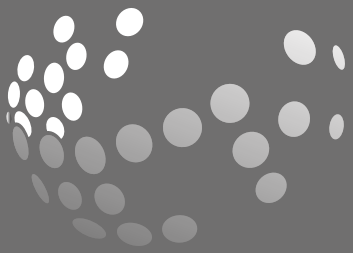

SCRIPTS LANGUAGE

wiki.larnitech.com

Content

data types

Data type	Descriptoin	Range
u8	8-bit unsigned integer, In the general case byte	0..255
i8	8-bit signed integer, In the general case byte	-128..127
u16	16-bit unsigned integer	0..65535
i16	16-bit signed integer	-32768..32767
u32	32-bit unsigned integer	0..4294967295
i32	32-bit signed integer	-2147483648..2147483647

Scripts language

FUNCTION

#ifdef

#ifdef usage

Example:

```
<item A1="" A2="" addr="100:1" body="

    #ifdef A1

    u8 A1d;
    #ifndef A2
        u8 A2d2;
    #endif
    #else
        u8 A1else;
    #endif"
type="script"/>
```

[ ]

go to content

[ ] — access to the word of device status.

Description:

`[ID:SUBID[.NU_OF_BYTE]]`
`[ID:SUBID[:{avg|min|max}{T}:{TIME}]]`

The function provides the access to the word of device ID:SUBID status or to the device NU_OF_BYTE byte. When operating with the sensors by means of this function the average (avg), minimum (min) or maximum (max) value during the period TIME (seconds or minutes T) could be obtained.

List of parameters:

`ID:SUBID` — Device address.
`NU_OF_BYTE` — ID:SUBID device byte number.
`avg|min|max` — Functions for receiving average (avg), minimum (min) or maximum (max) value of the device.
`T` — "m" or "s" are minutes and seconds correspondently.
`TIME` — Number of specified units of time.

Example:

```
<item addr="524:248" MS="512:32" RGB="524:16" name="Test for setStatus function" type="script">

V-ID/V-ADDR
{
    if(opt0())
    {
        u8 strForMess[200];
        u8 *sttOfRgb = [RGB];
        u8 syzeOfRgbstt = sizeof(sttOfRgb);

        sprintf(strForMess, "%cRGB status size - %d, the 2nd byte value -  %d", 1, syzeOfRgbstt, [RGB.1]);

        u16 msMin = [MS:mins:10];
        u16 msMax = [MS:maxs:10];
        u16 msAvg = [MS:avgs:10];

        sprintf(strForMess, "%s\10Move within 10 s.: min-%d max-%d avg-%d", strForMess, msMin, msMax,
[MS:avgs:10]);
        sprintf(strForMess, "%s\10Max move within 10 s.: %d", strForMess, [MS:maxm:10]);
        setStatus(@exciterId():32, &strForMess);
    }
}
</item>
```

Note:

When working with sensors while sending the time parameter (T) for 1 minute or more ([ID:SUBID:{avg|min|max}m:1]) the return value for all three functions (avg|min|max) will be the same. At that the temperature and motion sensors will display maximum value for a past minute and illumination sensor will display the average value for the same period of time.

The result of performing this example:

When pressing the element script, the message with the information about number of RGB element status word bytes and with the value of its 2nd byte will appear in the interface. Data on minimum, maximum and average values of motion sensor MS per 10 seconds as well as its maximum value for the last 10 minutes will be in the message too.

atol

go to content

atol — convert the string into integer type.

Description:

`u8 atol(u8 *string);`

Convert the string string into integer value.

List of parameters:

`string` — String or pointer to the string that contains the number in decimal or hexadecimal format.

Return values:

Function returns integer. If the integer number wasn"t found in the string the function returns value of zero.

Example:

```
<item addr="524:248" name="Test for atol function" type="script">
V-ID/V-ADDR {
            if(opt0()) {
                        u8 str_1[] = "123";
                        u8 str_2[] = "0xCF";
                        // Example 1
                        u8 res_1 = atol(&str_1);
                        // Example 2
                        u8 res_2 = atol(&str_2);
                        // Example 3
                        u8 res_3 = atol("37");
                        // Example 4
                        u16 res_4 = atol("0xFFFF");
                        // Result output to the interface
                        u8 strForMess[70];
                        sprintf(strForMess, "%cres_1 = %d\10res_2 = %d\10res_3 = %d\10res_4 = %d\10"
                              , 1, res_1, res_2, res_3, res_4);
                        setStatus(@exciterId():32, &strForMess);

            }
}
</item>
```

The result of performing the example in the interface:

res_1: 123
res_2: 207
res_3: 37
res_4: 65535

autoState

go to content

autoState — time of current automation deactivation.

Description: `u8 autoState(SUBID);`

Returns information about current time of SUBID unit automation deactivation.

List of parameters: `SUBID` — Device address located on device where the script is.

Return values: Returns value 0 if the device automation is activated.
Returns value more than 0 – time before automation activating.
Returns value -1 if the device automation is disactivated.

Example:

```
<!-- Example 1 -->
<item id="512" LAMP="512:8" name="Test for delayedCallR function" type="script">
V-ID/V-ADDR{

            if(opt0()){

                        // Activate automation
                        setAutoState(8, 0);
                        u8 strForMess[150];
                        // Automation status check
                        i8 sttOfAuto = autoState(8);
                        // Message output to the interface
                        if(sttOfAuto == 0) {
                                sprintf(strForMess, "%cAutomation is activated!", 1);
                                setStatus(@exciterId():32, &strForMess);
                        } else if(sttOfAuto > 0){
                                sprintf(strForMess, "%cTill activating the automation is %d seconds", 4,
                                sttOfAuto);
                                setStatus(@exciterId():32, &strForMess);
                        } else {
                                setStatus(@exciterId():32, {8, "Automation is disactivated"});
                        }
            }
}
</item>

<!-- Example 2 -->
<item id="512" LAMP="512:8" name="Test for delayedCallR function" type="script">
V-ID/V-ADDR{

            if(opt0()){
                        // Disactivate automation
                        setAutoState(8, -1);
                        u8 strForMess[150];
                        // Automation status check
                        i8 sttOfAuto = autoState(8);
                        // Message output to the interface
                        if(sttOfAuto == 0) {
                                sprintf(strForMess, "%cAutomation is activated!", 1);
                                setStatus(@exciterId():32, &strForMess);
                        } else if(sttOfAuto > 0){
                                sprintf(strForMess, "%cTill activating the automation is %d seconds", 4,
                                sttOfAuto);
                                setStatus(@exciterId():32, &strForMess);
                        } else {
                                setStatus(@exciterId():32, {8, "Automation is disactivated"});
                        }
            }
}
</item>
```

```
<!-- Example 3 -->
<item id="512" LAMP="512:8" name="Test for delayedCallR function" type="script">
V-ID/V-ADDR{

            if(opt0())
                        // Disactivate automation for 10 seconds
                        setAutoState(8, 10);
                        u8 strForMess[150];
                        // Automation status check
                        i8 sttOfAuto = autoState(8);
                        // Message output to the interface
                        if(sttOfAuto == 0) {
                                sprintf(strForMess, "%cAutomation is activated!", 1);
                                setStatus(@exciterId():32, &strForMess);
                        } else if(sttOfAuto > 0){
                                sprintf(strForMess, "%cTill activating the automation is %d seconds", 4,
                                sttOfAuto);
                                setStatus(@exciterId():32, &strForMess);
                        } else {
                                setStatus(@exciterId():32, {8, "Automation is disactivated"});
                        }
            }
}
</item>
```

The result of performing this example:

The result of example 1:
"Automation is activated!"

The result of example 2:
" Automation is disactivated!"

The result of example 3:
"10 seconds till automation activation!"

Note:

The function maps only the devices automation status which are located at the same unit as script.

bytesum

go to content

bytesum — returns the sum of bytes of the string.

Description:

```
u8 bytesum(u8 *string);
```

Returns the the sum of bytes of the string string.

Return values:

Sum of bytes of the string string.

Example:

```
<item addr="524:248" name="Test for bytesum function" type="script">
V-ID/V-ADDR {
            if(opt0()) {
                        u8 str[] = "12345";
                        u32 res_1 = bytesum(&str, 2);
                        u32 res_2 = bytesum(&str, 3);
                        u32 res_3 = bytesum(&str, 5);
                        // Result output to the interface
                        u8 strForMess[150];
                        sprintf(strForMess, "%cSum of first 2 bytes - %d\10Sum of first 3 bytes - %d\10Sum of
                        first 5 bytes - %d", 1, res_1, res_2, res_3);
                        setStatus(@exciterId():32, &strForMess);
            }
}
</item>
```

The result of performing the example in the interface:

Sum of first 2 bytes is 99
Sum of first 3 bytes is 150
Sum of first 5 bytes is 255

Code "1" is 49, code "2" is 50, the sum of the first two bytes is 99, code "3" is 51, the sum of the first three bytes is 150 etc.

cancelDelayedCall

go to content

cancelDelayedCall — cancellation of delayed function call.

Description: `void cancelDelayedCall(void *func);`

Cancel the delayed func function call.

List of parameters: `func` — Pointer to the function.

Example:

```
<import-script LAMP="512:8" id="524" name="Test for delayedCallM function">
u8 countOfBlink = 0;
u8 countOfBlink = 0;
void lampBlink()
{
    setStatus(LAMP, ![LAMP]);
    ++countOfBlink;

    if(countOfBlink > 4)
    {
        cancelDelayedCall(lampBlink);
        countOfBlink = 0;
    }
}

V-ID/LAMP
{
    if(opt0() && countOfBlink == 0)
    {
        delayedCallR(lampBlink, 2);
    }
}
</import-script>
```

The result of performing the example in the interface:

After activating the element LAMP the delayed function call delayedCallR will run which in return will start lampBlink function every two seconds. The element will be activated and disactivated twice whereafter cancelDelayedCall function that will stop lampBlink function call will run.

Note:

CancelDelayedCall function cancels all the delayed calls of the same function.

delayedCall

go to content

delayedCall — delayed function call.

Description:

`void delayedCall(void *func, u16 time[,u32 data]);`

Delayed func function call in time seconds. Data parameter is supported in the versions of modules and server firmware after 05.07.2016.

List of parameters:

`func` — Pointer to the function.

`time` — Time in seconds. Max number of seconds which is possible to specify 65534.

Example:

```
<import-script id="524" LAMP="543:45" name="Test for delayedCall function">
void offLamp() {
            setStatus(LAMP, 0);
            }
V-ID/LAMP {
            if(opt0()) {
                        delayedCall(offLamp, 5);
            }
}
</import-script>

<!--With the use of additional parameter:-->
<item type="script" addr="524:250" name="Test for delayedCall function">
void fn(u32 param)
    {
     srvMessage("fn param = %d", param);
    }
    V-ID/V-ADDR
    {
     delayedCall(fn, 1, opt(0));
    }
</item>
```

The result of performing the first example:

On expiry of 5 seconds after switching on the element LAMP will be turned off.

The result of performing the second example:

In a second after clicking the script the message with script status will be displayed in the server log.

Note:

Before the previous call is triggered the 2nd delayed call will be generated but not recorded over the previous one when assigning the delayed recall to one and the same function.

delayedCallM

go to content

delayedCallM — delayed function call in minutes.

Description:

```
void delayedCallM(void *func, u16 time[,u32 data]);
```

Delayed func function call in time minutes.

List of parameters:

`func` — Pointer to the function.
`time` — Time in minutes (maximum 1024).

Example:

```
<import-script id="524" LAMP="524:19" name="Test for delayedCallM function">

void offLamp()
{
    setStatus(LAMP, 0);
}

V-ID/LAMP
{
    if(opt0())
    {
        delayedCallM(offLamp, 1);
    }
}

</import-script>
```

The result of performing this example:

On expiry of a minute after switching on the element LAMP will be turned off.

Note:

Before the previous call is triggered the 2nd delayed call will be generated but not recorded over the previous one when assigning the delayed recall to one and the same function.

Maximum value for time is 1024.

delayedCallMR

go to content

delayedCallMR — delayed function call for each specified number of minutes.

Description:

`void delayedCallMR(void *func, u16 time[,u32 data]);`

Delayed func function call every time minutes.

List of parameters:

`func` — Pointer to the function.
`time` — Time in minutes (maximum 1024).

Example:

```
<item id="524" LAMP="524:19" name="est for delayedCallMR function" type="script">

void onOffLamp()
{
    setStatus(LAMP, ![LAMP]);
}

V-ID/V-ADDR
{
    if(opt0())
    {
        delayedCallMR(onOffLamp, 1);
    }

    if(!opt0())
    {
        cancelDelayedCall(onOffLamp);
    }
}
</item>
```

The result of performing this example:

Every minute the element LAMP will change its status to the opposite one (if it is activated – it will be disactivated and vice a versa)

Note:

Before the previous call is triggered the 2nd delayed call will be generated but not recorded over the previous one when assigning the delayed recall to one and the same function.

Use cancelDelayedCall for suspending the delayed function call.

delayedCallMs

go to content

delayedCallMs — delayed function call in milliseconds.

Description:

```
void delayedCallMs(void *func, u16 time[,u32 data]);
```

Delayed func function call in time milliseconds.

List of parameters:

`func` — Pointer to the function.

`time` — Time in milliseconds. Milliseconds can be specified only in multiples of 50. Max number of milliseconds which is possible to specify is 10000 units.

Example:

```
<import-script id="524" LAMP="524:19" name="Test for delayedCallMs function">

void offLamp()
{
    setStatus(LAMP, 0);
}

V-ID/LAMP
{
    if(opt0())
        delayedCallMs(offLamp, 500);

}

</import-script>
```

The result of performing the example in the interface:

On expiry of 500 milliseconds after switching on the element LAMP will be turned off.

Note:

Before the previous call is triggered the 2nd delayed call will be generated but not recorded over the previous one when assigning the delayed recall to one and the same function.

delayedCallMsR

go to content

delayedCallMsR — elayed function call for each specified number of milliseconds.

Description:

```
void delayedCallMsR(void *func, u16 time[,u32 data]);
```

Delayed func function call every time milliseconds.

List of parameters:

`func` — Pointer to the function.

`time` — Time in milliseconds. Milliseconds can be specified only in multiples of 50. Max number of milliseconds which is possible to specify is 10000 units.

Example:

```
<item id="524" LAMP="524:19" name="Test for delayedCallMsR function" type="script">

void onOffLamp()
{
    setStatus(LAMP, ![LAMP]);
}

V-ID/V-ADDR
{
    if(opt0())
    {
        delayedCallMsR(onOffLamp, 500);
    }

    if(!opt0())
    {
        cancelDelayedCall(onOffLamp);
    }
}

</item>
```

The result of performing this example:

Every 500 milliseconds the element LAMP will change its status to the opposite one (if it is activated – it will be disactivated and vice a versa)

Note:

Before the previous call is triggered the 2nd delayed call will be generated but not recorded over the previous one when assigning the delayed recall to one and the same function.

