v24.2.0
Derived objects

Table of Contents

Timing

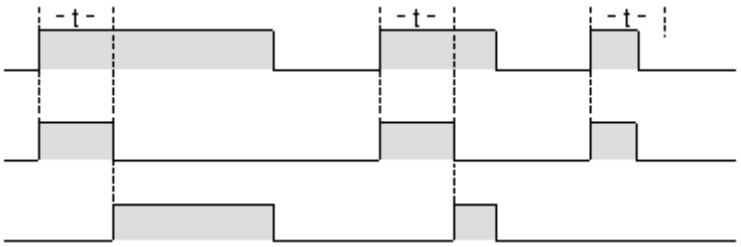
OnDelay

Evaluated on cycle ~10ms

Inputs:

Input	Boolean expression	TRUE/FALSE
Time [ms]	Analog expression	

Description:



Outputs:

Item 0:	time counting
Item 1:	as described above

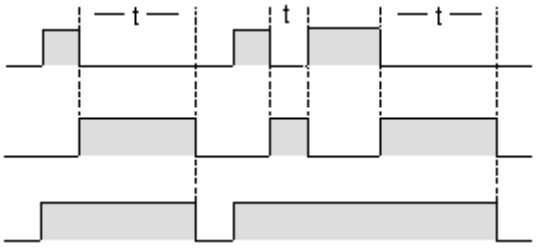
OffDelay

Evaluated on cycle ~10ms

Inputs:

Input	Boolean expression	TRUE/FALSE
Time [ms]	Analog expression	

Description:



Outputs:

Item 0:	time counting
Item 1:	as described above

TimeCounter

- ENABLE = FALSE
  Nothing will be evaluated, except for the RESET command.
- ENABLE = TRUE
  The item 0 is incremented by the elapsed time, every cycle that the INPUT event is TRUE.
- RESET = Depending on its event type
  Outputs forced to ZERO

Evaluated on cycle ~10ms or on inputs value changed

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Input	Boolean expression	TRUE/FALSE
Reset	Analog or boolean expression	If a unique tag is used, event can be Command
Reset event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)

Outputs:

Item 0:	Milliseconds with INPUT value TRUE

Pulse

It generates a pulse with the given periodicity and synchronized with the given time.
Mainly this variable is useful to execute timed actions using its output as a trigger for other tags.

Evaluated on cycle ~10ms or on inputs value changed

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Sync time		
Hour	Evaluated as constant	Value from 0 to 23
minute	Evaluated as constant	Value from 0 to 59
Second	Evaluated as constant	Value from 0 to 59

0,0,0 means midnight.

Periodicity

Days	Analog expression
Hours	Analog expression
Minutes	Analog expression
Seconds	Analog expression
Milliseconds	Analog expression

A positive value must be expressed for all components of periodicity

Outputs:

Item 0:	Events counter

Math

Counter

- ENABLE = FALSE
  Nothing will be evaluated, except for the RESET command.
- ENABLE = TRUE
  Every time that the INPUT event is TRUE, the item 0 is incremented using the Delta value.
- RESET = Depending on its event type
  Outputs forced to ZERO

Evaluated on inputs value changed

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Input	Analog or boolean expression	If a unique tag is used, event can be Command
Input event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Delta	Analog expression	
Reset	Analog or boolean expression	If a unique tag is used, event can be Command
Reset event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)

Outputs [Read only]:

Item 0:	Counter value
Item 1:	Events count

Function

This kind of variable is able to evaluate any analog or boolean expression.
Please refer to eScada.Notes.Expressions document for more information about this derived object

Evaluated on inputs value changed

Inputs:

Input	Analog or boolean expression

Comments:
C or C++ style comments can be used typing expression
examples:

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

/* my own comment .... */

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

Outputs:

Item 0:	Expression value

Integrator

This kind of variable is able to integrate the result of any analog expression.

Evaluated on cycle ~10ms or on inputs value changed

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Input	Analog expression	
T Reference	Evaluated as constant	0=ms, 1=seconds, 2=minutes, 3=hours
Reset	Analog or boolean expression	If a unique tag is used, event can be Command
Reset event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)

Outputs:

Item 0:	Integration of INPUT

Stats

- ENABLE = FALSE
  Nothing will be evaluated, except for the RESET command.
- ENABLE = TRUE
  If the TIMER expression is ZERO the output values will be evaluated every INPUT changing.
  If the TIMER expression is greater than ZERO the output values will be evaluated every TIMER expiration.
  TIMER value is expressed in milliseconds
- RESET = TRUE
  Outputs forced to ZERO

Evaluated on cycle ~10ms or on inputs value changed

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Signal	Analog expression	
Input	Analog or boolean expression	If a unique tag is used, event can be Command
Input event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Reset	Boolean expression	If a unique tag is used, event can be Command
Reset event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of the action) 4=Disabled (Evaluated in a script only)

Outputs:

Item 0:	Min value reached by the INPUT
Item 1:	Max value reached by the INPUT
Item 2:	Average value. (Sum / Samples)
Item 3:	Sum. (Sum = Sum + INPUT value)
Item 4:	Number of samples. (INPUT variation or TIMER expiration)

Series

It permits the calculation of sum, average, min and max value from a given series of numbers.

Evaluated on input tags change.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)
Start index	Analog expression	Tag starting index
Items	Analog expression	Number of items to consider for evaluation

Outputs:

Item 0:	0=OK, 1=Wrong starting index or items value
Item 1:	1=Error during values conversion
Item 2:	Valid items count
Item 3:	Sum
Item 4:	Average value. (Sum / valid Items)
Item 5:	Sum without min and max
Item 6:	Average without min and max
Item 7:	Min series value
Item 8:	Max series value

Interpolation

Interpolation provides a means of estimating the function at intermediate point; the reference.

Evaluated on input tags change.

Inputs:

X Series	Tag name	Tag name of numeric type but boolean
Y Series	Tag name	Tag name of numeric type but boolean
Reference	Analog expression	Reference value
Method	constant integer	0=Nearest value, 1=Linear, 2=Polynomial (Lagrange)

Outputs:

Item 0:	0=OK 1=The two series haven't got more then one element 2=The two series haven't got same elements 3=The X Series hasn't got incremental values
Item 1:	Reference value
Item 2:	Interpolated value
Item 3:	X Series, value a
Item 4:	X Series, value b
Item 5:	Y Series, value a
Item 6:	Y Series, value b
Item 7:	1=Value saturated using minimum value of Y Series
Item 8:	1=Value saturated using maximum value of Y Series

Derivative

This kind of variable is able to calculate the slope of a curve using two points.

Evaluated on trigger

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Trigger	Analog expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Value	Analog expression	

Outputs:

Item 0:	0=Output OK, 1=inf (number/0), 2=-nan (0/0)
Item 1:	Dv [Vunit] - (P2v-P1v)
Item 2:	Dt [ms] - (P2t-P1t)
Item 3:	Derivative value [Vunit/ms] – (P2v-P1v/P2t-P1t)

LinearScale

- ENABLE = FALSE
  Nothing will be evaluated. Output item 0 is forced to 0 and outputs from 1 to 3 are forced to FALSE

- ENABLE = TRUE
  The raw value INPUT is scaled using variable parameters.

Evaluated on inputs value changed.

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Raw value min	Analog expression	
Raw value max	Analog expression	
Engineering value min	Analog expression	
Engineering value max	Analog expression	
Input	Analog expression	
Value saturation	constant	1=Scaled value saturated to its Engineering limits

Outputs:

Item 0:	Scaled value
Item 1:	1=Value OK (in range)
Item 2:	1=Under range
Item 3:	1=Over range

Conversions

ToHex

Gets the value number in hexadecimal format

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)
Width	Number of characters needed.	constant – Integer - 0=No width

Outputs:

Item 0:	Hexadecimal number
Item 1:	0=Conversion OK, 1=Error

ToBinary

Converts the value number into a binary string

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type but boolean
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)
Width	constant integer	Integer from 0 to 64. (0 no width)

Outputs:

Item 0:	Binary string
Item 1:	0=Conversion OK, 1=Error

ToSInt8

Converts the given value into another value with a different type

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)

Outputs:

Item 0:	Converted value
Item 1:	0=Conversion OK, 1=Error

ToSInt16

Converts the given value into another value with a different type

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)

Outputs:

Item 0:	Converted value
Item 1:	0=Conversion OK, 1=Error

ToSInt32

Converts the given value in another value with adifferent type

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)

Outputs:

Item 0:	Converted value
Item 1:	0=Conversion OK, 1=Error

ToSInt64

Converts the given value into another value with a different type

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)

Outputs:

Item 0:	Converted value
Item 1:	0=Conversion OK, 1=Error

ToUInt8

Converts the given value into another value with a different type

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)

Outputs:

Item 0:	Converted value
Item 1:	0=Conversion OK, 1=Error

ToUInt16

Converts the given value into another value with a different type

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)

Outputs:

Item 0:	Converted value
Item 1:	0=Conversion OK, 1=Error

ToUInt32

Converts the given value into another value with a different type

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)

Outputs:

Item 0:	Converted value
Item 1:	0=Conversion OK, 1=Error

ToUInt64

Converts the given value into another value with a different type

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)

Outputs:

Item 0:	Converted value
Item 1:	0=Conversion OK, 1=Error

ToDouble

Converts the given value into another value with a different type

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)

Outputs:

Item 0:	Converted value
Item 1:	0=Conversion OK, 1=Error

ToFloat

Converts the given value into another value with a different type

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)

Outputs:

Item 0:	Converted value
Item 1:	0=Conversion OK, 1=Error

HexToValue

Composes a string based on given HEX values and converts them to the destination variable type.
In the event of destination being a string, the array of bytes obtained from the source values, will be converted into a text using an Auto conversion, able to evaluate ASCII or UNICODE texts.

Evaluated on inputs value changed.

Inputs:

Tag	Tag name	Tag name of any type remarks: The source TAG data type must be BYTE, INT16 or INT32, signed or unsigned, if you want to convert it into a string. S7 and AB strings are not supported as final data type.
Start Index	Number	Analog expression
End Index	Number	Analog expression
Invert	Boolean	TRUE list evaluation from 'end index' to 'start index' FALSE list evaluation from 'start index' to 'end index'

Outputs:

Item 0:	Converted value
Item 1:	0=Conversion OK, 1=Error 1='end index' < 'start index' 2='start index' out of bounds 3='end index' out of bounds 4=Invalid conversion

ConvertTo

It helps to convert an array of values to a different destination data type.

Data types supported: STRING, BIT, s/u BYTE (8 bits), s/u DWORD (32 bits), s/u QWORD (64 bits), FLOAT (32 bits), DOUBLE (64 bits); Float values follow the IEEE 754 standard.
Items: as much as declared by user

Inputs:

Source	Tag name	It indicates the tag which will be converted
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)
Start Index	Number	Analog expression
Elements	constant	Number of elements to convert
String length	constant	Strings length. (in case of destination type string)

Outputs:
Item 1 to Elements: Values converted

System

Notification

This derived object can be handled by the system only.
Evaluated on inputs value changed

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Input	Boolean expression	TRUE=Active, FALSE= Not Active
Acknowledge	Boolean expression	TRUE=ACK, FALSE=Not ACK
Description	Evaluated as string constant	Notification description
	This text parameter can contain the following tokens: %p1 this token will be replaced with p1 value %p2 this token will be replaced with p2 value %p3 this token will be replaced with p3 value %p4 this token will be replaced with p4 value %p5 this token will be replaced with p5 value %p6 this token will be replaced with p6 value e.g. Oil pump over pressure. %p1 bar → Oil pump over pressure. 5.8 bar	

Text message parameters

p1	Tag name of any type
p2	Tag name of any type
p3	Tag name of any type
p4	Tag name of any type
p5	Tag name of any type
p6	Tag name of any type

Outputs:

Item 0:	1 = Notification Active, 0 = Notification not Active
Item 1:	1 = Notification active and not ACK - (Status = ACT, Default colour RED)
Item 2:	1 = Notification ACK and still active - (Status = ACK, Default colour YELLOW)
Item 3:	1 = Notification reset but not ACK - (Status = RES, Default colour GREEN)
Item 4:	Notification code
Item 5:	Notification type (0 = Alarm, 1 = User message)
Item 6:	Notification status (Values: ACT, ACK, RES)
Item 7:	Priority code [a]
Item 8:	Priority description [a]
Item 9:	Group code [a]
Item 10:	Group description [a]
Item 11:	Notification message [a]
Item 12:	Generation event (ISO Date-time)
Item 13:	ACK event (ISO Date-time)
Item 14:	Reset event (ISO Date-time)

a) It is translated accordingly with project settings.

