



eScada

v24.2.0  
Script

## Table of Contents

Script.....	3
Assignments =.....	3
Evaluate( tagname, mode);.....	4
Sleep( time );.....	4
DeviceTag( tagname, mode);.....	5
DeviceObject( objecttype, objectname, mode);.....	5
AssignPointer( tagpointer, newreference);.....	6
AssignItem( itempointer, tag_newreference, itemid_newrefedrence);.....	7
Bitwise( item, action, bitnumber);.....	8
SetArray( tag ) { itemid = expr1, ... };.....	9
SetObjectProperty( objtype, objname, property, value);.....	10
SystemCommand( cmd );.....	12
Copy(source, sourceitem, destination, items, destinationitem, mode);.....	13
Translate( sourcestring, destinationstring, languageid);.....	14
Comments.....	15

# Script

A script is helpful to execute a list of functions sequentially instead of using triggers or other events. The script object itself can be executed using trigger, events or called even from another script using the function called “evaluate”.

## Assignments =

In order to assign a value to another tag element the operator = must be used.

For string assignments please observe that the right part of an assignment must be constant or single item only; in other words as it's easy to imagine, the right part can not be an expression.

String constants must be expressed using double quotes char.

remarks: if you need to set many items for the same tag, you should consider to use the *setarray* function instead as it is faster.

Examples:

```
//
// booleans
//
'bits:1' = 1; // numeric constant
'bits:2' = ('array.numb:19' > 10) and ('array.bit:19' = 1); // boolean expression
'bits:8' = ("text:1" like "*home*"); // boolean expression

//
// numbers
//
'numb:0' = '$SYS.Server.UITime:0'; // item assignment - hour
'numb:1' = '$SYS.Server.UITime:1'; // item assignment - minute
'numb:2' = '$SYS.Server.UITime:2'; // item assignment - seconds
'numb:3' = ('$SYS.Server.UITime:1' * '$SYS.Server.UITime:2') * 12.45; //expression

//
// strings
//
'text:0' = '$SYS.Server.SdateTime:5'; // item assignment
'text:1' = "Amazing string !"; // string constant
'text:2' = if (("texta:0" in "textb:0"), 'numb:0', "Not valid !"); // mixed
```

## Evaluate( tagname, mode);

Every object used as a derived variable is evaluated on events such as values changing or triggers. This function permits evaluation of a derived object even if its event or trigger is not raised.

Not all derived objects can be evaluated using this function, below is the list of unusable objects:

Public or Private tags, Counter, TimeCounter, Integrator, OnDelay, OffDelay, Notification, Stats, Pointer, Item, Pulse, Stack, Derivative, Series, Interpolation, Weihenstephan, ModbusTcp, OpcUA

All other objects not listed can be evaluated sequentially in a script using “evaluate” function.

### Parameters

#### **tagname**

A valid tag-name of type not listed above.

#### **mode**

Mode parameter can be one of these values:

**sync** it permits evaluation of objects waiting for their evaluation end

**async** it permits evaluation of objects asynchronously without waiting for their evaluation end

### Examples:

```
//  
// synchronous evaluation  
//  
Evaluate('WriteFile', sync);
```

```
//  
// asynchronous evaluation  
//  
Evaluate('ExecuteSql', async);
```

## Sleep( time );

This function is helpful to pause the script execution for the given milliseconds time.

#### **time**

It must be greater than zero; it's expressed in milliseconds.

### Examples:

```
Sleep('tagname:3');           // function parameterized using an item value  
Sleep('tagname:3' + 'tagname:1'); // function parameterized using an expression  
Sleep(1000);                  // function parameterized using a constant
```

## DeviceTag( tagname, mode);

This function is helpful to execute a synchronous read or write action for tag objects. The 'tagname' must be a device-tag object or a pointer to device-tag object.

### Parameters

#### **tagname**

A valid tag-name as defined above.

#### **mode**

Mode parameter can be one of these values:

**read**        it permits the force a read action  
**write**       it permits the force a write action

### Examples:

```
//  
// Read  
//  
DeviceTag('mytagname', read);  
  
//  
// Write  
//  
DeviceTag('mytagname', write);
```

## DeviceObject( objecttype, objectname, mode);

This function is helpful to execute a synchronous read or write action for a device object such as tag or group. The 'objectname' must be a device-tag or a device-group. You can use string constants or tags of type string to specify the object name.

### Parameters

#### **objecttype**

Object type parameter can be one of these values:

**tag**  
**group** (only **read** mode can be used for these objects)

#### **objectname**

A valid tag-name or device group-name as defined above.

#### **mode**

Mode parameter can be one of these values:

**read**        it permits the force a read action  
**write**       it permits the force a write action

Examples:

```
//  
// Read  
//  
DeviceObject(tag, 'mytagname:0', read);  
in this case the value of 'mytagname:0' will be used as tagname to read
```

```
DeviceObject(tag, "mytagname", read);  
in this case "mytagname" will be used as tagname to read
```

```
DeviceObject(group, "channel\device\group", read);  
It is important to observe that in case of groups the entire path must be specified
```

```
//  
// Write  
//  
DeviceObject(tag, 'mytagname:0', write)
```

**AssignPointer( tagpointer, newreference);**

This function is helpful to change a pointer to tag reference.  
The 'tagpointer' must be a pointer-tag object.