Use cancelDelayedCall for suspending the delayed function call.

delayedCallR

go to content

delayedCallR —delayed function call for each specified number of seconds.

Description:

```
void delayedCallR(void *func, u16 time[,u32 data]);
```

Delayed func function call every time seconds.

List of parameters:

`func` - Pointer to the function.

`time` - Time in seconds. Max number of seconds which is possible to specify is 0xFFFF.

Example:

```
<item id="524" LAMP="524:19" name="Test for function delayedCallR" type="script">

void onOffLamp()
{
    setStatus(LAMP, ![LAMP]);
}

V-ID/V-ADDR
{
    if(opt0())
    {
        delayedCallR(onOffLamp, 2);
    }

    if(!opt0())
    {
        cancelDelayedCall(onOffLamp);
    }
}

</item>
```

The result of performing this example:

Every 2 seconds the element LAMP will change its status to the opposite one (if it is activated – it will be disactivated and vice a versa)

Note:

Before the previous call is triggered the 2nd delayed call will be generated but not recorded over the previous one when assigning the delayed recall to one and the same function.

Use cancelDelayedCall for suspending the delayed function call.

dsec

go to content

dsec — number of seconds from the beginning of day.

Description:

```
u32 dsec();
```

Returns number of seconds from the beginning of day.

Return values: Returns number of seconds passed from the beginning of day. Type of return value u32.

Example:

```
<item addr="524:248" name="Test for dsec function"  type="script">

V-ID/V-ADDR
{
    if(opt0())
    {
        u32 Dsec = dsec();
        u8 strForMess[70];
        sprintf(strForMess, "%cFrom the beginning of day %d seconds have passed!", 1, Dsec);
        setStatus(@exciterId():32, &strForMess);
        setStatus(V-ADDR, 0);
    }
}

</item>
```

The result of performing the example in the interface:

When clicking the element script the message with number of seconds passed from the beginning of a day will appear in the interface.

eeEmulClean

go to content

eeEmulClean — clearance of all the cells of read-only memory.

Description:

```
u8 eeEmulClean();
```

Clears all the cells of read-only memory of and returns the result of cleanse process.

Return values:

Returns 0 if the error occurred when flushing the memory.
Returns not 0 if the memory flush was successful.

Example:

```
<item addr="524:248" name="Test for eeEmulClean function" type="script">
V-ID/V-ADDR {
            u8 addr = 1;
            if(opt0()) {
                        u8 testWriteVar = 41;
                        u8 resultOfWrite = eeEmulWrite(addr, testWriteVar);
                        u8 strForMess[70];
                        sprintf(strForMess, "%cValue %d", 1, testWriteVar);
                        if(resultOfWrite != 0) {
                                    strcat(&strForMess, " recorded");
                        } else {
                                    strcat(&strForMess, " not recorded");
                        }
                        sprintf(strForMess, "%s into a cell %d.", strForMess, addr);
                        setStatus(@exciterId():32, &strForMess);
                        u8 testClean = eeEmulClean();
                        if(testClean != 0) {
                                    sprintf(strForMess, "%Memory is erased!", 8);
                        } else {
                                    sprintf(strForMess, "%Memory is not erased!", 8);
                        }
                        setStatus(@exciterId():32, &strForMess);
                        u8 testReadVar = 0;
                        u8 resultOfRead = eeEmulRead(addr, &testReadVar);
                        sprintf(strForMess, "%cC cells%d", 4, addr);
                        if(resultOfRead != 0) {
                                    strcat(&strForMess, " is read");
                        } else {
                                    strcat(&strForMess, "the value is not read!");
                                return;
                        }
                        sprintf(strForMess, "%s value %d.", strForMess, testReadVar);
                        setStatus(@exciterId():32, &strForMess);
            }
}
</item>
```

The result of performing the example in the interface:

When clicking the element script in the interface the following messages will appear:
"Value 41 is stored to the cell 4" "The memory is erased!"

eeEmulRead

go to content

eeEmulRead — reading of one byte from the read-only memory.

Description:

```
u8 eeEmulRead(u8 addr, u32 &value);
```

Reads one byte from the cell addr of read-only memory and stores its value to value. Returns the result of reading process.

List of parameters:

`addr` - Cell address in the read-only memory. Value from 1 to 127.

`value` - Reference to variable u32.

Return values:

Returns 0 if the error occurred when reading.
Returns not 0 if reading was successful.

Example:

```
<item addr="524:248" name="Test for eeEmulRead function" type="script">
V-ID/V-ADDR {
            u8 addr = 4;
            if(opt0()) {
                        u32 testWriteVar = 41;
                        u8 resultOfWrite = eeEmulWrite(addr, testWriteVar);
                        u8 strForMess[70];
                        sprintf(strForMess, "%cValue %d", 1, testWriteVar);
                        if(resultOfWrite != 0) {
                                strcat(&strForMess, " recorded");
                        } else {
                                strcat(&strForMess, " not recorded");
                        }
                        sprintf(strForMess, "%s into a cell %d.", strForMess, addr);
                        setStatus(@exciterId():32, &strForMess);
            } else {
                        u32 testReadVar = 0;
                        u8 resultOfRead = eeEmulRead(addr, &testReadVar);
                        u8 strForMess[70];
                        sprintf(strForMess, "%cC cells%d", 4, addr);
                        if(resultOfRead != 0) {
                                strcat(&strForMess, " is read");
                        } else {
                                strcat(&strForMess, "the value is not read!");
                                return;
                        }
                        sprintf(strForMess, "%s value %d.", strForMess, testReadVar);
                        setStatus(@exciterId():32, &strForMess);
            }
}
</item>
```

The result of performing the example in the interface:

When clicking the element script the following messages will appear in the interface:
"Value 41 is stored to the cell 4"
"Value 41 is read from the cell 4"

Note:

This function supports only the variables of u32 type.

eeEmulWrite

go to content

eeEmulWrite — recording of one byte into read-only memory .

Description:

`u8 eeEmulWrite(u8 addr, u32 value);`
`Records the cell with the address addr`, one byte value to the read-only memory and returns the result of recording process.

List of parameters:

`addr` - Cell address in the read-only memory. Value from 1 to 127.

`value` - Information byte which is required to be stored in the read-only memory .

Return values:

Returns 0 if the error occurred when storing.
Returns not 0 if storage was successful.

Example:

```
<item addr="524:248" name="Test for eeEmulWrite function" type="script">
V-ID/V-ADDR {
            u8 addr = 4;
            if(opt0()) {
                        u32 testWriteVar = 41;
                        u8 resultOfWrite = eeEmulWrite(addr, testWriteVar);
                        u8 strForMess[70];
                        sprintf(strForMess, "%cValue %d", 1, testWriteVar);
                        if(resultOfWrite != 0) {
                                strcat(&strForMess, " recorded");
                        } else {
                                strcat(&strForMess, " not recorded");
                        }
                        sprintf(strForMess, "%s into a cell %d.", strForMess, addr);
                        setStatus(@exciterId():32, &strForMess);
            } else {
                        u32 testReadVar = 0;
                        u8 resultOfRead = eeEmulRead(addr, &testReadVar);
                        u8 strForMess[70];
                        sprintf(strForMess, "%cC cells%d", 4, addr);
                        if(resultOfRead != 0) {
                                strcat(&strForMess, " is read");
                        } else {
                                strcat(&strForMess, "the value is not read!");
                                return;
                        }
sprintf(strForMess, "%s value %d.", strForMess, testReadVar);
                        setStatus(@exciterId():32, &strForMess);
            }
}
</item>

```

The result of performing the example in the interface:
When clicking the element script in the interface the following messages will appear:
"Value 41 is stored to the cell 4"
"Value 41 is read from the cell 4"

Note:
This function supports only the variables of u32 type.

error

error — message sending when dealing with She logic emulator.

Description:

```
u8 error(u8 *array);
```

array message sending when dealing with She logic emulator.

Note:

To view messages She app with the parameter -console is required to run by means of logic emulator. Besides the emulator appears the window of console where the messages will be displayed.

List of parameters:

`string` - String or pointer to the string.

Example:

```
<item addr="524:298" RGB="524:16" name="Test for error function" type="script">

V-ID/V-ADDR
{
    error("Hello world!");
}
</item>
```

The result of example execution:

In the interface of She app the message "Hello world!" when clicking the element script.

Event handler

go to content

Event handler is the syntactic construction in LTScript that realizes trapping of the event and its handling occurring in Larnitech system.

General form:

```
[Events pre-processing: ]
{

...

[code]

...

}
```

Events pre-processing: V-ID@ID:SID{...}

Type of handlers:

```
V-ID/ID:SID { ... }
V-ID/ID1:SID1, ID2:SID2 { ... }
V-ID/V-ADDR { ... }
V-ID/ { ... }
V-ID/:SID { ... }
V-ID/[type]:[tyme] { ... }
```

exciterId

go to content

exciterId — determines the element that raised the event and returns its ID.

Description:

```
u16 exciterId();
```

Returns ID of the element raised the event:

Return values:

Returns ID of the element raised the event: Type of return value u16.

Example:

```
<item addr="524:248" name="Test for exciterId function" type="script">

V-ID/V-ADDR
{
    if(opt0())
    {
        u16 exciter_id = exciterId();
        u8 strForMess[70];
        sprintf(strForMess, "%cId of the element raised the event: %d", 1, exciter_id);
        setStatus(@exciterId():32, &strForMess);
        setStatus(V-ADDR, 0);
    }
}

</item>
```

The result of performing the example in the interface:

In this case the script will return the interface ID where the element script was pressed:
"Id of the element raised the event: 2023"

exciterSubId

go to content

exciterSubId — determines the element that raised the event and returns its SUB-ID.

Description:

```
u8 exciterSubId ();
```

Returns SUB-ID of the element raised the event.

Return values: Returns SUB-ID of the element raised the event: Type of return value u8.

Example:

```
<item addr="524:248" name="Test for exciterSubId function" type="script">
V-ID/V-ADDR
{
    if(opt0())
    {
        u16 exciter_sub_id = exciterSubId();
        u8 strForMess[70];
        sprintf(strForMess, "%cSub-Id of the element raised the event:: %d", 1, exciter_sub_id);
        setStatus(@exciterId():32, &strForMess);
        setStatus(V-ADDR, 0);
    }
}

</item>
```

The result of performing the example in the interface:

"Sub-Id of the element raised the event: 0"

flt2i32

go to content

flt2i32 — signed fractional number conversion into appropriate form for comparison operators.

Description:

```
i32 flt2i32(u32 value);
```

Returns signed fractional value converted to the form applicable for comparison operators (for example: converts fractional -20.5 into integer -5428). Later the obtained figure is possible to be used in comparison with data received from sensors.

List of parameters:

`value` - Unsigned fractional (fractional part is separated by a dot) Only literal.

Return values:

Returns signed integer. Type of return value u32.

Example:

```
<import-script id="512" TS="512:32" name="Test for flt2i32 function">
V-ID/s:2
{
    i32 ms = flt2i32(-1.52);
    u8 strForMess[70];

    if([TS] > ms)
    {
        sprintf(strForMess, "%c%d > %d", 1, [TS], ms);
    }
    else
        if([TS] == ms)
        {
            sprintf(strForMess, "%c%d = %d", 1, [TS], ms);
        }
        else
        {
            sprintf(strForMess, "%c%d less %d", 1, [TS], ms);
        }

    setStatus(2023:32, &strForMess);
}

</import-script>
```

The result of performing the example in the interface:

The message where the values of temperature sensor are compared with the value -1.52 is displayed in the interface every two seconds.

Note:

Only literal (figure not a variable) can be the parameter lvalue.

flt2u32

go to content

flt2u32 — unsigned fractional number conversion into appropriate form for comparison operators.

Description:

```
u32 flt2u32(u32 value);
```

Returns unsigned fractional value converted to the form applicable for comparison operators (for example: converts fractional 20.5 into integer 5428). Later the obtained figure is possible to be used in comparison with data received from sensors.

List of parameters:

`value` - Unsigned fractional (fractional part is separated by a dot) Only literal.

Return values:

Returns unsigned integer. Type of return value u32.

Example:

```
<import-script id="512" MS="512:32" name="Test for flt2u32 function">

V-ID/s:2
{
    u32 ms = flt2u32(20.5);
    u8 strForMess[70];

    if([MS] > ms)
    {
        sprintf(strForMess, "%c%d > %d", 1, [MS], ms);
    }
    else
        if([MS] == ms)
        {
            sprintf(strForMess, "%c%d = %d", 1, [MS], ms);
        }
        else
        {
            sprintf(strForMess, "%c%d less %d", 1, [MS], ms);
        }

    setStatus(2023:32, &strForMess);
}

</import-script>
```

The result of performing the example in the interface:

The message where the values of motion sensor are compared with the value 20.5 is displayed in the interface every two seconds.

Note:

Only literal (figure, not a variable) can be the parameter lvalue.

getStatus

go to content

getStatus —access to the array that describes the device status.

Description:

`u8 getStatus(ID:SUBID, u8 *array{, u8 arraysize});`

Entry into the array array of ID:SUBID device status value .

List of parameters:

`ID:SUBID` - Device address.
array - Reference to the array where the device status will be recorded.
arraysize - The size of requested array (optional parameter).

Return values:

Returns the ID:SUBID device status array size in bytes.

Example:

```
<item id="524" RGB="524:16" name="getStatus" type="script">

V-ID/V-ADDR
{
    if(opt0())
    {
        u8 strForMess[200];
        u8 stt[4];
        u16 sttOfRgb = getStatus(RGB, &stt);

        sprintf(strForMess, "%c RGB status size - %d\Byte - %d Byte - %d Byte - %d Byte %d",
            1, sttOfRgb, stt[0],stt[1], stt[2], stt[3]);

        setStatus(@exciterId():32, &strForMess);
    }
}
</item>
```

The result of performing this example:

When pressing the element script, the message with the information about number RGB element status word bytes and with the value of each byte will appear in the interface.

hour

go to content

hour — current value of hours.

Description:

```
u32 hour();
```

Returns the current value of hours.

Return values:

Returns the present value of integer hours passed from the beginning of a day. Type of return value u32.

Example:

```
<item addr="524:248" name="Test for hour function" type="script">

V-ID/s:1
{
    u32 Hour = hour();
    u32 Min = min();
    u32 Sec = sec();

    // Result output to the interface
    u8 strForMess[100];
    sprintf(strForMess, "%cTime in the format of HH:MM:SS  - %d:%d:%d ", 1, Hour , Min, Sec);
    setStatus(INTERFACEID:32, &strForMess);
}

</item>
```

The result of performing the example in the interface:

Every second the messages with current value of time appear in the interface.

isAutoOn

isAutoOn — device automation status.

go to content

Description:

`u8 isAutoOn(SUBID);`

Returns information whether SUBID device automation is activated.

