"
```

string PADRIGHT(string text, int totalwidth, string pad)

Use: *HTML, JavaScript, Script*

Description: *The function completes the string specified by the "text" parameter on the right with the character specified by the "pad" parameter to the length specified by the "totalWidth" parameter.*

Example:

```
PADRIGHT(x, 5, 0)
// Returns "x0000"
```

string PAGEIDENTIFIER(string page)

Use: *HTML, JavaScript, Script*

Description: *The function returns the identifier of the web page specified by the "page" parameter.*

Example:

```
PAGEIDENTIFIER(Calendar Action)
// Returns "calendar-action"
```

string PAGEIDENTIFIER2(string page)

Use: *HTML, JavaScript, Script*

Description: *The function returns the identifier of the web page specified by the "page" parameter.*

Example:

```
PAGEIDENTIFIER2(Calendar Action)
```

```
// Returns "CalendarAction"
```

string PLACESONMAP(int id)

Use: *Script*

Description: *The function returns the HTML code of the map specified by the "id" parameter, including all places on the map defined in the "Maps" application.*

Example:

```
PLACESONMAP(14)
```

double POWER(double number)

Use: *HTML, Script*

Description: *The function returns the square of the number specified by the "number" parameter.*

Example:

```
POWER(-1.23)
// Returns "1,5129"
POWER(5)
// Returns "25"
POWER(#A#)
```

int PRINTATTACHMENT(int button, int id)

Use: *Script*

Description: *The function starts the print defined on the control specified by the "button" parameter, saves the resulting print report as a new file attachment, and returns the ID of this attachment. The "button" parameter indicates the ID of the button that handles the printing, and the "id" parameter indicates the ID of the database record to be printed.*

Example:

```
PRINTATTACHMENT(123, 1)
```

void PRINTCOLUMNS(int[] cids)

Use: *Note database query*

Description: *The function limits the list of loaded columns when printing to the print template. The columns to be loaded are defined as an array of identifiers, where the column ID is the same as the control ID by which the column is defined (eg TextBox ID).*

Example:

```
PRINTCOLUMNS(1, 2, 3)
```

void PROTECTFORM(int form, int id, string message)

Use: *Script (where a comment is expected)*

Description: *The function verifies whether the record specified by the "id" parameter is open in the edit form specified by the "form" parameter. If so, it triggers the interrupt specified by the "message" parameter. The text "The record is locked by user {0} from {1}." Calling a function only makes sense if the form is set to lock records.*

Example:

```
PROTECTFORM(1, 1, Record editing not allowed.)
// Returns for example "Record editing is not allowed. The record is locked by the
Administrator user from 12:30."
```

void PROTECTSCRIPT(int form, int n, string message)

Use: *Script (where a comment is expected)*

Description: *The function verifies that the script specified by the "form" and "n" parameters is running. If so, it triggers the interrupt specified by the "message" parameter. The parameters "form" and "n" can be found in the administrator mode in the tooltip, which is displayed after hovering the mouse pointer over the link "Edit script".*

Example:

```
PROTECTSCRIPT(1, 10, ABC script execution not allowed. Please wait for it to complete.)
```

string READURL(string text)

Use: *HTML*

Description: *The function reads the content of the web page defined by the "text" parameter, and returns its content.*

Example:

```
READURL(https://www.netgenium.com)
```

void RECYCLEPOOL(string applicationpool)

Use: *Script*

Description: *The function recycles the application pool specified by the "applicationpool" parameter. The script that states the "RECYCLEPOOL" function must be run exclusively using the "RunScript.exe" program.*

Example:

```
RECYCLEPOOL(DefaultAppPool)
```

void REDUCEATTACHMENT(int file, int quality)

Use: *Script*

Description: *The function changes the quality of the image specified by the "file" parameter to the value specified by the "quality" parameter, and saves it in the "jpg" format.*

Example:

```
REDUCEATTACHMENT(#ng_image#, 25)
```

void REDUCEATTACHMENT(int file, int maxwidth, int maxheight)

Use: *Script*

Description: *The function reduces the image specified by the "file" parameter to the maximum size of the width specified by the "maxwidth" parameter and the height specified by the "maxheight" parameter, and saves in the "jpg" format. The highest possible quality will be used to reduce the image.*

Example:

```
REDUCEATTACHMENT(#ng_image#, 500, 500)
```

void REDUCEATTACHMENT(int file, int maxwidth, int maxheight, bool highquality)

Use: *Script*

Description: *The function reduces the image specified by the "file" parameter to the maximum size of the width specified by the "maxwidth" parameter and the height specified by the "maxheight" parameter, and saves in the "jpg" format. The "highquality" parameter set to "true" means that the highest possible quality will be used to reduce the image. Otherwise, standard quality will be used.*

Example:

```
REDUCEATTACHMENT(#ng_image#, 500, 500, true)
```

string REMOVEDIACRITICS(string text)

Use: *HTML, JavaScript, Script*

Description: *The function removes diacritics from the text string specified by the "text" parameter.*

Example:

```
REMOVEDIACRITICS(Příliš žluťoučký kůň pěl ďábelské ódy)
// Returns "Prilis zlutoucky kun pel dabelske ody"
```

string REMOVEIFCONTAINS(string text, string search1[, search2,…])

Use: *HTML, JavaScript, Script*

Description: *The function resets the "text" parameter if it contains the substring "search1" or "search2" or "search3", etc.*

Example:

```
REMOVEIFCONTAINS(Invoice is registered, is)
// Returns "" (empty string)
```

string REMOVEIFENDSWITH (string text, string search1 [, search2,…])

Use: *HTML, JavaScript, Script*

Description: *The function resets the "text" parameter if it ends with the substring "search1" or "search2" or "search3", etc.*

Example:

```
REMOVEIFENDSWITH(Good morning, morning, evening)
// Returns "" (empty string)
```

string REMOVEIFEQUALS(string text, string search1[, search2,…)

Use: *HTML, JavaScript, Script*

Description: *The function resets the "text" parameter if it matches the substring "search1" or "search2" or "search3", etc.*

Example:

```
REMOVEIFEQUALS(Approved, Approved)
// Returns "" (empty string)
```

string REMOVEIFSTARTSWITH(string text, string search1[, search2,…])

Use: *HTML, JavaScript, Script*

Description: *The function resets the "text" parameter if it starts with the substring "search1" or "search2" or "search3", etc.*

Example:

```
REMOVEIFSTARTSWITH(Invoice is registered, Fact, Fakt, Faak)
// Returns "" (empty string)
```

string REMOVETAGS(string html)

Use: *HTML, JavaScript, Script*

Description: *The function removes all tags from the string specified by the "html" parameter.*

Example:

```
REMOVETAGS(<p>abc</p><p>abc</p>)
// Returns "abcabc"
```

string REMOVEWHITESPACES(string value)

Use: *HTML, JavaScript, Script*

Description: *The function removes all spaces (spaces and tabs) from the string specified by the "value" parameter.*

Example:

```
REMOVEWHITESPACES(a b c);
// Returns "abc"
```

void RENAMEATTACHMENT(int file, string name)

Use: *Script (where a comment is expected)*

Description: *The function renames the file attachment specified by the "file" parameter to the name specified by the "name" parameter without the extension.*

Example:

```
RENAMEATTACHMENT(#ng_attachment#, New file name)
```

void RENAMEATTACHMENT(int file, string prefix, string postfix)

Use: *Script (where a comment is expected)*

Description: *The function renames the file attachment specified by the "file" parameter to a new name, which is created by combining the value of the "prefix" parameter, the original file name without extension, the value of the "postfix" parameter, and then the extension.*

Example:

```
RENAMEATTACHMENT(#ng_attachment#, A-, -B)
```

string REPLACE(string value, string oldvalue, string newvalue)

Use: *HTML, JavaScript, Script*

Description: *The function returns the text string specified by the "value" parameter, in which all occurrences of the string stored in the "oldvalue" parameter were replaced by the string stored in the "newvalue" parameter.*

Example:

```
REPLACE(Hello NET Genium, NET Genium, NG)
// Returns "Hello NG"
```

string REPLACEIFEQUALS(string compare1, string compare2, string text)

Use: *HTML, JavaScript, Script*

Description: *The function returns the value of the "text" parameter if the value of the "compare1" parameter is equal to the value of the "compare2" parameter. Otherwise, it returns the value of the "compare1" parameter.*

Example:

```
REPLACEIFEQUALS(abc, abc, def)
// Returns "def"
```

string REPLACETAB(string items, string olditem, string newitem)

Use: *HTML, JavaScript, Script*

Description: *The function returns a tab-delimited list of items specified by the "items" parameter, in which all items equal to the "oldvalue" (pattern) parameter have been replaced by the value specified by the "newvalue" (replacement) parameter.*

Example:

```
REPLACETAB(abc#tab#abcd#tab#def, abc, def)
// Returns "def#tab#abcd#tab#def"
```

string REPLYBODY(string from, string to, DateTime date, string body)

Use: *HTML, JavaScript, Script*

Description: *The function returns the value of the "body" parameter as a formatted email response. The body of the sent e-mail message starts with a free line where you can write the text in response to the received e-mail. The next line shows the current response time defined by the "date" parameter and the e-mail address stored in the "from" parameter. This parameter represents the e-mail address of the sender of the original message. On the next lines there is the text of the original message (parameter "points") bordered on the left by a vertical line. The "to" parameter is the email address of the recipient of the original message and the "REPLYBODY" function does not currently work with it.*

Example:

```
REPLYBODY(info@netgenium.com, info@netgenium.com, #now#, body)
```

string REQUESTFORM(string name)

Use: *Script*

Description: *The function returns the value of the form element specified by the "name" parameter when executing the HTML postback.*

Example:

```
<input name="C5$tb" type="text" value="New Invoice" id="C5_tb" style="width:110px;">
REQUESTFORM(C5$tb)
// Returns "New Invoice"
```

string REQUESTQUERYSTRING(string attribute)

Use: *Script*

Description: *The function returns the value of the attribute from the current URL. The attribute name is specified by the "attribute" parameter.*

Example:

```
https://www.netgenium.com?id=123&next=true
REQUESTQUERYSTRING(id)
// Returns "123"
```

void RESETDATAGRIDSETTINGS(int query)

Use: *Script (where a comment is expected)*

Description: *The function resets the datagrid setting specified by the "query" parameter by unchecking the search term, unchecking the search in each column, and sets the value of all filters to "(All)".*

Example:

```
RESETDATAGRIDSETTINGS(123)
```

void RESETUSERPASSWORD(string ng_checkbox, string from, string subject, string message)

Use: *Script (where a comment is expected)*

Description: *The function resets the password of the user who has the checkbox specified by the "ng_checkbox" parameter. The user is notified of this reset by an email from the address specified by the "from" parameter, with the subject specified by the "subject" parameter and the text specified by the "message" parameter. After the next login, the user is prompted to change the password.*

Example:

```
RESETUSERPASSWORD(ng_marktosendpassword, #ng_sender#, #ng_subject#, #ng_message#)
RESETUSERPASSWORD(ng_ marktosendpassword, noreply@netgenium.com, Password expiration, Please change your password.)
```

Example email:

```
Please change your password.

Login name: novak.petr
Password: kR2FP50h
```

string RESOURCE(int index)

Use: *HTML, JavaScript, Script*

Description: *The function returns an item from the list of sources specified by its index, resp. the "index" parameter. The resource must be defined in the "resources.txt" file in the "NETGenium\Config\Resources\language_settings" directory. The index uses zero numbering.*

Example file content:

- 0 = "10 minutes"
- 1 = "10 seconds"
- 2 = "1 decimal place"

Example:

```
RESOURCE(2)
```

void RESTARTSERVICE(string service)

Use: *Script*

Description: *The function restarts the service specified by the "service" parameter. The script that states the "RESTARTSERVICE" function must be run exclusively using the "RunScript.exe" program.*

Example:

```
RESTARTSERVICE(CRMService)
```

void RESTOREFROMHISTORY(int form, int id)

Use: *Script (where a comment is expected)*

Description: *The function restores the deleted record from the history specified by the "id" parameter in the edit form specified by the "form" parameter.*

Example:

```
RESTOREFROMHISTORY(16, 6620)
```

double ROUND(double number)

Use: *HTML, Script*

Description: *The function returns the rounded value of the number specified by the "number" parameter.*

Example:

```
ROUND(-1.63)
// Returns "-2"
ROUND(1.63)
// Returns "2"
```

double ROUND(double number, double decimals)

Use: *HTML, the function is not available in the script (it is necessary to use for example "ROUND (… * 100)/100")*

Description: *The function returns the number specified by the "number" parameter, rounded to the number of decimal places specified by the "decimals" parameter.*

Example:

```
ROUND(-1.636, 2)
// Returns "-1,64"
ROUND(1.636, 2)
// Returns "1,64"
```

string SCALEIMAGES(string html, int maxWidth, int maxHeight)

Use: *Script*

Description: *The function returns images with modified dimensions specified in the "html" parameter. The "width" and "height" parameters will be set proportionally for all edited images, depending on the "maxWidth" and "maxHeight" parameters.*

Example:

```
SCALEIMAGES(#ng_html#, 20, 15)
```

string SEARCHFILES(string subject, string content, int results)

Use: *HTML*

Description: *The function searches for file attachments according to the specified criteria, and returns a table with three columns, where the first column contains the name of the file attachment, the second its size, and the third column the creation date. File attachments are searched in full text according to their name defined by the "subject" parameter and according to the content defined by the "content" parameter. The "results" parameter determines how many occurrences of file attachments found should be displayed in the table. Using the "content" parameter only makes sense if the file attachments are indexed by the Indexing Service or Windows Search.*

Example:

```
SEARCHFILES(NETGenium.gif, , 100)
SEARCHFILES(Name, Content, 100)
```

string SEARCHFILES(string subject, string content, int results, DateTime date1)

Use: *HTML*

Description: *The function searches for file attachments according to the specified criteria and returns a table with three columns, where the first column contains the name of the file attachment, the second its size, and the third column the creation date. File attachments are searched in full text according to their name defined by the "subject" parameter and according to the content defined by the "content" parameter. The "results" parameter determines how many occurrences of file attachments found should be displayed in the table. The "date1" parameter further specifies that only attachments with a creation date greater than or equal to the specified date should be searched. Using the "content" and "date1" parameters only makes sense if the file attachments are indexed by the Indexing Service or Windows Search.*

Example:

```
SEARCHFILES(NETGenium.gif, , 100, 1.1.2013)
SEARCHFILES(Name, Content, 100, 1.1.2013)
```

string SEARCHFILES(string subject, string content, int results, DateTime date1, DateTime date2)

Use: *HTML*

Description: *The function searches for file attachments according to the specified criteria and returns a table with three columns, where the first column contains the name of the file attachment, the second its size, and the third column the creation date. File attachments are searched in full text according to their name defined by the "subject" parameter and according to the content defined by the "content" parameter. The "results" parameter determines how many occurrences of file attachments found should be displayed in the table. The "date1" and "date2" parameters further specify that only attachments with a creation date between these two specified dates will be searched. Using the "content", "date1", and "date2" parameters only makes sense if the file attachments are indexed by the Indexing Service or Windows Search.*

Example:

```
SEARCHFILES(NETGenium.gif, , 100, 1.1.2013, 31.12.2013)
SEARCHFILES(Name, Content, 100, 1.1.2013, 31.12.2013)
```

string SEARCHNEXTID(string array, string id)

Use: *HTML, JavaScript, Script*

Description: *The function accepts an array of values separated by a semicolon, which is specified by the "array" parameter. The value specified by the "id" parameter is searched for in this field, and the function then returns the value that is immediately after this value in the specified field. If the entered value "id" is not found in the field "array" or there is no value after the entered value in the field, the function returns "0". If the value "id" is "0", the function returns the first value of the specified field.*

Example:

```
SEARCHNEXTID(1;2;3, 2)
// Returns "3"
SEARCHNEXTID(1;2;3, 0)
// Returns "1"
SEARCHNEXTID(two;four;six, four)
// Returns "six"
```

string SEARCHPREVID(string array, string id)

Use: *HTML, JavaScript, Script*

Description: *The function accepts an array of values separated by a semicolon, which is specified by the "array" parameter. The value specified by the "id" parameter is searched for in this field, and the function then returns the value that is immediately before this value in the specified field. If the entered value "id" is not found in the "array" field or there is no value in the field before the entered value, the function returns "0". If the value "id" is "0", the function returns the last value of the specified field.*

Example:

```
SEARCHPREVID(1;2;3, 2)
// Returns "1"
SEARCHPREVID(1;2;3, 0)
// Returns "3"
SEARCHPREVID(two;four;six, four)
// Returns "two"
```

string SECONDS2TIME(double seconds)

Use: *HTML, JavaScript, Script*

Description: *The function returns a text string in the format "HH:mm:ss" expressed from the number of seconds specified by the "seconds" parameter.*

Example:

```
SECONDS2TIME(480)
// Returns "00:08:00"
SECONDS2TIME(0)
// Returns "00:00:00"
```

void SENDCHATMESSAGE(string message)

Use: *Script*

Description: *The function sends a chat message, the text of which is defined by the "message" parameter. The message is sent to all currently logged in users.*

Example:

```
SENDCHATMESSAGE(Hello)
```

void SENDCHATMESSAGE(string message, string loginname)

Use: *Script*

Description: *The function sends a chat message, the text of which is defined by the "message" parameter. The message is sent to the user specified by the "loginname" parameter. The user does not have to be currently logged in, the chat message will then be displayed after his next login.*

Example:

```
SENDCHATMESSAGE(Hello, Administrator)
```

string SENDMAILINVITATION(string from, string replyto, string to, string cc, string bcc, string subject, string body, string attachments, bool save, string summary, string location, DateTime start, DateTime end, string description [, string c0, string v0 [, string c1, string v1…]])

Use: *Script (where a comment is expected)*

Description: *The function sends the meeting request via the e-mail address specified by the "from" parameter to the e-mail address specified by the "to" parameter, and returns the UID of the invitation. The "replyto" parameter indicates to which e-mail address this invitation should be replied to by default. The "summary" parameter defines the subject of the meeting, the "location" parameter defines the place of the meeting. The "start" and "end" parameters indicate the start and end of the meeting. The "description" parameter offers the option of entering a more detailed and multi-line description of the meeting. Other parameters are explained for the "SCHEDULEMAILMESSAGE" function.*

Example:

```
SENDMAILINVITATION(info@mojefirma.cz, , info@netgenium.com, , , E-mail message, Dear Sir…, 0,
True, Meeting date, Venue, 25.4.2013 15:00, 25.4.2013 20:00, Meeting description)
```

void SENDMAILMESSAGE(string from, string replyto, string to, string cc, string bcc, string subject, string body, string attachments, bool save [, string c0, string v0 [, string c1, string v1…]])

Use: *Script (where a comment is expected)*

Description: *The function sends an e-mail message from the e-mail address specified by the "from" parameter to the e-mail address specified by the "to" parameter. The "replyto" parameter indicates which e-mail address should be replied to by e-mail by default. Other parameters are explained for the "SCHEDULEMAILMESSAGE" function.*

Example:

```
SENDMAILMESSAGE(info@mojefirma.cz, , info@netgenium.com, , , E-mail message, Dear Sir…, 0,
True)
SENDMAILMESSAGE(info@mojefirma.com, , info@netgenium.com, , , E-mail message, Dear Sir…,
1;#ng_attachment#;D:\attachment.txt, true)
```

void SENDMAILMESSAGE2(string from, string replyto, string to, string cc, string bcc, string subject, string body, string attachments, bool save [, string c0, string v0 [, string c1, string v1…]])

The function is identical to the SENDMAILMESSAGE function, but uses a second SMTP server – in the NET Genium settings "SMTP Server 2".

void SETCOOKIE(string key, string value, DateTime expires)

Use: *Script (where a comment is expected)*

Description: *The function sets the cookie value specified by the "key" parameter to the value defined by the "value" parameter, with the expiration date defined by the "expires" parameter. The "SameSite" parameter will be set to "Strict", the "Secure" parameter will be selected according to the NET Genium configuration (configuration file "SecureCookies.txt", "SSL.txt", or secure communication via the "https" protocol), and the "HttpOnly" will be set to "False".*

Example:

```
SETCOOKIE(abc, def, #today# + 6m)
```

void SETDATAGRIDSELECTION(string control, string ids)

Use: *Script (where a comment is expected)*

Description: *The function sets in the datagrid specified by the "control" parameter a list of primary keys of all selected records separated by semicolons specified by the "ids" parameter. The "control" parameter indicates the name of the control element according to the conventions used in variables or print templates:*

- *datagridX*
- *DGX*

Example:

```
SETDATAGRIDSELECTION(datagrid1, 1;2;3)
SETDATAGRIDSELECTION(DG123, 1;2;3)
```

void SETVARIABLE(string variable, string value)

Use: *Script (where a comment is expected)*

Description: *The function sets the value of the variable specified by the "variable" parameter to the value defined by the "value" parameter.*

Example:

```
SETVARIABLE(A, abc)
```

void SCHEDULEBACK()

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the previous page immediately after the successful execution of the script.*

Example:

```
SCHEDULEBACK()
```

void SCHEDULEBACK(string message)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the previous page immediately after the successful execution of the script. After opening the previous page, the user is presented with a message in the dialog box specified by the "message" parameter.*

Example:

```
SCHEDULEBACK(abc)
SCHEDULEBACK("Message that contains a comma.")
```

void SCHEDULEBACK(string message, bool dialog)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the previous page immediately after the successful execution of the script. After opening the previous page, the user is shown the message specified by the "message" parameter either in the dialog box or only as red highlighted text displayed at the bottom of the page.*

Example:

```
SCHEDULEBACK(abc, false)
SCHEDULEBACK("Message that contains a comma.", false)
```

void SCHEDULEDEFAULTVALUE(string dbname, string value)

Use: *Script (where a comment is expected)*

Description: *The function schedules the filling of the value specified by the "value" parameter into the control defined by the identifier "dbname" when opening the edit form immediately after the successful execution of the script.*

Example:

```
SCHEDULEDEFAULTVALUE(ng_status, Completed)
```

void SCHEDULEDEFAULTVALUE(string dbname, string dbtype, string value)

Use: *Script (where a comment is expected)*

Description: *The function schedules the filling of the value specified by the "value" parameter into the control defined by the identifier "dbname" when opening the edit form immediately after the successful execution of the script. The "dbtype" parameter specifies the database type of the value specified by the "value" parameter. The database type must be specified for values that are of type "Integer", "Double" or "Date". If not specified, the database type "String" will be used.*

Example:

```
SCHEDULEDEFAULTVALUE(ng_date, Date, #today# - 20)
```

void SCHEDULEDEFAULTVALUES(string dbname1, string dbname2, … , string values)

Use: *Script (where a comment is expected)*

Description: *The function schedules the completion of semicolon-separated values specified by the "values" parameter into the controls specified by the identifiers "dbname1", "dbname2", etc. open when the edit form is opened immediately after the successful execution of the script.*

Example:

```
SCHEDULEDEFAULTVALUES(ng_text1, ng_text2, #A#)
SCHEDULEDEFAULTVALUES(ng_text1, ng_text2, value1;value2)
```

void SCHEDULEENTRY(int id)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the record with the primary key specified by the "id" parameter in the currently open edit form immediately after the successful execution of the script.*

Example:

```
SCHEDULEENTRY(123)
```

void SCHEDULEENTRY(string column, int id)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the searched record in the currently open edit form immediately after the successful execution of the script. The record is retrieved in the associated database table from all records that have the value specified by the "id" parameter stored in the column specified by the "column" parameter. If no record matching this condition is found, a new record will be created, the value "id" will be stored in the "column", and this record will then be opened in the edit form.*

Example:

```
SCHEDULEENTRY(pid, 123)
```

void SCHEDULEFORM(int id)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the edit form specified by the "id" parameter. The "id" parameter can be found in the administrator mode in the tooltip, which is displayed after hovering the mouse pointer over the "Edit edit form" link. An empty edit form will open to create a new record.*

Example:

```
SCHEDULEFORM(1)
SCHEDULEFORM(#form#)
```

void SCHEDULEFORM(string dbname)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the edit form specified by the "dbname" parameter. An empty edit form will open to create a new record.*

Example:

```
SCHEDULEFORM(susers)
```

void SCHEDULEFORM(int id, int record)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the edit form specified by the "id" parameter, and the opening of the record with the primary key specified by the "record" parameter immediately after the successful execution of the script. The "id" parameter can be found in the administrator mode in the tooltip, which is displayed after hovering the mouse pointer over the "Edit edit form" link.*

Example:

```
SCHEDULEFORM(1, 1)
```

void SCHEDULEFORM(string dbname, int record)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the edit form specified by the "dbname" parameter, and the opening of the record with the primary key specified by the "record" parameter immediately after the successful execution of the script.*

Example:

```
SCHEDULEFORM(susers, 1)
```

void SCHEDULEFORM(int id, int record, bool back)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the edit form specified by the "id" parameter, and the opening of the record with the primary key specified by the "record" parameter immediately after the successful execution of the script. The "id" parameter can be found in the administrator mode in the tooltip, which is displayed after hovering the mouse pointer over the "Edit edit form" link. The "back" parameter defines whether the browser window redirects to the main page (view page/portlet) at "true" or only back to the parent form at "false".*

Example:

```
SCHEDULEFORM(1, 1, true)
```

void SCHEDULEFORM(string dbname, int record, bool back)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the edit form specified by the "dbname" parameter, and the opening of the record with the primary key specified by the "record" parameter immediately after the successful execution of the script. The "back" parameter defines whether the browser window redirects to the main page (view page/portlet) at "true" or only back to the parent form at "false".*

Example:

```
SCHEDULEFORM(susers, 1, false)
```

void SCHEDULEMAILMESSAGE(string to, string cc, string bcc, string subject, string body, string attachments, bool save [, string c0, string v0 [, string c1, string v1…]])

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of a page for entering a new e-mail message immediately after the successful execution of the script. The function on the new e-mail page fills in the input field:*

- "To" parameter "to",
- "Copy" with the parameter "cc",
- "Hidden" parameter "bcc",
- "Subject" parameter "subject",
- "Message" parameter "points".

Next, it attaches the file attachments specified by the "attachments" parameter to the message, and selects the "Save" check box if the value of the "save" parameter is equal to "True". "Attachments" can be a semicolon-

separated list of IDs or paths on a disk. Parameters “c0” resp. “V0” indicates the table of sent mail – identifier of the database control, resp. the value that will be inserted into the database control if the user selects the “Save” option before sending the e-mail message. This ensures that the sent message is stored in a database table that can contain these controls. The database table in which the sent mail is stored is set for the “E-Mail” control.

Example:

```
SCHEDULEMAILMESSAGE(info@netgenium.com, , , E-mail message, Dear Sir…, 0, true)
SCHEDULEMAILMESSAGE(info@netgenium.com, , , E-mail message, Dear Sir…,
1;#ng_attachment#;D:\attachment.txt, true)
```

void SCHEDULEPORTLETS()

Use: *Script (where a comment is expected)*

Description: *The function schedules the main page to open immediately after the script executes successfully.*

Example:

```
SCHEDULEPORTLETS()
```

void SCHEDULEPRINT(int button, int id)

Use: *Script (where a comment is expected)*

Description: *The function schedules printing to the print template immediately after the successful execution of the script. The “button” parameter indicates the ID of the button that handles the printing, and the “id” parameter indicates the ID of the record to be printed. The selection of the print template is controlled by the setting of the operating button. If multiple “SCHEDULEPRINT” function calls are used in the script, the output is multiple printed files packaged in a “zip” archive, which is then offered for download.*

Example:

```
SCHEDULEPRINT(123, 1)
```

void SCHEDULEPRINT(int button, int id, bool joinxls)

Use: *Script (where a comment is expected)*

Description: *The function schedules printing to the print template immediately after the successful execution of the script. The "button" parameter indicates the ID of the button that handles the printing, and the "id" parameter indicates the ID of the record to be printed. The selection of the print template is controlled by the setting of the operating button. If multiple "SCHEDULEPRINT" function calls are used in the script, the output is multiple printed files packaged in a "zip" archive, which is then offered for download. If all printed files are the result of printing to MS Excel print templates, or if all printed files are PDF files, the "mergefiles" parameter determines the output of the function. If the "mergefiles" parameter is set to "true", all printed files are merged into a single "xlsx" file with multiple sheets, or into a single PDF file.*

Example:

```
SCHEDULEPRINT(123, 1, true)
```

void SCHEDULEPRINTOFATTACHMENT(int id)

Use: *Script (where a comment is expected)*

Description: *The function schedules the printing of a file attachment immediately after the successful execution of the script. The "id" parameter indicates the ID of the file attachment that will be printed. If multiple "SCHEDULEPRINTOFATTACHMENT" function calls are used in the script, the output is multiple printed files packaged in a "zip" archive, which is then offered for download. The function call can also be freely combined with the "SCHEDULEPRINT" function.*

Example:

```
SCHEDULEPRINTOFATTACHMENT(1)
```

void SCHEDULEPRINTOFATTACHMENT(int id, string filename)

Use: *Script (where a comment is expected)*

Description: *The function schedules the printing of a file attachment immediately after the successful execution of the script. The "id" parameter indicates the ID of the file attachment that will be printed. Printing is performed to a file under the name specified by the "filename" parameter. The file extension is retained by renaming. If multiple "SCHEDULEPRINTOFATTACHMENT" function calls are used in the script, the output is multiple printed files packaged in a "zip" archive, which is then offered for download. The function call can also be freely combined with the "SCHEDULEPRINT" function.*

Example:

```
SCHEDULEPRINTOFATTACHMENT(1, Attachment)
```

void SCHEDULEPRINTNAME(string filename)

Use: *Script (where a comment is expected)*

Description: *The function schedules the print report file name specified by the "filename" parameter. The use of the function is only relevant in scripts that are used to print to a print template.*

Example:

```
SCHEDULEPRINTNAME(Invoice - #ng_numberfv#)
```

void SCHEDULESAVE()

Use: *Script (where a comment is expected)*

Description: *The function schedules the saving of the currently opened record in the edit form immediately after the successful execution of the script.*

Example:

```
SCHEDULESAVE()
```

void SCHEDULESAVE(string dbname)

Use: *Script (where a comment is expected)*

Description: *The function schedules the saving of the control of the currently opened record in the edit form immediately after the successful execution of the script.*

Example:

```
SCHEDULESAVE(ng_textbox)
```

void SCHEDULESCRIPT(int button)

Use: *Script (where a comment is expected)*

Description: *The function schedules the execution of the script assigned to the button in the view page specified by the “button” parameter immediately after the successful execution of the script. (The order of the comment line in the script does not matter, the scheduled script will only run after all script steps have been executed.) The “button” parameter can be found in the administrator mode in the tooltip, which appears after stopping the mouse pointer over the "Edit Button ID ..." link or over the "Edit script" link. If an error occurs during the execution of the scheduled script, this error is logged in the “Errors.log” log file. The normal user who ran the script is not alerted to this error in any way.*

Example:

```
SCHEDULESCRIPT(1)
```

void SCHEDULESCRIPT(int button, int id)

Use: *Script (where a comment is expected)*

Description: *The function schedules the execution of the script assigned to the button in the edit form specified by the "button" parameter immediately after the successful execution of the script. The "button" parameter can be found in the administrator mode in the tooltip, which appears after stopping the mouse pointer over the "Edit Button ID ..." link or over the "Edit script" link. The "id" parameter defines the ID of the record whose data will be used to run the script. After the completion of the script, the record with this ID will be saved in the database, if the button is set to save the record after the successful execution of the script. But the script "OnBeforeSave" and "OnAfterSave" will not run. If an error occurs during the execution of the scheduled script, this error is logged in the “Errors.log” log file. The normal user who ran the script is not alerted to this error in any way.*

Example:

```
SCHEDULESCRIPT(1, 1)
```

void SCHEDULESYNC()

Use: *Script (where a comment is expected)*

Description: *The function schedules a complete synchronization of all database controls (not just the changed ones) when saving the currently open record in the edit form.*

Example:

```
SCHEDULESYNC()
```

void SCHEDULEURL(string url)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the web page specified by the "url" parameter immediately after the successful execution of the script.*

Example:

```
SCHEDULEURL(https://www.netgenium.com)
```

void SCHEDULEVIEWPAGE(int viewpage)

Use: *Script (where a comment is expected)*

Description: *The function schedules the opening of the view page specified by the "viewpage" parameter immediately after the successful execution of the script. The "viewpage" parameter can be found in the administrator mode in the tooltip that appears after hovering the mouse pointer over the "Edit view page" link.*

Example:

```
SCHEDULEVIEWPAGE(1)
```

double SIN(double number)

Use: *HTML, Script*

Description: *The function returns the sine from the value specified by the "number" parameter.*

Example:

```
SIN(-1.23)
// Returns "-0,942488801931697"
SIN(1.23)
// Returns "0,942488801931697"
SIN(#A#)
```

void SKIPDELETE()

Use: *Script (where a comment is expected)*

Description: *The function prevents deleting the currently open record in the edit form.*

Example:

```
SKIPDELETE()
```

void SKIPSAVE()

Use: *Script (where a comment is expected)*

Description: *The function prevents the currently open record from being saved in the edit form.*

Example:

```
SKIPSAVE()
```

void SLEEP(int milliseconds)

Use: *Script (where a comment is expected)*

Description: *The function pauses the script for "milliseconds" milliseconds.*

Example:

```
SLEEP(1000)
```

int SPLIT(string value, string separator)

Use: *Script*

Description: *The function returns the number of elements in the text string specified by the "value" parameter divided by the value defined by the "separator" parameter.*

Example:

```
SPLIT(A;B;C, ";") // Returns "3"
SPLIT(1;2;5;6;8, ";") // Returns "5"
```

string SPLIT(string value, string separator, int index)

Use: *Script*

Description: *The function returns an element that is at the position specified by the "index" parameter in the text string specified by the "value" parameter, divided according to the value defined by the "separator" parameter. The index uses zero numbering.*

Example:

```
SPLIT(A;B;C, ";", 0) // Returns "A"
SPLIT(1;2;5;6;8, ";", 3) // Returns "6"
```

string SQL(string query)

Use: *HTML, JavaScript, Script*

Description: *The function returns the value from the first row and the first column of the database query specified by the "query" parameter. If commas are included in the database query, you must enclose the entire query in double quotes.*

Example:

```
SQL(SELECT * FROM susers)
```

string SQL(string query, string separator)

Use: *HTML, JavaScript, Script*

Description: *The function returns the result of the database query specified by the "query" parameter. If commas are included in the database query, you must enclose the entire query in double quotes. The query result values are separated by the separator specified by the "separator" parameter. The delimiter can be defined for each individual column separately as a semicolon-separated list of values. In this case, as many delimiters as the number of columns – 1 must be defined.*

Example:

```
SQL("SELECT id, loginname, lng FROM susers", " - ")
SQL("SELECT id, loginname, lng FROM susers", " - ; -- ")
```

string SQL(string query, string columnSeparator, string lineSeparator)

Use: *HTML, JavaScript, Script*

Description: *The function returns the result of the database query specified by the "query" parameter. If commas are included in the database query, you must enclose the entire query in double quotes. The columns of the query result values are separated by the "columnSeparator" parameter, and the individual rows are separated by the "lineSeparator" parameter. The "columnSeparator" can be defined for each individual column separately as a semicolon-separated list of values. In this case, as many delimiters as the number of columns – 1 must be defined.*

Example:

```
SQL("SELECT id, loginname, lng FROM susers", " - ", <br>)
SQL("SELECT id, loginname, lng FROM susers", " - ; -- ", <br>)
```

string SQLARRAY(string query)

Use: *JavaScript*

Description: *The function returns the result of the database query specified by the "query" parameter. If commas are included in the database query, you must enclose the entire query in double quotes. The database query returns an array of fields in the form of javascript.*

Example:

```
SQLARRAY(SELECT * FROM susers)
```

Example result:

```
[
   [1, 'True', 0, 0, 0, '', '', '', '', 'Administrator', 'pn1Xly6NKqI=', 'cs', '', ''],
   [2, 'True', 0, 0, 0, '', '', '', '', 'Anonymous', 'pn1Xly6NKqI=', 'cs', '', '']
]
```

string SQLQUERY(string control)

Use: *Script*

Description: *The function returns the SQL query of the control specified by the "control" parameter. The "control" parameter indicates the name of the control element according to the conventions used in variables or print templates:*

- *datagridX*
- *DGX*

Example:

```
SQLQUERY(datagrid1)
SQLQUERY(DG123)
```

string SQLTABLE(string query, string key1, [string key2, string key3…])

Use: *An expression in the virtual column of a statistical query. The "SQLTABLE" function is used as an optimized variant of the "SQL" function, whose call in each row of the statistical query leads to problems with*

the loading speed of the resulting statistical query – the "SQL" function is run as many times as there are rows in the loaded query.

- ***Description:*** *The function reads the database query specified by the "query" parameter, which is defined by grouping the data of the "GROUP BY" clauses, and returns the value of the last column from "query" for the row defined by the "key1", "key2", etc.… The query result is loaded into virtual table in memory, and a key-value dictionary is built from the individual rows of the table. The number of table columns that define a unique key is determined by the number of keys used in the "key1", "key2", etc. parameters. The column that defines the value is always the last column in the query.*

- ***Example*** *- non-optimized variant using the "SQL" function:*

```
Number of items@SQL("SELECT COUNT(*) FROM ng_item WHERE pid = FORMATINTSQL(#c0#)")
```

- ***Example*** *– optimized variant using the "SQLTABLE" function:*

```
Number of items@SQLTABLE("SELECT pid, COUNT(*) FROM ng_item GROUP BY pid", #c0#)
```

double SQUARE(double number)

- ***Use:*** *HTML, Script*

- ***Description:*** *The function returns the square root of the positive number specified by the "number" parameter.*

- ***Example:***

```
SQUARE(-5.22)
// Returns "Not a number"
SQUARE(5.22)
// Returns "2,28473193175917"
SQUARE(#A#)
```

string[] StringArray(string text1; string text2; …)

- ***Use:*** *Script*

- ***Description:*** *The function declares a script variable of the field type of values "String" from individual values specified by parameters "text1", "text2", etc. The function parameters are separated by a semicolon or a tab. The "StringArray" function call must be case sensitive in the function name.*

- ***Example:***

```
StringArray(a;b;c)
```

```
StringArray(a#tab#b#tab#c)
```

int SUBSTRACTDATES(DateTime date1, DateTime date2)

Use: *HTML, JavaScript, Script*

Description: *The function returns the number of minutes defined as the difference between the date specified by the "date1" parameter and the date specified by the "date2" parameter.*

Example:

```
SUBSTRACTDATES(#today# + 14, #today#)
// Returns "20160" (14 * 60 * 24)
```

double SUBSTRACTDATES2(DateTime date1, DateTime date2)

Use: *HTML, JavaScript, Script*

Description: *The function returns the number of seconds defined as the difference between the date specified by the "date1" parameter and the date specified by the "date2" parameter.*

Example:

```
SUBSTRACTDATES2(#today# + 14, #today#)
// Returns "1209600" (14 * 60 * 24 * 60)
```

int SUBSTRACTINTEGERS(int value1, in value2)

Use: *HTML, JavaScript, Script*

Description: *The function returns the difference of the two integers specified by the "value1" and "value2" parameters.*

Example:

```
SUBSTRACTINTEGERS(10, 1)
// Returns "9"
```

string SUBSTRING(string text, string startindex, int length)

Use: *HTML, JavaScript, Script*

Description: *The function returns a substring from the string specified by the "text" parameter, starting with the character at the position stored in the "startindex" parameter with the length stored in the "length" parameter. If the value of the "length" parameter is "-1", it returns a substring starting with the character at the position stored in the "startindex" parameter to the end of the string.*

Example:

```
SUBSTRING(Hello World, 6, 5)
// Returns "World"
SUBSTRING(Hello World, 6, -1)
// Returns "World"
```

double SUM(object[] array)

Use: *Script*

Description: *The function returns the sum of the values of the script variable specified by the "array" parameter, if the variable is an array of values. If the variable is not an array of values, the function returns the value of the variable.*

Example:

```
SUM(#A#)
```

void TABLE(string id)

Use: *HTML control name – The body of the control must be left blank*

Description: *The function generates an HTML tag "<table id="{id}" cellspacing="0" cellpadding="0"><tr>", Resp. start a new table. The tag ID defines the "id" parameter.*

Example:

```
TABLE(T1)
```

void /TABLE

Use: *HTML control name – The body of the control must be left blank*

Description: *The function generates an HTML tag "</table>", Resp. the end of the previous table that was generated using the "TABLE" function.*

Example:

```
/TABLE
```

double TAN(double number)

Use: *HTML, Script*

Description: *The function returns the tangent from the value specified by the "number" parameter.*

Example:

```
TAN(-0.56)
// Returns "-0,626949535052698"
TAN(1)
// Returns "1,557407724654902"
TAN(#A#)
```

void TD

Use: *HTML control name – The body of the control must be left blank*

Description: *The function generates an HTML tag "<td>", Resp. start a new table cell.*

Example:

```
TD
```

void TEFUIG(string message, string group1, string group2, …)

Use: *Script*

Description: *The function triggers an interrupt with an error message specified by the "message" parameter to all users who are members of the "group1", "group2" ... groups, and terminates the execution of the script. The number of parameters with a list of groups is arbitrary.*

Example:

```
TEFUIG(You are not authorized to perform this activity., 6, 21)
TEFUIG("Message containing a comma.", 6, 21)
```

string TEMPPASSWORD()

Use: *HTML, JavaScript, Script*

Description: *The function generates a random number that is used as a temporary password.*

Example:

```
TEMPPASSWORD()
```

void TESUIG(string message, string group1, string group2, …)

Use: *Script*

Description: *The function triggers an interrupt with an error message specified by the "message" parameter to all users who are not members of the "group1", "group2" ... groups, and terminates the execution of the script. The number of parameters with a list of groups is arbitrary.*

Example:

```
TESUIG(You are not authorized to perform this activity., 9, 3)
TESUIG("Message containing a comma.", 9, 3)
```

void THROWEXCEPTION(string exception)

Use: *HTML, Script*

Description: *The function triggers an interrupt with an error message specified by the "exception" parameter.*

Example:

```
THROWEXCEPTION(abc)
```

int TIME2INT(string time)

Use: *HTML, JavaScript, Script*

Description: *The function returns the number of minutes converted from the text string – time – specified by the "time" parameter in the format "HH:mm".*

Example:

```
TIME2INT(08:00)
// Returns "480"
TIME2INT(00:00)
// Returns "0"
```

int TIME2MINUTES(string time)

Use: *HTML, JavaScript, Script*

Description: *The function returns the number of minutes converted from the text string – time – specified by the "time" parameter in the format "HH:mm".*

Example:

```
TIME2MINUTES(08:00)
// Returns "480"
TIME2MINUTES(00:00)
// Returns "0"
```

double TIME2SECONDS(string time)

Use: *HTML, JavaScript, Script*

Description: *The function returns the number of seconds converted from the text string – time – specified by the "time" parameter in the format "HH:mm:ss".*

Example:

```
TIME2SECONDS(00:08:00)
// Returns "480"
TIME2SECONDS(00:00:00)
// Returns "0"
```

void TR

Use: *HTML control name – The body of the control must be left blank*

Description: *The function generates an HTML tag "<tr>", Resp. start a new table row.*

Example:

```
TR
```

string TRANSLATE(string text, string language)

Use: *JavaScript*

Description: *The function returns the language translation of the multilingual term specified by the "text" parameter. The text is translated according to the language setting defined by the "language" parameter.*

Example:

```
TRANSLATE(Schvalování bylo stornováno#en:Approval has been canceled,
#language#)
```

string TREECHILDNODES(string dbname, int id)

Use: *HTML, JavaScript, Script*

Description: *The function returns a list of IDs separated by semicolons, obtained from the table used in the tree structure. The IDs represent all descendants of the record specified by the "id" parameter. The search is performed recursively, which means that the function searches not only for direct descendants, but also for their further child records.*

Example:

```
TREECHILDNODES(ng_abc, 1)
// Returns "2;3;4"
```

string TRIM(string text)

Use: *HTML, JavaScript, Script*

Description: *The function returns a text string specified by the "text" parameter, in which spaces and unprintable characters have been removed at the beginning and at the end.*

Example:

```
TRIM(   Good morning   )
// Returns "Good morning"
```

string TRIM(string text, string trimchars)

Use: *HTML, JavaScript, Script*

Description: *The function returns a text string specified by the "text" parameter, in which all characters specified in the "trimchars" parameter have been removed at the beginning and at the end.*

Example:

```
TRIM(010, 023)
// Returns "1"
```

string TRIMEND(string text)

Use: *HTML, JavaScript, Script*

Description: *The function returns a text string specified by the "text" parameter, in which spaces and unprintable characters have been removed at the end.*

Example:

```
TRIMEND(Good morning )
// Returns "Good morning"
```

string TRIMEND(string text, string trimchars)

Use: *HTML, JavaScript, Script*

Description: *The function returns a text string specified by the "text" parameter, in which all characters specified in the "trimchars" parameter have been removed at the end.*

Example:

```
TRIMEND(010, 023)
// Returns "01"
```

string TRIMSTART(string text)

Use: *HTML, JavaScript, Script*

Description: *The function returns a text string specified by the "text" parameter, in which spaces and unprintable characters have been removed at the beginning.*

Example:

```
TRIMSTART( Good morning)
// Returns "Good morning"
```

string TRIMSTART(string text, string trimchars)

Use: *HTML, JavaScript, Script*

Description: *The function returns a text string specified by the "text" parameter, in which all characters specified in the "trimchars" parameter have been removed at the beginning.*

Example:

```
TRIMSTART(010, 023)
// Returns "10"
```

void UNLOCKPASSWORDS()

Use: *Script*

Description: *The function enables the setting of a new password in the user edit form.*

Example:

```
UNLOCKPASSWORDS()
```

void UPDATEAUTOSUM(int id, int form)

Use: *Script (where a comment is expected)*

Description: *The function updates all sums of nested records that are part of the record specified by the "id" parameter in the edit form specified by the "form" parameter.*

Example:

