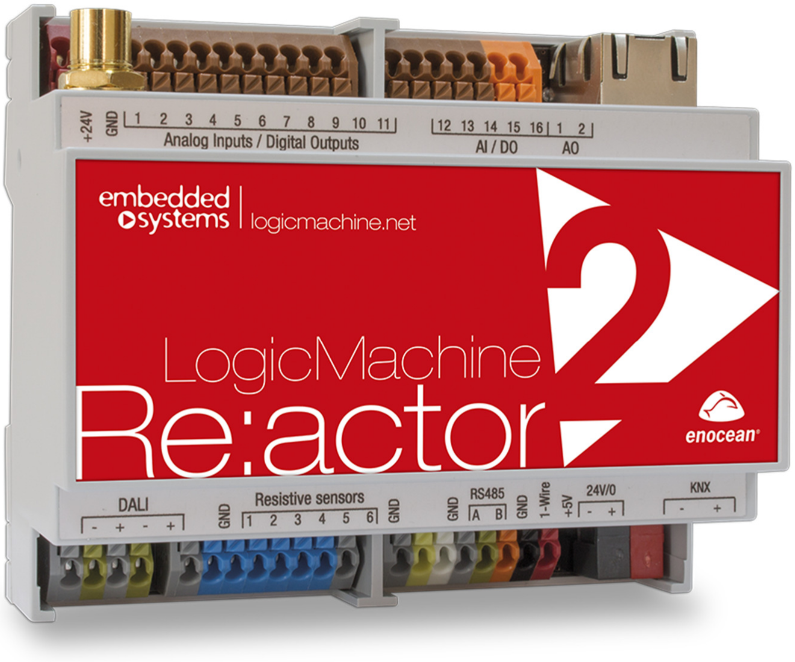
Embedded Systems SIA, VAT No LV40003411103
47. Katolu str., Riga, LV 1003, LATVIA
Phone: +371 67648888, fax: +371 67205036, e-mail: sales@openrb.com

LogicMachine3 Reactor V2

Product Manual



Document Issue 1.0
December, 2014

Technical Support:
support@openrb.com

Copyright

Notice

Embedded Systems SIA., reserves the right to modify the information contained herein as necessary. Embedded Systems SIA assumes no responsibility for any errors which may appear in this document. Information in this document is provided solely to enable system and software implementers to use KNX/EIB LogicMachine product.

Trademarks

LogicMachine is a trademark of Embedded Systems SIA. All other names and trademarks are the property of their respective owners and are hereby acknowledged.

Introduction

LogicMachine is your easiest way to program complex logic in KNX/EIB, Modbus, BACnet, EnOcean, DALI, 1-Wire networks. The LogicMachine will enable you to efficiently customize building automation processes, easily delivering unlimited flexibility benefit to end users in a cost-effective way.

LogicMachine is an embedded platform with integrated TPUART. LogicMachine allows to use it as IP Router, cross-standard gateway, logic engine, and visualization WEB SCADA server. Scripting templates provides user-friendly, flexible configuration interface. Via applying custom scripts the LogicMachine can simultaneously act as thermostat, security panel, lighting controller, etc

Technical support

Any faulty devices should be returned to Embedded Systems.

If there are any further technical questions concerning the product please contact our support, available Mon-Fri 9:00 – 17:00 GMT +02:00. Please write to support@openrb.com.

Firmware updates are available at www.openrb.com

Caution
Security advice

The installation and assembly of electrical equipment may only be performed by skilled electrician. The devices must not be used in any relation with equipment that supports, directly or indirectly, human health or life or with application that can result danger of people, animals or real value

Mounting advice

The devices are supplied in operational status. The cables connections included can be clamped to the housing if required.

Electrical connection

The devices are constructed for the operation of protective low voltage (SELV). Grounding of device is not needed. When switching the power supply on or off, power surges must be avoided.

Contents

Device specification

Application

Logical functions; WEB SCADA visualization for PC and touch-devices; cross-standard gateway between KNX, Modbus BACnet, EnOcean, 1-Wire, DALI and other protocols; integration with third party devices over RS232 or RS485 serial ports – AV, IR; Data logger with trends; HVAC

Types of product

LogicMachine3 Re:actor V2 LM3-REACTOR-2

Standards and norms compliance

CE conformity:	EMBS-CE-111231/01	Electromagnetic compatibility
EMC:	EN61000-6-1 EN61000-6-3	
PCT	Certificate	

Technical data:

Power supply:	7-36V DC	
Power consummation:	1.5W	
Interface:	EnOcean 868MHz	1
	10BaseT/100BaseTX	1
	RS485	1
	USB2.0	1
	TPUART2	KNX/EIB compatible
	DALI	1
	1-Wire	1
	Analog input / Digital output	16 (Configurable). 380 mA continues current on output. Optoislated from KNX/EIB bus. Additional 24V power source is supported. When used as analog input - 0-30V with configurable threshold voltage, 12bit resolution.
	Resistive sensor inputs	6 (PT100, PT1000, 0 Ω...20M Ω)
	Analog outputs	2 (0-10V, 12bit resolution, 20mA max current)
Connections:	KNX bus:	Bus Connection Terminal 0.8mm2

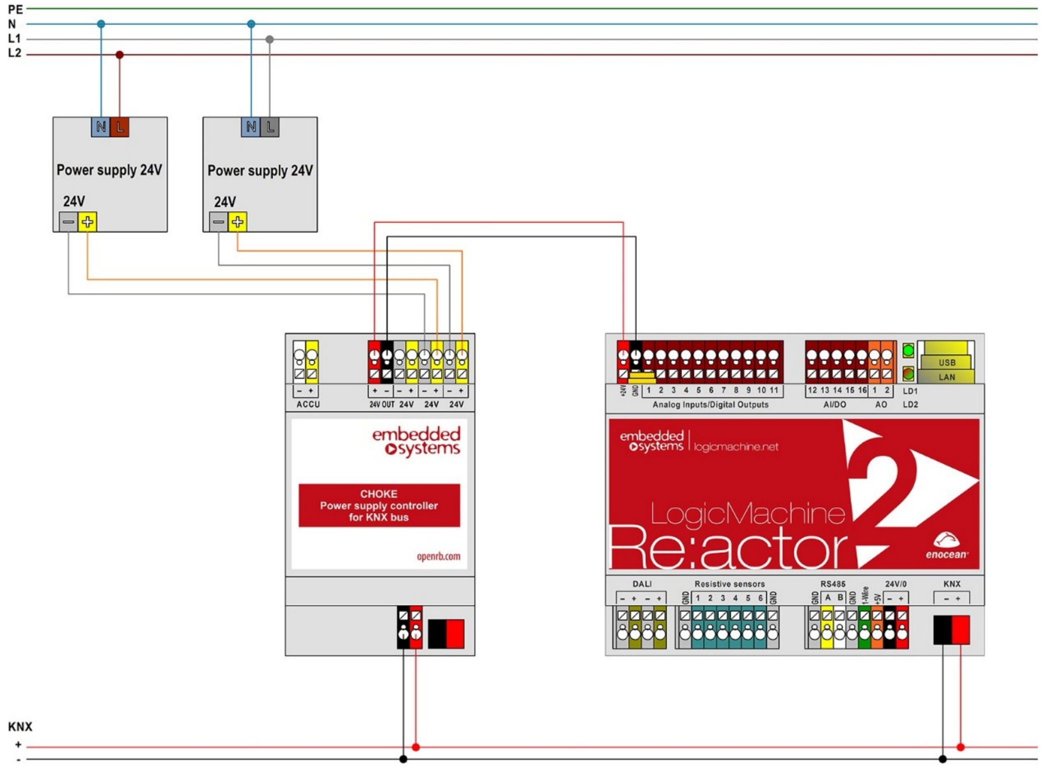
	Power supply:	Clamp, 1.5mm2
	Serial:	Clam, 1.5mm2
	IO:	Clam, 1.5mm2
	DALI:	Clam, 1.5mm2
	1-Wire:	Clam, 1.5mm2
Operating elements	LED	1 – CPU load
		1 - Activity
Enclosure:	Material:	Polyamide
	Color:	Gray
	Dimensions:	104(W)x90(H)x51(L) mm
Usage temperature:	-5C ... +45C	
Storage temperature:	-25C ... +55C	
Weight:	150g	
Warranty:	2 years	

LogicMachine3 Reactor V2 kit contains:

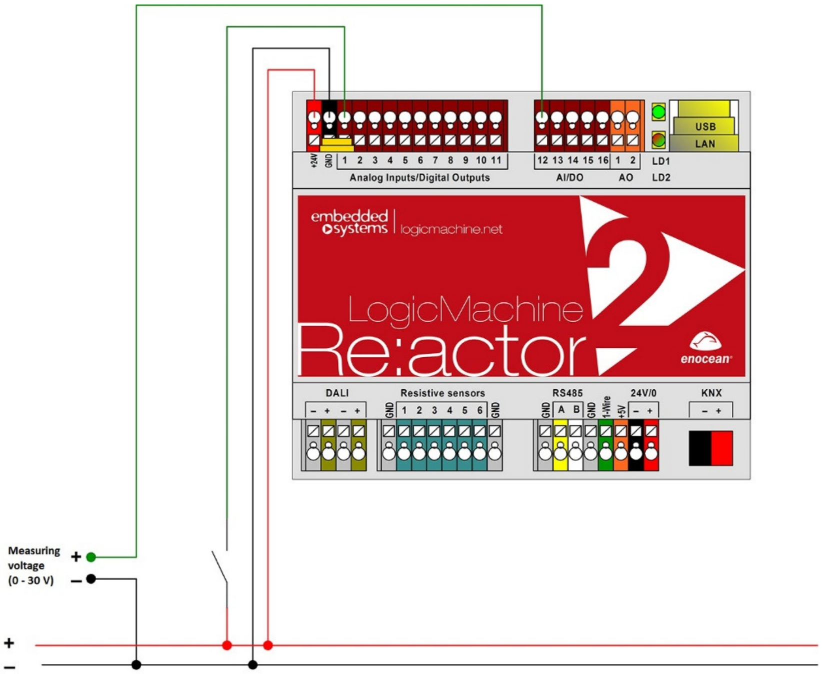
- Embedded board with preinstalled software
- Plastic DIN-rail case
- EnOcean antenna with cable 868MHz
- WAGO mounting connectors