Strings

Pad

This variable is useful to pad a value

Evaluated on inputs value changed

Inputs:

STag	Tag name	Tag name of any type
Width	Number of characters needed.	constant - Integer
Pad	0=Left 1= Right	constant – Integer (Starting from 1)
Filling char	Char used to fill the final string	constant - String

Outputs:

Item 0:	Padded string

Append

Append two values as strings. S1+S2

Evaluated on inputs value changed.

Inputs:

S1Tag	Tag name or constant string	constant - Tag name of any type
S2Tag	Tag name or constant string	constant - Tag name of any type

The two tags name can't be both constant

Outputs:

Item 0:	S1Tag + S2Tag

Prepend

Prepend two values as strings. S2+S1

Evaluated on inputs value changed.

Inputs:

S1Tag	Tag name or constant string	constant - Tag name of any type
S2Tag	Tag name or constant string	constant - Tag name of any type

The two tags name can't be both constant

Outputs:

Item 0:	S2Tag + S1Tag

Compose

Compose a string using the given tag values and constants.

Evaluated on inputs value changed.

Inputs:

STag1	Tag name or constant string	constant - Tag name of any type
…		
STag32	Tag name or constant string	constant - Tag name of any type

It is possible to insert up to 32 tags

Outputs:

Item 0:	S1Tag + S2Tag + S3Tag + …. (up to 32)

StartsWith

This function can be used to test if the string 'S1' starts with the specified prefix 'S2'

Evaluated on inputs value changed.

Inputs:

S1Tag	Tag name	Tag name of any type
S2Tag	Tag name or constant string	constant - Tag name of any type

The two tags name can't be both constant

Outputs:

Item 0: 1=String 'S1' starts with the given 'S2' value

EndsWith

This function can be used to test if the string 'S1' ends with the specified suffix 'S2'

Evaluated on inputs value changed.

Inputs:

S1Tag	Tag name	Tag name of any type
S2Tag	Tag name or constant string	constant - Tag name of any type

The two tags name can't be both constant

Outputs:

Item 0: 1=String 'S1' ends with the given 'S2' value

AfterFirst

Gets all the characters in 'Source' after the first occurrence of the given 'Delimiter'.

Evaluated on inputs value changed.

Inputs:

Source	Tag name	It indicates the tag which will contain the string to evaluate.
Delimiter	Tag name or constant string	constant - Tag name of string type **remarks**: please observe that if you want to specify the char | (pipe) or char ; as delimiters in a constant value, it will result as an error and the server won't start. In order to avoid this behaviour you should adopt these alternative methods: 1) use the system variable 'SYS_Utility_FieldSeparators' 2) type these strings as constant value, instead of their symbols: "@(vp)" as | or "@(dc)" as ;

Outputs:

Item 0: String value

AfterLast

Gets all the characters in 'Source' after the last occurrence of the given 'Delimiter'.

Evaluated on inputs value changed.

Inputs:

Source	Tag name	It indicates the tag which will contain the string to evaluate.
Delimiter	Tag name or constant string	constant - Tag name of string type **remarks**: please observe that if you want to specify the char | (pipe) or char ; as delimiters in a constant value, it will result as an error and the server won't start. In order to avoid this behaviour you should adopt these alternative methods: 1) use the system variable 'SYS_Utility_FieldSeparators' 2) type these strings as constant value, instead of their symbols: "@(vp)" as | or "@(dc)" as ;

Outputs:

Item 0: String value

BeforeFirst

Gets all the characters in 'Source' before the first occurrence of the given 'Delimiter'.

Evaluated on inputs value changed.

Inputs:

Source	Tag name	It indicates the tag which will contain the string to evaluate.
Delimiter	Tag name or constant string	constant - Tag name of string type **remarks**: please observe that if you want to specify the char | (pipe) or char ; as delimiters in a constant value, it will result as an error and the server won't start. In order to avoid this behaviour you should adopt these alternative methods: 1) use the system variable 'SYS_Utility_FieldSeparators' 2) type these strings as constant value, instead of their symbols: "@(vp)" as | or "@(dc)" as ;

Outputs:

Item 0: String value

BeforeLast

Evaluated on inputs value changed.
Variable type: String
Items: 1

Inputs:

Source	Tag name	It indicates the tag which will contain the string to evaluate.
Delimiter	Tag name or constant string	constant - Tag name of string type **remarks**: please observe that if you want to specify the char | (pipe) or char ; as delimiters in a constant value, it will result as an error and the server won't start. In order to avoid this behaviour you should adopt these alternative methods: 1) use the system variable 'SYS_Utility_FieldSeparators' 2) type these strings as constant value, instead of their symbols: "@(vp)" as | or "@(dc)" as ;

Outputs:

Item 0: String value

Description:
Gets all the characters in 'Source' before the last occurrence of the given 'Delimiter'.

Find

Get the beginning in S1 of S2; -1 if not found

Evaluated on inputs value changed.

Inputs:

S1Tag	Tag name or constant string	constant - Tag name of any type
S2Tag	Tag name or constant string	constant - Tag name of any type

The two tags name can't be both constant

Outputs:

Item 0: Starting index, or -1 if not found

Matches

Check if the string contents matches a mask containing '*' and '?'

Evaluated on inputs value changed.

Inputs:

S1Tag	Tag name or constant string	constant - Tag name of any type
S2Tag	Tag name or constant string	constant - Tag name of any type

The two tags name can't be both constant

Outputs:

Item 0:	1=S2 mask matches S1 content

IsSameAs

Test for the string equality, either considering case or not

Evaluated on inputs value changed.

Inputs:

S1Tag	Tag name or constant string	constant - Tag name of any type
S2Tag	Tag name or constant string	constant - Tag name of any type
CaseSensitive	constant integer, 1=Case sensitive 0=No	constant

The two tags name can't be both constant

Outputs:

Item 0:	1=S1 Contains S2

Replace

Replace all occurrences in S1 of substring S2 with S3

Evaluated on inputs value changed.

Inputs:

S1Tag	Tag name or constant string	constant - Tag name of any type
S2Tag (old)	Tag name or constant string	constant - Tag name of any type
S3Tag (new)	Tag name or constant string	constant - Tag name of any type

The tags name can't be all constant

Outputs:

Item 0:	S1 with all occurrences of S2 replaced with S3

GetChar

Returns the character at position *index*
It could be a value which belong to the ASCII table or even an UNICODE value.
If 0 means that the string is empty or index if out of bounds.

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Index	Analog expression	Analogical → UINT16

Outputs:

Item 0:	Character value at position *index*

LastChar

Returns the last character *value*
It could be a value which belong to the ASCII table or even an UNICODE value.
If 0 means that the string is empty.

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type

Outputs:

Item 0:	Last character value *of the string*

IsNumber

Returns 1 if the string is a number

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type

Outputs:

Item 0:	1=String is a number

IsEmpty

Evaluated on inputs value changed.
Variable type: Boolean
Items: 1

Inputs:

STag	Tag name	Tag name of any type

Outputs:

Item 0:	1=String is empty

Description:
Returns 1 if the string is empty

Length

Returns the string length

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type

Outputs:

Item 0:	String length

Size

Returns the string size in bytes
Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type

Outputs:

Item 0:	String size

Transform

It transforms the string by type: 0=Lower 1=Upper 2=Capitalize

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Type	Type of transformation	constant - Integer

Outputs:

Item 0:	Returns the transformed string

Trim

Removes white-space (space, tabs, form feed, newline and carriage return)
Type: 0=Left side 1=Right side 2=Both

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Type	Trimming type	constant - Integer

Outputs:

Item 0:	Returns the trimmed string

Left

Returns the first count characters of the string

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Count	Analog expression	Analogical → UINT16

Outputs:

Item 0:	The first count characters of the string

Right

Returns the last count characters of the string

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
Count	Analog expression	Analogical → UINT16

Outputs:

Item 0:	The last count characters of the string

Mid

Returns a substring starting at *first*, with length *count*

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
First	Analog expression	Analogical → UINT32
Count	Analog expression	Analogical → UINT32

Outputs:

Item 0:	Substring

Substring

Returns the part of the string between the indices *from* and *to* inclusive

Evaluated on inputs value changed.

Inputs:

STag	Tag name	Tag name of any type
From	Analog expression	Analogical → UINT16
To	Analog expression	Analogical → UINT16

Outputs:

Item 0:	Substring

Insert

insert the string S2 into the string S1 starting at given index.

Evaluated on inputs value changed.

Inputs:

S1Tag	Tag name or constant string	constant - Tag name of any type
Start	Analog expression	Analogical → UINT16
S2Tag	Tag name or constant string	constant - Tag name of any type

The two tags name can't be both constant

Outputs:

Item 0:	S1Tag + S2Tag

Format

Format the provided numeric value using the given format: [sign]integers[.][decimals]
Examples:

23	→	0000	→	0023
23	→	s0	→	+23
-23	→	0	→	-23
23	→	0	→	23
12.4	→	s000.000	→	+012.400
231.129	→	0.00	→	231.13

Evaluated on inputs value changed.

Inputs:

S1Tag	Tag name	Tag name of numeric type
Format	Tag name or constant string	constant - Tag name of string type

Outputs:

Item 0:	Formatted value

ValidateString

It permits a check of the string content matching the given method type

Evaluated on inputs value changed.

Inputs:

Item	Item name	Item to validate
Validation	Constant	Validation mode 0=Contains 1=Not contains
Chars	Tag name or constant string	String of chars used by validation mode

Outputs:

Item 0:	1=Validation OK

Utility

Public

Data types supported: STRING, BIT, s/u BYTE (8 bits), s/u DWORD (32 bits), s/u QWORD (64 bits), FLOAT (32 bits), DOUBLE (64 bits); Float values follow the IEEE 754 standard.
Items: as much as declared by user
Mode: Read & Write

Inputs:

Elements	Evaluated as constant value	Analogical → UINT16
String length	Evaluated as constant value	Analogical → UINT16
Read-only	Boolean expression	Boolean → TRUE=RO, FALSE=Not RO

Description:
This kind of variable is exactly as a variable which comes from peripherals, but will be used the PC memory.

Private

Same as PUBLIC but with the difference that no data will be sent between SERVER and its CLIENTS.
The variable can be written by SERVER or CLIENT actions maintaining different values on both sides.

DataColumn

This kind of variable must be used to extract a data column from a tag source containing the rows of a data table. In case of numeric type a conversion can be made using the parameter called 'base'.