List of parameters:

`SUBID` - Device address located on unit where the script is.

Return values:

Returns value 0 if the device automation is disactivated.
Returns value not 0 if the device automation is activated.

Example:

```
<item id="512" LAMP="512:8" name="IsAutoOn" type="script">
V-ID/V-ADDR{

            if(opt0()){
                        u8 strForMess[50];
                        u8 sttOfAuto = isAutoOn(8);
                        if(sttOfAuto != 0) {
                                    setStatus(@exciterId():32, {1, "Automation is activated"});
                        } else {
                                    setStatus(@exciterId():32, {4, "Automation is disactivated"});
                        }
                        setAutoState(8, 10);
                        sttOfAuto = isAutoOn(8);
                        if(sttOfAuto != 0) {
                                    setStatus(@exciterId():32, {1, "Automation is activated"});
                        } else {
                                    setStatus(@exciterId():32, {4, "Automation is disactivated"});
                        }
            }
}
</item>
```

The result of performing this example:

When clicking the element script in the interface the following message will appear:
"Automation is activated!"
"Automation is disactivated!"

Note:

The function maps only the devices automation status which are located at the same module as script.

isDefined

go to content

isDefined — definition of the device operability.

Description:

`u8 isDefined(ID:SUBID[,PERIOD]);`

Returns information about ID:SUBID device operability.

List of parameters:

`ID:SUBID` - Sensor address.

`PERIOD` - Period after which the device is considered to be non operable, milliseconds. Maximum 10 minutes.

Return values:

Returns 0 – if ID:SUBID device doesn"t respond.
Returns value not 0 – if ID:SUBID device responds.

Attention:

time out for detecting the inoperability is 10 minutes by default.

Example:

```
<item addr="524:248" LAMP="512:8" MS="512:32" name="Test for isDefined function"
            type="script">

V-ID/V-ADDR
{
         if(opt0())
         {
             u8 definedMs = isDefined(MS);
             u8 definedLamp = isDefined(LAMP);
             u8 strForMess[100];

             sprintf(strForMess, "%cLamp ", 1,);

             if(definedLamp != 0)
                 strcat(&strForMess, "available, sensor");
             else
                strcat(&strForMess, "not available, sensor ");

             if(definedMs != 0)
                strcat(&strForMess, "available.");
             else
                 strcat(&strForMess, "not available.");

             setStatus(@exciterId():32, &strForMess);
         }
}

</item>
```

The result of performing the example in the interface:

When clicking the element the messages whether LAMP and MS elements are available will be displayed in the interface.

ltoa

go to content

ltoa — convert the integer type into string.

Description:

```
u8 ltoa(u8 *string, u8 value);
```

Convert the integer number value into string string and returns the string length of string.

List of parameters:

string - Pointer to the string into which the number is converted.
value - Convertable number.

Return values:

Function returns the string length of string after converting into it the number value.

Example:

```
<item addr="524:248" name="Test for ltoa function" type="script">
V-ID/V-ADDR
{
    if(opt0())
    {
        u16 int_1 = 1233;
        u8 int_2 = 0xCF;
        u8 str_1[10];
        u8 str_2[10];
        u8 str_3[10];
        u8 str_4[10];

        // Example: 1
        u8 len_1 = ltoa(&str_1, int_1);

        // Example: 2
        u8 len_2 = ltoa(&str_2, int_2);

        // Example: 3
        u8 len_3 = ltoa(&str_3, 37);

        // Example: 4
        u8 len_4 = ltoa(&str_4, 0xFFFF);

        //output
        u8 strForMess[70];
        sprintf(strForMess, "%cstr_1 - %s\10str_2 - %s\10str_3 - %s\10str_4 - %s\10", 1, str_1, str_2,
str_3, str_4);
        setStatus(@exciterId():32, &strForMess);
        sprintf(strForMess, "%clen_1 - %d\10len_2 - %d\10len_3 - %d\10len_4 - %d\10", 4, len_1, len_2,
len_3, len_4);
        setStatus(@exciterId():32, &strForMess);
    }
}
```

The result of performing the example in the interface:

str_1: 1233

str_2: 207

str_3: 37

str_4: 65

len_1: 4

len_2: 3

len_3: 2

len_4: 5

LTScript namespace

When writing scripts it is possible to address the variables that are located in other sections of scrip using namespace.

Connecting the scripts in file logic.xml

<import-script id="512" name="Script with variable" path="scripts/test_namespace_1.txt"/>

<item addr="512:247" name="Script" type="script" path="scripts/test_namespace_2.txt"/>

File test_namespace_1.txt:

```
//File test_namespace_1.txt:

namespace test;
u8 testVar = 15;

//File test_namespace_2.txt:

namespace test;
V-ID/V-ADDR
{
    srvError("%d", testVar);
}

//Result:

namespace S1;
namespace test;
u8 testVar = 15;
namespace S2;
namespace test;

512/512:247
{
   srvError("%d", testVar);
}
```

The result of performing the example:

When clicking the element script in the server log the following message will be displayed:
512:247.0: 15
512:247.0: 15

Notes:

1. Scripts with the variables are required to declare before the scripts where they are used, for example, if declare the script "test_namespace_2.txt" first and only after it "test_namespace_1.txt" the result of compilation will be the following

```
namespace S1;
namespace test;

512/512:247
{
    srvError("%d", testVar);
}

namespace S2;
namespace test;

u8 testVar = 15;
```

that would lead to error. Namespace functions only within one device

Macros

go to content

The following macros are supported: #ifdef, #ifndef,#if, #else, #endif, #define, #undef.

memcpy

go to content

memcpy — copying from memory to memory.

Description:

```
u8 memcpy(u8 *addr_1, u8 *addr_2, u8 size);
```

Copies the values of addr_2 array, in the number of size, into addr_1 array.

List of parameters:

`addr_1` - Destination address of copying (use & for connecting the address).

`addr_2` - Source address of copying (use & for connecting the address).

`size` - Number of bytes required to copy.

Example:

```
<item addr="524:248" name="Test for memcpy function" type="script">

V-ID/V-ADDR {
    if(opt0())
    {
        u8 mas1[5] = {1, 2, 3, 4, 5};
        u8 mas2[5];
        memcpy(&mas2, &mas1, 5);
        u8 strForMess[70];
        sprintf(strForMess, "%cArray mas2 = {%d, %d, %d, %d, %d}", 1, mas2[0], mas2[1], mas2[2], mas2[3],
mas2[4]);
        setStatus(@exciterId():32, &strForMess);
    }
}

</item>
```

The result of performing the example in the interface:

Array mas2 = {1, 2, 3, 4, 5}

message

message — message sending when dealing with She logic emulator.

Description:

```
u8 message(u8 *array);
```

array message sending when dealing with She logic emulator.

Note:

To view messages She app with the parameter -console is required to run by means of logi emulator. Besides the emulator appears the window of console where the messages will be displayed.

List of parameters:

string - String or pointer to the string.

Example:

```
<item addr="524:298" RGB="524:16" name="Test for message function" type="script">

V-ID/V-ADDR
{
     message("Hello world!");
}

</item>
```

The result of performing the example:

In the interface of She app the message "Hello world!" will appear when clicking the element script.

min

go to content

min — current value of minutes.

Description:

```
u32 min ();
```

Returns the current value of minutes.

Return values:

Returns the present value of integer minutes passed from the beginning of an hour.
Type of return value u32.

Example:

```
<item addr="524:248" INTERFACEID="2023"  name="Test for min function" type="script">
V-ID/m:1
{
   u32 Min = min();

   // Result output to the interface
   u8 strForMess[100];
   sprintf(strForMess, "%cMinutes from the beginning of an hour: %d", 1, Min);
   setStatus(INTERFACEID:32, &strForMess);
}
</item>
```

The result of performing the example in the interface:

Every minute the messages with current value of minutes passed from the beginning of an hour appear in the interface.

month

month — current month.

go to content

Description:

```
u32 month();
```

Returns the current month.

Return values:

Returns the current month of the year. The numbering of months starts from 0 (0 – January, 1 – February, 11 – December).

Example:

```
<!-- Example 1 -->
<item addr="524:248" name="Test for month function" type="script">
V-ID/V-ADDR {
            if(opt0()) {
                        u8 Month = month();
                        u8 strForMess[70] = {1, "Now"};
                        if(Month == 0) {
                                strcat(strForMess, "January!");
                        }
                        if(Month == 1) {
                                strcat(strForMess, "February!");
                        }
                        if(Month == 2) {
                                strcat(strForMess, "March!");
                        }
                        if(Month == 3) {
                                strcat(strForMess, "April!");
                        }
                        if(Month == 4) {
                                strcat(strForMess, "May!");
                        }
                        if(Month == 5) {
                                strcat(strForMess, "June!");
                        }
                        if(Month == 6) {
                                strcat(strForMess, "July!");
                        }
                        if(Month == 7) {
                                strcat(strForMess, "August!");
                        }
                        if(Month == 8) {
                                strcat(strForMess, "September!");
                        }
                        if(Month == 9) {
                                strcat(strForMess, "October!");
                        }
                        if(Month == 10) {
                                strcat(strForMess, "November!");
                        }
                        if(Month == 11) {
                                strcat(strForMess, "December!");
                        }
                        setStatus(@exciterId():32, &strForMess);
                        setStatus(V-ADDR, 0);
            }
}
</item>

<item addr="524:249" name="Test for month function2" type="script">
V-ID/V-ADDR
{
    if(opt0())
    {
        u8 strOfMonth[] ="JanuaryFebruaryMarchAprilMayJuneJulyAugustSeptemberOctoberNovemberDecembe";
        u16 sttOfcount[] = {12, 1214, 2608, 3412, 4606, 5208, 6008, 6812, 8016, 9614, 11012, 12214};
        u8 Month = month();
        u16 count = (sttOfcount[Month] / 100);
        u16 countOfLetters = (sttOfcount[Month] % 100);

        u8 strForMess[70];
        sprintf(strForMess, "%cNow %.*s!", 4, countOfLetters, &strOfMonth + count);
        setStatus(@exciterId():32, &strForMess);
        setStatus(V-ADDR, 0);
    }
}

</item>
```

The result of performing the example in the interface:

Both in the first and second exampled the result of clicking the element script the message with current month will appear in the interface.

ms()

go to content

ms() — number of milliseconds from module being started (resets to zero after u32 counter overflow).

Description:

```
u32 ms();
```

Returns number of milliseconds from unit being started (resets to zero after u32 counter overflow)

Returns number of milliseconds from unit being started (resets to zero after u32 counter overflow).

Return values:

Returns number of milliseconds from unit being started (resets to zero after u32 counter overflow). Type of return value u32.

Example:

```
<item addr="524:248" INTERFACEID="2023" name="Test for ms function" type="script">

V-ID/s:1
{
    u32 msec = ms();

    // output
    u8 strForMess[100];
    sprintf(strForMess, "%cMilliseconds since the start of unit: %d", 1, msec);
    setStatus(INTERFACEID:32, &strForMess);
}

</item>
```

The result of performing the example in the interface:

Every second the messages with current value of milliseconds passed from unit being started appear in the interface.

msWithinDay()

go to content

msWithinDay() — number of milliseconds from the beginning of day.

Description:

```
u32 msWithinDay();
```

Returns number of milliseconds from the beginning of day.

Return values:

Returns the present number of milliseconds from the beginning of day.
Type of return value u32.

Example:

```
<item addr="524:248" INTERFACEID="2023" name="Test for msWithinDay function" type="script">

V-ID/s:1
{
    u32 msec = msWithinMin();

    // Result output to the interface
    u8 strForMess[100];
    sprintf(strForMess, "%cMilliseconds from the beginning of a day: %d", 1, msec);
    setStatus(INTERFACEID:32, &strForMess);
}
</item>
```

The result of performing the example in the interface:

Every second the messages with current value of milliseconds passed from the beginning of day appear in the interface.

msWithinMin()

go to content

msWithinMin() — number of milliseconds from the beginning of a minute.

Description:

```
u32 msWithinMin();
```

Returns number of milliseconds from the beginning of a minute.

Return values:

Returns the present number of milliseconds from the beginning of a minute.
Type of return value u32.

Example:

```
<item addr="524:248" INTERFACEID="2023" name="Test for msWithinMin function " type="script">

V-ID/s:1
{
    u32 msec = msWithinMin();

    // Result output to the interface
    u8 strForMess[100];
    sprintf(strForMess, "%cMilliseconds from the beginning of a minute: %d", 1, msec);
    setStatus(INTERFACEID:32, &strForMess);
}

</item>
```

The result of performing the example in the interface:

Every second the messages with current value of milliseconds passed from the beginning of a minute appear in the interface.

Negative numbers proper operation

go to content

Taking into account the fractional part the processing should be performed as follows:

```
V-ID/TEMP
{
    i32 t = opt16(0);
    i32 n = flt2i32(-10.6);

    if (t &lt n)
    ...
}
```

onInit

go to content

onInit — actuation at module loading (during overwriting the logics or supplying the power).

Description:

```
void onInit() {

...

}
```

Actions specified in braces are performed during module reboot. It is applied for setting initial parameters, conditions, for correct system initialization, initializing it to operate or to operate together with eeEmulRead, eeEmulWrite functions.

Example:

```
<item addr="524:248" addr="524:248" LAMP1="512:8" LAMP2="512:9" name="Test for onInit() function"
type="script">

void onInit()
{
    setStatus(LAMP1, 1);
    setStatus(LAMP2, 1);
    u32 val = 0;
    eeEmulRead(1, &val);

    if(val > 200)
    {
        setStatus(2023:32, {1, "unit was restarted more than 200 times!"});
    }
    else
    {
        u8 strForMess[70];
        sprintf(strForMess, "% unit was restarted %d times!", 4, val);
        setStatus(2023:32, &strForMess);
        ++val;
        eeEmulWrite(1, val);
    }
}

V-ID/V-ADDR
{
    if(opt0())
    {
        eeEmulWrite(1, 1);
        setStatus(2023:32, {1, "Countdown has been started!"});
    }
}

</item>
```

The result of performing the example in the interface:

When clicking the element script and further unit rebooting with the loss of power the elements LAMP1 and LAMP2 will be actuated as well as the messages containing the data on how many times unit was rebooted will be displayed in the interface.

Note:

When connecting several scripts containing this function to one unit all the connected functions are executed.

opt

go to content

opt — access to byte of event status.

Description:

`u8 opt (u8 byte);`

Returns the value of specified word byte (byte) of element status sent the event.

List of parameters:

`value` - Byte number in the word of event status starting from zero.

Return values:

Returns the value of word byte of element status. Type of return value u8.

Note:

opt is also the pointer to the array of device status, for example:
u8 *status=opt;
status – array of device status.