Terminal connection schemes

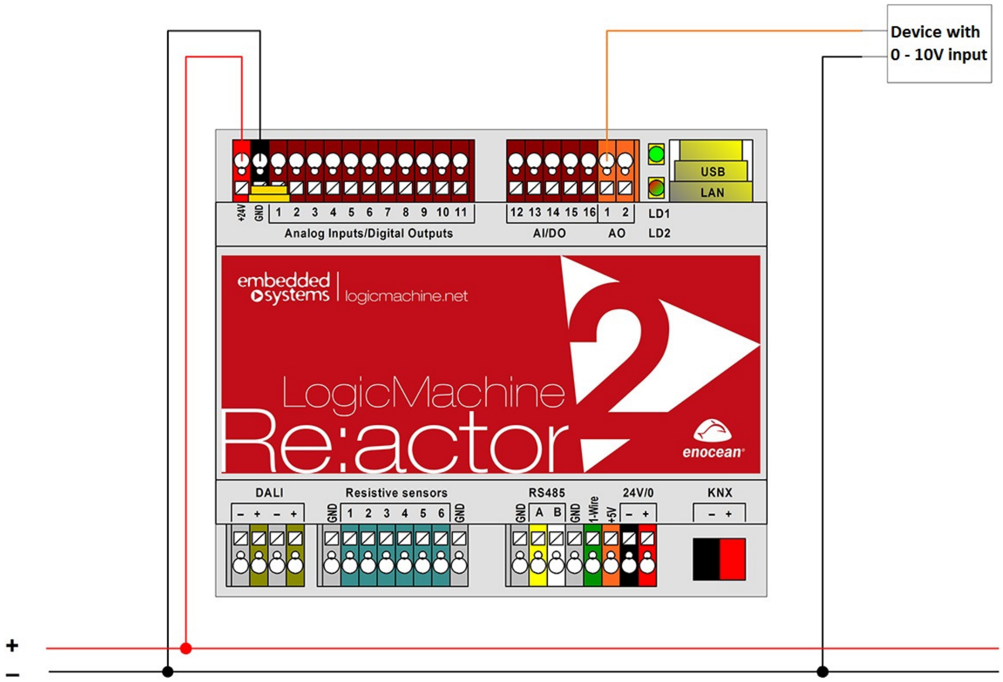
KNX/EIB TP1 connection



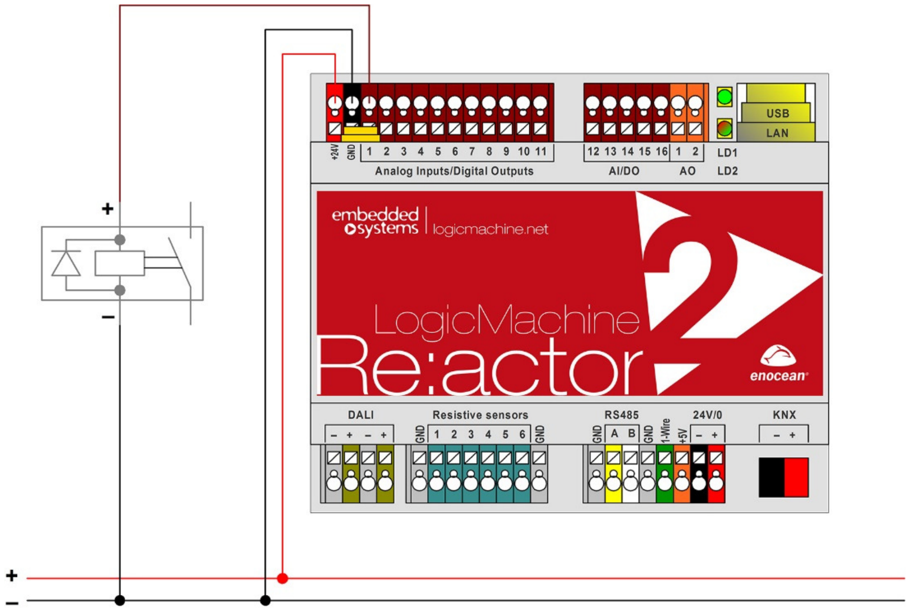
Analog input 0 – 30 V



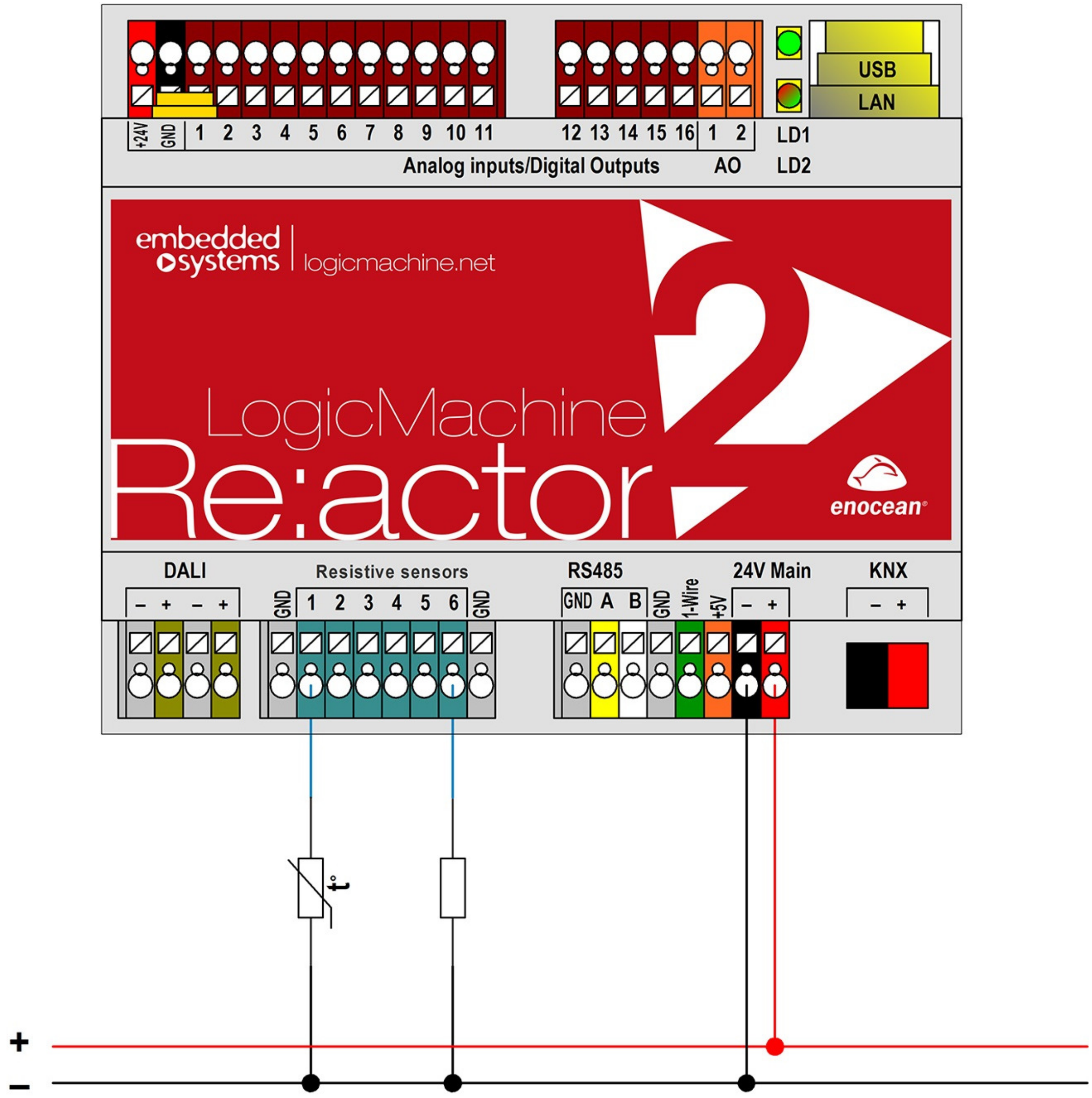
Analog output 0 – 10 V



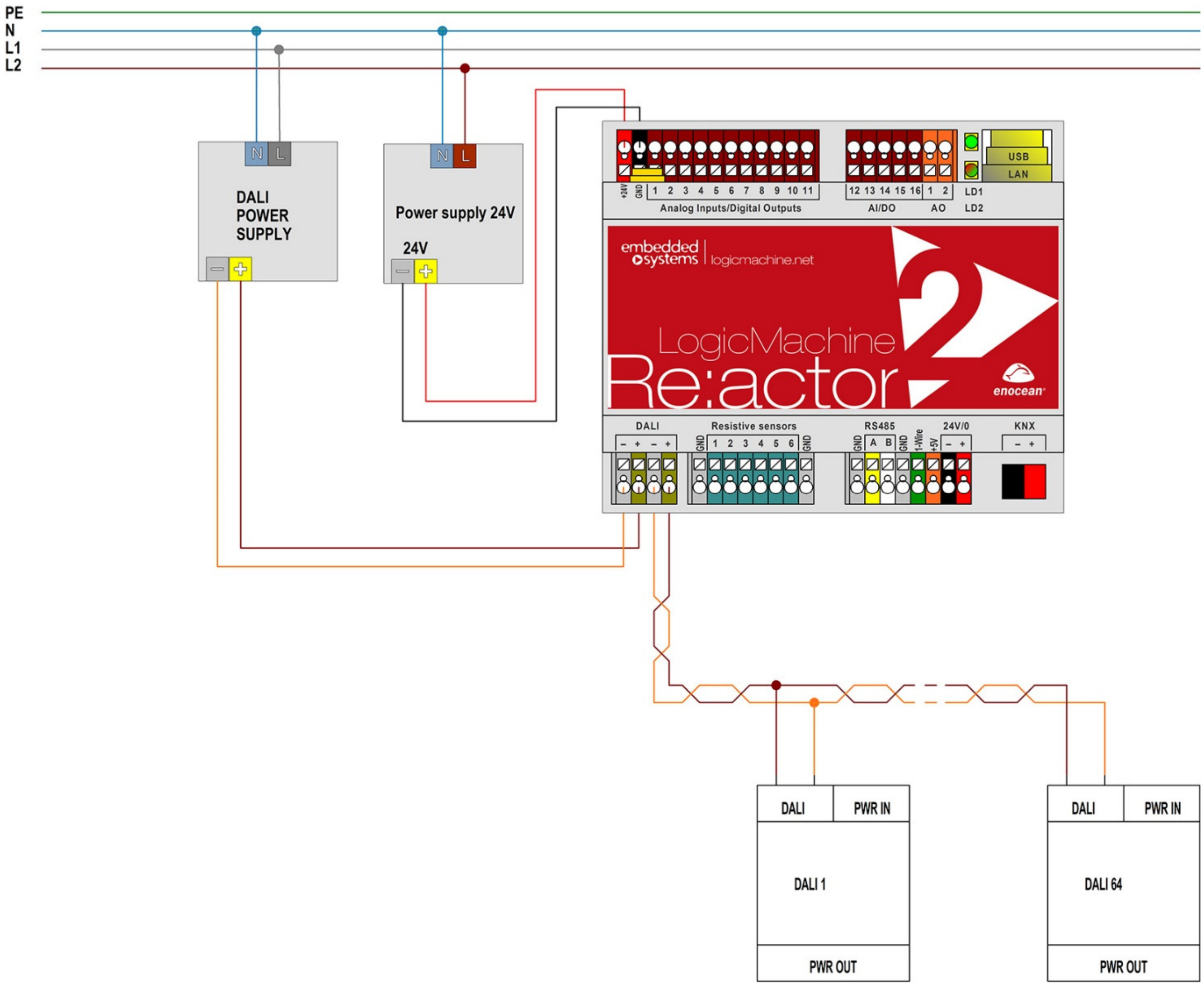
Digital output (e.g. external relay)



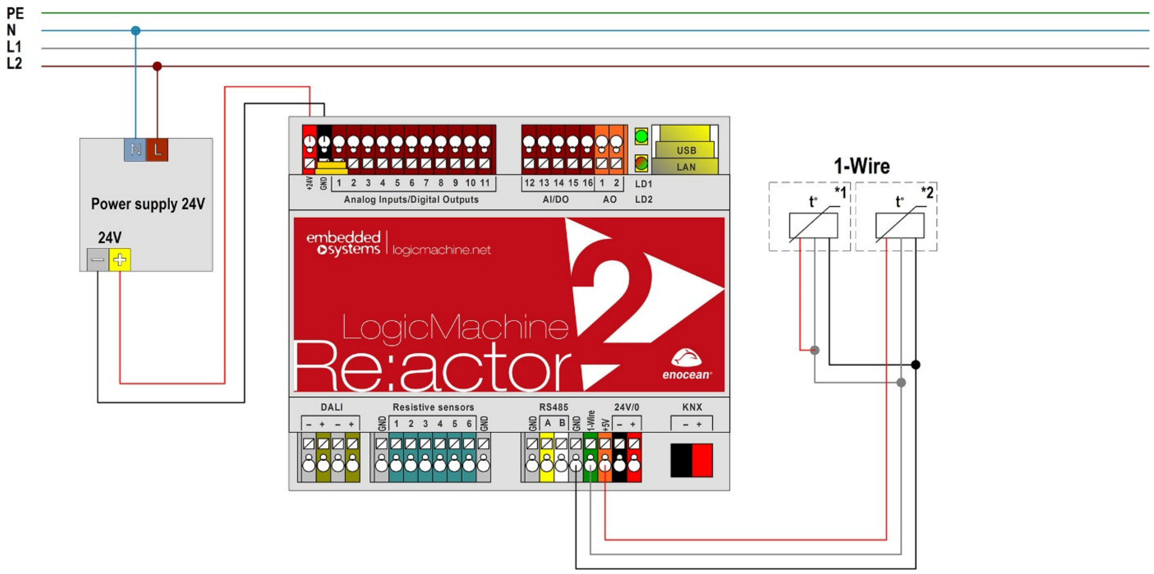
Resistive sensor input (e.g. PT1000)



DALI line connection



1-wire fieldbus interconnection



Factory default, discover IP

There is a reset button on the side of LogicMachine. You can either reboot the device by pressing this button or reset the configuration to factory defaults:

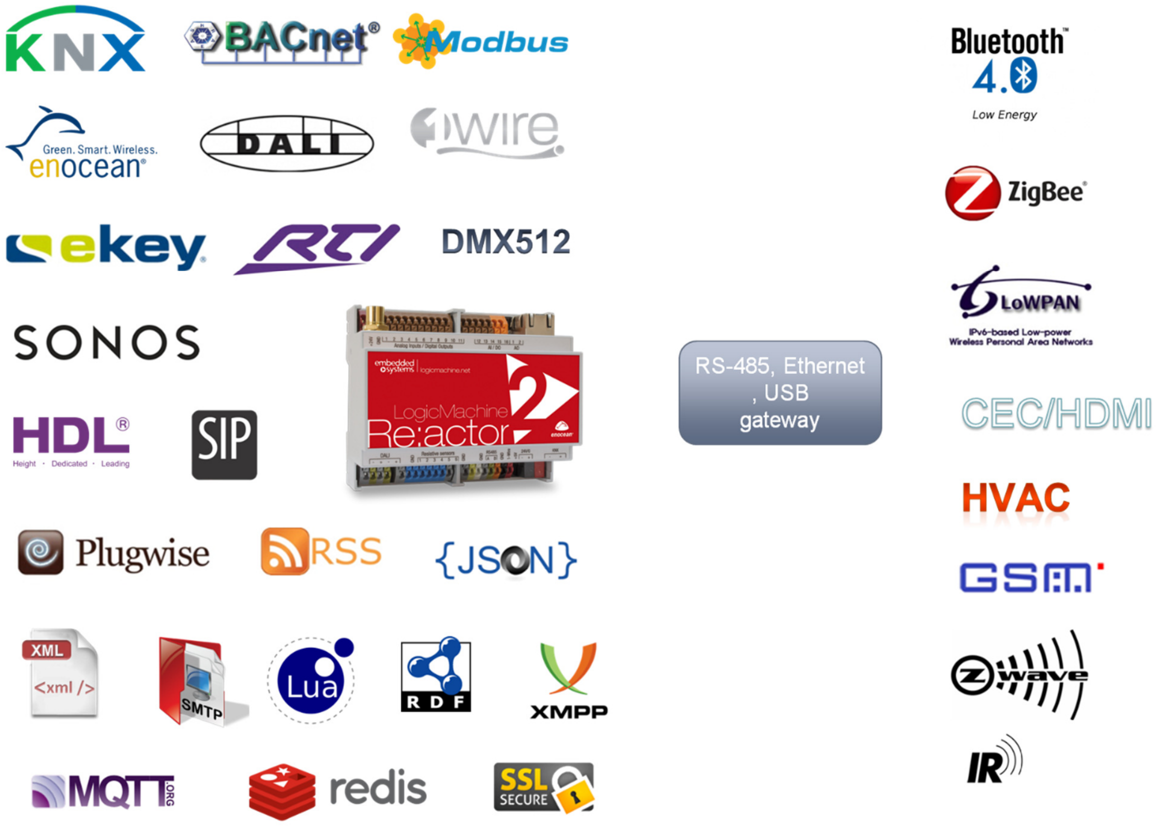
- *Press and hold for <10 sec* – reboot the device
- *Press and hold for >10 sec* – reset networking with IP to factory default
- *Press and hold for >10 sec and again press and hold for >10 sec* – full reset of configuration to factory defaults

There is also another possibility to discover IP address – LM3 has built-in zeroconf utility by default, so using the following applications you can find out the IP:

- Windows PC – *ServiceBrowser*
- Linux PC – *Avahi*
- Android – *ZeroConf Browser*
- iOS – *Discovery*

For more info please see here: http://openrb.com/discover-ip-of-logic-machine-or-streaming-player/

Standards supported



LogicMachine is compatible with the following standards:

- KNX/EIB TP, KNXnet/IP
- Modbus TCP, Modbus RTU Client/Server
- BACnet IP, Client/Server
- GSM (Huawei E173 and similar modem support through USB) for sending SMS notifications and controlling the installation by receiving SMS commands.
- DMX512 (in the box, through RS485)
- DALI
- 1-Wire
- Ekey biometrical access systems (RS485)
- HVAC systems can be controller through RS232 interface by using scripting
- SMTP/Email, SSL
- SIP
- XML (export object values, alerts or errors; integration with Fidelio)
- RSS (read Error or Alert tab content)
- JSON, XMPP
- ..