Data types supported: STRING, BIT, s/u BYTE (8 bits), s/u DWORD (32 bits), s/u QWORD (64 bits), FLOAT (32 bits), DOUBLE (64 bits); Float values follow the IEEE 754 standard.
Items: as much as declared by user
Mode: Read only. (it can be modified by server only)

Inputs:

Elements	constant	Column rows
String length	constant	Strings length
Source	Tag name	It indicates the tag which will contain the rows from which is necessary to extract the column.
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)
Delimiters	Tag name or constant string	constant - Tag name of string type **remarks**: please observe that if you want to specify the char | (pipe) or char ; as delimiters in a constant value, it will result as an error and the server won't start. In order to avoid this behaviour you should adopt these alternative method: 1) use the system variable 'SYS_Utility_FieldSeparators' 2) type these strings as constant value, instead of their symbols: "@(vp)" as | or "@(dc)" as ;
Column	Analogical expression	Data column ID, align to ZERO
Item offset	Analogical expression	Source item offset

Outputs:
Item 1 to Elements: Column values

Crc

This variable is useful to calculate the CRC code for the given string or array of numbers.
CRC stands for: Cyclic redundancy check

Evaluated on inputs value changed. (STag)

Inputs:

STag	Tag item	Tag name of any type
Order	constant	Is the CRC Poly Order, counted without the leading '1' bit
Polynom	constant	Is the CRC Poly without leading '1' bit
CRC Init	constant	Is the initial CRC value belonging to that algorithm
XOR Out	constant	Is the final XOR value
Direct	constant	1=Specifies the kind of algorithm: 1=direct, no augmented zero bits
Reflection In	constant	1=Specifies if a data byte is reflected before processing (UART) or not
Reflection Out	constant	1=Specifies if the CRC will be reflected before XOR

Outputs:

Item 0:	Calculated CRC code
Item 1:	0=OK, else error 1=Invalid order, it must be between 1..32 2=Invalid polynom 3=Invalid CRC Init 4=Invalid CRC XorOut

Stack

Every trigger event the value of tag configured as source will be added to the stack

Data types supported: STRING, BIT, s/u BYTE (8 bits), s/u DWORD (32 bits), s/u QWORD (64 bits), FLOAT (32 bits), DOUBLE (64 bits); Float values follow the IEEE 754 standard.
Items: as much as declared by user
Mode: Read only. (it can be modified by server only)

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Elements	constant	Column rows
String length	constant	Strings length
Trigger	Expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Source	Tag name	It indicates the value which will be added to the stack
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)
New value	constant integer	0=ON Top 1=At bottom
Empty	Expression	If a unique tag is used, event can be Command
Empty event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)

Outputs:
Item 1 to Elements: Values stack

Split

It helps to break up a string into a number of tokens

Data types supported: STRING, BIT, s/u BYTE (8 bits), s/u DWORD (32 bits), s/u QWORD (64 bits), FLOAT (32 bits), DOUBLE (64 bits); Float values follow the IEEE 754 standard.
Items: as much as declared by user
Mode: Read only. (it can be modified by server only)

Inputs:

Elements	constant	Column rows
String length	constant	Strings length
Source	Tag name	It indicates the tag which will contain the string to split.
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)
Delimiters	Tag name or constant string	constant - Tag name of string type **remarks**: please observe that if you want to specify the char | (pipe) or char ; as delimiters in a constant value, it will result as an error and the server won't start. In order to avoid this behaviour you should adopt these alternative methods: 1) use the system variable 'SYS_Utility_FieldSeparators' 2) type these strings as constant value, instead of their symbols: "@(vp)" as | or "@(dc)" as ;

Outputs:
Item 1 to Elements: Source string tokens

Thresholds

- ENABLE = FALSE
  Nothing will be evaluated. Output items from 0 to 3 are forced to FALSE, item 4 is TRUE
- ENABLE = TRUE
  The result of INPUT expression, is compared with thresholds enabled.
  The value is considered OK when no thresholds are reached.

Evaluated when: On cycle ~100ms

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Options	Evaluated as bitwise on its value	bit 0: HH enabled, bit 1: H enabled bit 2: L enabled, bit 3: LL enabled
HH Threshold	Analog expression	
H Threshold	Analog expression	
L Threshold	Analog expression	
LL Threshold	Analog expression	
Input	Analog expression	

Outputs:

Item 0:	TRUE = HH Threshold reached, FALSE = normal
Item 1:	TRUE = H Threshold reached, FALSE = normal
Item 2:	TRUE = L Threshold reached, FALSE = normal
Item 3:	TRUE = LL Threshold reached, FALSE = normal
Item 4:	TRUE = Value OK, FALSE = alarm, out of ranges

Contains

It searches for a given text (SFind) into an array of values (Svector)

Evaluated on input tags change.

Inputs:

SVector	Tag name	Tag name of any type
Start index	Analog expression	Tag starting index
Items	Analog expression	Number of items to consider for evaluation
SFind	Tag name or constant	Tag name of any type or constant value
From	constant integer	(0=Top, 1=Bottom)
Mode	constant integer	(0=Start with, 1=End with, 2=Contains, 3=Match, 4=Equal)

Outputs:

Item 0:	0=OK, 1=Wrong starting index or items value
Item 1:	Item ID, or -1 if not found
Item 2:	Occurrences

RgbColour

Creates a colour from the given components

Evaluated when: on inputs value changed.

Inputs:

Red	Analog expression. Value from 0 to 255
Green	Analog expression. Value from 0 to 255
Blue	Analog expression. Value from 0 to 255
Alpha (Transparency)	Analog expression. Value from 0 to 255

Outputs:

Item 0:	Colour value

SwapBytes

It permits the swap of bytes in the given value

Evaluated when: on inputs value changed.

Inputs:

Tag	Tag name	It indicates the value which will be manipulated

Outputs:

Item 0:	value

ArrayChanged

It evaluates if the given array content has been changed.

Evaluated on input tags change.

Inputs:

STag	Tag name	Tag name of any type
Start index	Analog expression	Tag starting index
Items	Analog expression	Number of items to consider for evaluation A value of 0 will use Stag total iems.

Outputs:

Item 0:	0=OK, 1=Wrong starting index or items value
Item 1:	1=Array initialized
Item 2:	1=Array content changed
Item 3:	Changes counter

ValueChanged

It evaluates if the given values has been changed.

Evaluated on input tags change.

Inputs:

Value [1 to 16]	Tag name	Tag name of any type

Up to 16 values can be evaluated

Outputs:

Item 0:	Changes counter

DeviceTagInfo

It permits reading of runtime information about a device tag

Evaluated on inputs value changed. (Tag name)

Inputs:

Trigger	Analog expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)

Remarks: By default the trigger is disabled, thus in this case the action will be executed on every change of Tag content, on the contrary it will be executed on trigger event.

Tag	Tag name	Device tag name only

Outputs:

Item 0:	Polling time [ms]
Item 1:	1=Read only
Item 2:	Elements
Item 3:	1=Communications OK
Item 4:	Minimum reading time
Item 5:	Maximum reading time
Item 6:	Last reading time
Item 7:	Minimum writing time
Item 8:	Maximum writing time
Item 9:	Last writing time

DerivedTagInfo

It permits the reading of runtime information about a device tag

Evaluated on inputs value changed. (Tag name)

Inputs:

Trigger	Analog expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)

Remarks: By default the trigger is disabled, thus in this case the action will be executed on every change of Tag content, on the contrary it will be executed on trigger event.

Tag	Tag name	Derived tag name only

Outputs:

Item 0:	0=Read only
Item 1:	Elements
Item 2:	Minimum execution time
Item 3:	Maximum execution time
Item 4:	Last execution time
Item 5:	Evaluation calls

EditArray

This variable is useful to edit items in a given array.

Evaluated on trigger

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Trigger	Analog expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
array		
Tag	Tag name	Tag name of any type
Action	constant integer	1=Insert above, 2=Insert below 3=Delete 4=Clear
Start index	constant integer	Start item index
Items	constant integer	Amount of items affected by action
Value	constant or tag item	Value used to initialize array items affected by action

Outputs:

Item 0:	Last action status: 0=OK 1=The given array is not enabled 2=The given array is read only 3=Iems value is less than 0 4=Starting index is outside the array limits 5=The action on items array is exceeding array limits 6=Unrecognised action type

ParametrizedText

This variable is useful to get a parametrized text

Evaluated on trigger

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Trigger	Analog expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag resetted at the end of action) 4=Disabled (Evaluated in a script only)

text

Type	Type of text	0=Generic text, 1=SQL
Content	Parametrized text	Text containing tag values

Outputs:

Item 0:	Last action status: 0=Value

Translate

Returns the translated text

Evaluated on inputs value changed.

Inputs:

Text	Tag name	Tag name of any type
Language ID	Analog expression	Analogical → UINT16

Outputs:

Item 0:	Translated text

Bits

SplitBits

This variable is useful to split every single bit of the given tag.

Evaluated on source tag change.

Inputs:

STag	Tag name	Tag name, it must be numeric
Bits	constant integer	(8,16 default,32,64)

Outputs:

Item 0 to Bits-1:	Bit status

Complement

This variable is useful to realize a bit complement of the given tag.

Evaluated on source tag change.

Inputs:

STag	Tag name	Tag name, it must be numeric.
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)
Value type	constant integer	(8,16, 32)

Outputs:

Item 0:	Complement operation result
Item 1:	0=OK, 1=Error

Mask

This variable is useful to realize a bit mask using the two given tags.

Evaluated on source tags change.

Inputs:

STag1	Tag name	Tag name, it must be numeric.
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)
STag2	Tag name	Tag name, it must be numeric.
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)
Mask	Mask type	And, Or, XOr
Value type	constant integer	(8,16, 32)

Outputs:

Item 0:	Mask operation result
Item 1:	0=OK, 1=Error

PackBits

This variable is useful to pack a series of bits

Evaluated on source tags change.

Inputs:

STag	Tag name	Tag name, it must be boolean.
Start index	Analog expression	Tag starting index
Bits	constant integer	(8,16 default,32,64)

Outputs:

Item 0:	Pack bits operation value
Item 1:	0=OK, 1=Wrong starting index value, 2=Conversion error

Shift

This variable is useful to realize a bit shift using the two given tags.

Evaluated on source tags change.

Inputs:

STag	Tag name	Tag name, it must be numeric.
Base	constant integer	(2=Binary, 8=Octal, 10=Decimal, 16=Hexadecimal)
Shift	Shift type	Left, Right
Bits	Analog expression	Valid value range from 1 to 63
Value type	constant integer	(8,16, 32)

Outputs:

Item 0:	Shift operation result
Item 1:	0=OK, 1=Error

Description:

Pointers

Item

This kind of variable creates a reference with a variable item.
Every action on the item is transferred to the pointed one.

Variable type: as its referenced variable

Inputs:

STag	Tag name	Tag name of any type, but ITEMS

Pointer

This kind of variable creates a reference with another one.
Every action on the pointer is transferred to the pointed variable.

Variable type: as its referenced variable
Items: as its referenced variable

Inputs:

STag	Tag name	Tag name of any type, but POINTERS nor ITEMS

AssignItem

It changes the reference to a tag item.

Evaluated on input tags change.

Inputs:

SItemItem	pointer	Tag name of Pointer type
Tag name	Analog expression	Tag name which creates the reference It can be empty, in this case the last tag will be used
Tag index	Analog expression	Tag item index

Outputs:

Item 0:	0=OK, 1=Index beyond tag elements, 2=Not existing tag name 3=Something went wrong creating reference

AssignPointer

It changes the reference to a tag.

Evaluated on input tags change.

Inputs:

SPointer	Tag pointer	Tag name of Item type
Tag name	Analog expression	Tag name which creates the reference

Outputs:

Item 0:	0=OK, 1=Not existing tag name, 2=Something went wrong creating reference

Actions

Bitwise

Bitwise operations

Evaluated on trigger

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Trigger	Analog expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
STag	Tag name	Tag name (Integers 8, 16, 32 or 64 bit)
Action	Evaluated as constant	0=Set bit, 1=Reset bit, 2=Toggle bit
Bit number	Analog expression	Valid value range from 0 to 63

Outputs [Read only]:

Item 0:	Actions executed successfully	integer
Item 1:	Errors counter	integer
Item 2:	Last action status: 0=OK 1=Error writing tag value 2=Bit value out of range 3=Tag is disabled 4=Error converting Stag value	integer

Bitshift

Bitshift operations

Evaluated on trigger

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Trigger	Analog expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
STag	Tag name	Tag name (Integers 8, 16, 32 or 64 bit)
Action	Evaluated as constant	0=Left, 1=Right
Bits number	Analog expression	Valid value range from 1 to 64

Outputs [Read only]:

Item 0:	Actions executed successfully	integer
Item 1:	Errors counter	integer
Item 2:	Last action status: 0=OK 1=Error writing tag value 2=Bit value out of range 3=Tag is disabled 4=Error converting Stag value	integer

BitsMask

Bits mask operations

Evaluated on trigger

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Trigger	Analog expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
DTag	Tag name (Destination)	Tag name (Integers 8, 16, 32 or 64 bit)
Mask	Mask type	And, Or, XOr
Value	Analog expression	Integers 8, 16, 32 or 64 bit

Outputs [Read only]:

Item 0:	Actions executed successfully	integer
Item 1:	Errors counter	integer
Item 2:	Last action status: 0=OK 1=Error writing tag value 2=Bit value out of range 3=Tag is disabled 4=Error converting Dtag value	integer

SetItem

This variable is useful to set an item with the value of another one.