Example:

```
<!-- Example: 1 -->
<item addr="524:248" name="Test for opt function" type="script">

V-ID/V-ADDR
{
    u8 status = opt(0);
    if(status)
    setStatus(@exciterId():32, {1, "Device is activated"});
    else
    setStatus(@exciterId():32, {1, "Device is disactivated"});
}

</item>

<!-- Example 2 usage of opt as the pointer to the array of device event status  -->
<item addr="524:248" COND="524:200" name="Test for opt function" type="script">

V-ID/COND
{
    u8 *status = opt;
    u8 syzeOfStatus = optl;
    u8 strForMess[200];

    sprintf(strForMess, "%cEvent size of conditioner - %d. ", 1, syzeOfStatus);

    u8 i = 0;

    for(i = 0; i < optl; ++i)
    {
        sprintf(strForMess, "%sBYTE[%d] = %d. ", strForMess, i + 1, status[i]);
    }
        setStatus(@exciterId():32, &strForMess);
}
</item>
```

The result of performing the example in the interface:

Example 1: when clicking the element script the message with its current status will appear in the interface.

Example 2: when clicking the element COND in the interface the message with its current status will appear in the interface.

opt0

go to content

Returns opt(0)&7.

Description: `u8 opt0();`

Return values: Returns opt(0)&7. Type of return value u8.

Example:

```
<item addr="524:248" LAMP="509:16" name="Test for opt0() function" type="script">
V-ID/LAMP
{
    u8 status = opt0();

    if(status == 0)
        setStatus(2014:32, {1, "disactivated"});

    if(status == 1)
        setStatus(2014:32, {1, "activated"});}
</item>
```

The result of performing the example in the interface:

The message with LAMP status is displayed in the interface.

opt16

opt16 — access to two bytes of event status.

go to content

Description:

```
u16 opt16(u8 byte);
```

Returns two word bytes of element status sent the event.

List of parameters:

`value` - Byte number in the word of event status starting from zero.

Return values:

Returns the value of two word byte of element status. Type of return value u16.

Example:

```
<item addr="524:248" MS="509:16" name="Test for opt16 function" type="script">

V-ID/MS
{
    u16 status = opt16(0);

    if(status > 0x0300)
        setStatus(2047:32, {1, "Movement excesses  3%"});
}

</item>
```

The result of performing the example in the interface:

When motion sensor MS values are exceeded the corresponding message will be displayed.

optl

optl — element status number of bytes.

go to content

Description: `u8 optl;`

Returns number of bytes of element status sent the event.

Return values: Returns number of bytes of element status sent the event. Type of return value u8.

Example:

```
<item addr="524:248" MS="509:16" name="Test for optl function" type="script">

V-ID/509:16
{
    u8 status = optl;

    if(status == 1)
        setStatus(2014:32, {1, "1 byte"});

    if(status == 2)
        setStatus(2014:32, {1, "2 byte"});

    if(status == 3)
        setStatus(2014:32, {1, "3 byte"});
}

</item>
```

The result of performing the example in the interface:

The message with number of element status bytes is displayed in the interface.

rand()

go to content

rand() — returns random u32.

Description:

```
u32 rand();
```

Returns random u32.

Return values: Returns random u32. Type of return value u32.

Example:

```
<item addr="524:248" INTERFACEID="2023" name="Test for rand function" type="script">

V-ID/s:1
{
    u32 s = rand();

    // Result output to the interface
    u8 strForMess[100];
    sprintf(strForMess, "%cRandom number: %d", 1, s);
    setStatus(INTERFACEID:32, &strForMess);
}

</item>
```

The result of performing the example in the interface:

Every second the messages with random number appear in the interface.

sec

sec — current value of seconds.

Description:

```
u32 sec();
```

Returns the current value of seconds.

Return values:

Returns the present value of integer seconds passed from the beginning of a minute.
Type of return value u32.

Example:

```
<item addr="524:248" INTERFACEID="2023" name="Test for sec function" type="script">

V-ID/s:1
{
    u32 Sec = sec();

    // Result output to the interface
    u8 strForMess[100];
    sprintf(strForMess, "%cSeconds from the beginning of a minute:%d", 1, Sec);
    setStatus(INTERFACEID:32, &strForMess);
}

</item>
```

The result of performing the example in the interface:

Every second the messages with current value of seconds passed from the beginning of a minute appear in the interface.

senderId

go to content

senderId — determines the element that sent the event and returns its ID.

Description:

```
u16 senderId();
```

Returns ID of the element sent the event:

Return values: Returns ID of the element sent the event: Type of return value u16.

Example:

```
<item addr="524:248" name="Test for senderId function" type="script">

V-ID/V-ADDR
{
    if(opt0())
    {
        u16 id = senderId();
        u8 strForMess[70];
        sprintf(strForMess, "%cElement sent the event: %d", 1, id);
        setStatus(@exciterId():32, &strForMess);
        setStatus(V-ADDR, 0);
    }
}

</item>
```

The result of performing the example in the interface:

"Element sent the event: 524"

senderSubId

go to content

senderSubId — determines the element that sent the event and returns its SUB-ID.

Description:

```
u8 senderSubId();
```

Returns SUB-ID of the element sent the event:

Return values:

Returns SUB-ID of the element sent the event: Type of return value u8.

Example:

```
<item addr="524:248" name="Test for senderSubId function" type="script">

V-ID/V-ADDR
{
    if(opt0())
    {
        u8 sub_id = senderSubId();
        u8 strForMess[70];
        sprintf(strForMess, "%cSub-Id of the element sent the event: %d", 1, sub_id);
        setStatus(@exciterId():32, &strForMess);
        setStatus(V-ADDR, 0);
    }
}

</item>
```

The result of performing the example in the interface:

"Sub-Id of the element sent the event: 248"

setAutoState

go to content

setAutoState — device automation control from script.

Description:

`u8 setAutoState(ID:SUBID, u8 time);`

Activates or disactivates the automation on the device for a specific time interval time.

List of parameters:

`SUBID` - device address where the script is located.

`ID:SUBID` - Device address.

`time`

Value more than 0 – disactivate the device automation for time seconds;

0 – activate the device automation;

-1 or -2 – disactivate the device automation until it will be activated by another setAutoState or till unit will be powered off.

Example:

```
<!-- Example 1 -->
<item id="512" LAMP="512:8" name="Test for delayedCallR function" type="script">
V-ID/V-ADDR{

            if(opt0()){
                        // Activate automation
                        setAutoState(8, 0);
                        u8 strForMess[150];
                        // Automation status check
                        i8 sttOfAuto = autoState(8);
                        // Message output to the interface
                        if(sttOfAuto == 0) {
                                sprintf(strForMess, "%cAutomation is activated!", 1);
                                setStatus(@exciterId():32, &strForMess);
                        } else if(sttOfAuto > 0){
sprintf(strForMess, "%cTill activating the automation is %d seconds", 4, sttOfAuto);
                                setStatus(@exciterId():32, &strForMess);
                        } else {
                                setStatus(@exciterId():32, {8, "Automation is disactivated"});
                        }
            }
}
</item>

<!-- Example 2 -->
<item id="512" LAMP="512:8" name="Test for setAutoState function" type="script">
V-ID/V-ADDR{

            if(opt0()){
                        // Disactivate automation
                        setAutoState(8, -1);
                        u8 strForMess[150];
                        // Automation status check
                        i8 sttOfAuto = autoState(8);
                        // Message output to the interface
                        if(sttOfAuto == 0) {
                                sprintf(strForMess, "%cAutomation is activated!", 1);
                                setStatus(@exciterId():32, &strForMess);
                        } else if(sttOfAuto > 0){
sprintf(strForMess, "%cTill activating the automation is %d seconds", 4, sttOfAuto);
                                setStatus(@exciterId():32, &strForMess);
                        } else {
                                setStatus(@exciterId():32, {8, "Automation is disactivated"});
                        }
            }
}
</item>
```

```
<!-- Example 3 -->
<item id="512" LAMP="512:8" name="Test for setAutoState function" type="script">
V-ID/V-ADDR{

            if(opt0()){
                        // Disactivate automation for 10 seconds
                        setAutoState(8, 10);
                        u8 strForMess[150];
                        // Automation status check
                        i8 sttOfAuto = autoState(8);
                        // Message output to the interface
                        if(sttOfAuto == 0) {
                                sprintf(strForMess, "%cAutomation is activated!", 1);
                                setStatus(@exciterId():32, &strForMess);
                        } else if(sttOfAuto > 0){
sprintf(strForMess, "%cTill activating the automation is %d seconds", 4, sttOfAuto);
                                setStatus(@exciterId():32, &strForMess);
                        } else {
                                setStatus(@exciterId():32, {8, "Automation is disactivated"});
                        }
            }
}
</item>
```

The result of performing the example:

The result of example 1:
"Automation is activated!"
The result of example 2:
" Automation is disactivated!"
The result of example 3:
"10 seconds till automation activation!"

Note:

ATTENTION: having switched off the automation that uses setAutoState(ID:SUBID, -1) function in case of changing the slave status the automation will be activated (upon the expiry of autoperiod).

setStatus

go to content

setStatus — device status word setup.

Description:

```
void setStatus(ID:SUBID, u8 *status[, u8 sizeOfstatus]);
```

Set up of status status(with size sizeOfstatus) for ID:SUBID device. Depending on the device one byte is sent for simple devices (TRX, HYT) and several bytes (array) for complicated ones.

List of parameters:

ID:SUBID - Device address.

status - Status forwarded to the device. Literal, variable or pointer to the array can be forwarded as the parameter.

sizeOfstatus - The size of forwarded status (optional parameter).

Example:

```
<item LAMP1="512:8" LAMP2="512:9" DIM="524:19" RGB="524:16" addr="524:248" COND="524:200" name="Test for
setStatus function" type="script">

V-ID/V-ADDR
{
if(opt0())
{
// Example: 1
setStatus(LAMP1, 1);

// Example: 2
u8 stt_1 = 1;
setStatus(LAMP2, stt_1);

// Example: 3
setStatus(DIM, {1, 175, 3});

// Example: 4
u8 stt_2[] = {0x21, 0x08, 0, 0x22, 0x02};
setStatus(COND, &stt_2);

// Example: 5
u8 stt_3[] = {1, 200, 200, 0, 10};
setStatus(RGB, &stt_3, 5);
}
}
</item>
```

The result of performing this example:

When clicking the element script the lamps LAMP1 and LAMP2 will be switched on, the dimmable lamp DIM will be switched on with brightness of 175 within 3 seconds, RGB lamp RGB within 10 seconds will be switched on and will change the colour to red. Virtual device of conditioner COND will be switched on with the temperature of 24 degrees in dry mode with airflow capacity 3 and position of horizontal and vertical fins 2.

sizeof

go to content

sizeof — receiving the memory size of variable or array in bytes.

Description: `u8 sizeof (u8 *var);`

Returns the memory size of variable or var array in bytes.

List of parameters: `var` - Variable or array for which the memory size is measured.

Return values: var memory size in bytes.

Example:

```
<item addr="524:248" name="Test for sizeof function" type="script">

    V-ID/V-ADDR
    {
        if(opt0())
        {
                u8 testU8;
                u16 testU16;
                u32 testU32;
                u8 testMasOfU8[4];
                u16 testMasOfU16[4];
                u32 testMasOfU32[4];
                // Output the result to the interface
                u8 strForMess[200];

                sprintf(strForMess, "%ctestU8 - %d\10testU16 - %d\10testU32 - %d\10testMasOfU8 - %d\10test
                MasOfU16 -  %d\10testMasOfU32- %d", 1, sizeof(testU8), sizeof(testU16), sizeof(testU32),
                sizeof(testMasOfU8), sizeof(testMasOfU16), sizeof(testMasOfU32));

                setStatus(@exciterId():32, &strForMess);

        }
    }

</item>
```

The result of performing the example in the interface:

testU8 – 1

testU16 – 2

testU32 – 4

testMasOfU8 – 4

testMasOfU16 – 8

testMasOfU32 – 16

sprintf

sprintf — formats and stores sets of symbols and values..

Description:

```
u8 sprintf(u8 *buf, u8 *string[, u8 arg-list…]);
```

Saves the arguments from arg-list list into buf under the control of string string and returns the length of converted string buf. Every argument arg-list (if any) is converted and displayed in accordance with corresponding format specification specified in string.

List of parameters:

`buf` - Pointer to the string.

`string` - String or pointer to the format control string.

Format control string consists of normal characters, escape-sequences and if the arguments follow the format string then it includes format specification. If the arguments arg-list follow the format string then this string is also have to contain the format specifications determining the outputs of these arguments. Format specification always starts from percent symbol (%).

Format string is read left-to-right. When the format specification (if any) occurs the value of the first argument after the format string is converted and is displayed in accordance with specified specification. The second format specification causes the conversion and displaying of the second argument and so on so forth till the end of the format string. If arguments more than format specifications then the additional arguments are ignored.

Format specification takes the following form: %[flags][wigth][.precision]type.

Where:

% – format specification symbol;

[flags] – Turning on of displaying and printing the characters, spaces, octal and hexadecimal prefixes (Table flags).

[wigth] – Minimum number of displayed characters (Description of wigth is below).

[.precision] – Maximum number of characters (for %s) or minimum number of digits (for printing the integer values %d %u %i).

type – format specification (Table type).

Table of format specification (type)

Symbol	Description:
s	String of symbols.
p	Pointer value
d	Signed decimal number of integer type
i	Signed decimal number of integer type
e	Scientific notation (e of lowercase)
E	Scientific notation (E of uppercase)
o	Unsigned octal integer
u	Unsigned decimal number of integer type
x	Unsigned hexadecimal integer (letters of lowercase)
X	Unsigned hexadecimal integer (letters of uppercase)
%	Symbol%
c	Symbols

Values flags

Flag	Value	Default value
–	Result outdent within the field wigth.	right
+	Symbol appending to the displayed value if it has character type..	The symbol "-" appears only for negative character values.
#	When using with o, x, X formats the flag # appends 0, 0x, 0X correspondently to any nonzero displayed value .	Without prefix.

`Width` – non-negative decimal integer that controls the number of printed characters. If the number of characters in the output is less than in width, then the spaces are added on the left and right (depending on where the flag "-" is specified) unless the minimum width is reached. If 0 is appended to width then 0 will be added till the minimum width is reached. (It is not applicable to the outdent numbers).
The star (*) may occur in specification width when instead of value the argument corresponding to it is put from the list of arguments . Argument width is to precede the corresponding value.

`arg-list` – Variable number of parameters.

Return values: The function returns buf string length after conversion.