The system is made so that each of the standards can be used with each other, so LogicMachine can act as BACnet to DALI gateway or Modbus to GSM etc.

Quick startup guide

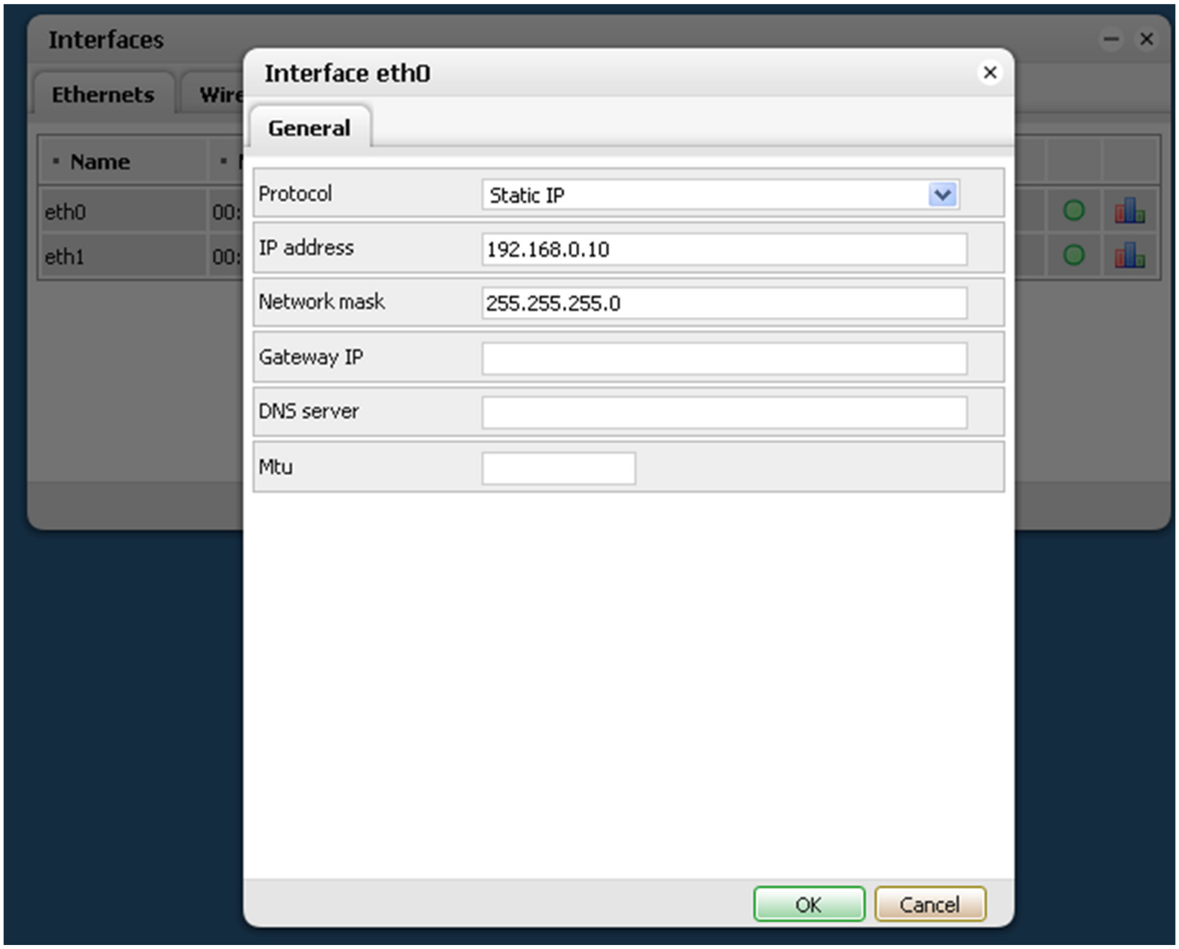
1) Mount the device on DIN rail
2) Connectthe KNX bus cable
3) Connect 24V power supply to the device (red pole to *24V+*, grey pole to *GND*)
4) Connect Ethernet cable coming from the PC

Default IP configuration

*Logic Machine/System Configuration*Login name	**admin**
*Logic Machine/System Configuration*Password	**admin**
User mode visualization/Touch visualization Login name	Read-only: **visview** Write: **viscontrol** Write + admin level: **visadmin**
User mode visualization/Touch visualization Password	Read-only: **visview** Write: **viscontrol** Write + admin level: **visadmin**
IP address on LAN	**192.168.0.10**
Networks mask on LAN	255.255.255.0

Change IP settings

In *System configuration* →*Network* → *Interfaces* window click on the specific interface to change the IP settings.



Protocol: Static IP
IP address: 192.168.0.10
Network mask: 255.255.255.0
Gateway IP:
DNS server:
Mtu:

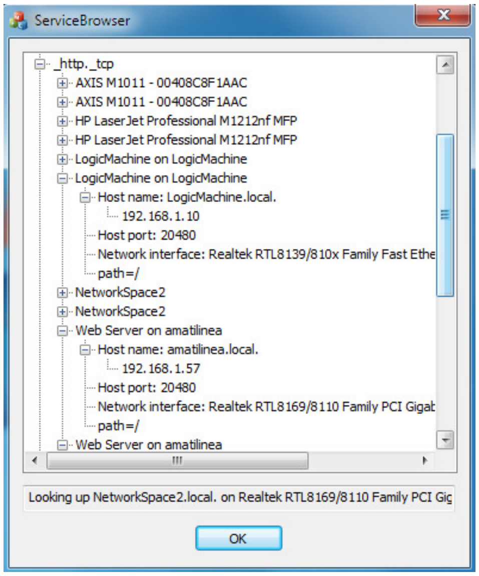
- ***Protocol***– specific protocol used for addressing
  - ***Static IP*** – static IP address. By default 192.168.0.10
  - ***DHCP*** – use DHCP protocol to get IP configuration.
    - ***Current IP***– the IP address got from DHCP server. This field appears only if the IP address is given otherwise it's hidden.
- ***Network mask*** – network mask. By default 255.255.255.0 (/24)
- ***Gateway IP*** – gateway IP address
- ***DNS server*** – DNS server IP address
- ***MTU***– maximum transmission unit, the largest size of the packet which could be passed in the communication protocol. By default 1500

When changes are done, the following icon appears in the top-right corner. This should be applied changes to take effect. Apply changes

Discover LogicMachine IP address

Windows PC

Easiest way is by using the utility **ServiceBrowser** which can be downloaded here: *http://marknelson.us/2011/10/25/dns-service-discovery-on-windows/*

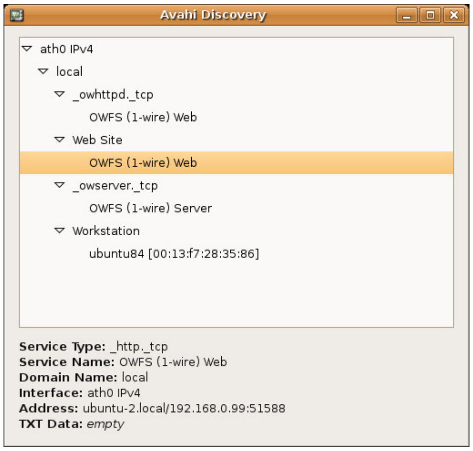


```
_http._tcp
  AXIS M1011 - 00408C8F1AAC
  AXIS M1011 - 00408C8F1AAC
  HP LaserJet Professional M1212nf MFP
  HP LaserJet Professional M1212nf MFP
  LogicMachine on LogicMachine
  LogicMachine on LogicMachine
    Host name: LogicMachine.local.
      192.168.1.10
    Host port: 20480
    Network interface: Realtek RTL8139/810x Family Fast Ethe
    path=/
  NetworkSpace2
  NetworkSpace2
  Web Server on amatilinea
    Host name: amatilinea.local.
      192.168.1.57
    Host port: 20480
    Network interface: Realtek RTL8169/8110 Family PCI Gigab
    path=/
  Web Server on amatilinea
```

Looking up NetworkSpace2.local. on Realtek RTL8169/8110 Family PCI Gig

Linux PC

The utility called **Avahi**, can be downloaded here:
www.avahi.org

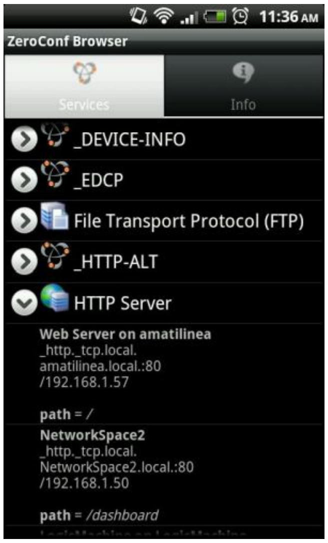


```
ath0 IPv4
  local
    _owhttpd._tcp
      OWFS (1-wire) Web
    Web Site
      OWFS (1-wire) Web
    _owserver._tcp
      OWFS (1-wire) Server
    Workstation
      ubuntu84 [00:13:f7:28:35:86]
```

Service Type: _http._tcp
Service Name: OWFS (1-wire) Web
Domain Name: local
Interface: ath0 IPv4
Address: ubuntu-2.local/192.168.0.99:51588
TXT Data: *empty*

Android

The freely available app called **ZeroConf Browser**, can be downloaded in *Play Store*:
https://play.google.com/store/apps/details?id=com.grokkt.android.bonjour&hl=en



_DEVICE-INFO

_EDCP

File Transport Protocol (FTP)

_HTTP-ALT

HTTP Server

Web Server on amatilinea
_http._tcp.local.
amatilinea.local.:80
/192.168.1.57

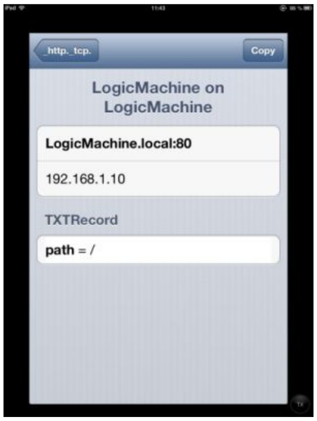
path = /

NetworkSpace2
_http._tcp.local.
NetworkSpace2.local.:80
/192.168.1.50

path = /dashboard

<u>iOS/Mac OS</u>
The freely available app called **Discovery**, can be downloaded in *App Store*:
https://itunes.apple.com/en/app/discovery-bonjour-browser/id305441017?mt=8

For iPad install the iPhone/iPod version of the utility.