Evaluated on inputs value changed. (Enable, Trigger)

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Source	Numeric or string constant, Analogical expression or string item	
Destination	Tag item	Tag name of any type

Outputs:

Item 0:	Actions executed successfully	integer
Item 1:	Errors counter	integer
Item 2:	Last action status: 0=OK 1=Error writing destination tag values 2=Destination item is disabled 3=Destination item is read-only	integer

ValueToHex

It permits to split the given HEX value to sub-packets, their number depends on destination tag type.
Every packet obtained will be written into the given destination array.
Only integer destination tag can be used.

Evaluated on trigger

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Source value	Numeric or constant,	Analogical expression
Source value type	constant	0=16 bit, 1=32 bit, 2=64 bit, 3=Float, 4=Double
Tag	Tag name	Tag name of any type remarks: The source TAG data type must be integer
Start Index	Number	Analog expression
End Index	Number	Analog expression
Invert	Boolean	TRUE invert packets before writing
WriteAll	constant (0=FALSE 1=TRUE)	TRUE The entire destination array will be write. FALSE Items in destination array will be write one by one

Outputs:

Item 0:	Actions executed successfully	integer
Item 1:	Errors counter	integer
Item 2:	Last action status: 0=OK 1=Error writing destination tag values 2=Destination tag is disabled 3=Invalid start or end index 4=Invalid destination tag 5=Start index is out of destination array limits 6=End index is out of destination array limits	integer

Copy

This variable is useful to copy some values from a source variable to another destination.

Evaluated on inputs value changed. (Enable, STag)

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Trigger	Analog expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)

Remarks: By default the trigger is disabled, thus in this case the action will be executed on every change of STag content, on the contrary it will be executed on trigger event.

source

STag	Tag name	Tag name of any type
SIndex	constant integer	Start item index

destination

DTag	Tag name	Tag name of any type
DIndex	constant integer	Destination item index
Items	constant integer	Number of items to copy
WriteAll	constant integer (0=FALSE 1=TRUE)	TRUE The entire destination array will be write. FALSE Items in destination array will be write one by one.

Outputs:

Item 0:	Actions executed successfully	integer
Item 1:	Errors counter	integer
Item 2:	Last action status: 0=OK 1=Error writing destination tag values 2=Source tag is disabled 3=Destination tag is disabled 4=Source tag communication error 5=Destination tag is a Read-only tag 6=Source start index or items number doesn't match source variable elements number 7=Destination start index or items number doesn't match destination variable elements number	integer

WriteTextFile

This variable is useful to write data into a text file.

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Path	Tag name or constant string	Path where to write the file
Name	Tag name or constant string	File name, e.g. data.log
On missing path	constant	0=Do nothing 1=Make the entire path on destination disk
Header	File header content. It will be written one time only in the event the file is opened in append mode. Leave this property empty if you don't need a file header.	
Body	File body content. In the event of the file being opened using append modality, this text will be written every time and each record will be separated using the “Line break” property.	
Mode	constant	0=Append, 1=Overwrite
Encoding	File chars encoding.	ASCII, UTF-7, UTF-8, UTF-16 or UTF-32
Line break	constant	File line break CR+LF (windows style), LF (Unix/Linux, iOS style), CR (old Mac OS style)

Path and File names parametrisation

Concerning Path and Name properties, they can contain these tags in their text:

%d Actual day
%m Actual month
%y Actual year

%H Actual hour
%M Actual minute
%S Actual second
%t Actual millisecond

It means you can create parametric paths and file names.

e.g. Path = C:/Temp-%y/%m/%d → C:/Temp-2015/06/25 (*assuming an actual date as 25th June 2015*)
File = data-%H.txt → data-09.txt (*assuming an actual time as 09 AM*)
File = data.%H → data.15 (*assuming an actual time as 3 PM*)

Remarks

in case constants will be used for Path or Name, please use this char **/** in order to separate folders names.
e.g. C:/Temp/Data

If you need to insert these chars **{**, **}** typing header or body text, you must prepend this char **/** to them.
e.g. **1/} daily production**, it will result in **1} daily production** inside the file.

Outputs:

Item 0:	Date and time of last execution (ISO format: yyyy-mm-dd HH:MM:SS)	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time [ms]	float
Item 4:	Maximum execution time [ms]	float
Item 5:	Last execution time [ms]	float
Item 6:	Path	string
Item 7:	File name	string
Item 8:	Path + File name	string
Item 9:	File size (bytes)	integer
Item 10:	0=OK, else error 1=Path doesn't exist 2=Error making file path 3=Can't open file for writing operation 4=Can't write file 5=Directory not writeable	integer
Item 11:	Error description. (it could be always empty even in case of error)	string

LoadTextFile

This variable is useful to load the content of a text file.

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Path	Tag name or constant string	Path where to write the file
Name	Tag name or constant string	File name, e.g. data.log
Encoding	File chars encoding.	ASCII, UTF-7, UTF-8, UTF-16 or UTF-32
Delete file	constant	FALSE=Do nothing TRUE=Remove file after reading
Row offset	Analogical expression	It indicates from which row start to read
Content	Tag name	It indicates the tag which will contain the file content It can be only a tag of string type, and it must have got as much as items as the rows you want to read from the text file.
On error	constant	0=Keep last content 1=Empty content

Path and File names parametrisation

Concerning Path and Name properties, they can contain these tags in their text:

%d Actual day
%m Actual month
%y Actual year

%H Actual hour
%M Actual minute
%S Actual second
%t Actual millisecond

It means you can create parametric paths and file names.

e.g. Path = C:/Temp-%y/%m/%d → C:/Temp-2015/06/25 (*assuming an actual date as 25th June 2015*)
File = data-%H.txt → data-09.txt (*assuming an actual time as 09 AM*)
File = data.%H → data.15 (*assuming an actual time as 3 PM*)

Remarks

in case constants will be used for Path or Name, please use this char **/** in order to separate folders names.
e.g. C:/Temp/Data

Outputs:

Item 0:	Date and time of last execution (ISO format: yyyy-mm-dd HH:MM:SS)	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time [ms]	float
Item 4:	Maximum execution time [ms]	float
Item 5:	Last execution time [ms]	float
Item 6:	Path	string
Item 7:	File name	string
Item 8:	Path + File name	string
Item 9:	File size (bytes)	integer
Item 10:	Last modification date time (ISO format: yyyy-mm-dd HH:MM:SS)	string
Item 11:	0=OK, else error 1=Path doesn't exist 2=Can't find the file 3=Can't open the file 4=The file is empty 5=The content tag is not enabled 6=The file is not readable 7=The content tag is not writeable	integer
Item 12:	Error description. (it could be always empty even in case of error)	string

SqlCommand

This variable is useful to retrieve data, execute SQL commands or both.
Remarks: The SQL syntax depend on the selected engine.

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Connector	constant	0=MsOleDb (available only for Windows OS) 1=MySql (Based on C connector, libmysql.dll) 2=FireBird (Version 3.0.4) 3=SQLite (Version 3.x.x) 4=SqlCmd (Microsoft SQL Server tool sqlcmd[1])
String connection	constant	Connection string required for the selected engine
SQL Command	Any valid SQL command	It can be parametrized using TAG values.
Execution	constant	Command execution mode (SYNC or ASYNC)
Row offset	Analogical expression	It indicates from which row start to read
Content	Tag name	It indicates the tag which will contain the dataset It can be only a tag of type string, and it must have got as much as items as the records contained in dataset
Delimiter	Tag name or constant string **remarks**: please observe that if you want to specify the char | (pipe) or char ; as delimiters in a constant value, it will result as an error and the server won't start. In order to avoid this behaviour you should adopt these alternative method: 1) use the system variable 'SYS_Utility_FieldSeparators' 2) type these strings as constant value, instead of their symbols: "@(vp)" as | or "@(dc)" as ;	constant - Tag name of string type
On error	constant	0=Keep last content 1=Empty content

[1]
- Follow this link in order to install SQL Server on LINUX platform.
https://docs.microsoft.com/en-GB/sql/linux/quickstart-install-connect-ubuntu

There are two sections:

1) Install SQL Server
Useful to install SQL server engine

2) Install the SQL Server command-line tools
Useful to install SQL server tool client

- Follow this link in order to install sqlcmd on Windows platform.
https://docs.microsoft.com/en-GB/sql/tools/sqlcmd-utility

Outputs:

Item 0:	Date and time of last execution (ISO format: yyyy-mm-dd HH:MM:SS)	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time [ms]	float
Item 4:	Maximum execution time [ms]	float
Item 5:	Last execution time [ms]	float
Item 6:	Connection string	string
Item 7:	Records count	integer
Item 8:	Fields count	integer
Item 9:	Fields names (only if SQL command gets a valid recordset from DB)	string
Item 10:	0=OK, else error 1=The content tag is not enabled 2=Couldn't open the connection 3=Couldn't execute sql query 4=The content tag is not writeable **remarks:** In case of error more information can be found in log folder.	integer
Item 11:	Error description. (it could be always empty even in case of error)	string

Comments:

C or C++ style comments can be used in SQL Code

examples:

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

/* my own comment .... */

// my own comment ....

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

HttpCommand

This variable is useful to send HTTP requests to a server, able to parse them, and get the server response. It can be used to send or get data for example using JSON format.

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Authentication	constant	0=Basic, 1=Auto, 2=Login
User[1]	text	User name
Password[1]	text	Password
Certificate[1]	file (optional)	Certificate file (*.pem)
Url	Tag name or constant string	End point URL
Content type	constant	0=Text, 1=Html, 2=Java, 3=JSon, 4=Xml, 5=Form
Accept type	constant	0=Text, 1=Html, 2=Java, 3=JSon, 4=Xml, 5=Form
Request type	constant	0=Get, 1=Head, 2=Upload[4], 3=Post, 4=Custom
Custom request[2]	text	Custom request (e.g. PUT, DELETE, PATCH)
Request content[3]	text	HTTP Body

Remarks

[1] Only with authentication equal to Login
[2] Only with a custom request type
[3] Only with these requests: Upload, Post and Custom
[4] The files to upload must be expressed in the Content

Outputs:

Item 0:	Date and time of last execution (ISO format: yyyy-mm-dd HH:MM:SS)	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time [ms]	float
Item 4:	Maximum execution time [ms]	float
Item 5:	Last execution time [ms]	float
Item 6:	0=OK, else error 1=Invalid URL 2=Invalid HTTP header 3=Invalid content length 4=Wrong response from server 255=System error	integer
Item 7:	Error description.	String
Item 8:	End point URL	String
Item 9:	Request type	String
Item 10:	Request content	String
Item 11:	Process time	Integer
Item 12:	HTTP Responce code	Integer
Item 13:	HTTP Responce code description	String
Item 13:	Responce header	String
Item 14:	Responce content	String

Execute

Executes another program or command in a synchronous or asynchronous process.

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event[4]	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Command[1]	Tag name or constant string	Command name. It can be an executable, system command[5], batch file, a file with a default application to open it, ecc...
Working path[1]	Tag name or constant string	Working path if needed by command
Parameters[2]	Tag name or constant string	Parameters for command if it needs them
Show interface	Boolean	TRUE=Show application interface FALSE=Don't show
Execution[3]	constant	Command execution mode (SYNC or ASYNC)
Outputs	Tag name	It indicates the tag which will contain the outputs provided by the command used. It can be only a tag of string type, and it must have got as much as items as the outputs you expect from command.

Remarks

[1] in case constants will be used for Command or Working path, please use this char **/** in order to separate folders names.
e.g. C:/Temp/Data

[2] please use these chars, (# preferred, ~ or %) as separators, in order to separate multiple parameters
e.g. aaa#bbb#ccc#ddd eeee#ffff - aaa~bbb~ccc~ddd eeee~fff - aaa%bbb%ccc%ddd eeee%fff

[3] You have to be careful using this kind of modality; in any case the command must terminate.

[4] Do not use a trigger less than a second to execute commands, this could result in danger for the server execution.