Examples:

```
<--! Example 1 -->
<item addr="524:248" name="Test for sprintf function"; type="script">
u8 count = 0;

V-ID/V-ADDR {
            if(opt0()){
                        // string for result output to the interface
                        u8 strForMess[100];
                        u8 str[] = "hello";
                        if(count == 0) {
                                // %s  String of symbols
                                u8 resultOfFunc = sprintf(strForMess, "%cString - %s and %s, len = ", 1,
                                str, "world");
                                ltoa(strForMess+strlen(strForMess), resultOfFunc);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 1) {
                                // %p - pointer value
                                sprintf(strForMess, "%cAddresses - %p %p %p %p", 4, str, str + 1, str + 2,
                                str + 3);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 2) {
                                // %d - Signed decimal number of integer type
                                u8 value_1 = 55;
                                i8 value_2 = -55;
                                sprintf(strForMess, "%c%d %d %d %d", 8, 1, -2, value_1, value_2);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 3) {
                                // %i  - Signed decimal number of integer type
                                u8 value_1 = 22;
                                i8 value_2 = -33;
                                sprintf(strForMess, "%c%i %i %i %i", 1, 1, -2, value_1, value_2);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
```

```
                        if(count == 4) {
                                // %e   Scientific notation  (e of lowercase)
                                sprintf(strForMess, "%c%e", 4, 3000, 2000);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 5) {
                                // %e   Scientific notation  (E of uppercase)
                                sprintf(strForMess, "%c%E", 8, 3000, 2000);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 6) {
                                // %o   Unsigned octal integer
                                sprintf(strForMess, "%cOctal system: %o", 1, 10);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 7) {
                                // %u  Signed decimal number of integer type
                                i8 value_1 = 10;
                                i8 value_2 = -10;
                                sprintf(strForMess, "%c%u %u", 4, value_1, value_2);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 8) {
                                // %x   Unsigned hexadecimal integer (letters of lowercase)
                                sprintf(strForMess, "%cHexadecimal number systema lov: 0x%x", 8, 255);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 9) {
                                // %X  Unsigned hexadecimal integer (letters of uppercase)
                        sprintf(strForMess, "%cHexadecimal number system hi: 0x%X", 1, 255);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 10) {
                                // %% Displays symbol %
                                sprintf(strForMess, "%c%%", 4);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 11) {
                                // %c   Displays symbols
                                u8 val_1 = 0x41; // A
                                u8 val_2 = 0x40; // @
                                u8 val_3 = 0x55; // U
                                u8 *val_4 = "+- %#";
                                sprintf(strForMess, "%cSymbols: %c %c %c %c %c %c", 8, val_1, val_2, val_3,
                                val_4[4], 0x67);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                count = 0;
                                return;
                        }
                }
        }
        </item>
```

```
<--! Example 2 -->
<item addr="524:248" name="Test for sprintf function" type="script">
u8 count = 0;

V-ID/V-ADDR {
            if(opt0()) {
                        // string for result output to the interface
                        u8 strForMess[200];
                        if(count == 0) {
                                // usage of width
                                u8 *str = "Hello World";
                        sprintf(strForMess, "%cwidth with a string (with and without -): _%20s_ _%-20s_",
                                1, str, str);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 1) {
                                // usage of width
                        sprintf(strForMess, "%cwidth with a number (with and without -): _%20d_ _%-20d_",
                                4, 33, 33);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 2) {
                                // usage of width
                        sprintf(strForMess, "%cwidth with a number (*with and without -): _%*d_ _%-*d_",
                                8, 7, 33, 7, 33);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 3) {
                                // flag#
                                sprintf(strForMess, "%cOctal(%%o) and hexadecimal(%%x, %%X) without and
with the flag #: \10%%o - %o, %#o\10%%x - %x, %#x\10%%X - %X,  %#X", 1, 200, 200, 200, 200, 200, 200);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 4) {
                                // precision
                                u8 *str = "Hello world";
                        sprintf(strForMess, "%c.precision with a string  %s\10%%s - %s\10%%.4s - %.4s",
                        4, str, str, str);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 5) {
                                // precision
                        sprintf(strForMess, "%c.precision with a number %d\10%%d - %d\10%%.6d - %.6d",
                        8, 43, 43, 43);
                                setStatus(@exciterId():32, &strForMess);
                                setStatus(V-ADDR, 0);
                                ++count;
                                return;
                        }
                        if(count == 6) {
                               // Flag#
                               i8 val_1 = -33;
                               i8 val_2 = 55;
                               sprintf(strForMess, "%cFlag +\10Number -33 with flag %+d, and without3 %d.\
10Number 55 with flag %+d, and without %d.", 1, val_1, val_1, val_2, val_2);
                               setStatus(@exciterId():32, &strForMess);
                               setStatus(V-ADDR, 0);
                               count = 0;
                               return;
                        }
            }
}
</item>
```

The result of performing the examples in the interface::

The result of example 1 when the element script is clicked sequentially
"String – Hello World, len = 38″
"Addresses – current value of pointers"
"1 -2 55 -55″
"1 -2 22 -33"
"9.881313e-321"
"9.881313E-321"
"Octal system: 12"
"10 4294967286"
"Hexadecimal system lov: 0xff"
"Hexadecimal system hi:: 0xFF"
"%"
"Symbols: A @ U # g"

The result of example 2 when the element script is clicked sequentially

"cwidth with a string (with and without -): _ Hello World_ _ Hello World _"
"width with a number (with and without -): _ 33_ _33 _"
"width with a number (with and without -): _ 33_ _33 _"

" Octal(%o) and Hexadecimal(%x, %X) without and with the flag #:
%o – 310, 0310
%x – c8, 0xc8
%X – C8, 0XC8"

".precision with the string Hello world
%s – Hello world
%.4s – Hell"

".precision with number 43
%d – 43
%.6d – 000043"

"Flag+
Number -33 with flag -33,and without -33.
Number 55 with flag +55,and without 55"

srvError

go to content

srvError — message output to the server log.

Description:

```
void srvError(u8 *string[, u8 arg-list…]);
```

string message sending to the server log.

List of parameters:

string - String or pointer to the format control string.
arg-list - Variable number of parameters (up to 8).
The format control string operation is detailed in sprintf function.

Example:

```
<item id="524" RGB="524:16" name="Test for srvError function" type="script">

V-ID/V-ADDR
{
    u8 str[] = "Hello world!";
    srvError(&str);
    srvError("Hello %s! %d", "world", 12345);
}

</item>
```

The result of performing the example:

B6606460 2015/03/26 14:51:24.452 Error! 512:248: Hello world!
B6606460 2015/03/26 14:51:24.458 Error! 512:248: Hello world! 12345

srvMessage

go to content

srvMessage — message output to the server log.

Description:

```
void srvMessage (u8 *string[, u8 arg-list…]);
```

string message sending to the server log.

List of parameters:

string - String or pointer to the format control string.

arg-list - Variable number of parameters (up to 8).
The format control string operation is detailed in sprintf function.

Example:

```
<item id="524" RGB="524:16" name="Test for srvMessage function" type="script">

V-ID/V-ADDR
{
    u8 str[] = "Hello world!";
    srvMessage(&str);
    srvMessage("Hello %s! %d", "world", 12345);
}

</item>
```

The result of performing the example:
B6606460 2015/03/26 14:53:53.922 512:248: Hello world!
B6606460 2015/03/26 14:53:53.928 512:248: Hello world! 12345

str2low

go to content

str2low — string characters conversion into lowercase.

Description: `void str2low(u8 *string);`

Convert the string symbols of string into lowercase.

List of parameters: `string` - String or pointer to the string.

Example:

```
<item addr="524:248" name="Test for str2low function" type="script">

V-ID/V-ADDR
{
        if(opt0())
        {
            u8 str[] = "HELLO WORLD!";
            str2low(&str);
            u8 strForMess[50];
            sprintf(strForMess, "%cstr: %s", 1, str);
            setStatus(@exciterId():32, &strForMess);
        }
}

</item>
```

The result of performing the example in the interface:

str: hello world!

Note:

Attempt to send literal (str2low("TEXT")) as the parameter will occur the error.

strcmp

go to content

strcmp — compares the strings lexicographically.

Description:

```
u8 strcmp(u8 *string_1, u8 *string_2);
```

Compares the string string_1 with the string string_2 and returns numeric value.

List of parameters:

`string_1` - String or pointer to the string.

`string_2` - String or pointer to the string.

Return values:

Returns 0 – if string_1 and string_2 are equal.
Returns the value more than 0 – if string_1 is more than string_2.
Returns the value less than 0 – if string_1 is less than string_2.

Example:

```
<item addr="524:248" name="Test for strcmp function" type="script">

        V-ID/V-ADDR
        {
            if(opt0())
            {
                i8 cmp_1 = strcmp("hello!", "hello!");
                u8 str_1[] = "abc";
                u8 str_2[] = "aac";
                i8 cmp_2 = strcmp(&str_1, &str_2);
                i8 cmp_3 = strcmp(&str_2, "abc");

                // Result output to the interface
                u8 strForMess[50];
                sprintf(strForMess, "%ccmp1 = %d\10cmp2 = %d\10cmp3 = %d", 1, cmp_1, cmp_2, cmp_3);
                setStatus(@exciterId():32, &strForMess);
            }
        }

</item>
```

The result of performing the example in the interface:

cmp1 = 0

cmp2 = 1

cmp3 = -1

Note:

strcmp function considers the character case.

strcpy

go to content

strcpy — string copying.

Description:

`u8 strcpy(u8 *string_1, u8 *string_2)`

Copies the string string_2 into the string string_1 and return the target string length.

List of parameters:

string_1 - String or pointer to the string where it will be copied to.

string_2 - String or pointer to the string which will be copied.

Return values:

Returns the target string size in bytes.

Example:

```
<item addr="524:248" name="Test for strcpy function" type="script">

        V-ID/V-ADDR
        {
            if(opt0())
            {
                u8 str_1[50] = "first";
                u8 str_2[50] = "second";

                // example 1
                u8 copyResult_1 = strcpy(&str_1, &str_2);

                // Result output to the interface
                u8 strForMess[100];
                sprintf(strForMess, "%cstr_1: %s (len_1 - %d)", 1, str_1, copyResult_1);
                setStatus(@exciterId():32, &strForMess);

                // example 2
                u8 copyResult_2 = strcpy(&str_2, "third");

                // Result output to the interface
                sprintf(strForMess, "%cstr_2: %s (len_2 - %d)", 4, str_2, copyResult_2);
                setStatus(@exciterId():32, &strForMess);
             }
        }

</item>
```

The result of performing the example in the interface:

cstr_1: second (len_1 – 6)

cstr_2: third (len_2 – 5)

strlen

go to content

strlen — receiving the length of string.

Description:

```
u8 strlen(u8 *string);
```

Returns the string length of string in bytes.

List of parameters:

`string` - Input string or pointer to the string for which the length is measured.

Return values:

The string length of string in bytes in case of success and 0 if string is empty.
For double byte character set string the result will be twice as large.

Example:

```
<item addr="524:248" name="Test for strlen function" type="script">
V-ID/V-ADDR {
    if(opt0()) {
        // Example 1

        u8 result_1 = strlen("It"s the first string!");

        // Example 2

        u8 str[] = "what length of this string?";
        u8 result_2 = strlen(&str);

        // Result output to the interface
        u8 strForMess[50];
        sprintf(strForMess, "%c result_1 = %d, result_2 = %d", 1, result_1, result_2);
        setStatus(@exciterId():32, &strForMess);
    }
}
</item>
```

The result of performing the example in the interface:

result_1 = 22, result_2 = 27

strlen2

go to content

strlen2 — receiving the length for double byte character set string.

Description:

```
u8 strlen2(u8 *string);
```

Returns the string length of string in bytes.

List of parameters:

`string` - Input string or pointer to the string for which the length is measured.

Return values:

The string length of string in bytes in case of success and 0 if string is empty.

Example:

```
<item addr="524:248" name="Test for strlen2 function" type="script">

    V-ID/V-ADDR
    {
        if(opt0())
        {
            u8 str_1[] = toUcs2(65001, "Test string");

            // Usage of strlen2
            u8 result_1 = strlen2(&str_1);

             // Usage of strlen for the same string
            u8 result_2 = strlen(&str_1);

            // Result output to the interface
            u8 strForMess[50];
            sprintf(strForMess, "%cstrlen2(&str_1) = %d\10strlen(&str_1) = %d", 1, result_1, result_2);
            setStatus(@exciterId():32, &strForMess);
        }
    }
</item>
```

The result of performing the example in the interface:

strlen2(&str_1) = 22

strlen(&str_1) = 12

strtsr

strstr — finds the first insertion of substring.

Description:

```
u8 *strstr (u8 *string_1, u8 *string_2);
```

Returns the pointer to the substring of string string_1 starting from the first insertion of string_2 (including it) and till the end of string string_1.

List of parameters:

string_1 - Input string.
string_2 - String or pointer to the search string.

Return values:

Returns the pointer to the part of string or 0 if there is no string string_2 in string_1.

Example:

```
<item addr="524:248" name="Test for strstr function" type="script">

V-ID/V-ADDR
{
    if(opt0())
    {
        u8 str_1[50] = "SmartHouse Smart";
        u8 str_2[50] = "House";

        // Example: 1
        u8 *result_1 = strstr(&str_1, &str_2);

        // Example: 2
        u8 *result_2 = strstr("Larnitech", &str_2);

        // Example: 3
        u8 *result_3 = strstr("Larnitech", "Sm");

        // Example: 4
        u8 *result_4 = strstr(&str_1, "Larni");

        // Result output to the interface
        u8 strForMess[100];
        sprintf(strForMess, "%cresult_1: %s\10result_2: %d\10result_3: %s\10result_4: %d", 1, result_1,
result_2, result_3,
        result_4);

         setStatus(@exciterId():32, &strForMess);
    }
}
</item>
```

The result of performing the example in the interface:

result_1: House Larnitech

result_2: 0

result_3: tech

result_3: 0

Note:

strstr function considers the character case.

time()

go to content

time() — number of seconds since 01.01.1970.

Description:

```
u32 time();
```

Returns number of seconds followed 01.01.1970.

Return values:

Returns number of seconds followed 01.01.1970. Type of return value u32.

Example:

```
<item addr="524:248" INTERFACEID="2023" name="Test for time function" type="script">

V-ID/s:1
{
    u32 s = time();

    // Result output to the interface
    u8 strForMess[100];
    sprintf(strForMess, "%cSeconds followed 01.01.1970: %d", 1, s);
    setStatus(INTERFACEID:32, &strForMess);
}

</item>
```

The result of performing the example in the interface:

Every second the messages with current value of seconds passed from 01.01.1970 appear in the interface.

timeInRange(s-e)

go to content

timeInRange — checking if the present time hits the specified interval.

Description:

`u8 timeInRange(HH:MM-HH:MM);`

Checking if the the specified interval of time present time hits the interval HH:MM-HH:MM.

List of parameters:

HH:MM-HH:MM - Start time and end time of checked interval of time. Time is set in the format of HH:MM.