LogicMachine on
LogicMachine

LogicMachine.local:80

192.168.1.10

TXTRecord

path = /

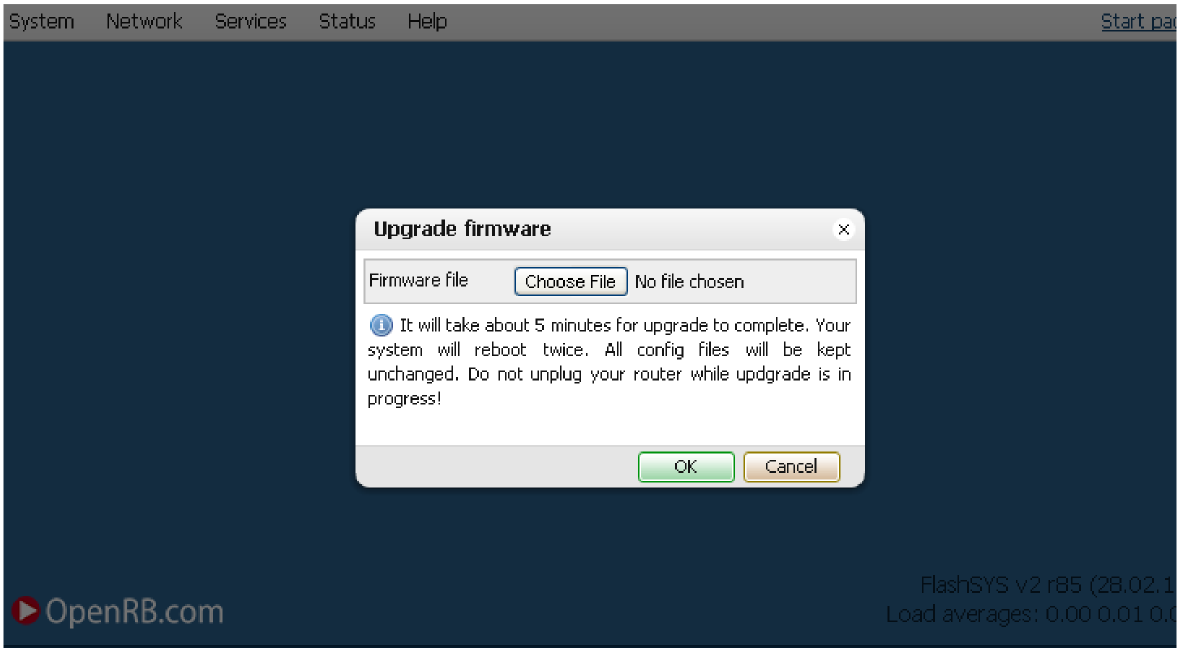
Firmware upgrade

Note! Before each upgrade please backup your visualization, scripts and object in *Logic Machine → Tools → Backup.*

Note! After each upgrade, we strongly recommend to clean your browser cache.

Use web browser to perform upgrade of the software of Logic Machine. Firmwares are available in a form of images and could be downloaded from support page ofwww.openrb.com.

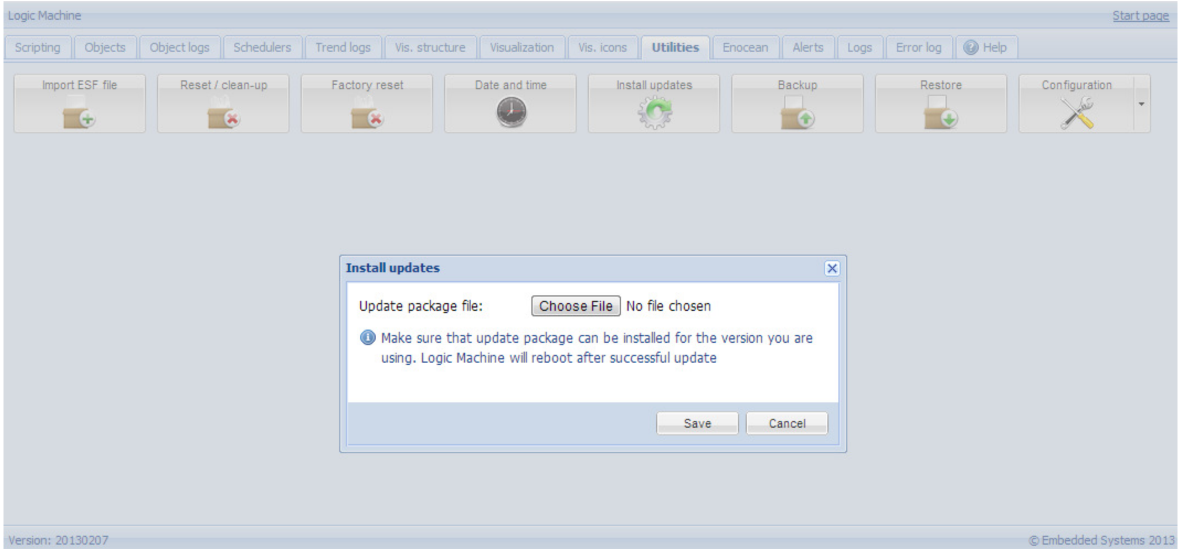
<u>Complete system upgrade</u> can be done in *System Configuration →System →Upgrade firmware*



Firmware file Choose File No file chosen

It will take about 5 minutes for upgrade to complete. Your system will reboot twice. All config files will be kept unchanged. Do not unplug your router while updgrade is in progress!

LogicMachine visualization upgrade or patch installation can be done in *Utilities* tab and press on *Install updates* icon. After *.LMU file is chosen from the corresponding location press *Save* button. The device will be rebooted after 5 seconds and new firmware will be installed.



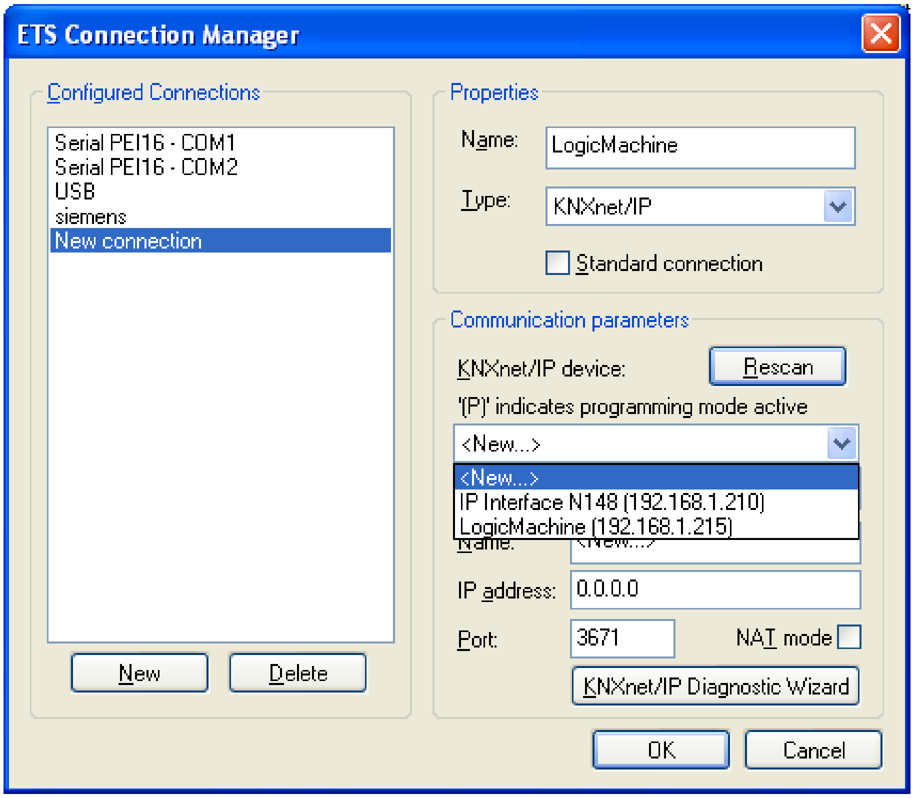
Update package file: Choose File No file chosen

Make sure that update package can be installed for the version you are using. Logic Machine will reboot after successful update

LogicMachine for KNX/EIB network configuration management with ETS

To use LogicMachine with KNXnet/IP functionality and program other KNX bus devices, the device should be added into *ETS Connection Manager.*

- Go to *Extras* → *Options* →*Communication* →*Configure interfaces*



Serial PEI16 - COM1
Serial PEI16 - COM2
USB
siemens
New connection

Name: LogicMachine

Type: KNXnet/IP

□ Standard connection

KNXnet/IP device:

'[P]' indicates programming mode active

<New...>

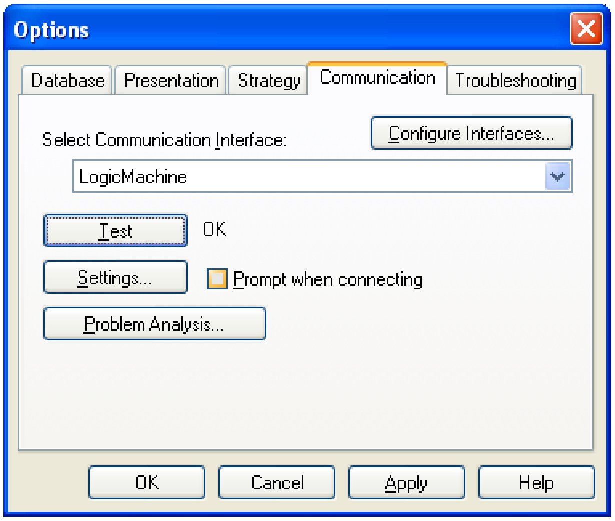
<New...>
IP Interface N148 (192.168.1.210)
LogicMachine (192.168.1.215)

Name: <New...>

IP address: 0.0.0.0

Port: 3671 NAT mode □

- Put some freely chosen *Name* for the connection
- Chose ***Type* = *KNXnet/IP***
- Press ***Rescan*** button and then choose from the drop down menu found LogicMachine
- Press ***OK***
- Back in ***Options*** → ***Communication*** window select newly created interface as ***Communication Interface*** from the drop-down menu.
- To test the communication with ETS, press ***Test*** button.



Select Communication Interface: LogicMachine

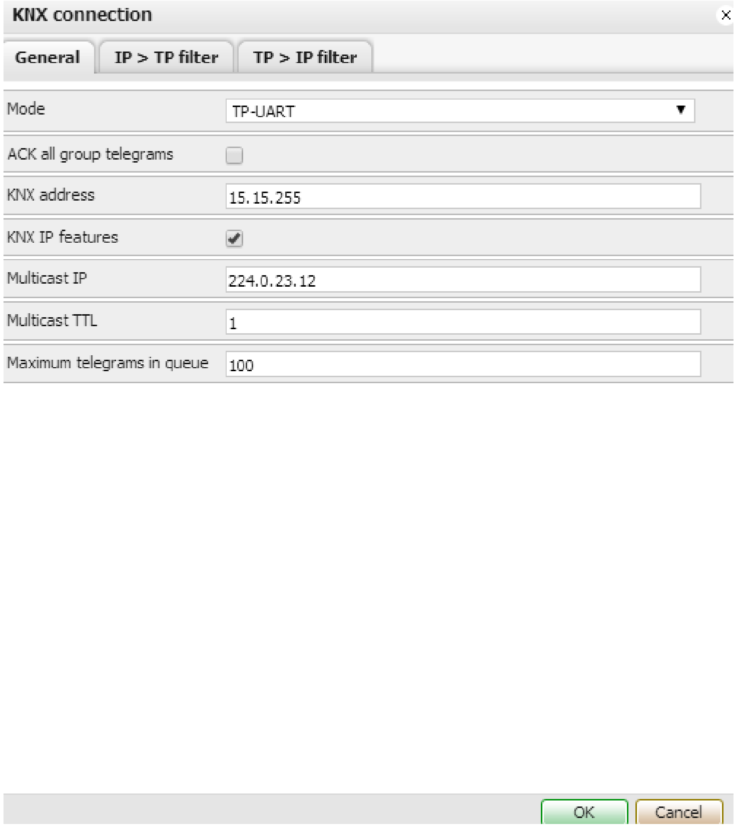
Test OK

□ Prompt when connecting

- Make sure that bus status is Online – press button in ETS.

KNX and IP Router settings

KNX specific configuration is located in *System configuration*→ *Network* → *KNX connection* window.



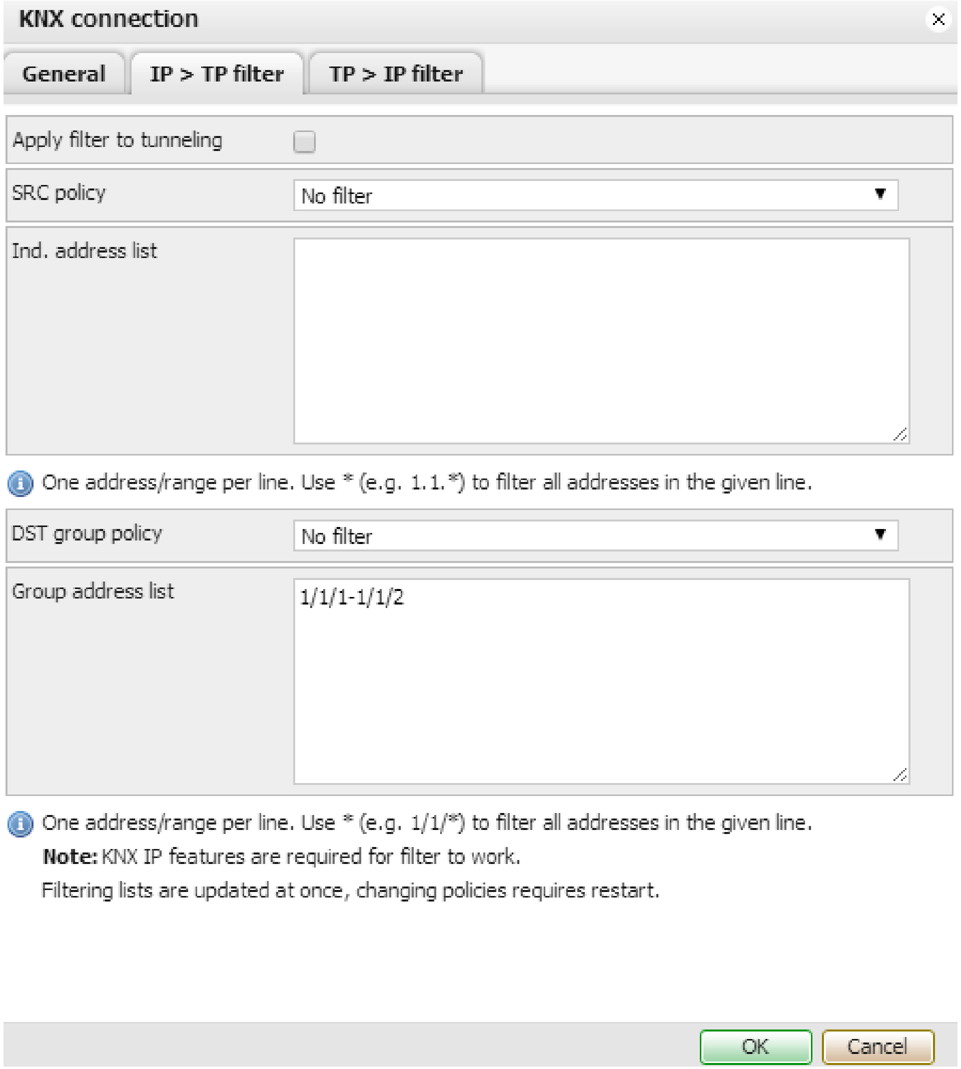
Mode	TP-UART ▼
ACK all group telegrams	☐
KNX address	15.15.255
KNX IP features	☑
Multicast IP	224.0.23.12
Multicast TTL	1
Maximum telegrams in queue	100