[5] Windows users should read cmd.exe documentation in order to directly execute system commands.
Linux user should refer to their distribution manuals in order to directly execute system commands.

Command, path and parameters parametrisation

Concerning Path and Name properties, they can contain these tags in their text:

%d	Actual day
%m	Actual month
%y	Actual year
%H	Actual hour
%M	Actual minute
%S	Actual second
%t	Actual millisecond
{PP}	Project path

It means you can create parametric paths and file names.

e.g.	Path = C:/Temp-%y/%m/%d → C:/Temp-2015/06/25	(*assuming an actual date as 25th June 2015*)
	File = data-%H.txt → data-09.txt	(*assuming an actual time as 09 AM*)
	File = data.%H → data.15	(*assuming an actual time as 3 PM*)

Outputs:

Item 0:	Date and time of last execution (ISO format: yyyy-mm-dd HH:MM:SS)	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time [ms]	float
Item 4:	Maximum execution time [ms]	float
Item 5:	Last execution time [ms]	float
Item 6:	Command	string
Item 7:	Working path	string
Item 8:	Parameters	string
Item 9:	0=OK, Greater or less than zero ERROR	integer
Item 10:	Error description. (it could be always empty even in case of error)	string

Useful LINUX tip
It could be useful to disable sudo password request in order to execute shell scripts in a correct way from an action, thus you can achieve it following this procedure:

Open the file *etc/sudoers* with super user privileges and add the following row at the end of the file:
YourUserName ALL=(ALL) NOPASSWD:ALL

After saving it is necessary to reboot your system.

System

It executes a system command

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Type	constant	Command type. Log OFF all users Acknowledge alarms Acknowledge user messages Edit recipes Unlock recipes (server side) Disconnect all clients Disconnect all clients but first Shutdown all clients Shutdown all clients but first Shutdown server Save tag values Shows or Hides server GUI

Outputs:

Item 0:	Date and time of last execution (ISO format: yyyy-mm-dd HH:MM:SS)	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time [ms]	float
Item 4:	Maximum execution time [ms]	float
Item 5:	Last execution time [ms]	float
Item 6:	0=OK, Greater or less than zero ERROR Edit recipes command 1=Recipes not enabled 2=Recipes editor already opened	integer
Item 7:	Error description.	String

DeviceTag

It permits to force a read or write action for a device tag.

Evaluated on inputs value changed. (Enable, Trigger)

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Source	Tag name	Device-tag name of any type, or pointer to device-tag
Mode	constant	0=Read 1=Write

Outputs:

Item 0:	Actions executed successfully	integer
Item 1:	Errors counter	integer
Item 2:	Last action status: 0=OK 1=Communication error 2=Source tag is disabled 3=Source tag is read-only 4=The pointed tag is not a device-tag	integer

Sequencer (deprecated, please use Script or Lua modules instead)

It permits to execute actions in a sequence.

Evaluated on inputs value changed. (Enable, Trigger)

Inputs:

Enable	Boolean expression	TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)

The following parameters are repeated 32 times, it means you can execute a sequence of 32 actions.

Action	Tag name	Tag name of any type; it must be a device tag.
Evaluation	constant	0=Private (executed only in sequence), 1=Public (normal cycle)
On error	constant	0=Stop sequence, 1=Skip action and continue
Wait time	constant	Time to wait after executing action. 0=no wait

Outputs:

Item 0:	Actions executed successfully	integer
Item 1:	Errors counter	integer
Item 2:	Last action status: 0=OK From 1 to 32, Sequence number executed with errors. **remarks**: read Action output details in order to understand what has happened.	integer

SendEmail

It permits to send an email

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Server	constant string	SMTP Server name
Port	constant integer	SMTP Server port
Security type	constant integer	0=None, 1=SSL/TLS, 2=STARTTLS
User	constant string	User name
Password	constant string	User password
Certificate	fle	Certificate file (*.pem)
From	string	email address which eventually reply messages are sent to
To	parametric text	list of valid address to send messages
Cc	parametric text	list of valid address to send messages as carbon copy
Subject	parametric text	Email subject
Message	parametric text	Email message
Content type	constant	0=Text, 1=Html, 2=Xml, 3=Csv, 4=Rtf, 5=Sgml, 6=Tabs
Attachements	parametric text	list of valid files to attach

Outputs:

Item 0:	Date and time of last execution (ISO format: yyyy-mm-dd HH:MM:SS)	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time [ms]	float
Item 4:	Maximum execution time [ms]	float
Item 5:	Last execution time [ms]	float
Item 6:	0=OK, else error 1=Error sending email	integer
Item 7:	Error description.	string
Item 8:	Process time	Integer

TrendLog

It permits to log data in a trend.

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Trend name	constant string	Trend object name

Outputs:

Item 0:	Date and time of last execution (ISO format: yyyy-mm-dd HH:MM:SS)	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time [ms]	float
Item 4:	Maximum execution time [ms]	float
Item 5:	Last execution time [ms]	float
Item 6:	0=OK, else error 1=Trend name not valid	integer

UploadValues

This variable is useful to save or to load the tag items with the attribute 'Upload' active.

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
File path	Tag name or constant string	Path to save file
File name	Tag name or constant string	File name.

Paths and File name parametrisation
Concerning Path and Name properties, they can contain these tags in their text:

%d Actual day
%m Actual month
%y Actual year

%H Actual hour
%M Actual minute
%S Actual second
%t Actual millisecond
{PP} Project path (**remarks:** only for source path)

Remarks
in case constants will be used for File path or File name, please use this char **/** in order to separate folders names.
e.g. C:/Temp/Data

Outputs:

Item 0:	Date and time of last execution (ISO format: yyyy-mm-dd HH:MM:SS)	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time [ms]	float
Item 4:	Maximum execution time [ms]	float
Item 5:	Last execution time [ms]	float
Item 6:	File path	string
Item 7:	File name	string
Item 8:	File path+name	string
Item 9:	File size [bytes]	integer
Item 10:	0=OK, else error 1=Path doesn't exist 2=Can't open file for writing 3=Can't open file for reading 4=The folder is not writeable	integer
Item 11:	Error description.	String

Script

It permits the execution of a file containing structured language.
Every object of this type can be considered as a module.
Please refer to eScada.Notes.Script document for more information about this derived object

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Script	Script content	

Outputs:

Item 0:	Last execution	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time	float
Item 4:	Maximum execution time	float
Item 5:	Last execution time	float
Item 6:	0=OK, else error	integer
Item 7:	Errors description all errors encountered during execution	string

LuaModule

It permits the execution of a program written by using Lua language.
Please refer to eScada.Development.Lua document for more information about this derived object

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Loa code	Script content	

Outputs:

Item 0:	Last execution	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time	float
Item 4:	Maximum execution time	float
Item 5:	Last execution time	float
Item 6:	0=OK, else error	integer
Item 7:	Errors description all errors encountered during execution	string

File system

ZipFolder

This variable is useful to zip a folder and its content.

Evaluated on trigger

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Source path	Tag name or constant string	Path to compress
Destination path	Tag name or constant string	Path where to write the file
File name	Tag name or constant string	(Optional) Final file name. If empty the final file name will be same at Source path + .zip extension

Paths and File name parametrisation

Concerning Path and Name properties, they can contain these tags in their text:

%d	Actual day
%m	Actual month
%y	Actual year
%H	Actual hour
%M	Actual minute
%S	Actual second
%t	Actual millisecond
{PP}	Project path (**remarks:** only for source path)

Remarks

in case constants will be used for Source path, Destination path or File name, please use this char **/** in order to separate folders names.
e.g. C:/Temp/Data

Outputs:

Item 0:	Date and time of last execution (ISO format: yyyy-mm-dd HH:MM:SS)	string
Item 1:	Actions executed successfully	integer
Item 2:	Errors counter	integer
Item 3:	Minimum execution time [ms]	float
Item 4:	Maximum execution time [ms]	float
Item 5:	Last execution time [ms]	float
Item 6:	Folder name to zip	string
Item 7:	Destination folder	string
Item 8:	File name	string
Item 9:	File size (bytes)	integer
Item 10:	Zipped files count	string
Item 11:	0=OK, else error 1=Path to compress doesn't exist 2=Destination path doesn't exist 3=Could not possible make a relative path creating zip file 4=Could not possible instance zip file object 5=Could not possible add a new entry in zip file 6=Destination path is not writeable 7=Source folder and destination folder can not be same	integer
Item 12:	Error description.	string

Folder

It permits to list the content of a folder

Evaluated on trigger, folder name or files filter changed

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Elements	constant	Number of elements to keep
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
Folder name	Tag name or constant string	
Content type	constant	0=Files only, 1=Folders only, 2=Files and folders
Files filter	Tag name or constant string	Common filter syntax can be used

Outputs:

Item 0:	Folder name	string
Item 1:	Files filter	string
Item 2:	1=Has files	boolean
Item 3:	1=Has subfolders	boolean
Item 4:	Folders count	integer
Item 5:	Files count	integer
Item 6:	Files size	integer
Item 7:	Actions executed successfully	integer
Item 8:	Errors counter	integer
Item 9:	Minimum executing time	float
Item 10:	Maximum executing time	float
Item 11:	Last executing time	float
Item 12:	0=OK, else error	integer
Item 13:	Error description	string

Item 14 to elements contain parameters for every single object found.
Objects type depends on the selected content type during parametrization.
Parameters are strings following the format below described.

```
parameter format:X;X;X;X;X;X;X
                 | | | | | | |
                 | | | | | | |Modification time
                 | | | | | |
                 | | | | | |File size
                 | | | | |
                 | | | | |File extension
                 | | | |
                 | | | |Name
                 | | |
                 | | |1=Writeable 0=NO
                 | |
                 | |1=Readable 0=NO
                 |
                 |0=Folder 1=File
```

In order to extract parameters from items, the following derived object can be used:
Split, DataColumn

File

It permits retrieval of the file or directory information.

Evaluated on trigger or file name changed

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
Trigger	Analogical expression	If a unique tag is used, event can be Command
Trigger event	constant	0=Rising edge (from 0 to a value not equal to 0) 1=Falling edge (from a value not equal to 0 to 0) 2=Every changing 3=Command (Used tag reset at the end of action) 4=Disabled (Evaluated in a script only)
File name	Tag name or constant string	File or directory name

Outputs:

Item 0:	File or directory name	string
Item 1:	1=Object exists	boolean
Item 2:	1=Is Folder	boolean
Item 3:	1=Is File	boolean
Item 4:	1=Is readable	boolean
Item 5:	1=Is writeable	boolean
Item 6:	1=Is File executable	boolean
Item 7:	1=Has volume	boolean
Item 8:	Volume name	char
Item 9:	File extension	string
Item 10:	Path	string
Item 11:	Forbidden chars	string
Item 12:	File size	integer
Item 13:	Human readable file size	string
Item 14:	Modification time	string

Extended

Weihenstephan Standards Server Version 08

Weihenstephan server instance.
Every instance of this kind of object is resident in its own thread.
Object Evaluated on socket TCP Event (Client ↔ Server connection)

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
IP Address	constant	Server IP address, it must be a valid IP configured on the host system.
TCP port	constant	TCP port number, default is 50000
Clients	constant	Number of client sessions connected at the same time
Allowed addresses	constant	List of address allowed to communicate with this instance
XML File		XML configuration file, see paragraph below. It can contain this parameter {PP} in its name {PP} stands for Project path
Timeout	constant	Watchdog time to detect a communication breakdown. After this time of inactivity, the server will disconnect the correspondent connected client.

Remarks
in case constants will be used for XML File, please use this char **/** in order to separate folders names.