Return values:

Returns 0 if the present time doesn"t hit the interval HH:MM-HH:MM.
Returns the value not 0 if the present time hits the interval HH:MM-HH:MM.

Example:

```
<item addr="524:248" name="Test for timeInRange function" type="script">
V-ID/V-ADDR
{
    if(opt0())
    {
        u8 res = timeInRange(6:00-10:00);

        if(res != 0)
        {
            setStatus(@exciterId():32, {1, "Good morning!"});
        }

        res = timeInRange(10:01-16:00);

        if(res != 0)
        {
            setStatus(@exciterId():32, {1, "Good afternoon!"});
        }

        res = timeInRange(16:01-22:00);

        if(res != 0)
        {
            setStatus(@exciterId():32, {1,  "Good evening!"});
        }

        res = timeInRange(22:01-5:59);

        if(res != 0)
        {
            setStatus(@exciterId():32, {1,  "Good night!"});
        }
    }
}

</item>
```

The result of performing the example:

Depending on the time of day when clicking the element script the following messages will be displayed in the interface: "Good morning!", "Good afternoon!", "Good evening!" or "Good night!".

timeInRange(s-e|wd)

go to content

timeInRange — checking if the present time hits the specified interval of time and day of week.

Description:

`u8 timeInRange(HH:MM-HH:MM|day1[, day2…]);`

Checking if the present time hits the specified interval of time HH:MM-HH:MM and one or more specified days of week day1.

List of parameters:

`HH:MM-HH:MM` - Start time and end time of checked interval of time. Time is set in the format of HH:MM.

`day1` - Days of week separated by commas. Two-character symbols according to American standard are use for denoting the days of week:

`su` – Sunday
`mo` – Monday
`tu` – Tuesday
`we` – Wednesday
`th` – Thurthday
`fr` – Friday
`sa` – Saturday

Return values:

Returns 0 in case if the present time do not hit the specified interval of time or the present day do not correspond to the specified days.

Returns value not 0 in case if the present time hits the specified interval of time within the days specified.

Example:

```
<item addr="524:248" name="Test for timeInRange function" type="script">

V-ID/V-ADDR
{
    if(opt0())
    {
        u8 res = timeInRange(9:00-18:00|mo, tu, we, th, fr);

        if(res != 0)
        {
            setStatus(@exciterId():32, {1, "Working time!"});
        }

        res = timeInRange(18:01-8:59|mo, tu, we, th, fr);

        if(res != 0)
        {
            setStatus(@exciterId():32, {1, "Non-working period!"});
        }

        res = timeInRange(00:00-23:59|su, sa);

        if(res != 0)
        {
            setStatus(@exciterId():32, {1, "Weekends!"});
        }

    }
}

</item>
```

The result of performing the example:

Depending on the day of week and time of day when clicking the element script the following messages will be displayed in the interface: "Working time!", "Non-working period!" or "Weekends!".

V-ID@ID:SID

go to content

Pre-processing of setting the statuses that enter "ID:SID" element. Allows to process and change if required the status that is set for the device and its length optl.

Note:

if the status changes due to internal reasons (e.g. leakage or temperature valve) the pre-handler is not got called.

V-ID and ID should be equal, in other words the script which contains element"s status pre-handler should be processed on the same module as element.

Return values:

return 0; – ignore

return opt; – use the current

return newStatus; – new status on the stack (up to 256 bytes)

return may be missed then return opt adds automatically;

For example:

```
V-ID@V-ADDR
{
     u8 newStatus[2];
     optl = 2;
     newStatus[0] = time();
     return newStatus;
}
```

V-ADDR device status setting pre-processing, new status 2 bytes in length setting and changing the status setting for newStatus are shown in the example.

V-ID/ID:SID

go to content

"V-ID" device event handling which generate "ID:SID" element.

Example:

```
<item addr="524:248" ID="524" SID="32" LAMP="524:32" name="Message processing" type="script">

// example 1
524/524:32
{
setStatus(2023:32, {1, "Lamp status is changed."});
}

// example 2
V-ID/524:32
{
setStatus(2023:32, {1, "Lamp status is changed."});
}

// example 3
V-ID/ID:SID
{
setStatus(2023:32, {1, "Lamp status is changed."});
}

// example 4
V-ID/LAMP
6{
setStatus(2023:32, {1, "Lamp status is changed."});
}
</item>


//After compilation:

524/524:32
{
setStatus(2023:32, {1, "Lamp status is changed."});
}

524/524:32
{
setStatus(2023:32, {1, "Lamp status is changed."});
}

524/524:32
{
setStatus(2023:32, {1, "Lamp status is changed."});
}

524/524:32
{
setStatus(2023:32, {1, "Lamp status is changed."});
}
```

Result:

In all the examples when switching on or off the lamp 524:32 the following messages will be displayed in the interface: "Lamp status is changed."

V-ID/:SID

go to content

"V-ID" device events handling which generates the element with sub-id ":SID"in it.

Example:

```
<item addr="524:248" SID="24" name="Message processing" type="script">

    V-ID/:SID
    {
        if(opt0())
        {
            setStatus(@exciterId():32,{1, "RGB is activated"});
        }
    }
</item>


//After compilation:


        524/:24
        {
            if(opt0())
            {
            setStatus(@exciterId():32,{1, "RGB is activated"});
            }
        }
```

Result:

When clicking the element 524:24 the message "RGB is activated" will be displayed in the interface.

V-ID/[type]:[time]

go to content

Construct "V-ID/[tipe]:[time] { ... }" generates the event at specified intervals.

Parameters:

tipe - "m", "s" or "ms" are minutes, seconds or milliseconds correspondently.

time - Number of specified units of time. Maximum number of seconds which could be set when using tipe "s" is 0xFFFF and maximum number of minutes which could be specified when using tipe "m" si 0x0444. When using tipe "ms" milliseconds are to be specified in increments of 50 milliseconds, 10000 milliseconds is the maximum value.

Example:

```
<item addr="524:248" LAMP="512:16"  MS="524:24" name="Message processing" type="script">

V-ID/s:15
{
    setStatus(LAMP,![LAMP.0]);
}

V-ID/m:1
{
    u8
    strForMess[100];
    sprintf(strForMess,"%cMovement - %d%%", 1, [MS.1]);
    setStatus(2023:32,&strForMess);
}
</item>

//After compilation:

524/s:15
{
    setStatus(512:16,![512:16.0]);
}

524/m:1
{
    u8
    strForMess[100];
    sprintf(strForMess,"%cMovement - %d%%", 1, [524:32.1]);
    setStatus(2023:32,&strForMess);
}
```

Result:

Every 15 seconds the element LAMP will change its status to the opposite one and every minute the messages with motion level value from MS sensor will be displayed.

V-ID/ and ~

go to content

"V-ID/ { … }" construct when when combined with symbol "~" allows to trap the events. As in the following examples the symbol "~" is required to be used for trapping the event from the concrete element. At that the code V-ID/{…} will be processed when the events of the element or elements preceded by the symbol "~" would occur.

Example:

```
<item addr="524:248" LAMP512="512:16"  RGB524="524:24" DIM524=":27" name="Message processing"
type="script">
V-ID/ {
if([~LAMP512.0] == 1) {
setStatus(2020:32, {1, "Lamp from  512"});
}
if([~RGB524.0] == 1) {
setStatus(2020:32, {1, "RGB from  524"});
}

if([~DIM524.0] == 1) {
setStatus(2020:32, {1, "Dim  from  524"});
}
}
</item>
```

After compilation:

```
524/ {
if([~512:16.0] == 1) {
setStatus(2020:32, {1, "Lamp from 512"});
}

if([~524:24.0] == 1) {
setStatus(2020:32, {1, "RGB from 524"});
}

if([~:27.0] == 1) {
setStatus(2020:32, {1, "Dim  from 524"});
}
}
```

Result:

When clicking one of the elements LAMP512, RGB524 orDIM524 the corresponding message will be displayed:

```
"Lamp from 512"
"RGB from 524"
"Dim  from 524"
```

V-ID/ID1:SID1, ID2:SID2

go to content

"V-ID" device event handling which generate "ID1:SID1" element or "ID2:SID2" element.

Example:

```
<item addr="524:248" LAMP01="524:24" LAMP01="524:27" name="Message processing" type="script">

V-ID/LAMP02,
LAMP01
{
    if(opt0())
    {
        if(senderId() == 524 && senderSubId() == 24)
        setStatus(@exciterId():32,{1, "RGB is activated"});

        if(senderId() == 524 && senderSubId() == 27)
        setStatus(@exciterId():32,{1, "Dimmer is activated"});
    }
}
</item>


//After compilation:


524/524:27, 524:24
{
    if(opt0())
    {
        if(senderId() == 524 && senderSubId() == 24)
            setStatus(@exciterId():32,{1, "RGB is activated"});

        if(senderId()== 524 && senderSubId() == 27)
            setStatus(@exciterId():32,{1, "Dimmer is activated"});
    }
}
```

Result:

The messages "RGB is activated" or "Dimmer is activated" will be displayed in the interface depending on whether element LAMP01 or element LAMP02 is switched on.

V-ID/V-ADDR

go to content

The construct "V-ID/V-ADDR { ... }" can be used when writing scripts for message tracking and handling .
In this case the compiler would put device id where the script is located instead of "V-ID" and it will put the element script id and sub-id divided by ":" instead of "V-ADDR" .

Example:

```
<item addr="524:248" name="Message processing" type="script">
V-ID/V-ADDR
{
    u8
    strForMess[100];
    sprintf(strForMess,"%cID:SID - %d:%d", 1, senderId(), senderSubId());
    setStatus(@exciterId():32,&strForMess);
}

</item>


//After compilation:


524/524:248
{
    u8 strForMess[100];

    sprintf(strForMess,"%cID:SID - %d:%d", 1, senderId(), senderSubId());

    setStatus(@exciterId():32,&strForMess);
}
```

Result:

When clicking the element script in the interface the following message will be displayed:
"ID:SID – 524:248"

wday

wday — current week day.

go to content

Description: `u8 wday();`

Returns the current week day.

Return values: Returns the current week day starting from Sunday (Sunday is 0).

Example:

```
<item addr="524:248" name="Test for wday function" type="script">
V-ID/V-ADDR
{
    if(opt0())
    {
        u8 Wday = wday();
        u8 strForMess[70] = {1, "Today "};

        if(Wday == 0)
            strcat(strForMess, "Sunday!");

        if(Wday == 1)
            strcat(strForMess, "Monday!");

        if(Wday == 2)
            strcat(strForMess, "Tuesday!");

        if(Wday == 3)
            strcat(strForMess, "Wednesday!");

        if(Wday == 4)
            strcat(strForMess, "Thursday!");

        if(Wday == 5)
            strcat(strForMess, "Friday!");

        if(Wday == 6)
            strcat(strForMess, "Saturday!");

        setStatus(@exciterId():32, &strForMess);
        setStatus(V-ADDR, 0);
    }
}

</item>
```

The result of performing the example in the interface:

When clicking the element script the message with current day of week will appear in the interface.

weekTimeInRange

go to content

weekTimeInRange — checking if the present time hits the specified interval of time and day of week.

Description:

`u8 weekTimeInRange(000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000);`

Checking if the present time hits the specified interval of time of the week.

List of parameters:

Parameter in the format of 7 consecutive repeating successions of hexadecimal notations like XXXXXXXXXXXX The length of each succession is 12 characters for each day of week starting from Sunday. They represent bit-to-bit half-hour intervals (every character equals to 2 hours, every character bit equals to 30 minutes).

For example:

0001FFC00000 – from 7:30 till 13:00,

000000FE0380 – from 12:00 till 15:30 and from 19:00 till 20:30.

Return values:

Returns 0 if the present time doesn"t hit the interval .

Returns value not 0 if the present time hits the interval .

Example:

```
<item addr="524:248" name="Test for weekTimeInRange function" type="script">

V-ID/V-ADDR
{
    if(opt0())
    {
        u8 res = weekTimeInRange(000000000000000000000000FFFFFFFFFFFF000000000000FFFFFFFFFFFF000000000000FF
FFFFFFFFFF);

        if(res != 0)
            setStatus(@exciterId():32, {1, "Today is TU or TH or SA"});

        res = weekTimeInRange(000000000000FFFFFF000000000000000000000000FFFFFF000000000000FFFF
FF0000000000000000);

        if(res != 0)
            setStatus(@exciterId():32, {1, "Today is MO or WE up to 12:00 or FR after 12:00!"});

        res = weekTimeInRange(000000000000000000FFFFFF000000000000FFFFFF000000000000000000000000FFFF
FF000000000000);

        if(res != 0)
            setStatus(@exciterId():32, {1, "Today is MO or WE after 12:00 or FR up to 12:00!"});
    }
}

</item>
```

The result of performing the example:

Depending on the day of week and time of day when clicking the element script the following messages will be displayed in the interface: " Today is TU or TH or SA!", "Today is MO or WE up to 12:00 or FR after 12:00!" or " Today is MO or WE after 12:00 or FR up to 12:00!".

year

year — current year.

go to content

Description:

```
u32 year();
```

Returns the current year starting from 1900.

Return values:

Returns the current value of the year starting from 1900. The value is the integer number onsisting of three figures where the last two specify the current year (for example: 2012 – 112, 2014 – 114, 2015 – 115). Type of return value u32.

Example:

```
<item addr="524:248" name="Test for year function" type="script">

V-ID/V-ADDR
{
    if(opt0())
    {
        u32 Year = year();
        u32 Month = month();
        Year = Year % 100;
        u8 strForMess[70] = {1, "Now "};

                if(Month == 0)
                    strcat(strForMess, "January!");

                if(Month == 1)
                    strcat(strForMess, "February!");

                if(Month == 2)
                    strcat(strForMess, "March!");

                if(Month == 3)
                    strcat(strForMess, "April!");

                if(Month == 4)
                    strcat(strForMess, "May!");

                if(Month == 5)
                    strcat(strForMess, "June!");

                if(Month == 6)
                    strcat(strForMess, "July!");

                if(Month == 7)
                    strcat(strForMess, "August!");

                if(Month == 8)
                    strcat(strForMess, "September!");

                if(Month == 9)
                    strcat(strForMess, "October!");

                if(Month == 10)
                    strcat(strForMess, "November!");

                if(Month == 11)
                    strcat(strForMess, "December!");

        sprintf(strForMess, "%s 20%d year!", strForMess, Year);
        setStatus(@exciterId():32, &strForMess);
        setStatus(V-ADDR, 0);
    }
}

</item>
```

The result of performing the example in the interface:

When clicking the element script the message with current month and year will appear in the interface.