General tab

- ***Mode*** *[TP-UART / EIBnet IP Tunneling / EIBnet IP Tunneling(NAT mode) / EIBnet IP Routing]* – KNX connection mode. LogicMachine3 Reactor has TPUART interface by default built-in. Note! If there is no KNX TP connected to the device, it will automatically offer to switch to KNXnet/IP mode.
- ***ACK all group telegrams*** – acknowledge receipt of telegram to all group communication
- ***Parameter***–KNX corresponding interface in OS of the system
- ***KNX address*** – KNX physical address of the device
- ***KNX IP features*** – Use this device with KNX IP features e.g. for KNXnet/IP network configuration
- ***Multicast IP*** – multicast IP address
- ***Multicast TTL*** – Time to live for multicast telegram in seconds
- ***Maximum telegrams in queue*** – count of maximum telegrams in the queue

IP > TP filter

Filtering table for telegrams going from IP network to KNX TP1 is located in this submenu.



Apply filter to tunneling □

SRC policy: No filter ▼

Ind. address list

One address/range per line. Use * (e.g. 1.1.*) to filter all addresses in the given line.

DST group policy: No filter ▼

Group address list: 1/1/1-1/1/2

One address/range per line. Use * (e.g. 1/1/*) to filter all addresses in the given line.

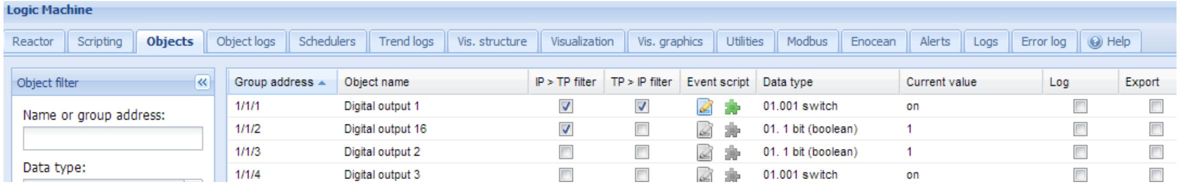
Note: KNX IP features are required for filter to work.

Filtering lists are updated at once, changing policies requires restart.

- ***Apply filter to tunneling*** – either to apply filter policy to telegrams in tunneling mode. If ETS is used it is recommended to turn this feature off.
- ***SRC policy*** *[No filter / Accept selected individual addresses / Drop selected individual addresses]*– policy to apply to the list of source addresses
- ***Ind. address list*** – list of individual addresses. One address/range per line. Use * (e.g. 1.1.*) to filter all addresses in the given line.
- ***DST group policy****[No filter / Accept selected group addresses / Drop selected group addresses]*– policy to apply to the list of destination group addresses
- ***Group address list*** – list of group addresses. One address/range per line. Use * (e.g. 1/1/*) to filter all addresses in the given line.

Note! *KNX IP features* should be on for filter to work. Filtering lists are updated at once, changing policies requires restart.

Note that group address list can be filled automatically by checking necessary group addresses in *LogicMachine* → *Objects* list



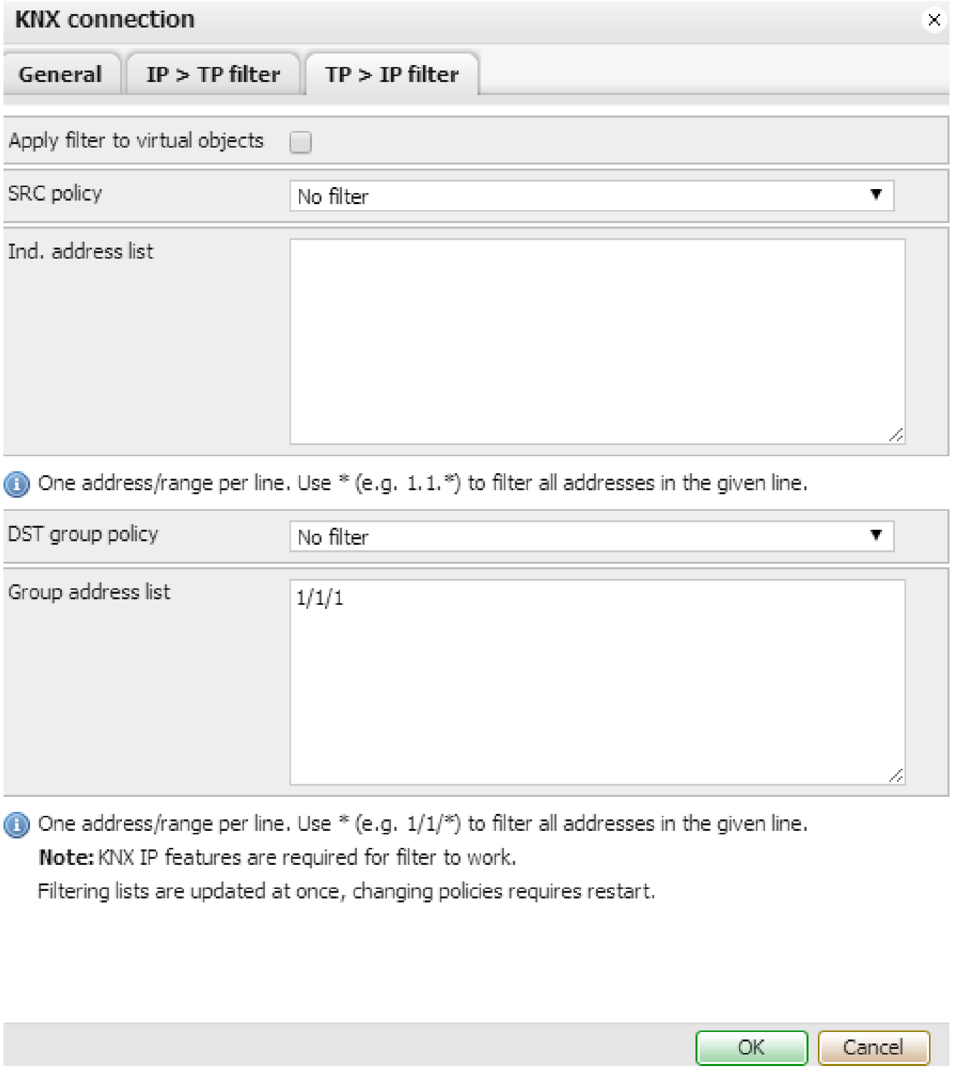
Name or group address:

Data type:

Group address ▲	Object name	IP > TP filter	TP > IP filter	Event script	Data type	Current value	Log	Export
1/1/1	Digital output 1	☑	☑		01.001 switch	on	□	□
1/1/2	Digital output 16	☑	□		01. 1 bit (boolean)	1	□	□
1/1/3	Digital output 2	□	□		01. 1 bit (boolean)	1	□	□
1/1/4	Digital output 3	□	□		01.001 switch	on	□	□

TP > IP filter

Filtering table for telegrams going from KNX TP1 to IP network is located in this submenu.



Apply filter to virtual objects □

SRC policy: No filter ▼

Ind. address list

One address/range per line. Use * (e.g. 1.1.*) to filter all addresses in the given line.

DST group policy: No filter ▼

Group address list: 1/1/1

One address/range per line. Use * (e.g. 1/1/*) to filter all addresses in the given line.
Note: KNX IP features are required for filter to work.
Filtering lists are updated at once, changing policies requires restart.

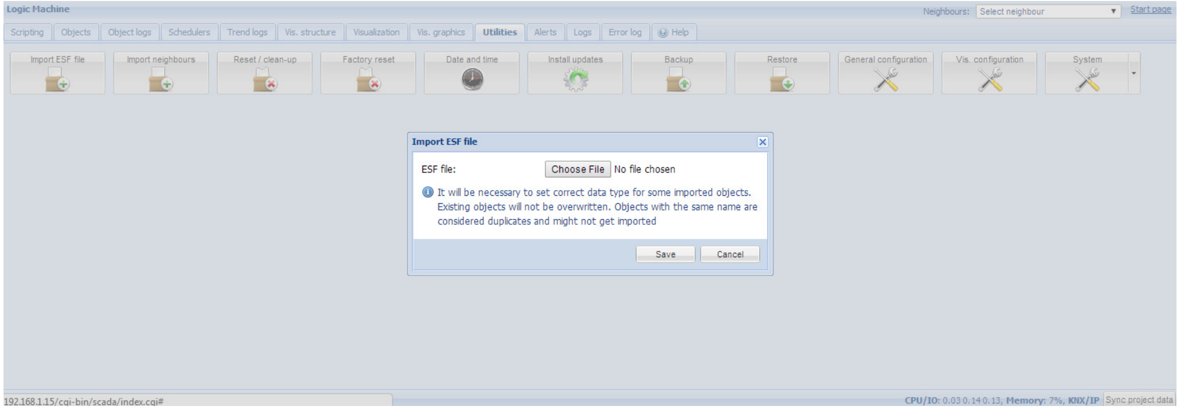
- ***Apply filter to virtual objects*** – either to apply filter policy to objects added in Objects tab as virtual objects without attraction to bus
- ***SRC policy*** *[No filter / Accept selected individual addresses / Drop selected individual addresses]*– policy to apply to the list of source individual addresses
- ***Ind. address list*** – list of individual addresses. One address/range per line. Use * (e.g. 1.1.*) to filter all addresses in the given line.
- ***DST group policy*** *[No filter / Accept selected group addresses / Drop selected group addresses]*– policy to apply to the list of destination group addresses
- ***Group address list*** – list of group addresses. One address/range per line. Use * (e.g. 1/1/*) to filter all addresses in the given line.

Note! *KNX IP features* should be on for filter to work. Filtering lists are updated at once, changing policies requires restart.

Quick guide - create visualization for iPad/PC

1. Import objects

Fastest way is to import *.ESF file from ETS in *Logic Machine →Utilities → Import ESF file*.



Import ESF file

ESF file: Choose File No file chosen

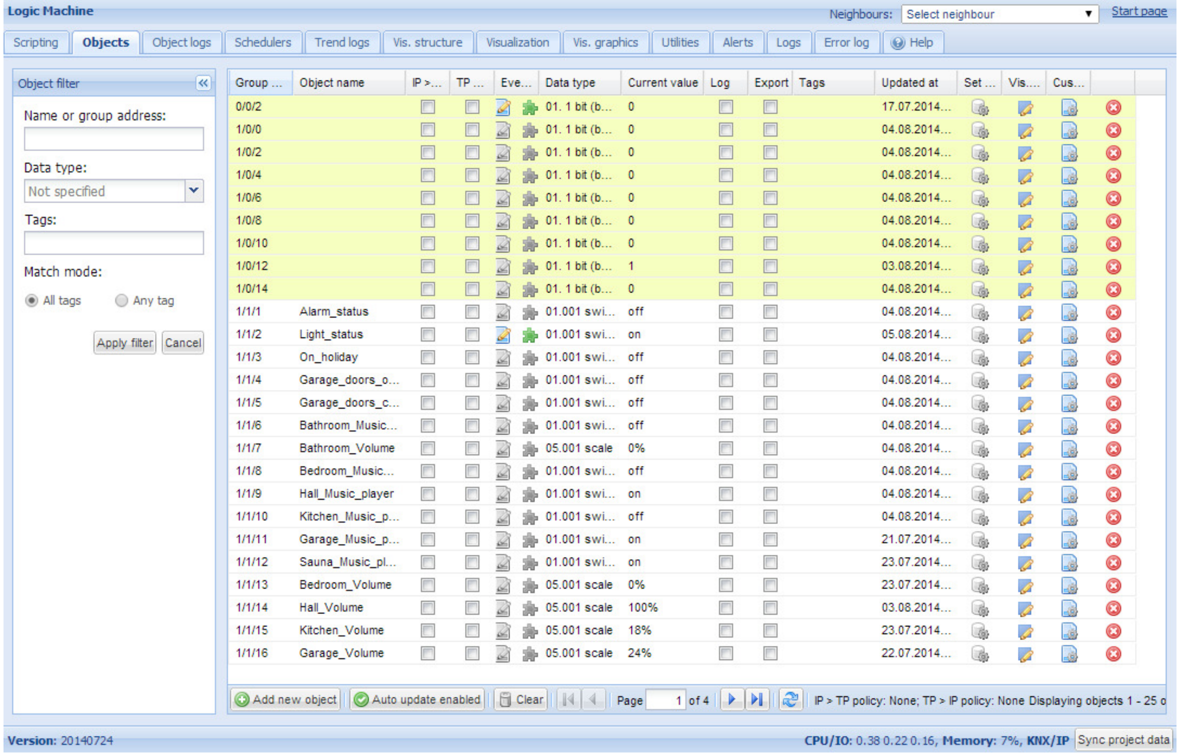
It will be necessary to set correct data type for some imported objects. Existing objects will not be overwritten. Objects with the same name are considered duplicates and might not get imported

Save Cancel

192.168.1.15/cgi-bin/scada/index.cgi#

CPU/IO: 0.03 0.14 0.13, Memory: 7%, KNX/IP Sync project data

Or connect LM to the bus and it will detect objects automatically (in yellow) in *Objects* tab once they are activated. Objects can be added manually as well.