Parameters***tagpointer***

A valid tag-name as defined above.

***newreference***

The new pointer to tag reference can be expressed using a string constant or an item object.  
In the event of string constant please observe that its existence cannot be checked during script editing.  
In the event of item object, it must be of string type.

Examples:

```
//  
// New reference as constant string  
//  
AssignPointer('mytagname', "newtagname");
```

```
//  
// New reference as item content  
//  
AssignPointer('mytagname', 'references:2');
```

## AssignItem( itempointer, tag\_newreference, itemid\_newreference);

This function is helpful to change a pointer to item reference.  
The 'itempointer' must be a pointer-item object.

### Parameters

#### ***itempointer***

A valid tag-name as defined above.

#### ***tag\_newreference***

The new pointer to item reference can be expressed using a string constant or an item object.  
In the event of string constant please observe that its existence can't be checked during script editing. In the event of item object, it must be of string type.  
In order not to change the original tag reference, this parameter can assume the value of **null**

#### ***itemid\_newreference***

The new item-ID reference can be expressed using a numeric constant or an expression.

### Examples:

```
//  
// New reference as constant string and item-ID as numeric constant  
//  
AssignItem('mytagname', "newtagname", 10);  
  
//  
// New reference as constant string and item-ID as expression  
//  
AssignItem('mytagname', "newtagname", ('tagname:0' + 'othertagname:2'));  
  
//  
// New reference as item content and item-ID as numeric constant  
//  
AssignItem('mytagname', 'references:2', 0);  
  
//  
// New reference as item content and item-ID as expression  
//  
AssignItem('mytagname', 'references:2', 'tagname:10');  
  
//  
// This keep the original tag reference and changes the item-ID only  
//  
AssignItem('mytagname', null, 'tagname:10');
```

## Bitwise( item, action, bitnumber);

This function is helpful to execute a bitwise action

### Parameters

#### **item**

A valid item of integer type.  
8bit, 16bit, 32bit or 64bit

#### **action**

Action parameter can be one of these values:

**set**            it permits the setting of the given bit  
**reset**        it permits resetting of the given bit  
**toggle**       it permits toggling of the given bit

#### **bitnumber**

A valid expression which returns an integer value or constant number.  
This parameter value can not exceed the amount of bits related to given item.

### Examples:

```
//  
// It will set the first bit of given item  
//  
Bitwise('itemname:0', set, 0);
```

```
//  
// It will toggle the 32nd bit of given item  
//  
Bitwise('itemname:0', toggle, 31);
```

```
//  
// It will reset the 2nd bit of given item  
//  
Bitwise('itemname:0', reset, 1);
```



## **SetArray( tag ) { itemid = expr1, ... };**

If you need to set many tag items, this function is a faster alternative to many single assignments. The higher the amount of array items you want to set, the faster the execution will be. String constants must be expressed using double quotes char.

### Parameters

#### **tag**

A valid not read-only tag-name.

#### **itemid**

It is the item ID you want to set.

It is expressed as a constant number.

#### **expr1, ...**

It is the expression which its value will be set as item value

### Example:

// Instead of setting every single elements using this code

```
'mytagname:0' = 'tagname:0' + 'othertagname:2';
```

```
'mytagname:1' = 12.34;
```

```
'mytagname:4' = 'tagname:10';
```

```
'mytagname:8' = 0;
```

// You should take in consideration to use *Setarray* function

// Because it will result in a faster execution.

```
SetArray('mytagname')
```

```
{
```

```
    0 = 'tagname:0' + 'othertagname:2',
```

```
    1 = 12.34,
```

```
    4 = 'tagname:10',
```

```
    8 = 0
```

```
};
```

## SetObjectProperty( objtype, objname, property, value);

If you need to change properties to runtime objects this function must be used.  
Runtime objects are listed below.

### Parameters

#### **objtype**

<b>protocol</b>	Channel protocol
<b>device</b>	Device
<b>devicetag</b>	Device tag
<b>derived</b>	Derived channel
<b>derivedtag</b>	Derived tag
<b>item</b>	Tag item

#### **objname**

Object name must be an item of type string containing the object name or a string constant.  
For objects of type **device**, **devicegrp**, **derivedgrp** object name must contain the entire path.

e.g. myprotocolchannel\mydevice, myprotocolchannel\mydevice\mygroup,  
myderivedchannel\mygroup

#### **property**

Property name must be an item of type string containing the property name or a string constant  
Below is valid property names for each available object.