XML configuration file of a Weihenstephan Server instance
In order to prepare an XML file to configure your own instance, it is recommanded to refer to WS Protocol specifications on www.weihenstephaner-standards.de

Outputs:

Item 0:	Library WS Version
Item 1:	Server address
Item 2:	1=Server ready
Item 3:	Connected clients
Item 4:	Clients list
Item 5:	Device Description File
Item 6:	1=Device Description File OK
Item 7:	Source path
Item 8:	WS File Version
Item 9:	Vendor version
Item 10:	Project version
Item 11:	Allowed commands
Item 12:	Tag Objects count
Item 13:	List Objects count
Item 14:	Last command's event (ISO format: yyyy-mm-dd HH:MM:SS)
Item 15:	Commands executed successfully
Item 16:	Errors counter
Item 17:	Minimum execution time [ms]
Item 18:	Maximum execution time [ms]
Item 19:	Last execution time [ms]
Item 20:	Last command description
Item 21:	Last command ID
Item 22:	Last point ID

Item 23: 0=OK, else error
1, TCP SERVER ERROR - Generic error: %s
2, TCP SERVER ERROR - %s
3, TCP CLIENT ERROR - General error - %s
4, Variable name '%s' is not present in database.
5, List '%s', this tag name '%s' is not defined in configuration file.
6, This list number '%s' is already defined in configuration file.
7, This tag number '%s' is already defined in configuration file.
8, In this list '%s' this tag number '%s' is defined more than one time.
9, No tags defined in XML config file.

Item 24: Last error description

Modbus TCP Server

Modbus TCP server instance.
Every instance of this kind of object is resident in its own thread.
Object Evaluated on socket TCP Event (Client ↔ Server connection)

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
IP Address[2]	constant	Modbus TCP instance IP address, it must be a valid IP configured on the host system
TCP port[2]	constant	Modbus TCP instance TCP port number, default is 502[1]
Clients	constant	Number of client sessions connected at the same time
Allowed addresses	constant	List of address allowed to communicate with this Modbus TCP instance
XML File		XML configuration file, see paragraph below. It can contain this parameter {PP} in its name {PP} stands for Project path

Remarks
in case constants will be used for XML File, please use this char / in order to separate folders names.

Outputs:

Item 0:	Library Version
Item 1:	Server address
Item 2:	1=Server ready
Item 3:	Connected clients
Item 4:	Connected clients list
Item 5:	XML Configuration file
Item 6:	1= XML Configuration file OK
Item 7:	Tag Objects count
Item 8:	0=OK, else error 1, Modbus address not valid for item '%s'. (item definition) 2, Item name '%s' is not present in database. (item definition) 3, This item '%s' is already defined in configuration file. (item definition) 4, Modbus address not valid for item '%s'. (tag definition) 5, Item name '%s' is not present in database. (tag definition) 6, This item '%s' is already defined in configuration file. (tag definition) 7, No tags defined in configuration file. 8, Error creating client socket 9, Socket server select function failure 10, Error creating server socket 11, Error creating modbus context 12, Error mapping modbus tables 13, Missing attribute for XML object
Item 9:	Last error description

---

[1] For LINUX users please remember that TCP port from 0 to1024 are not usable as they are reserved for the system.

[2] These parameters will be ignored in case the endpoint string is specified in XML file.
The endpoint string must be like this example: 192.168.85.4.1:502

XML configuration file of a Modbus TCP Server instance

The following example can be use to parametrize your own instance.
You can copy and paste it in an empty file with xml extension. e.g. modbustags.txl
Just put after DECLARATIONS section your own tags and items you want to share using this kind of server

```
<?xml version="1.0" encoding="UTF-8"?>
<CModbusSrv version="1.0" sleep="10" endpoint="">

  <!-- The http://www.modbus.org site provides documentation about the protocol at http://www.modbus.org/specs.php         -->

  <!--                                                                                                                    -->
  <!-- Modbus mapping: The fist value of each area is accessible from 0 address                                           -->
  <!--                                                                                                                    -->
  <coils maxaddress="999" />
  <discretesinput maxaddress="999" />
  <holdingregisters maxaddress="999" />
  <inputregisters maxaddress="999" />

  <!-- areatype values:                                                                                                  -->
  <!--   1 = Coils             (DO) Read/Write (BOOLEAN), this type of data can be alterable by an application program.    -->
  <!--   2 = Discretes Input   (DI) Read-Only  (BOOLEAN), this type of data can be provided by an I/O system               -->
  <!--   3 = Holding Registers (RO) Read/Write (WORD 16 bit), this type of data can be alterable by an application program -->
  <!--   4 = Input Registers   (RI) Read-Only  (WORD 16 bit), this type of data can be provided by an I/O system           -->

  <!-- address: (modbus address)                                          -->
  <!--   It must be a valid address from 0 to maxaddress                  -->
  <!--   maxaddress can't be defined over the value 65535                 -->
  <!--   e.g. maxaddress = 999 means 1000 addresses                       -->
  <!--                                                                    -->
  <!--   address gap among consecutive elements                           -->
  <!--   1, boolean                                                       -->
  <!--   1, byte (8 bit) signed, or unsigned                              -->
  <!--   1, word (16 bit) signed, or unsigned                             -->
  <!--   2, dword (32 bit) signed, or unsigned                            -->
  <!--   2, float (32 bit)                                                -->
  <!--   4, qword (64 bit) signed, or unsigned                            -->
  <!--   4, double(64 bit)                                                -->
  <!--   The step of strings, must be the string length / 2               -->
  <!--   Please see parameter 'chars' description                         -->

  <!-- name:                                                              -->
  <!--   It must be a valid TAG or ITEM declared in database              -->
  <!--   Syntax (tag): TagName e.g. "$SYS.Server.UITime"                  -->
  <!--   Syntax (item): TagName:ID e.g. "$SYS.Server.UITime:0"            -->
  <!--                                                                    -->
  <!--   In case of TAG it is necessary to specify these two parameters:  -->
  <!--   firstitem = First tag item to address                            -->
  <!--   items = amount of values from first item                         -->

  <!-- chars: (Ignored for numeric values)                                -->
  <!--   It specifies the string length,                                  -->
  <!--   it must be an even number and grether then 0                     -->
  <!--   remarks: Two chars in one word                                   -->
  <!--            In case of S7 Strings the real text length will be      -->
  <!--            chars-2 because of siemens strings specification        -->

  <!-- encoding: (Ignored for numeric values)                             -->
  <!--   String endoding                                                  -->
  <!--   0=ASCII                                                          -->
  <!--   1=Auto (default if it's not declared)                            -->
  <!--   2=UTF-7                                                          -->
  <!--   3=UTF-8 (recomended in case of unicode strings)                  -->
  <!--   4=UTF-16                                                         -->
  <!--   5=UTF-32                                                         -->

  <!--                                                                    -->
  <!--                         DECLARATIONS                               -->
  <!--                                                                    -->

  <tag areatype="3" address="0" name="MDB_Test_01" firstitem="0" items="10" chars="0" encoding="0" />
  <item areatype="3" address="10" name="MDB_Test_02:0" chars="0" encoding="0" />

  <item areatype="4" address="0" name="MDB_Test_01:0" chars="0" encoding="0" />
  <item areatype="4" address="0" name="MDB_Test_String:4" chars="12" encoding="1" />

</CModbusSrv>
```

OPC-UA TCP Server

OPC-UA TCP server instance.
Every instance of this kind of object is resident in its own thread.
Object Evaluated on socket TCP Event (Client ↔ Server connection)

Inputs:

Enable	Boolean expression	Boolean → TRUE=Enable ON, FALSE=Not enabled
End point[1]	constant	Server end point URL e.g. opc.tcp://192.168.1.120:4840
Sampling rate	constant	Minimum supported sampling rate Increase it in case of a high CPU % load
Session timeout	constant	Session inactivity time, to consider a client offline.
Sessions	constant	Maximum amount of sessions connected at same time A value of 0 means no sessions limit.
XML File		XML configuration file, see paragraph below. It can contain this parameter {PP} in its name. {PP} stands for Project Path

Remarks
in case constants will be used for XML File, please use this char **/** in order to separate folders names.

Outputs:

Item 0:	Library Version
Item 1:	Server end point
Item 2:	1=Server ready
Item 3:	Sessions count
Item 4:	XML Configuration file
Item 5:	1=XML Configuration file OK
Item 6:	Tag Objects count
Item 7:	0=OK, else error 1, Unexpected object name. 2, eScada object ‘%s’ is already defined. (tag or item already defined) 3, Invalid eScada object ‘%s’. (tag or item definition) 4, Empty tag name for object '%s'. (tag or item definition) 5, Group object without a valid name. (group object definition) 6, Empty group 7, Invalid object '%s'. (objects definitions definition)
Item 8:	Last error description

---

[1] This parameter will be ignored in case the endpoint string is specified in XML file.
The endpoint string must be like this example: opc.tcp://192.168.1.120:4840

XML configuration file of a OPC-UA TCP Server instance

The following example can be use to parametrize your own instance.
You can copy and paste it in an empty file with xml extension. e.g. opcuatags.txl
Just put after DECLARATIONS section your own tags and items you want to share using this kind of server.

```
<?xml version="1.0" encoding="UTF-8"?>
<COpcUaSrv version="1.0" endpoint="">

  <!-- group:                                                          -->
  <!--   It is a container for opctag objects.                         -->
  <!--                                                                 -->
  <!-- attributes:                                                     -->
  <!--   name: String, REQUIRED                                        -->
  <!--   description: String, OPTIONAL                                 -->
  <!--   ns: String, OPTIONAL                                          -->
  <!--   minsamprate: Integer, OPTIONAL [ms]                           -->

  <!-- opctag:                                                         -->
  <!--   It could be a tag or an item belonging to a tag declared      -->
  <!--   in eScada.                                                    -->
  <!--   They can belong to these contexts, device objects, derived    -->
  <!--   objects or notifications.                                     -->
  <!--                                                                 -->
  <!-- remarks:                                                        -->
  <!--   In case of tag, if it is a tag with more than one item,       -->
  <!--   the final OPCUA object will be an array, otherwise it will be -->
  <!--   a single value (scalar) object.                               -->
  <!--                                                                 -->
  <!--   In case of item, the final OPCUA object will be always a single -->
  <!--   value (scalar) object.                                        -->
  <!--                                                                 -->
  <!-- attributes:                                                     -->
  <!--   name: String, REQUIRED                                        -->
  <!--   tag example: name="tagname"                                   -->
  <!--   item example: name="tagname:4"                                -->
  <!--                                                                 -->
  <!--   description: string, OPIONAL                                  -->
  <!--   Object description, it will replace eScada object description -->
  <!--                                                                 -->
  <!--   ns: String, OPTIONAL                                          -->
  <!--   Namespace normally used to separate tags with same name       -->
  <!--                                                                 -->
  <!--   readonly: integer, OPTIONAL                                   -->
  <!--   Valid values: "0" (default) or "1"                            -->
  <!--                                                                 -->
  <!--   readonly="0" A Write action will depend on eScada object      -->
  <!--   configuration                                                 -->
  <!--                                                                 -->
  <!--   readonly="1" Eventough the eScada object it is defined as a   -->
  <!--   read and write object, using readonly attribute setted to 1   -->
  <!--   means the OPCUA tag will be a read only one                   -->
  <!--                                                                 -->
  <!--   skiphmirange: integer, OPTIONAL                               -->
  <!--   Valid values: "0" (default) or "1"                            -->
  <!--                                                                 -->
  <!--   skiphmirange="0" A Write action using a value outside the HMI -->
  <!--   range will result in a OpcUa_BadOutOfRange error              -->
  <!--                                                                 -->
  <!--   skiphmirange="1" A Write action using a value outside the HMI -->
  <!--   range will be truncated to its range MIN value or range MAX   -->
  <!--   value                                                         →
  <!--                                                                 -->
  <!--   readdata: integer, OPTIONAL (Device tag only)                 -->
  <!--   Valid values: "0" (default) or "1"                            -->
  <!--                                                                 -->
  <!--   readdata="0" No reading action will be performed before       -->
  <!--   retrieving data                                                -->
  <!--                                                                 -->
  <!--   readdata="1" The device tag will be read before retrieving data  -->
  <!--                                                                 -->
  <!--                          DECLARATIONS                           →
  <!--                                                                 -->
  <group name="MyGroup 1" description="Some tags" ns="MyNameSpace" minsamprate="100">
    <opctag name="uaUint8" />
    <opctag name="uaUint16" />
    <opctag name="uaUint32" skiphmirange="1" />
    <opctag name="uaUint16:5" />
    <opctag name="uaUint32:5" />
    <opctag name="uaUint64:5" />

      <group name="Nested 1" description="Nested group 1" minsamprate="50">
        <opctag name="uaSint8" />
        <opctag name="uaSint16" />

        <group name="Nested 2" description="Nested group 2" >
          <opctag name="uaSint64" ns="TagNameSpace1" />
          <opctag name="uaSint64" ns="TagNameSpace2" />
        </group>
      </group>
  </group>
```

```
  <group name="MyGroup 2" description="Other tags" ns="AnotherNameSpace" >
    <opctag name="uaBit" />
    <opctag name="uaString" minsamprate="200" />
    <opctag name="uaBit:5" readonly="1" />
    <opctag name="uaString:5" />
  </group>
</COpcUaSrv>
```

System variables

$SYS.Accesss (User access information)

Evaluated on internal thread

$SYS.Local.Saccess
Local user information.
String – Read Only

remarks Items values are useful only on the client side, on the server side they are always ZERO.