Object filter

Name or group address:

Data type: Not specified

Tags:

Match mode: ● All tags ○ Any tag

Group ...	Object name	IP >...	TP ...	Eve...	Data type	Current value	Log	Export	Tags	Updated at	Set ...	Vis....	Cus...
0/0/2		□	□		01. 1 bit (b...	0	□	□		17.07.2014...			
1/0/0		□	□		01. 1 bit (b...	0	□	□		04.08.2014...			
1/0/2		□	□		01. 1 bit (b...	0	□	□		04.08.2014...			
1/0/4		□	□		01. 1 bit (b...	0	□	□		04.08.2014...			
1/0/6		□	□		01. 1 bit (b...	0	□	□		04.08.2014...			
1/0/8		□	□		01. 1 bit (b...	0	□	□		04.08.2014...			
1/0/10		□	□		01. 1 bit (b...	0	□	□		04.08.2014...			
1/0/12		□	□		01. 1 bit (b...	1	□	□		03.08.2014...			
1/0/14		□	□		01. 1 bit (b...	0	□	□		04.08.2014...			
1/1/1	Alarm_status	□	□		01.001 swi...	off	□	□		04.08.2014...			
1/1/2	Light_status	□	□		01.001 swi...	on	□	□		05.08.2014...			
1/1/3	On_holiday	□	□		01.001 swi...	off	□	□		04.08.2014...			
1/1/4	Garage_doors_o...	□	□		01.001 swi...	off	□	□		04.08.2014...			
1/1/5	Garage_doors_c...	□	□		01.001 swi...	off	□	□		04.08.2014...			
1/1/6	Bathroom_Music...	□	□		01.001 swi...	off	□	□		04.08.2014...			
1/1/7	Bathroom_Volume	□	□		05.001 scale	0%	□	□		04.08.2014...			
1/1/8	Bedroom_Music...	□	□		01.001 swi...	off	□	□		04.08.2014...			
1/1/9	Hall_Music_player	□	□		01.001 swi...	on	□	□		04.08.2014...			
1/1/10	Kitchen_Music_p...	□	□		01.001 swi...	off	□	□		04.08.2014...			
1/1/11	Garage_Music_p...	□	□		01.001 swi...	on	□	□		21.07.2014...			
1/1/12	Sauna_Music_pl...	□	□		01.001 swi...	on	□	□		23.07.2014...			
1/1/13	Bedroom_Volume	□	□		05.001 scale	0%	□	□		23.07.2014...			
1/1/14	Hall_Volume	□	□		05.001 scale	100%	□	□		03.08.2014...			
1/1/15	Kitchen_Volume	□	□		05.001 scale	18%	□	□		23.07.2014...			
1/1/16	Garage_Volume	□	□		05.001 scale	24%	□	□		22.07.2014...			

Page 1 of 4 IP > TP policy: None; TP > IP policy: None Displaying objects 1 - 25 o

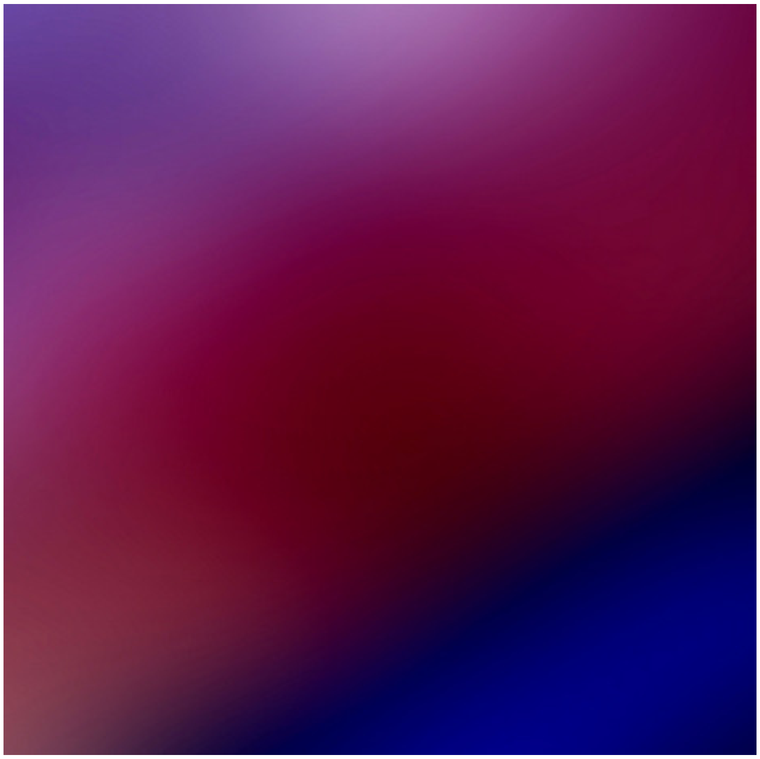
Version: 20140724

CPU/IO: 0.38 0.22 0.16, Memory: 7%, KNX/IP Sync project data

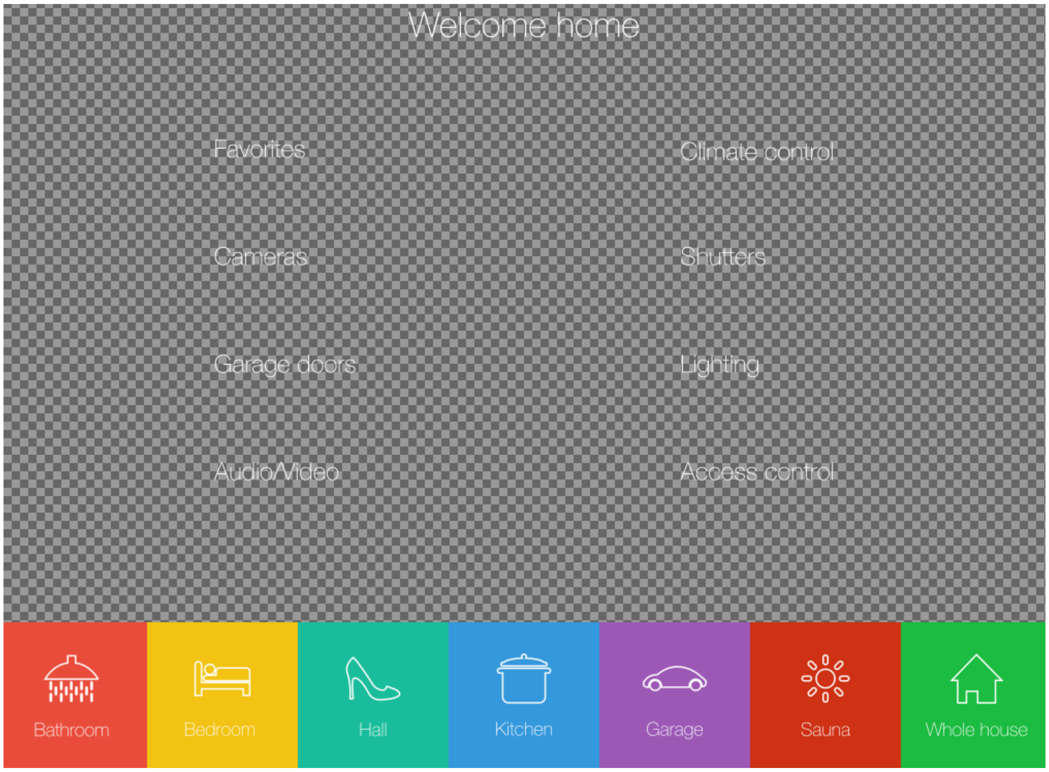
2. Prepare graphics

Either in Adobe Illustrator or any ready images can be used. In this example we use professionally created designs in Illustrator in SVG form (so we can do scaling depending of the screen size and not losing the quality)

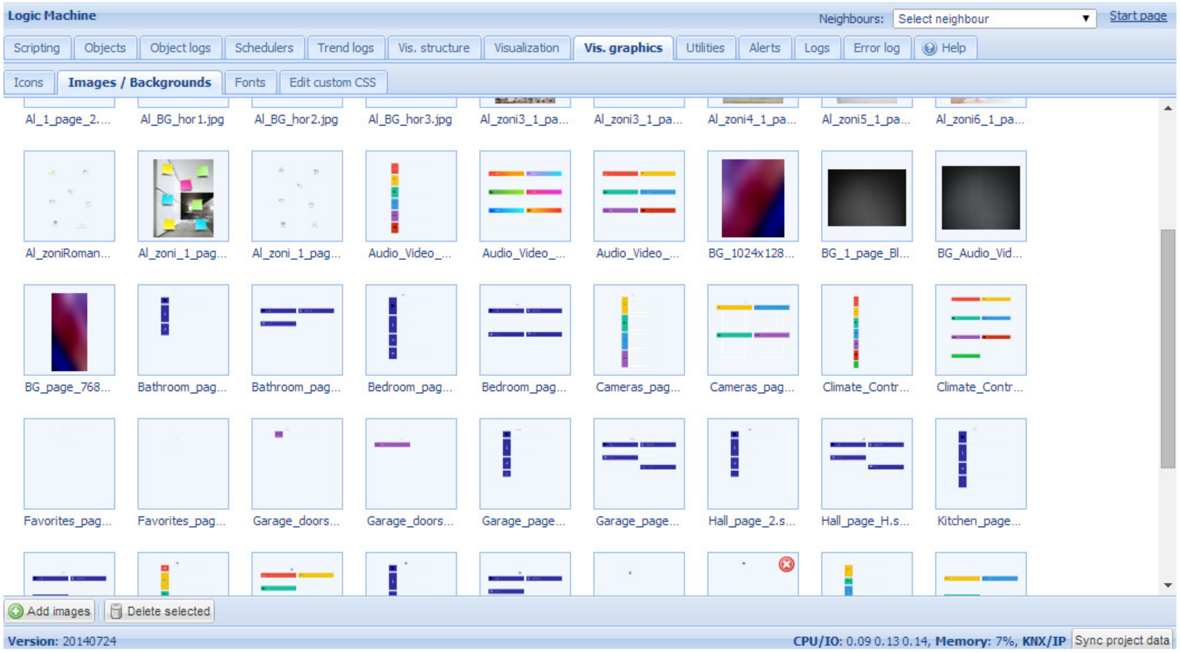
a) basic background which can be changed by necessity



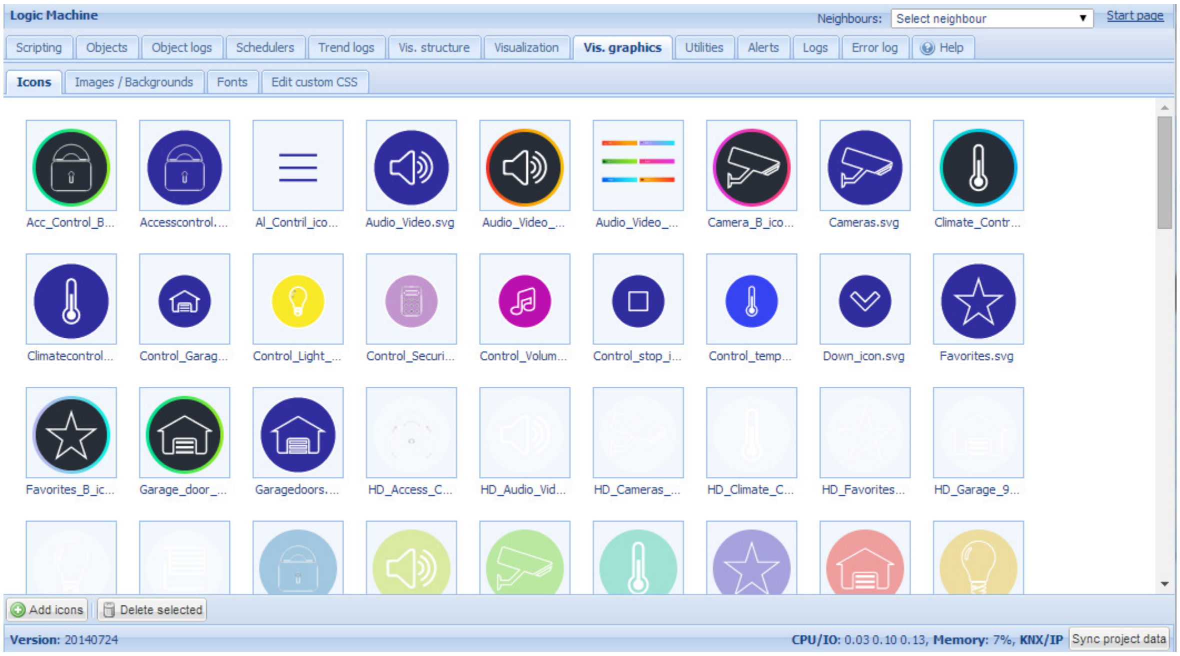
b) foreground which will stay unchanged



Add both files in *Logic Machine* → *Vis. Graphics* → *Images/Backgrounds*

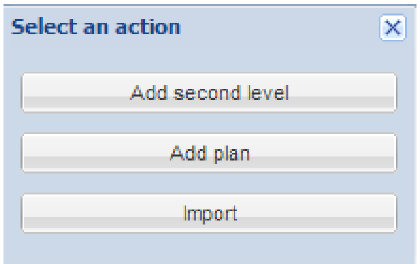


Prepare set of icons (preferably in SVG form) and add them in *Logic Machine* → *Vis. Graphics* → *Icons*. Or you can use icons predefined in LogicMachine by default.