Scripts language

OTHER

Device storage

8 Kbyte device storage available for scripts and automations.

Troubleshooting

Element is not displayed in the interface

- check if the tablet is connected to server
- check if the server log while processing XML file is error-free
- check if the element in XML file was recorded correct
- check system field value (if available)

VoIP module. Sending of notifications

go to content

SetStatus(NOTIFY, message);

Message format:

`type|dest|message`
type: SMS, MMS, CALL, EMAIL
dest: dest1[,dest2[,dest3]]
....
`message`: message in UTF-8 Format
MMS message: [urls]|Message
EMAIL message: [urls]|Message

Examples:

SMS|+123456,+654321,some@email.com|Alarm! Water leakage detected
MMS|+123456|http://video/1.mpg|Intercom message
EMAIL|some@email.com|http://files/attachmen|Email message
EMAIL|some@email.com||Email message
EMAIL|some@email.com|Email message
CALL|+123456|http://file/message.mp3
CALL|+123456|/message.mp3
CALL|+123456|Message text

SRV-ID

Construct SRV-ID is changed into server ID

VOIP

Construct VOIP is used for sending SMS Calls over TCP/IP protocol

For example:

```
setStatus(VOIP, "SMS|123456789|Alarm! Water leakage detected");
```

Mediapoint control

MULTIROOM OUTPUT POINT CONTROL INSTRUCTIONS:

setStatus(MRID, {SpeakerCmd});

MRID – media point ID, for example 297,

SpeakerCmd – command.

List of commands:

```
SpeakerCmdPlay = 1, file/link play, setStatus(MRID:30, {1, VOLUME, PRIORITY, "MIDURL"});,
here VOLUME - sound level, PRIORITY - priority, MIDURL - link to media file. Example: setStatus(297:30, {1,
120, 0, "http://192.168.1.125/voice/sirena.mp3"});.
For reproducing from interface to the loudspeaker - setStatus(MRID:30, {1, 180, 0, "mr://INTERFACE_
ID:31"}); and to stop broadcasting - setStatus(MRID:30, 0);.
For reproducing from interface to the interface - setStatus(INTERFACE_ID1:30, {1, 180, 0, "mr://INTERFACE_
ID2:31"}); and to stop broadcasting - setStatus(INTERFACE_ID1:30, 0);.

SpeakerCmdStop = 0, stop. setStatus(MRID:30, 0);
 SpeakerCmdPause = 2, pause. setStatus(MRID:30, 2);.
 SpeakerCmdContinue = 3, continue. setStatus(MRID:30, 3);.
 SpeakerCmdVolume = 4, sound level setting VOLUME, 0-250.  setStatus(MRID:30, {4,VOLUME});.
 SpeakerCmdVolumeRel = 5, increment/decrement sound for VOLUME value from the current. setStatus(MRID:30,
{5,-VOLUME});.
 SpeakerCmdMute = 6, sound off. setStatus(MRID:30, {6,1})- mute off, setStatus(MRID:30, {6,0})- mute on.
SpeakerCmdBalanse = 7, balance.
 SpeakerCmdSetUrl = 8, URL play. setStatus(297:30, {8,"http://stream1.lux.fm:8088"});.
 SpeakerCmdSeek = 9, playback setting from the position specified in the command, milliseconds from track
start, 4 bytes. setStatus(MRID:30, {9,0,1,0,0}); - playback starting from 256 milliseconds.
SpeakerCmdSeekRel = 10, forward/backward winding.
SpeakerCmdPauseReverse = 11, change of the command "pause" into the command "continue" and change from
"continue" into "pause" setStatus(MRID:30, 11);
SpeakerCmdSetSyncMaster = 32, synchronise, setStatus(MRID:30,{32,MasterSubID,MasterID_low,MasterID_hi});
 SpeakerCmdPlayingSync = 33, synchronise the reproduction on the devices, setStatus(MRID:30,{32,SlaveSub-
ID,SlaveID_low,SlaveID_hi,SlaveSubID1,SlaveID1_low,SlaveID1_hi,...,SlaveSubIDN,SlaveIDN_low,SlaveIDN_hi});
 SpeakerCmdPlayingMove = 34, relocate the reproduction on the devices, setStatus(MRID:30,{32,,SlaveSub-
ID,SlaveID_low,SlaveID_hi,SlaveSubID1,SlaveID1_low,SlaveID1_hi,...,SlaveSubIDN,SlaveIDN_low,SlaveIDN_hi});
 SpeakerCmdError = 48, error.
 SpeakerCmdVolume2  = 0xfe, sound level for binding to dimmer, setStatus(MRID:30,{0xFE,VOLUME});.
SpeakerCmdSeek = 0xFF, change of the command "stop" into the command "continue" and change from "continue"
into "stop" setStatus(MRID:30, 0xFF);
```

Mediapoint control PRIORITY (0-lowest level, 250-highest)
If you play with priority 1 and higher, then the interface will have priority 0. Control from the application will not work.

MEDIA POINT STATUS:

Byte media point status:
0 – SpeakerStatusOffsetState, 0 – off, 1 – is playing, 2 – error, 4 – pause;
1 – SpeakerStatusOffsetVolume, sound level;
2 – SpeakerStatusOffsetMute, Mute status;
3 – SpeakerStatusOffsetBalanse, balance;
4 – SpeakerStatusOffsetPriority, priority;
5 – SpeakerStatusOffsetMsDuration, track length;
9 – SpeakerStatusOffsetMsDuration, track current position;
13 – SpeakerStatusOffsetSyncMasterAddr, device address synchronised with media point;
13 – SpeakerStatusOffsetSyncMasterSubId, device SubID synchronised with media point;
14 – SpeakerStatusOffsetSyncMasterId, device ID synchronised with media point;
16 – SpeakerStatusOffsetSyncSlaveCount, number of devices synchronised with media point. If non-zero values then the devices addresses will be recorded after URL;
17 – SpeakerStatusOffsetURL, tracked URL reference.

The comment as for Multiroom tools tab in the interface:

Message: reproduction of the message recorded to the speaker (speakers) via the microphone of interface. For recording the messages select speaker (speakers), volume and press to start speaking. After recording the message press the button "stop speaking" and the message will be displayed on the selected speaker.
Synchronize: reproduction is synchronized between 2 speakers. Upon that two speakers replay media files synchronically, reproduction is managed from the speaker it is synchronized with. For stopping press "stop" on the synchronized speaker.
Move reproduction: reproduction is moved. At that the reproduction is stopped on the speaker from which it is moved. It is possible to manage the track from the speaker where the reproduction is moved.
Synchronize play list: play list synchronization.

Playing on external upnp renderers from script

go to content

Playing on external upnp renderers from script (supported only DE-MG)

Play video on renderer:

```
setStatus(2043:1,{"uuid:7607c8bea69e36846e4163767fee6e62","cmdply","v:http://192.168.1.125/Dubstep.mp4"});
```

Play audio on renderer:

```
setStatus(2043:1,{"uuid:7607c8be-a69e-3684-6e41-63767fee6e62",
"cmdply","http://192.168.1.243/files/Audiofile.mp3"});
```

Stop playing:

```
setStatus(2043:1, {"uuid:7607c8be-a69e-3684-6e41-63767fee6e62","cmdstp"});
```

Heating profile operation from script

setStatus(id:sub-id,"Area(address)\AutoN(command)");

Name	type and variety	description	default value
id*	number	server ID	1000
sub-id*	number,br>100...102	element address on the device	
Area(address)	line	Area for running the command. If there is no recording then give the command to all connected to id:sub-id.	
Address – id:sub-id of connected device where the status is set			
AutoN(command)	string	Automation mode setting or command (0 – manual mode; "always-off" means permanently disabled)	

In case if set automation type is absent in the heating circuit the server will return the error and will assign to the circuit the default values with a setpoint of 21 degrees.

Examples:

setStatus(1000:100, 0); – for all: manual mode;
setStatus(1000:100, "always-off"); – for all: permanently disabled
setStatus(1000:100, "Komfort"); – for all: change to automation "Komfort ";
setStatus(1000:101, "Area1\0Komfort "); – for all in "Area1": change to automation "Komfort ";
setStatus(1000:101, "Area1\10SubArea11\0always-off"); – for all in "Area1" and in subarea "SubArea11": permanently disabled;
setStatus(1000:102, "16:1\0Komfort"); – heating circuit with the address 16:1, where 16 – heating control module ID, and 1 – its SUBID, specified in file logic.xml.
In a similar way the temperature can be specified for the current mode (but it is not changed in within specified intervals, only general temperature of mode). ts – temperature value is presented textually, t – simply u8.
setStatus(1000:101, "Area1\0ts:25"); – set the temperature of current mode as 25 degrees for all in "Area1";
setStatus(1000:101, {"Area1\0t:"A, temp,0}); – set the temperature recorded in temp variable for all in "Area1";
setStatus(1000:102, "891:1\0ts:25"); – heating with the address 891:1 set the temperature 25 degrees.

In a similar way the automation can be specified for the current mode as – automation value is presented textually, a – simply u8.
setStatus(1000:101, "Area1\0as:0"); – set the autimation mode;
setStatus(1000:101, {"Area1\0a:"A, temp,0}); – set the automation recorded in temp variable for all in "Area1";
setStatus(1000:102, "891:1\0as:0"); – heating with the address 891:1.

Set mode by it number
-1 mean always-off
-2 means previosly mode
-3 means manual mode

Message output to interface from script

go to content

```
setStatus(2047:32, {N, "Message"});
```

(2047 – broadcast)

N:

1. – Usual
4. – Important
8. – Critical

Video message output to the interface:

```
setStatus(ID:40,"url");
```

For example:

```
setStatus(2029:40, "http://192.168.1.125/wbp.mp4");
```

Dialogue output to the interface:

```
setStatus(ID:34, "text1|VIRT^1^text2|VIRT^0^text3|VIRT^2^text4");
```

where text1 – dialogue"s title;
VIRT – element to which setStatus 1, 0 or 2 will be sent accordingly due to pressing text2, text3 or text4 in the ID interface.

Major attributes description

go to content

Name	Value	Description
escription	string	Script description Is shown in the script selection list
tag	item or import-script	Tag name that will be added to xml file
Item will be added with type="script" attribute	Tag name that will be added to xml file	Area for running the command. If there is no recording then give the command to all connected to id:sub-id
selectArea	true or false	List of rooms. The script will be added to the selected room
addItems	array of objects or single object	List of variable configuration for web-form that will be added to the script as attributes
vars	array of objects or arrays	List of variable configuration for web-form that will be added to the script as attributes
name	line	Device name. The value of "name" attribute in "item" tag or "import-script" tag

Web form configuration description

Configuration for script and camera form

go to content

Configuration is set in xml format and is placed as a comment at the beginning of script or camera profile.

Example of configuration structure for script form:

```
/*
<?xml version="1.0" encoding="UTF-8"?>
<xml>
  <description>
    <b>Script for remote control LG</b>
    <br/>
   6711A20010B from conditioner LG
  </description>
  <tag value="import-script"/>
  <target value="IRT"/>
  <selectArea value="1"/>

  <additems>
    <additem tag="item" id="%TARGET%" name="Conditioner_LG_6711A20010B" sub-id="%SUBID%" type="conditioner"
     t-delta="14" t-min="16" vane-hor="0" vane-ver="0"/>
  </additems>

  <item type="devices-list" name="IRT" filter="ir-transmitter" required="1" comment="IR-transmitter"/>
   <item type="string" name="NAME" comment="Conditioner name" value="Conditioner"  min="3" max="40"
    regexp=""  required="1"/>
  <item type="hidden" name="COND" value="%TARGET%:%SUBID%"/>

</xml>
*/
```

Example of configuration structure for Foscam FI8918W camera form:

```
<?xml version="1.0" encoding="UTF-8"?>
<profile>
    <fields>
        <name value="Foscam FI8918W"/>
        <line>
            <item type="string" name="LHOST" comment="Local address of camera" value="192.168.1.151"
             required="1" min="7" max="15" regexp="" mask="099.099.099.099" weight="6"/>
            <item type="number" name="LPORT" comment="Port for video stream" value="554" required="1"
             min="1" max="65535" regexp="" weight="3"/>
        </line>
        <line>
            <item type="string" name="RHOST" comment="Remote address of camera" value="192.168.1.158"
             required="1" min="7" max="100" regexp="" weight="6"/>
            <item type="number" name="RPORT" comment="Port for video stream" value="554" required="1"
             min="1" max="65535" regexp="" weight="3"/>
        </line>
        <line>
            <item type="string" name="VLOGIN" comment="Login" value="admin" required="1" min="3" max="50"
             regexp="" weight="4"/>
            <item type="string" name="VPASSWD" comment="Password" value="123456" required="0" min="1"
             max="30" regexp="" weight="4"/>
            <item type="checkbox" name="AUTOPLAY" comment="Auto playback" value="0" required="0"
             weight="4"/>
        </line>
    </fields>

<xml><item>
name="%NAME%"
id="2048"
type="rtsp"
url="http://%LOGIN%:%PASSWD%@%LHOST%:%LPORT%"
url-remote="http://%LOGIN%:%PASSWD%@%RHOST%:%RPORT%"
http-auth="%LOGIN%:%PASSWD%"
video-uri="/videostream.cgi"
ffmpeg-opt="fmt=mjpeg;analyzeduration=100"
top-uri="/decoder_control.cgi?command=0"
bottom-uri="/decoder_control.cgi?command=2"
left-uri="/decoder_control.cgi?command=6"
right-uri="/decoder_control.cgi?command=4"
stop-uri="/decoder_control.cgi?command=1"
auto-play="%AUTOPLAY%"/>
</xml>
</profile>
```

vars block description

go to content

In this block the fields of script variables, their positioning in the form of addendum and editing are described.

Example of a single field layout in the string:

```
<item type="string" name="VAR1" comment="String" regexp="[0-9]{6,12}" value="12120873"  min="5" max="10"
width="300" required="1"/>
```

The text box (optional for filling) will be displayed in the form. Name attribute contains the name of script variable which has to contain capital letters and numbers, the name of variable for camera is to be enclosed in percent symbols %LOGIN%. Only 5 to 10 symbols specified in "filter" attribute can be entered in the field. Its description will be displayed to the left of field. The field length is 300 px.

Example of several fields layout in one string:

```
<item type="weekday" name="DAY1" comment="Days of week" value="" required="1"/>
<item type="time" name="TIME11" comment="Time from" required="0"/>
```

Two fields will be positioned in one string. The description of field is taken from the first found "desc" attribute. If the language is selected the description will be taken from "desc-language" attribute (e.g.,: desc-ru, desc-en, desc-fr, desc-ukr). For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .

Field type is specified in "type" attribute.

addItems block description

In this block the elements that will be added after the added tag of script in xml are described.

List of attributes:

Attribute	Value	Description
tag	line	Tag name. For example: item
name	line	Device name
type	string	Device type. For example: lamp or dim-mer-lamp
id	integer value	device id
sub-id	integer value	device sub-id

Scripts language

OTHER

Types of fields

Field type: devices-list

Field with the drop-down list of devices. Devices are selected from "item" tags with "type" attribute. The list of devices will contain only those devices which have "type" attribute value in the list of "filter" attribute values if in the field description "filter" attribute is present.