Item 0: Group name
Item 1: Group description
Item 2: User name
Item 3: User description
Item 4: User full name

$SYS.Local.UIAccess
Local user information ID.
Signed Integer 16 bit – Read Only

remarks Items values are useful only on the client side, on the server side they are always ZE

Item 0: Group ID
Item 1: User ID
Item 2: 1=Remote user
(user connected from a remote client, it means a machine different from server machine)
Item 3: Project language ID
Item 4: User language ID
Item 5: Interface language ID

$SYS.Local.UOED
Local user options – Editing flags.
Boolean – Read Only

remarks Items values are useful only on the client side, on the server side they are always ZE

Item 0: Project parameters
Item 1: Devices
Item 2: Derived
Item 3: Trends
Item 4: Notifications
Item 5: Recipes
Item 6: Access
Item 7: Pictures
Item 8: Resources
Item 9: Run tools

$SYS.Local.UORT
Local user options – Runtime flags.
Boolean – Read Only

remarks Items values are useful only on the client side, on the server side they are always ZERO

Item 0: Read-only user
Item 1: Edit recipes
Item 2: Upload recipes
Item 3: Download recipes
Item 4: Copy & Paste dataset
Item 5: Export recipes
Item 6: Resources
Item 7: Close runtime
Item 8: Tools.Client – Browse tags
Item 9: Tools.Client – Set values
Item 10÷23: none
Item 24: User option 1
Item 25: User option 2
Item 26: User option 3
Item 27: User option 4
Item 28: User option 5
Item 29: User option 6
Item 30: User option 7
Item 31: User option 8

$SYS.Server.SUsers
Connected users
String – Read Only

Item 0: Connected users.
(String with users separated using ,)

$SYS.Server.Users
String – Read Only
Connected users one by one

Item 0÷100: User name

$SYS.Server.UIAccess
Unsigned integer – Read Only
Server users information

Item 0: Unique user is active

$SYS.Client (System variables, client side)

Evaluated on internal thread

$SYS.Client.Memory
Client memory information
Signed Integer 32 bit – Read Only

remarks Items values are useful only on the client side, on the server side they are always ZERO

Item 0: Total disk size (MB)
Item 1: Free disk size (MB)
Item 2: Free disk percentage (%)
Item 3: Free memory (MB)

$SYS.Client.SDateTime
Current client date-time in string format
String – Read Only

remarks Items values are useful only on the client side, on the server side they are always ZERO

Item 0: Date
Item 1: Time
Item 2: Date-Time
Item 3: ISO Date
Item 4: ISO TIme
Item 5: ISO Date-Time
Item 6: english month name
Item 7: english week day name
Item 8: month name
Item 9: week day name
Item 10: UTC
Item 11: Time Zone

$SYS.Client.SHost
Client information - Strings
String – Read Only

remarks Items values are useful only on the client side, on the server side they are always ZERO

Item 0: host machine name
Item 1: Ethernet card mac address
Item 2: directory for storing temporary files
Item 3: operating system description
Item 4: operating system main version
Item 5: information about a Linux distribution
Item 6: operating system name of the OS
Item 7: operating system family name

$SYS.Client.SIHost
Client information in signed integer format
Signed Integer 32 bit – Read Only

remarks Items values are useful only on the client side, on the server side they are always ZERO

Item 0: 1=64bit OS
Item 1: 1=little endian OS
Item 2: CPU count
Item 3: OS Type (1=Linux 2=Window)

$SYS.Client.SUser
Current client system user information
String – Read Only

remarks Items values are useful only on the client side, on the server side they are always ZERO

Item 0: user name
Item 1: full user name
Item 2: user home directory
Item 3: directory for the user-dependent application data files
Item 4: directory containing the current user's documents

$SYS.Client.UIDate
Current client date
Signed Integer 16 bit – Read Only

remarks Items values are useful only on the client side, on the server side they are always ZERO

Item 0: year day
Item 1: week day
Item 2: day
Item 3: month
Item 4: year
Item 5: century
Item 6: 1=Leap year
Item 7: days in this month
Item 8: days in this year

$SYS.Client.UITime
Signed Integer 16 bit – Read Only
Current client time

remarks Items values are useful only on the client side, on the server side they are always ZERO

Item 0: hour
Item 1: minute
Item 2: second
Item 3: millisecond

$SYS.Client.Confirmation
Unsigned Integer 16 bit – Read Only
User confirmation parameters

remarks Items values are useful only on the client side, on the server side they are always ZERO

Item 0: Confirmation dialog-box, event ID[1]
Item 1: Confirmation dialog-box, user selection (1=OK or YES, 2=NO, 3=CANCEL)[1]
Item 2: Confirmation dialog-box, amount of dialog-box already opened. (0 Means none)[1]
Item 3: Indirection mode, selected item index[2]
Item 4: Indirection mode, value to apply[2]

[1] Available using a confirmation action in “OnConfirmation” picture event.

[2] Available for radiobuttons, combobox, textchoice, imagecombobox and checkbox widgets
These two items are available using “OnSelectedItem” event related to the widget you are selecting from.

$SYS.Client.IP4
Unsigned Integer 16 bit – Read Only
Client local IP4 address

remarks Items values are useful only on the client side, on the server side they are always ZERO

Item 0: IP4 Byte 3
Item 1: IP4 Byte 2
Item 2: IP4 Byte 1
Item 3: IP4 Byte 0

$SYS.Client.Blinking
Boolean – Read Only
Blinking bits

remarks Items values are useful only on the client side, on the server side they are always ZERO

Item 0: Blinking mode 1 (1=ON 0=OFF)
Item 1: Blinking mode 2 (1=ON 0=OFF)
Item 2: Blinking mode 3 (1=ON 0=OFF)

$SYS.Notifications (System notifications variables)

Evaluated on internal thread

$SYS.Alarms
Alarms counters
Unsigned integer 32 bit – Read Only

Item 0: Total alarms defined
Item 1: Active alarms
Item 2: Acknowledged alarms
Item 3: Unacknowledged alarms

$SYS.UserMessages
User messages counters
Unsigned integer 32 bit – Read Only

Item 0: Total messages defined
Item 1: Active messages
Item 2: Acknowledged messages
Item 3: Unacknowledged messages

$SYS.Notifications.Info
Information about notifications context
String – Read Only

Item 0: Runtime alarm messages
Item 1: Runtime user messages

Messages format:
The content of items above listed could be formatted using more than a single row.
Every row is separated with the ASCII char value 10 (\n)
Every single field composing a row is separated by the char | (pipe)

```
ALARM01|2018-11-04 11:30:30|1541327430289|ACK|Notification text|Priority code|Group code
|       |                    |             |   |                |            |
|       |                    |             |   |                |            |(c)
|       |                    |             |   |                |(c)
|       |                    |             |   |
|       |                    |             |   |Message text (c)
|       |                    |             |
|       |                    |             |Message status (b)
|       |                    |
|       |                    |Ticks date time x 1000 (a)
|       |
|       |ISO Date time
|
|Alarm code. (It is unique among alarms)
```

a) Date time using ticks value but including milliseconds
You need to divide by 1000 to get the integer ticks value, the rest represents the milliseconds.

b) Values:
ACT=Activated and not yet acknowledged
ACK=Still active but acknowledged
RES=Resetted but not yet acknowledged

c) It is translated accordingly with project settings.

e.g. (content)
ALARM01|2018-11-04 11:30:30|1541327430289|ACK|Alarm text 1|HIGH|MCH1
ALARM02|2018-11-04 11:30:30|1541327430798|ACT|Alarm text 2|LOW|MCH1

$SYS.Recipes (System recipes variables)

Evaluated on internal thread

$SYS.Recipes.Action
Signed Integer 32 bit- Read & Write
Recipe data parameters

Item 0: Recipe ID
Item 1: Dataset ID
Item 2: Action [0=None 1=Download 2=Upload 3=Load 4=Save values 5=Save all 6=Upload and save 7=Upload and save all 8=Load and download]

0=None	Ready to execute command
1=Download	DB → FIELD
2=Upload	$DTS.ITEMS ← FIELD
3=Load	$DTS.ITEMS ← DB
4=Save values	$DTS.ITEMS → DB values
5=Save all	$DTS.ITEMS → DB values + title + description
6=2 + 4	
7=2 + 5	
8=3 + 1	

$SYS.Recipes.Flags
Boolean – Read Only
Last download status

Item 0: 1 = Editing action active
Item 1: 1 = Action active
Item 2: 1 = Error during last action (*)
Item 3: 1 = Updating data (SERVER → CLIENT or CLIENT → SERVER)

$SYS.Recipes.Titles
String – Read Only
Downloaded recipe information

Item 0: Recipe ID
Item 1: Recipe title
Item 2: Recipe notes
Item 3: Dataset ID
Item 4: Dataset title
Item 5: Dataset notes
Item 6: Download time-stamp

$SYS.Recipes.Status
Signed Integer 32 bit – Read Only
Recipes status

Item 0: Executed action [0=None 1=Download 2=Upload 3=Load 4=Save values 5=Save all 6=Upload and save 7=Upload and save all 8=Load and download]
Item 1: Action time [ms]
Item 2: Action status [0=OK, Other value means error (*)]

(*) Error codes
1 Recipe ID not valid
2 Error getting recipe data from DB
3 recipe not enabled

4 Dataset ID not valid
5 Error getting dataset data from DB
6 dataset not enabled

7 General error writing recipe parameters
8 Error getting recipes values from DB
9 Recipe ID or Dataset ID not valid

10 Error getting variables pointers during an upload action
11 Field variable communication error
12 General error uploading recipe parameters
13 No elements
14 Error saving value

9001 Bit specification not allowed
9002 The specified Tag does not exists
9003 The specified item index is not valid.
9004 Tag name syntax error

$SYS.Server (System variables, server side)

Evaluated on internal thread

$SYS.Server.Clients
String – Read Only
Connected clients one by one

Item 0÷100: Client IP address and tcp port

$SYS.Server.Memory
Server memory information
Signed Integer 32 bit – Read Only

Item 0: Total disk size (MB)
Item 1: Free disk size (MB)
Item 2: Free disk percentage (%)
Item 3: Free memory (MB)

$SYS.Server.SClients
Connected clients
String – Read Only

Item 0: Connected clients.
(String with users separated using ,)

$SYS.Server.SDateTime
Current server date-time in string format
String – Read Only

Item 0: Date
Item 1: Time
Item 2: Date-Time
Item 3: ISO Date
Item 4: ISO TIme
Item 5: ISO Date-Time
Item 6: english month name
Item 7: english week day name
Item 8: month name
Item 9: week day name
Item 10: UTC
Item 11: Time Zone

$SYS.Server.SHMI
Server information in string format
String – Read Only

Item 0: HMI Server name
Item 1: IP address and TCP port
Item 2: Project file
Item 3: Project path
Item 4: Server version
Item 5: Project version
Item 6: Project version date
Item 7: Project version author
Item 8: wxWidgets version
Item 9: ASIO Library version
Item 10: SQLite version

$SYS.Server.SHost
Server information - Strings
String – Read Only

Item 0: host machine name
Item 1: Ethernet card mac address
Item 2: directory for storing temporary files
Item 3: operating system description
Item 4: operating system main version
Item 5: information about a Linux distribution
Item 6: operating system name of the OS
Item 7: operating system family name