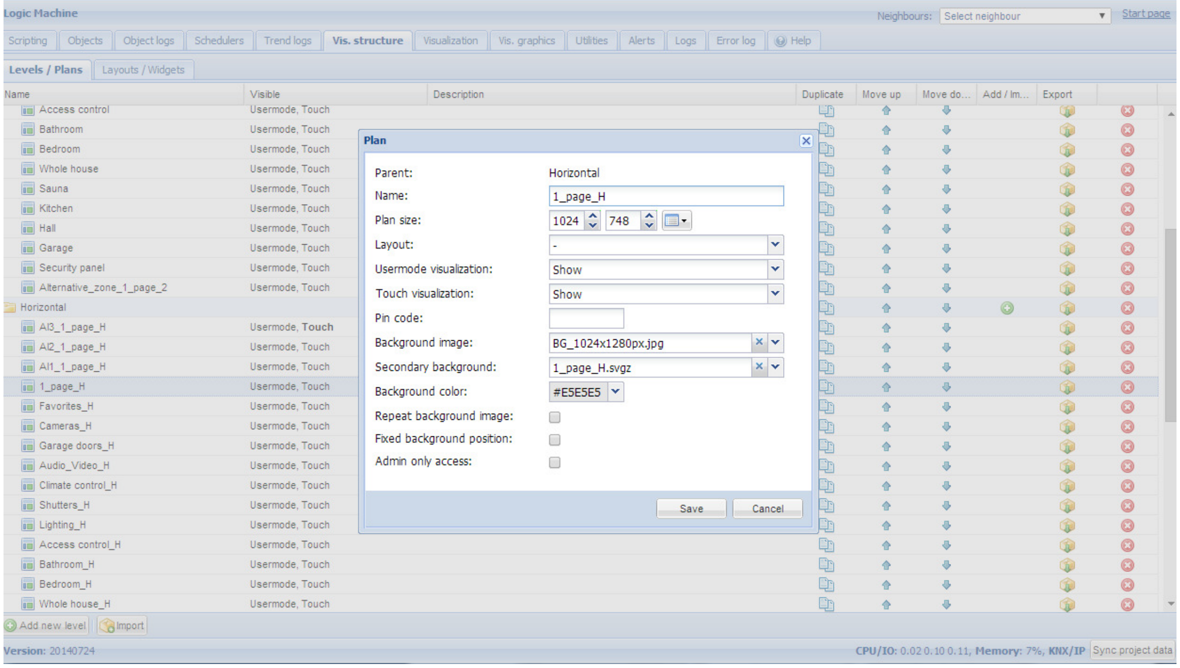


Create “floor” structure and add objects to the map

In *Logic Machine →Vis.structure* menu the structure of the visualization is defined and visualization backgrounds are uploaded. Use icon to add ⊕floor.



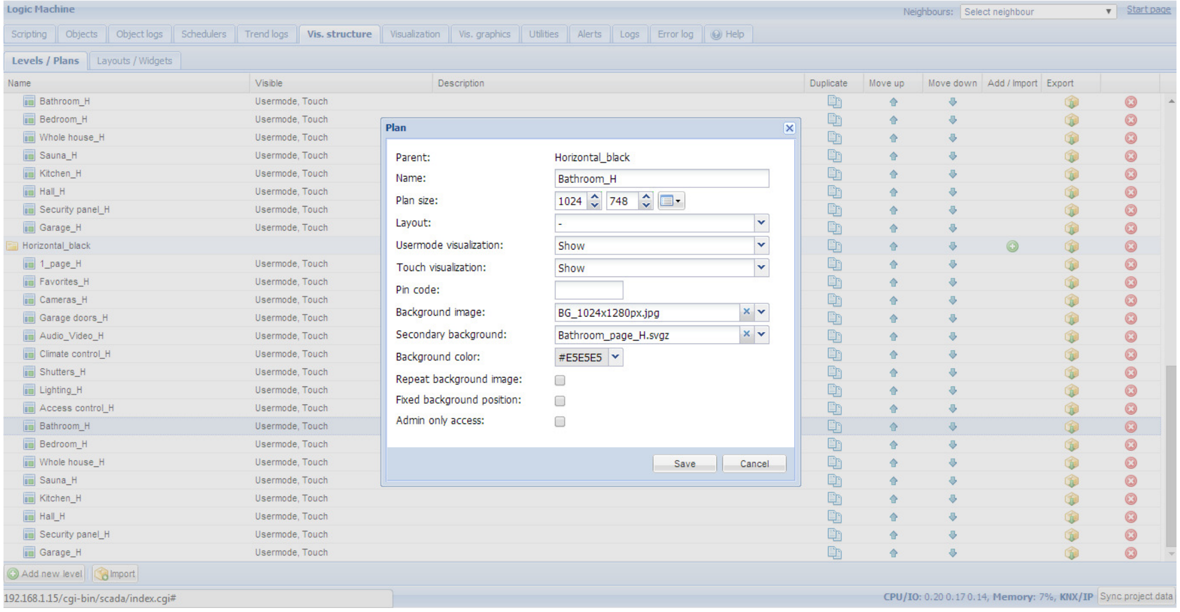
In this example we will create a new floor named “1_page_H” and “Bathroom_H”. First Floor will be a dashboard with link to other rooms and functions. Choose screen resolution for which you are creating this visualization, choose first and second background images from the ones added before.



Name	Visible	Description
Access control	Usermode, Touch	
Bathroom	Usermode, Touch	
Bedroom	Usermode, Touch	
Whole house	Usermode, Touch	
Sauna	Usermode, Touch	
Kitchen	Usermode, Touch	
Hall	Usermode, Touch	
Garage	Usermode, Touch	
Security panel	Usermode, Touch	
Alternative_zone_1_page_2	Usermode, Touch	
Horizontal		
AI3_1_page_H	Usermode, Touch	
AI2_1_page_H	Usermode, Touch	
AI1_1_page_H	Usermode, Touch	
1_page_H	Usermode, Touch	
Favorites_H	Usermode, Touch	
Cameras_H	Usermode, Touch	
Garage doors_H	Usermode, Touch	
Audio_Video_H	Usermode, Touch	
Climate control_H	Usermode, Touch	
Shutters_H	Usermode, Touch	
Lighting_H	Usermode, Touch	
Access control_H	Usermode, Touch	
Bathroom_H	Usermode, Touch	
Bedroom_H	Usermode, Touch	
Whole house_H	Usermode, Touch	

Plan

Parent:	Horizontal
Name:	1_page_H
Plan size:	1024 748
Layout:	-
Usermode visualization:	Show
Touch visualization:	Show
Pin code:	
Background image:	BG_1024x1280px.jpg
Secondary background:	1_page_H.svgz
Background color:	#E5E5E5
Repeat background image:	□
Fixed background position:	□
Admin only access:	□



Name	Visible	Description	Duplicate	Move up	Move down	Add / Import	Export
Bathroom_H	Usermode, Touch						
Bedroom_H	Usermode, Touch						
Whole house_H	Usermode, Touch						
Sauna_H	Usermode, Touch						
Kitchen_H	Usermode, Touch						
Hall_H	Usermode, Touch						
Security panel_H	Usermode, Touch						
Garage_H	Usermode, Touch						
Horizontal_black							
1_page_H	Usermode, Touch						
Favorites_H	Usermode, Touch						
Cameras_H	Usermode, Touch						
Garage doors_H	Usermode, Touch						
Audio_Video_H	Usermode, Touch						
Climate control_H	Usermode, Touch						
Shutters_H	Usermode, Touch						
Lighting_H	Usermode, Touch						
Access control_H	Usermode, Touch						
Bathroom_H	Usermode, Touch						
Bedroom_H	Usermode, Touch						
Whole house_H	Usermode, Touch						
Sauna_H	Usermode, Touch						
Kitchen_H	Usermode, Touch						
Hall_H	Usermode, Touch						
Security panel_H	Usermode, Touch						
Garage_H	Usermode, Touch						

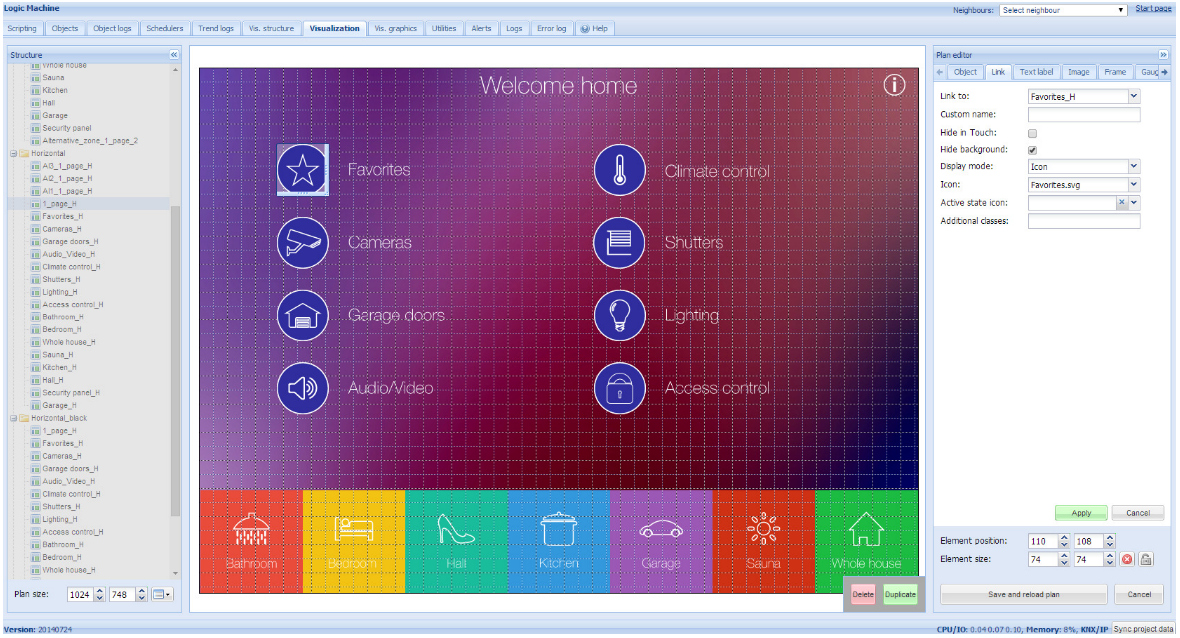
Plan

Parent: Horizontal_black
Name: Bathroom_H
Plan size: 1024 748
Layout: -
Usermode visualization: Show
Touch visualization: Show
Pin code:
Background image: BG_1024x1280px.jpg
Secondary background: Bathroom_page_H.svgz
Background color: #E5E5E5
Repeat background image: □
Fixed background position: □
Admin only access: □

Save Cancel

Add objects to newly created visualization map

After the building and floor structure is defined, it is visualized in *Visualization* tab. Controlled and monitored objects can be added and managed in this section. Both side bars can be minimized by pressing on left/right arrow icon making the map more visible especially on small displays.