For example:
filter="door-sensor,leak-sensor,switch"
devices with "type" attribute the value of which equals to door-sensor or leak-sensor or switch will be included into the list

If the filter attribute is not specified then in the drop-down list all the devices with "item" tag will be displayed.

Example:

```
<item type="devices-list" name="SENS1" filter="leak-sensor,door-sensor,switch"
 comment="Leakage sensor sensing element " required="1" />
```

List of attributes:

Attribute	Value	Description
name	string	Include the name of script variable which has to contain capital letters and numbers For example: SENS1 or VAR5
type	devices-list	
required	true, false	Denote that the field is to be filled in obviously
filter	strings array	Select the devices with values of types specified in the array. For example: filter:["door-sensor","leak-sensor","switch"]. If in the filter only "switch" type is specified then after device selection the button for autotraining the button will appear. It provides the button quick search in a long list.
desc	line	Field description. It is displayed to the left of field by default. For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .
descWidth	integer value	The width of field description, pixels. Equals to 100 by default.
descAlign	left, right, top	Field description layout respectively to field. It can be placed leftward, rightward and at the top.
width	integer value	Field width, pixels.
flex	floating-point number, it can be integer	Field width respectively to screen size and sizes of other fields that are placed in one string with this field. For example: flex:1 – if the string contains one field then it will be stretched up to the right hand border of the form. flex:0.5 – it will be stretched up to the middle of the form flex:2 – field width will be greater than other fields have with flex:1 If "width" attribute is present then "flex" attribute won"t be applicable.

Field type: areas-list

Field with the drop-down list of rooms. Path of the chosen room will be added to the value of a variable.

Example:

```
<item type="areas-list" name="PATH2" comment="Path"/>
```

List of attributes:

Attribute	Value	Description
type	areas-list	
name	string	nclude the name of script variable which has to contain capital letters and numbers For example: SENS1 or VAR5
required	true, false	Denote that the field is to be filled in obviously
desc	string	Field description. It is displayed to the left of field by default. For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .
descWidth	integer value	The width of field description, pixels. Equals to 100 by default.
descAlign	left, right, top	Field description layout respectively to field. It can be placed leftward, rightward and at the top.
weight	integer value	Field width, pixels.
flex	floating-point number, it can be integer	Field width respectively to screen size and sizes of other fields that are placed in one string with this field. For example: flex:1 – if the string contains one field then it will be stretched up to the right hand border of the form flex:0.5 – it will be stretched up to the middle of the form flex:2 – field width will be greater than other fields have with flex:1

Field type: text, string

Free text field. Text entering can be filtered. Minimum and maximum length of entered text can be specified.

Example:

```
<item type="devices-list" name="SENS1" filter="leak-sensor,door-sensor,switch"
 comment="Leakage sensor sensing element " required="1" />
```

List of attributes:

Attribute	Value	Description
name	string	Include the name of script variable which has to contain capital letters and numbers For example: SENS1 or VAR5 Obvious attribute.
type	text, string	Obvious attribute.
filter	string	Only symbols specified in the string could be entered
min	integer value	Minimum character lengths to be entered
max	integer value	Maximum character lengths to be entered
Value	string	Text shown in the field by default
required	true, false	Denote that the field is to be filled in obviously
desc	string	Field description. It is displayed to the left of field by default. For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .
descWidth	integer value	The width of field description, pixels. Equals to 100 by default.
descAlign	left, right, top	Field description layout respectively to field. It can be placed leftward, rightward and at the top.
weight	integer value	Field width, pixels.
flex	floating-point number, it can be integer	Field width respectively to screen size and sizes of other fields that are placed in one string with this field. For example: flex:1 – if the string contains one field then it will be stretched up to the right hand border of the form. flex:0.5 – it will be stretched up to the middle of the form flex:2 – field width will be greater than other fields have with flex:1 If "width" attribute is present then "flex" attribute won"t be applicable.

Field type: number

go to content

Numbers entry field.

Example:

```
1|    <item type="number" name="VAR2" value="30" required="0" min="10" max="50" comment="Number" width="170"/>
```

List of attributes:

Attribute	Value	Description
name	string	Include the name of script variable which has to contain capital letters and numbers For example: SENS1 or VAR5 Obvious attribute.
type	number	Obvious attribute.
min	integer value	Minimum number to be entered It can be negative.
max	integer value	Maximum number to be entered. It can be negative.
Value	number	Number shown in the field by default
required	true, false	Denote that the field is to be filled in obviously
desc	string	Field description. It is displayed to the left of field by default. For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .
descWidth	integer value	The width of field description, pixels. Equals to 100 by default.
descAlign	left, right, top	Field description layout respectively to field. It can be placed leftward, rightward and at the top.
weight	integer value	Field width, pixels.
flex	floating-point number, it can be integer	Field width respectively to screen size and sizes of other fields that are placed in one string with this field. For example: flex:1 – if the string contains one field then it will be stretched up to the right hand border of the form. flex:0.5 – it will be stretched up to the middle of the form flex:2 – field width will be greater than other fields have with flex:1 If "width" attribute is present then "flex" attribute won"t be applicable.

Field type: hidden

go to content

Hidden field. Its value may contain the parameter of script variable which is not subject to edit.

Example:

```
1|    <item type="hidden" name="VAR4" value="data"/>
```

List of attributes:

Attribute	Value	Description
name	string	Includes the name of script variable which has to contain capital letters and numbers For example: SENS1 or VAR5 Obvious attribute.
type	hidden	Obvious attribute.
value	string	The value of a variable. Names of script variables that exist in the form could be added to the value. These names will be changed into values entered into these fields. Name should be enclosed in percent symbols. Example: {value:"%VAR1%:%VAR2%"} {value:"apple"}

Field type: list

Field with the drop-down list of specified values.

Example:

```
<item type="list" name="VAR3" comment="List
of values" required="1" width="350">
    <option key="banana" value="Banana"/>
    <option key="pear" value="Pear"/>
    <option key="plum" value="Plum"/>
    <option key="apple" value="Apple"/>
    <option key="apricot" value="Apricot"/>
    <option key="cherry" value="Cherry"/>
    <option key="peach" value="Peach"/>
  </item>
```

List of attributes:

Attribute	Value	Description
name	string	Includes the name of script variable which has to contain capital letters and numbers For example: SENS1 or VAR5Obvious attribute.
type	list	Obvious attribute.
data	hash	Value:name list . Value is inserted into the value of script variable. Name is shown in the list. Example: {data:{"banana":"Banana","pear":"Pear","plum":"Plum"}} Obvious attribute.
Value	string	Value chosen in the list by default
required	true, false	Denote that the field is to be filled in obviously
desc	string	Field description. It is displayed to the left of field by default. For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .
descWidth	integer value	The width of field description, pixels. Equals to 100 by default.
descAlign	left, right, top	Field description layout respectively to field. It can be placed leftward, rightward and at the top.
weight	integer value	Field width, pixels
flex	floating-point number, it can be integer	Field width respectively to screen size and sizes of other fields that are placed in one string with this field. For example: flex:1 – if the string contains one field then it will be stretched up to the right hand border of the form. flex:0.5 – it will be stretched up to the middle of the form flex:2 – field width will be greater than other fields have with flex:1 If "width" attribute is present then "flex" attribute won"t be applicable.

Field type: irt

Field with the drop-down list of IR-transmitter signals.

Example:

```
<item type="irt" item-type="remote-control" name="COMMAND1_2,IRT1_2" comment="IR comand 2" weight="9"
 required="0"/>
```

List of attributes:

Attribute	Value	Description
name	2 elements array	Includes 2 names of script variables which have to contain capital letters and numbers. The first variable value will be substituted for signal value. The second variable value will be substituted for value from transmitter-addr attribute. For example: name:["IRT","COMMAND"] Obvious attribute.
type	irt	Obvious attribute.
itemType	string	Type of item tag. Remote by default. Signal values are added from remote-signal tag which belong to item tag with specified type.
required	true, false	Denote that the field is to be filled in obviously
desc	line	Field description. It is displayed to the left of field by default. For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .
descWidth	integer value	The width of field description, pixels. Equals to 100 by default.
descAlign	left, right, top	Field description layout respectively to field. It can be placed leftward, rightward and at the top.
weight	integer value	Field width, pixels.
flex	floating-point number, it can be integer	Field width respectively to screen size and sizes of other fields that are placed in one string with this field. For example: flex:1 – if the string contains one field then it will be stretched up to the right hand border of the form. flex:0.5 – it will be stretched up to the middle of the form flex:2 – field width will be greater than other fields have with flex:1 If "width" attribute is present then "flex" attribute won"t be applicable.

Field type: weekday

go to content

Field with weekdays list. Is displayed ad check boxes.

Example:

```
1|    <item type="weekday" name="DAY1" comment="Days of week" value="" required="1"/>
```

List of attributes:

Attribute	Value	Description
name	string	Includes the name of script variable which has to contain capital letters and numbers For example: SENS1 or VAR5 Obvious attribute.
type	weekday	Obvious attribute.
cols	integer value	Number of columns (from 1 to 7) Equals to 7 by default.
required	true, false	Denote that the field is to be filled in obviously
desc	string	Field description. It is displayed to the left of field by default. For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .
descWidth	integer value	The width of field description, pixels. Equals to 100 by default.
descAlign	left, right, top	Field description layout respectively to field. It can be placed leftward, rightward and at the top.
weight	integer value	Field width, pixels.
flex	floating-point number, it can be integer	Field width respectively to screen size and sizes of other fields that are placed in one string with this field. For example: flex:1 – if the string contains one field then it will be stretched up to the right hand border of the form. flex:0.5 – it will be stretched up to the middle of the for mflex:2 – field width will be greater than other fields have with flex:1 If "width" attribute is present then "flex" attribute won"t be applicable.

Field type: time

Field with drop-down list of hours and minutes (24-hour format).

Example:

```
1|    <item type="time" name="TIME11" comment="Time from" required="0"/>
```

List of attributes:

Attribute	Value	Description
name	String or array of 2 elements	Include the name of script variable which has to contain capital letters and numbers For example: TIME2 If the array of 2 variables is set then the value in the second variable will be one minute greater than the value from the first variable. Example: {name:["TIME","M1TIME"]} Obvious attribute.
type	time	Obvious attribute.
required	true, false	Denote that the field is to be filled in obviously
desc	string	Field description. It is displayed to the left of field by default. For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .
descWidth	integer value	The width of field description, pixels. Equals to 100 by default.
descAlign	left, right, top	Field description layout respectively to field. It can be placed leftward, rightward and at the top.
weight	integer value	Field width, pixels.
flex	floating-point number, it can be integer	Field width respectively to screen size and sizes of other fields that are placed in one string with this field. For example: flex:1 – if the string contains one field then it will be stretched up to the right hand border of the form. flex:0.5 – it will be stretched up to the middle of the form flex:2 – field width will be greater than other fields have with flex:1 If "width" attribute is present then "flex" attribute won"t be applicable.

Field type: checkbox

Single check box. If it is selected then the value is true by default.

Example:

```
<item type="checkbox" name="VAR10" comment="Choose checkbox" value="1" required="1" checkValue=""
 uncheckValue="" checked="true"/>
```

List of attributes:

Attribute	Value	Description
name	string	Includes the name of script variable which has to contain capital letters and numbers For example: SENS1 or VAR5 Obvious attribute.
type	checkbox	Obvious attribute.
checked	true, false	If the value is true the field is selected by default.
checkValue	string	If the field is selected it is assigned to the value of variable.
uncheckValue	line	If the field is not selected it is assigned to the value of variable.
required	true, false	Denote that the field is to be filled in obviously
desc	string	Field description. It is displayed to the left of field by default. For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .
descWidth	integer value	The width of field description, pixels. Equals to 100 by default.
descAlign	left, right, top	Field description layout respectively to field. It can be placed leftward, rightward and at the top.

Field type: comment

Text string, free text It can be used for describing the sections with fields or field description when one string contains several fields.

Example:

```
1|    <item type="comment" name="comment1" comment="%%color:#393;%%Any text" value=""/>
```

List of attributes:

Attribute	Value	Description
type	number	Obvious attribute.
style	string	String style description with the help of Css
text	line	Displayed text For description in another language add to "text" attribute the hyphen and language name and enclose them in quotation marks. For example: "text-ru":"", "text-en":"", "text-ukr":"" .
desc	line	Field description. It is displayed to the left of field by default. For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .
descWidth	integer value	The width of field description, pixels. Equals to 100 by default.
descAlign	left, right, top	Field description layout respectively to field. It can be placed leftward, rightward and at the top.
weight	integer value	Field width, pixels.
flex	floating-point number, it can be integer	Field width respectively to screen size and sizes of other fields that are placed in one string with this field. For example: flex:1 – if the string contains one field then it will be stretched up to the right hand border of the form. flex:0.5 – it will be stretched up to the middle of the for mflex:2 – field width will be greater than other fields have with flex:1 If "width" attribute is present then "flex" attribute won"t be applicable.

Field type: interval-time (xm)

go to content

Field for selecting the intervals of time. Usually is used for automation.

Example:

```
<item type="interval-time" name="VAR13,VAR14" comment="interval of time" value="19:21 - 19:22"
 required="1"/>
```

List of attributes:

Attribute	Value	Description
name	string	Includes the name of script variable which has to contain capital letters and numbers For example: SENS1 or VAR5 Obvious attribute.
type	interval-time	Obvious attribute.
value	string	Text shown in the field by default
required	true, false	Denote that the field is to be filled in obviously
desc	string	Field description. It is displayed to the left of field by default. For description in another language add to "desc" attribute the hyphen and language name and enclose them in quotation marks. For example: "desc-ru":"", "desc-en":"", "desc-ukr":"" .
descWidth	integer value	The width of field description, pixels. Equals to 100 by default.
descAlign	left, right, top	Field description layout respectively to field. It can be placed leftward, rightward and at the top.
weight	integer value	Field width, pixels.
flex	floating-point number, it can be integer	Field width respectively to screen size and sizes of other fields that are placed in one string with this field. For example: flex:1 – if the string contains one field then it will be stretched up to the right hand border of the form. flex:0.5 – it will be stretched up to the middle of the form flex:2 – field width will be greater than other fields have with flex:1 If "width" attribute is present then "flex" attribute won"t be applicable.

%TARGET% and %SUBID%

Template variables set in the attribute values of tags where id and subid of script are to be specified in the value.

%TARGET% – id attribute value of the selected device specified in the main configuration attribute – target.

%SUBID% – the value is generated by selecting from 250 to 1 and if it is absent in the existing sub-id then it is generated according to selected device id specified in the main configuration attribute – target. %SUBID01%, %SUBID02%..%SUBID99% are also can be added.