<u>Object</u>	<u>Valid property name and data type</u>
<b>protocol</b>	<b>offline</b> (boolean)
<b>device</b>	<b>offline</b> (boolean)
<b>devicetag</b>	<b>offline</b> (boolean), <b>polling</b> (integer)
<b>derived</b>	<b>skip</b> (boolean)
<b>derivedtag</b>	<b>skip</b> (boolean)
<b>item</b>	<b>note</b> (string), <b>unit</b> (string), <b>log</b> (boolean), <b>publish</b> (boolean)

### Property name meaning

<b>offline</b>	It permits to put offline an object
<b>polling</b>	It permits to change the polling time of a tag object
<b>skip</b>	It permits to skip the evaluation of a derived object
<b>note</b>	It permits to set an item note
<b>unit</b>	It permits to set an item unit
<b>log</b>	It permits to set an item log flag
<b>publish</b>	It permits to set an item publish flag

**value**

Property value must be an item of any type containing the property value or a string constant.

For properties using Boolean values you can use these values:

- to specify TRUE: **1** or **true** or **on**
- to specify FALSE: **0** or **false** or **off**

For properties using Numeric values you can set numbers using a valid syntax.

Valid syntax examples: **1 0.034 1234.56 -23**

For properties using String values you can simply type the string you need.

Examples:

String constants instead of items of type String.

```
SetObjectProperty(protocol, "myprotocolchannel", "offline", "0");  
SetObjectProperty(device, "myprotocolchannel\\mydevice", "offline", "false");  
SetObjectProperty(devicetag, "mytagname", "offline", "false");  
SetObjectProperty(devicetag, "mytagname", "polling", "2000");
```

```
SetObjectProperty(derived, "myderivedchannel", "skip", "true");  
SetObjectProperty(derivedtag, "mytagname", "skip", "on");
```

```
SetObjectProperty(item, "mytagname:0", "note", "my item note");  
SetObjectProperty(item, "mytagname:12", "unit", "mm");  
SetObjectProperty(item, "mytagname:22", "log", "false");  
SetObjectProperty(item, "mytagname:25", "publish", "false");
```

Items of type String instead of string constants.

```
SetObjectProperty(device, 'mytagname:1', 'matagname:2', "false");  
SetObjectProperty(devicetag, 'mytagname:1', "offline", 'mytagname:2');  
SetObjectProperty(item, 'mytagname:1', "unit", 'mytagname:2');
```

## SystemCommand( cmd );

This function is helpful to execute a system command.

### **cmd**

- 0 = Log-off all connected users
- 1 = Acknowledge all alarm messages
- 2 = Acknowledge all user messages
- 3 = Unlock recipes (*Useful only in case of problem using recipes handler*)
- 4 = Disconnect all connected clients
- 5 = Disconnect all connected clients, except the first one
- 6 = Shutdown all connected clients
- 7 = Shutdown all connected clients, except the first one
- 8 = Shutdown server
- 9 = Save tag values
- 10 = Toggle server GUI

Examples:

```
SystemCommand('tagname:3');    // function parameterized using an item value  
SystemCommand(8);              // function parameterized using a constant
```

**Copy(source, sourceitem, destination, items, destinationitem, mode);**

This function helps to copy an array portion to another array.

Parameters**source**

A valid tag-name.

**sourceitem**

Source starting item.

It could be a constant or an expression.

**destination**

A valid tag-name.

It must be a not read-only tag.

**items**

Items to copy from source.

It could be a constant or an expression.

**destinationitem**

Destination starting item.

It could be a constant or an expression.

**mode**

**all** = The entire destination array will be write with a single protocol frame.

**single** = Items in destination array will be write one by one.

Examples:

```
copy(  
    'dbg03.source',           // Source tag  
    2,                       // Source start item  
    'dbg03.destination',     // Destination tag  
    10,                      // Items to copy  
    4,                       // Destination start item  
    all                      // Copy items mode  
);  
  
copy(  
    'dbg03.source',           // Source tag  
    'dbg03.srcitem:0',       // Source start item  
    'dbg03.destination',     // Destination tag  
    'dbg03.items:0',         // Items to copy  
    'dbg03.desitem:0',       // Destination start item  
    all                      // Copy items mode  
);
```

## Translate( sourcestring, destinationstring, languageid);

This function is helpful to translate an user string.

### Parameters

#### **sourcestring**

Source single item string.

#### **destinationstring**

Destination single item string.

It must be a not read-only item.

#### **languageid**

Language ID (from 1 to 16)

It could be a constant or an expression (single item).

### Examples:

```
//  
// Translate a string using language id as numeric constant  
//  
Translate('mysourcestring:3', 'mydestinationstring:0', 10);  
  
//  
// Translate a string using language id as numeric constant  
//  
Translate('mysourcestring:3', 'mydestinationstring:0', 'mylanguageid:0');
```

## Comments

C or C++ style comments can be used

examples:

```
/*  
    my own comment ....  
    bla, bla, bla ....  
*/
```

```
/* my own comment .... */
```

```
// my own comment ....
```

```
//  
// my own comment ....  
// bla, bla, bla ....  
//
```