$SYS.Server.SIHMI
HMI server information in integer format
Signed Integer 32 bit – Read Only

Item 0: 1=Server ready, initialization completed
Item 1: Connected clients
Item 2: Total Tags
Item 3: Total Elements
Item 4: System Tags
Item 5: System Elements
Item 6: User Tags
Item 7: User Elements
Item 8: Device Tags
Item 9: Device Elements
Item 10: String Tags
Item 11: Numeric Tags
Item 12: Boolean Tags
Item 13: Pointer Tags
Item 14: Item Tags
Item 15: Disabled Tags

$SYS.Server.SIHost
server information in signed integer format
Signed Integer 32 bit – Read Only

Item 0: 1=64bit OS
Item 1: 1=little endian OS
Item 2: CPU count
Item 3: OS Type (1=Linux 2=Window)

$SYS.Server.SUser
Current server system user information
String – Read Only

Item 0: user name
Item 1: full user name
Item 2: user home directory
Item 3: directory for the user-dependent application data files
Item 4: directory containing the current user's documents

$SYS.Server.UIDate
Current server date
Signed Integer 16 bit – Read Only

Item 0: year day
Item 1: week day
Item 2: day
Item 3: month
Item 4: year
Item 5: century
Item 6: 1=Leap year
Item 7: days in this month
Item 8: days in this year

$SYS.Server.UITime
Signed Integer 16 bit – Read Only
Current server time

Item 0: hour
Item 1: minute
Item 2: second
Item 3: millisecond

$SYS.Server.Licence
String – Read Only
Server licence information

Item 0: 1=Licence OK
Item 1: 1=Server shutdown on invalid licence
Item 2: 1=Warning message or server shutdown action on invalid licence
Item 3: Remaining time
Item 4: Customer ID
Item 5: Company name
Item 6: EMail
Item 7: Address
Item 8: Registration code
Item 9: Activation code

$SYS.Utility (System utilities)

Evaluated on internal thread

$SYS.Utility.FieldSeparators
String – Read Only
Field separators

Item 0: ;
Item 1: ,
Item 2: Tab
Item 3: Pipe (|)
Item 4: .
Item 5: :
Item 6: /
Item 7: \
Item 8: *
Item 9: =
Item 10: -
Item 11: +
Item 12: #
Item 13: @
Item 14: §
Item 15: _
Item 16: Back space
Item 17: Form feed
Item 18: Vertical tab

$RCP.xxxxx (Added by manager to edit recipes using pictures)

Evaluated on internal thread

These sections are automatically created by the manager during a normal recipes configuration.
They represent a recipe dataset and the content of this recipe group is filled with dataset parameter names.

The first three tags are always:

$DTS.RCPX.Description	Dataset description
$DTS.RCPX.Title	Dataset name
$DTS.RCPX.ID	Dataset ID

Then follow dataset parameters TAGs

$DTS.xxxxx	Dataset parameter name. Its name is the same name used by the configured TAG.

All of them are writeable, it means you can use them directly on HMI pages instead of using system recipes editor.

$PIC.Pictures (Pictures names, server side)

Inside this group, the system creates pictures names variables.
User should use these variables instead of writing a constant name whenever it is needed.

$SYS.Server.Devices (Server devices)

Evaluated on internal thread

$SYS.Server.CHA.x.DEV.y
String – Read Only
Device information, where: x=Channel name, y=Device name

Item 0: Is enabled
Item 1: Is connected
Item 2: Is off line
Item 3: Device name
Item 4: Device description
Item 5: Protocol version
Item 6: Tags count
Item 7: Elements count
Item 8: Bytes count

$SYS.Client.Threads (Client threads)

Evaluated on internal thread

$SYS.Client.THS.Information
String – Read Only
Derived variables for client information activities

Item 0: Priority
Item 1: Polling time [ms]
Item 2: Thread wait time [ms]
Item 3: Number of cycles
Item 4: Cycles per second
Item 5: Processed tags number
Item 6: Number of cycles in overload
Item 7: Number of cycles in overload [%]
Item 8: Last code execution time [ms]
Item 9: Min code execution time [ms]
Item 10: Max code execution time [ms]
Item 11: Average code execution time [ms]
Item 12: Total cycles time [ms]

$SYS.Server.Threads (Server threads)

Evaluated on internal thread

$SYS.Server.THS.Key
String – Read/Write
Server side key specification to observe channels threads

Item 0: Client IP address
(set here the client IP address in order to read channel threads information)

$SYS.Server.THS.DER.Channel.x
String – Read Only
Derived channel information, x is the channel number.
Values content depends on $SYS.Server.THS.Key value.

Item 0: Enabled
Item 1: Priority
Item 2: Polling time [ms]
Item 3: Thread wait time [ms]
Item 4: Number of cycles
Item 5: Cycles per second
Item 6: Processed tag number
Item 7: Number of cycles in overload

Item 8: Number of cycles in overload [%]
Item 9: Last code execution time [ms]
Item 10: Min code execution time [ms]
Item 11: Max code execution time [ms]
Item 12: Average code execution time [ms]
Item 13: Total cycles time [ms]
Item 14: Channel name
Item 15: Channel description
Item 16: Configured tags
Item 17: Total elements
Item 18: Total bytes
Item 19: Timeout event counter
Item 20: Connected clients
Item 21: Socket client

$SYS.Server.THS.DEV.Channel.x
String – Read Only
Device channel information, x is the channel number.
Values content depends on $SYS.Server.THS.Key value.

Item 0: Enabled
Item 1: Priority
Item 2: Polling time [ms]
Item 3: Thread wait time [ms]
Item 4: Number of cycles
Item 5: Cycles per second
Item 6: Processed tag number
Item 7: Number of cycles in overload
Item 8: Number of cycles in overload [%]
Item 9: Last code execution time [ms]
Item 10: Min code execution time [ms]
Item 11: Max code execution time [ms]
Item 12: Average code execution time [ms]
Item 13: Total cycles time [ms]
Item 14: Driver name, version
Item 15: Base library
Item 16: Channel name
Item 17: Channel description
Item 18: Configured devices
Item 19: Configured tags
Item 20: Total elements
Item 21: Total bytes
Item 22: Timeout event counter
Item 23: Connected clients
Item 24: Socket client

$SYS.Server.THS.Data.Notifications
String – Read Only
Export alarms and user messages

Item 0: Enabled
Item 1: Last execution date time
Item 2: Priority
Item 3: Polling time [ms]
Item 4: Thread wait time [ms]
Item 5: Number of cycles
Item 6: Cycles per second
Item 7: Processed tag number
Item 8: Number of cycles in overload
Item 9: Number of cycles in overload [%]
Item 10: Last code execution time [ms]
Item 11: Min code execution time [ms]
Item 12: Max code execution time [ms]
Item 13: Average code execution time [ms]
Item 14: Total cycles time [ms]

$SYS.Server.THS.Data.Recipes
String – Read Only
Recipes actions execution (Save, load, download, ecc...)

Item 0: Enabled
Item 1: Last execution date time
Item 2: Priority
Item 3: Polling time [ms]
Item 4: Thread wait time [ms]
Item 5: Number of cycles
Item 6: Cycles per second
Item 7: Processed tag number
Item 8: Number of cycles in overload
Item 9: Number of cycles in overload [%]
Item 10: Last code execution time [ms]
Item 11: Min code execution time [ms]
Item 12: Max code execution time [ms]
Item 13: Average code execution time [ms]
Item 14: Total cycles time [ms]

$SYS.Server.THS.TRD.Data.x
String – Read Only
Trends logging where x is ternd name

Item 0: Enabled
Item 1: Last execution date time
Item 2: Priority
Item 3: Polling time [ms]
Item 4: Thread wait time [ms]
Item 5: Number of cycles
Item 6: Cycles per second
Item 7: Processed tag number
Item 8: Number of cycles in overload
Item 9: Number of cycles in overload [%]
Item 10: Last code execution time [ms]
Item 11: Min code execution time [ms]
Item 12: Max code execution time [ms]
Item 13: Average code execution time [ms]
Item 14: Total cycles time [ms]

$SYS.Server.THS.Information
String – Read Only
Derived variables for server information activities

Item 0: Priority
Item 1: Polling time [ms]
Item 2: Thread wait time [ms]
Item 3: Number of cycles
Item 4: Cycles per second
Item 5: Processed tag number
Item 6: Number of cycles in overload
Item 7: Number of cycles in overload [%]
Item 8: Last code execution time [ms]
Item 9: Min code execution time [ms]
Item 10: Max code execution time [ms]
Item 11: Average code execution time [ms]
Item 12: Total cycles time [ms]

$SYS.Server.THS.Notifications
String – Read Only
Derived variables for notifications activities

Item 0: Priority
Item 1: Polling time [ms]
Item 2: Thread wait time [ms]
Item 3: Number of cycles
Item 4: Cycles per second
Item 5: Processed tag number
Item 6: Number of cycles in overload
Item 7: Number of cycles in overload [%]
Item 8: Last code execution time [ms]
Item 9: Min code execution time [ms]
Item 10: Max code execution time [ms]
Item 11: Average code execution time [ms]
Item 12: Total cycles time [ms]

$SYS.Server.THS.NOT.Channel
String – Read Only
Alarms and user messages management

Item 0: Enabled
Item 1: Priority
Item 2: Polling time [ms]
Item 3: Thread wait time [ms]
Item 4: Number of cycles
Item 5: Cycles per second
Item 6: Number of cycles in overload
Item 7: Number of cycles in overload [%]
Item 8: Last code execution time [ms]
Item 9: Min code execution time [ms]
Item 10: Max code execution time [ms]
Item 11: Average code execution time [ms]
Item 12: Total cycles time [ms]
Item 13: Channel name
Item 14: Channel description
Item 15: Timeout event counter
Item 16: Connected clients
Item 17: Socket client
Item 18: Data tipe
Item 19: Last action date time
Item 20: Range ID – Start
Item 21: Range date time – Start
Item 22: Range ID – End
Item 23: Range date time – End
Item 24: Records number

Item 25: Not compressed bytes
Item 26: Compressed bytes
Item 27: Compression level
Item 28: Reading date time [ms]
Item 29: Sending date time [ms]

$SYS.Server.THS.TRD.Channel
String – Read Only
Trends management channel

Item 0: Enabled
Item 1: Priority
Item 2: Polling time [ms]
Item 3: Thread wait time [ms]
Item 4: Number of cycles
Item 5: Cycles per second
Item 6: Number of cycles in overload
Item 7: Number of cycles in overload [%]
Item 8: Last code execution time [ms]
Item 9: Min code execution time [ms]
Item 10: Max code execution time [ms]
Item 11: Average code execution time [ms]
Item 12: Total cycles time [ms]
Item 13: Channel name
Item 14: Channel description
Item 15: Timeout event counter
Item 16: Connected clients
Item 17: Socket client
Item 18: Trend name
Item 19: Trend description
Item 20: Last action date time
Item 21: Range ID – Start
Item 22: Range date time – Start
Item 23: Range ID – End
Item 24: Range date time – End
Item 25: Records number
Item 26: Not compressed bytes
Item 27: Compressed bytes
Item 28: Compression level
Item 29: Reading date time [ms]
Item 30: Sending date time [ms]

$SYS.Server.THS.RCP.Channel
String – Read Only
Recipes management

Item 0: Enabled
Item 1: Priority
Item 2: Polling time [ms]
Item 3: Thread wait time [ms]
Item 4: Number of cycles
Item 5: Cycles per second
Item 6: Number of cycles in overload
Item 7: Number of cycles in overload [%]
Item 8: Last code execution time [ms]
Item 9: Min code execution time [ms]
Item 10: Max code execution time [ms]
Item 11: Average code execution time [ms]
Item 12: Total cycles time [ms]
Item 13: Channel name
Item 14: Channel description
Item 15: Timeout event counter
Item 16: Connected clients
Item 17: Socket client
Item 18: Last action date time
Item 19: Records number
Item 20: Not compressed bytes
Item 21: Compressed bytes
Item 22: Compression level
Item 23: Reading date time [ms]
Item 24: Sending date time [ms]