```
UPDATEAUTOSUM(1, 1)
UPDATEAUTOSUM(#id#, #form#)
```

void UPDATEAUTOSUM(string id, int form)

Use: *Script (where a comment is expected)*

Description: *The function updates all sums of nested records that are part of the records specified by the "id" parameter (field of values separated by a semicolon) in the edit form specified by the "form" parameter.*

Example:

```
UPDATEAUTOSUM(1;2;3, 1)
```

void UPDATEAUTOSUM(int id, string dbname)

Use: *Script (where a comment is expected)*

Description: *The function updates all sums of nested records that are part of the record specified by the "id" parameter in the edit form specified by the "dbname" parameter.*

Example:

```
UPDATEAUTOSUM(1, susers)
```

void UPDATEAUTOSUM(string id, string dbname)

Use: *Script (where a comment is expected)*

Description: *The function updates all sums of nested records that are part of the records specified by the "id" parameter (semicolon-separated value fields) in the edit form specified by the "dbname" parameter.*

Example:

```
UPDATEAUTOSUM(1;2;3, susers)
```

string UPPER(string text)

Use: *HTML, JavaScript, Script*

Description: *The function returns the text string specified by the "text" parameter, in which all characters are converted to uppercase.*

Example:

```
UPPER(abc)
```

```
// Returns "ABC"
```

string URLDECODE(string text)

Use: *Script*

Description: *The function decodes all URL hexadecimal characters in the text string specified by the "text" parameter.*

Example:

```
URLDECODE(https%3a%2f%2fwww.netgenium.com)
// Returns "https://www.netgenium.com"
```

string URLENCODE(string text)

Use: *Script*

Description: *The function encodes all URL invalid characters (eg a space) in the text string specified by the "text" parameter after the corresponding sequence of characters in hexadecimal.*

Example:

```
URLENCODE(https://www.netgenium.com)
// Returns "https%3a%2f%2fwww.netgenium.com"
```

string USERDETAILS(string columnname)

Use: *HTML, JavaScript, Script*

Description: *The function returns the value stored in the user database table in the column specified by the "columnname" parameter on the row that relates to the currently logged in user.*

Example:

```
USERDETAILS(id)
USERDETAILS(loginname)
```

string USERGROUP(string loginname)

Use: *HTML, JavaScript, Script*

Description: *The function returns a semicolon-separated list of all user group names in which the user specified by the "loginname" parameter is a member.*

Example:

```
USERGROUP(Administrator)
USERGROUP(#loginname#)
```

string USERINGROUP(string group1 [,… string group2]))

Use: *HTML, JavaScript, Script*

Description: *The function returns the value "x" if the currently logged in user is a member of the user group specified by the "group1" parameter, or a member of more groups specified by other parameters. The function returns the value "" if it is not a member of this user group. "Group" can be the ID of the user group or its name.*

Example:

```
USERINGROUP(1)
USERINGROUP(Administrators)
USERINGROUP(1, 2)
USERINGROUP(Administrators, Users)
// Returns "" or "x"
```

string USERSINGROUP(int group1 [,… int group2])

Use: *HTML, JavaScript, Script*

Description: *The function returns a list of all user IDs separated by a semicolon who are members of a group with the specified ID in the "group1" parameter, or multiple groups specified by other parameters.*

Example:

```
USERSINGROUP(1)
USERSINGROUP(1, 2)
// Returns for example "1", "1;2" etc.
```

string USERSINGROUP(string group1 [,… string group2])

Use: *HTML, JavaScript, Script*

Description: *The function returns a list of all user IDs separated by a semicolon who are members of a group with the specified ID in the "group1" parameter, or multiple groups specified by other parameters.*

Example:

```
USERSINGROUP(Administrators)
USERSINGROUP(Administrators, Users)
// Returns for example "1", "1;2" etc.
```

string USERSINGROUP2(int group1 [,… int group2])

Use: *HTML, JavaScript, Script*

Description: *The function returns a list of all user-separated user logins that are members of a group with the specified ID in the "group1" parameter, or multiple groups specified by other parameters.*

Example:

```
USERSINGROUP2(1)
USERSINGROUP2(1, 2)
// Returns for example "Administrator", "Administrator;Anonymous" etc.
```

string USERSINGROUP2(string group1 [,… string group2])

Use: *HTML, JavaScript, Script*

Description: *The function returns a list of all user-separated user logins that are members of a group with the specified ID in the "group1" parameter, or multiple groups specified by other parameters.*

Example:

```
USERSINGROUP2(Administrators)
USERSINGROUP2(Administrators, Users)
// Returns for example "Administrator", "Administrator;Anonymous" etc.
```

string VALIDEMAIL(string email)

Use: *HTML, JavaScript, Script*

Description: *The function tests the correct format of the e-mail address specified by the "email" parameter. If so, the function returns an "x". If the e-mail address is not in the correct format, the function returns an empty string.*

Example:

```
VALIDEMAIL(info@netgenium.com)
// Returns "x"
```

void WEBFORM

Use: *Note of the edit form or HTML control name (the body of the control must be left blank)*

Description: *The function starts a block of controls that do not use a colored background in the left column with the control's name, and that use a reverse design – the controls do not have a frame, have a colored background, and display a colored bar below the control.*
A started block that is not terminated with the "/WEBFROM" function anchors the "Save", "Delete" and "Back" buttons. These buttons will then not be floating, and will always be displayed at the end of the edit form.

Example:

```
WEBFORM()
```

void /WEBFORM

Use: *HTML control name – The body of the control must be left blank*

Description: *The function ends a block of controls that do not use a colored background in the left column with the control's name, and that use a reverse design – the controls do not have a frame, have a colored background, and display a colored bar below the control when focused.*

Example:

```
/WEBFORM
```

void WRITEMESSAGE(string message)

Use: *Script (where a comment is expected)*

Description: *The function prints the message specified by the "message" parameter immediately after the successful execution of the script. The function cannot be used in the "OnBeforeSave" and "OnAfterSave" scripts.*

Example:

```
WRITEMESSAGE(Script execution completed successfully.)
WRITEMESSAGE("Message, that contains a comma.")
```

void WRITEMESSAGE(string message, bool dialog)

Use: *Script (where a comment is expected)*

Description: *The function prints the message specified by the "message" parameter immediately after the successful execution of the script, and also displays it in the dialog box. The function cannot be used in the "OnBeforeSave" and "OnAfterSave" scripts.*

Example:

```
WRITEMESSAGE(Script execution completed successfully., true)
WRITEMESSAGE("Message, that contains a comma.", true)
```

int YEAR(DateTime date)

Use: *HTML, JavaScript, Script*

Description: *The function returns the year number from the date specified by the "date" parameter.*

Example:

```
YEAR(15.2.2013)
// Returns "2013"
YEAR(#now#)
```