Plan size: 1024 748

Link to: Favorites_H
Custom name:
Hide in Touch: □
Hide background: ☑
Display mode: Icon
Icon: Favorites.svg
Active state icon:
Additional classes:

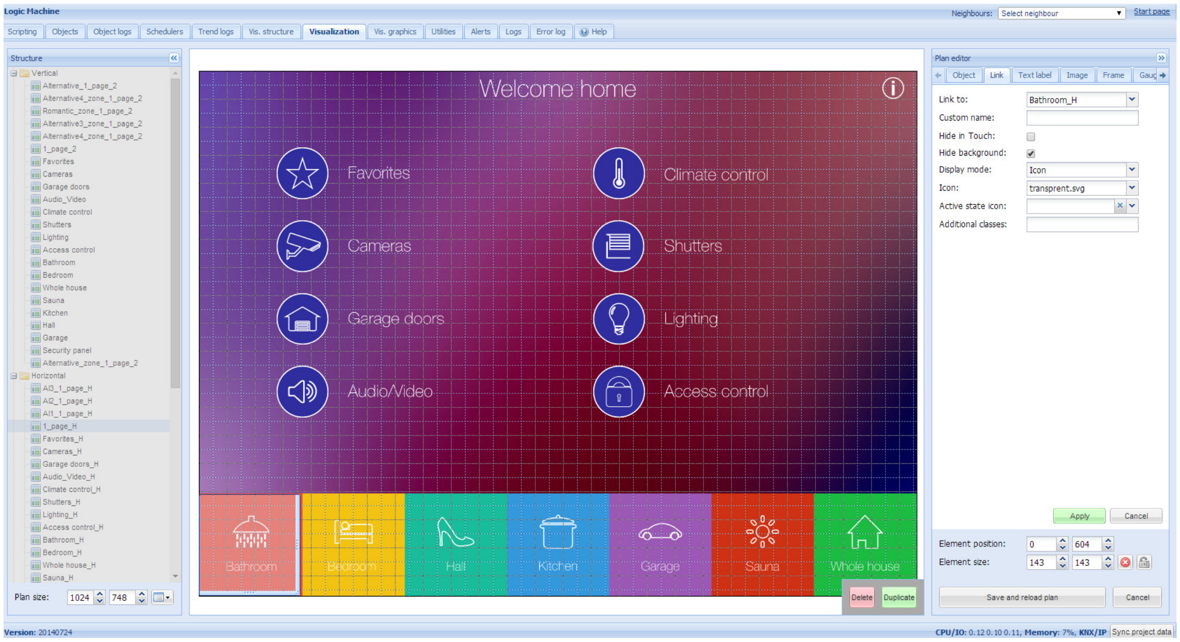
Apply Cancel

Element position: 110 108
Element size: 74 74

Save and reload plan Cancel

Objects can be added to the map by clicking on *Unlock current floor plan for editing* button. In this example we are creating first page of visualization which will link to other Floors with specific object control. Add link by clicking on Link tab, choosing specific icon, scale it and place in desired location.

This example’s secondary background already contains icons on it, so what is needed, is to add transparent image in *Vis.graphics* and add this image on top of every icon.

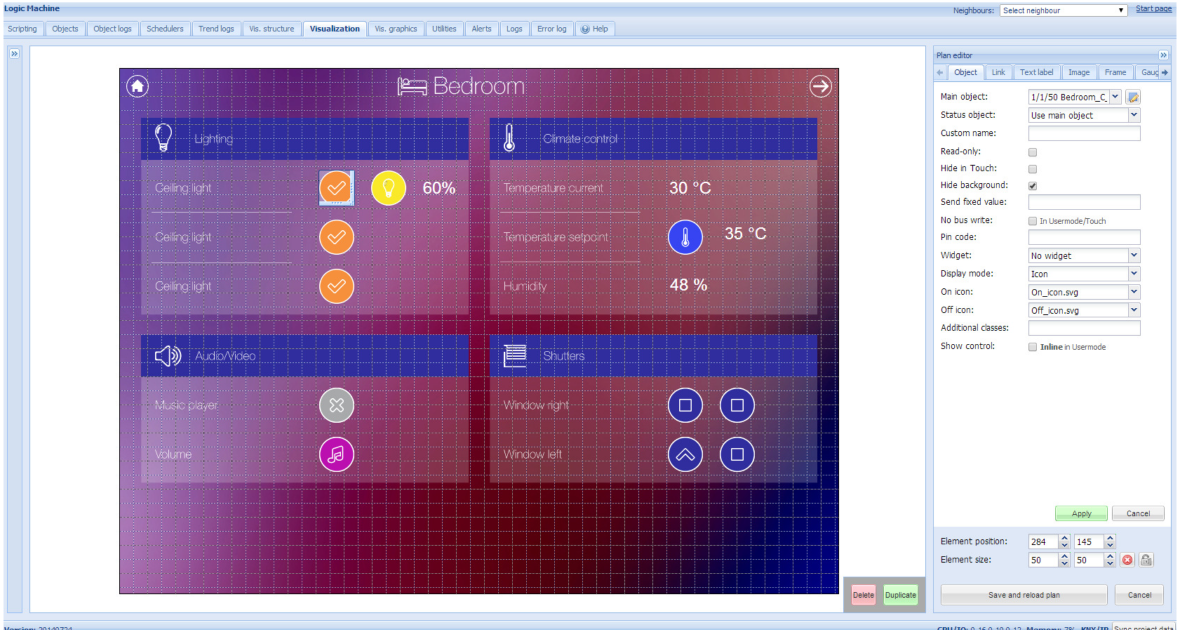


Link to: Bathroom_H
Custom name:
Hide in Touch: □
Hide background: ☑
Display mode: Icon
Icon: transprent.svg
Active state icon:
Additional classes:

Element position: 0 604
Element size: 143 143

Plan size: 1024 748

When all links are defined, press *Save and reload floor plan* button.
In same way fill the Bedroom plan with object parameters in Object tab.

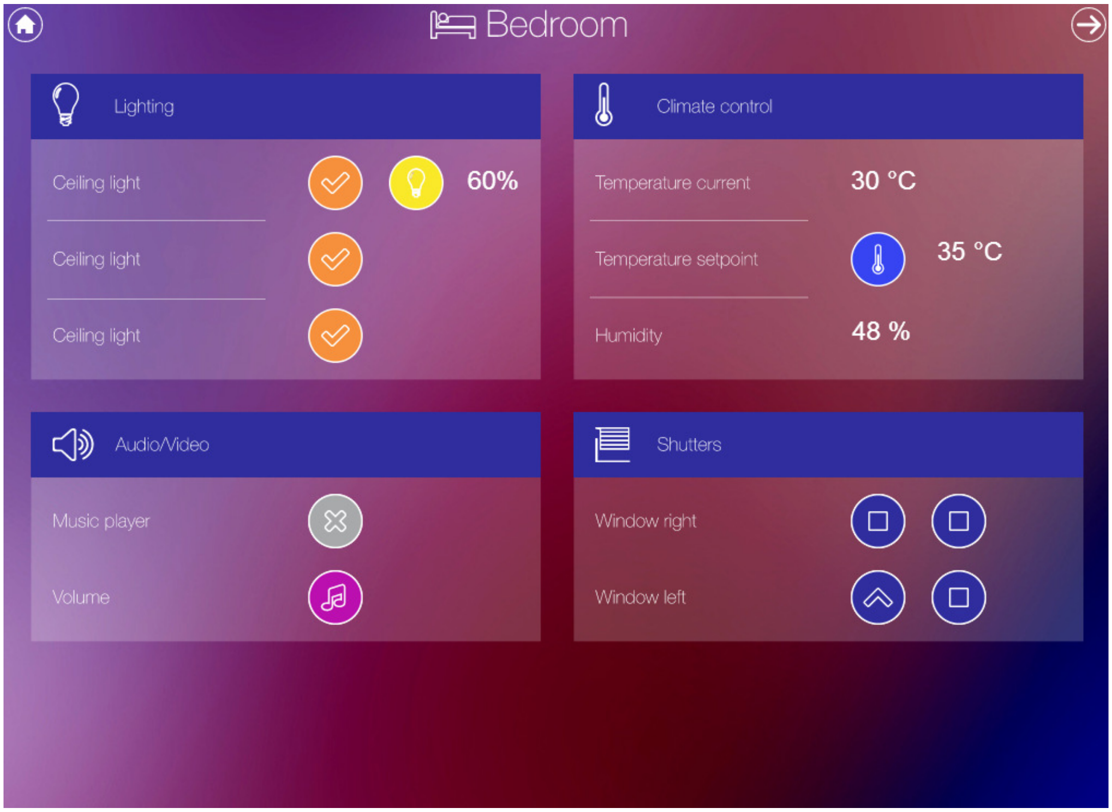


Main object: 1/1/50 Bedroom_C
Status object: Use main object
Custom name:
Read-only: □
Hide in Touch: □
Hide background: ☑
Send fixed value:
No bus write: □ In Usermode/Touch
Pin code:
Widget: No widget
Display mode: Icon
On icon: On_icon.svg
Off icon: Off_icon.svg
Additional classes:
Show control: □ **Inline** in Usermode

Element position: 284 145
Element size: 50 50

Launching visualization on touch device (iPad in this case)

- Make sure your iPad is connected wirelessly to the Logic Machine
- In the browser enter Logic Machine's IP (default 192.168.0.10).
- Click on the User *mode visualization*
- Save the application as permanent/shortcut in your iPad



Bedroom

Lighting

Ceiling light 60%

Ceiling light

Ceiling light

Climate control

Temperature current 30 °C

Temperature setpoint 35 °C

Humidity 48 %

Audio/Video

Music player

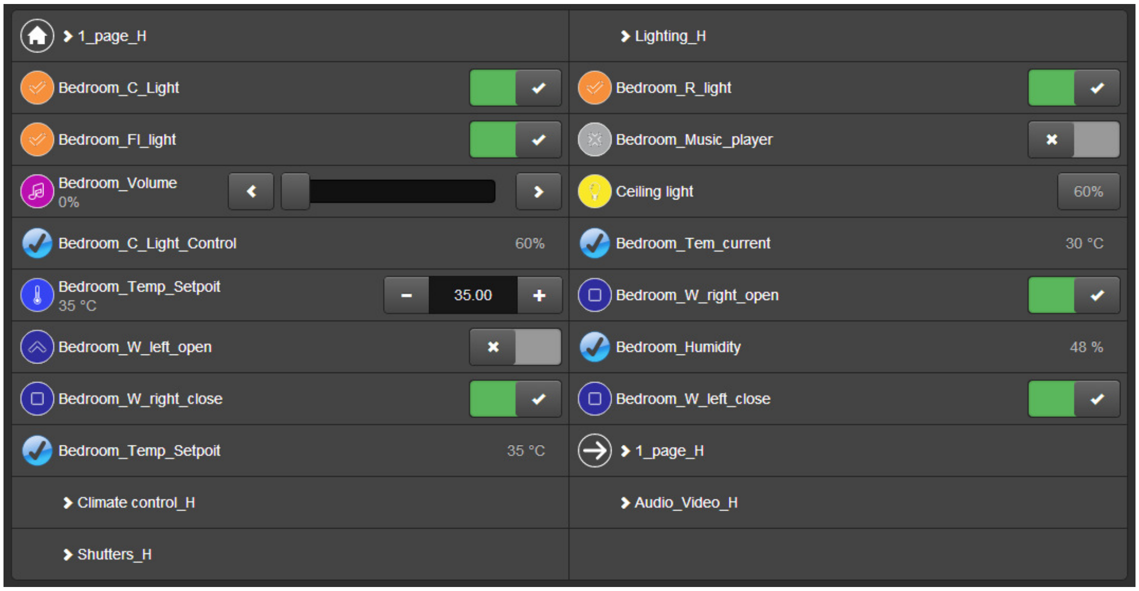
Volume

Shutters

Window right

Window left

Touch visualization is also automatically created with list of Floor objects.



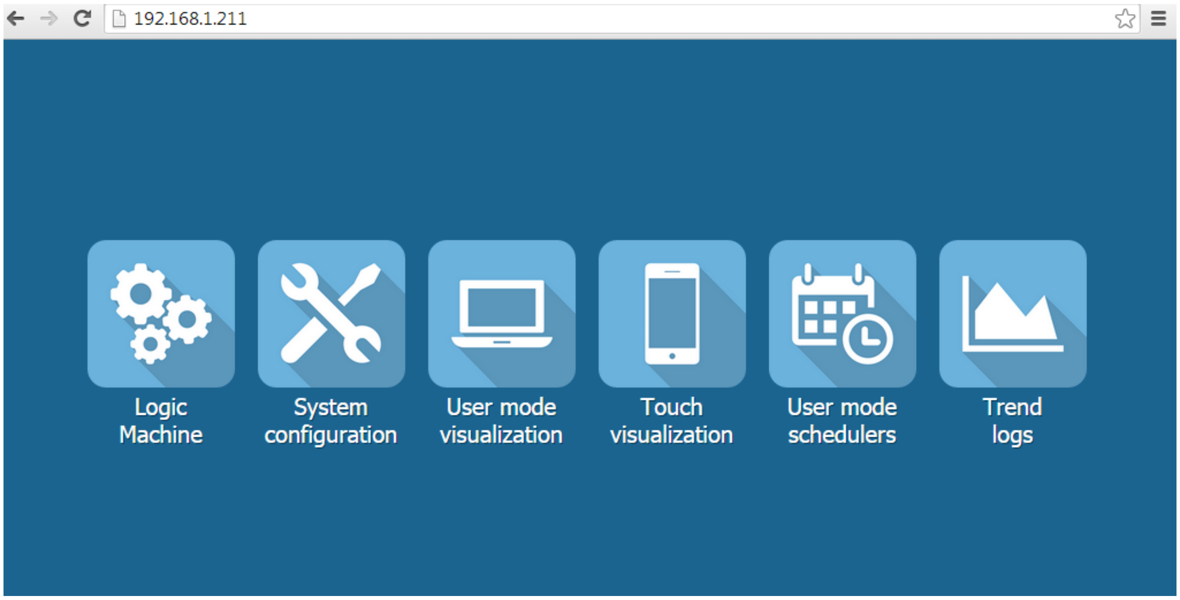
1_page_H	Lighting_H
Bedroom_C_Light ✓	Bedroom_R_light ✓
Bedroom_Fl_light ✓	Bedroom_Music_player ✖
Bedroom_Volume 0%	Ceiling light 60%
Bedroom_C_Light_Control 60%	Bedroom_Tem_current 30 °C
Bedroom_Temp_Setpoit 35 °C − 35.00 +	Bedroom_W_right_open ✓
Bedroom_W_left_open ✖	Bedroom_Humidity 48 %
Bedroom_W_right_close ✓	Bedroom_W_left_close ✓
Bedroom_Temp_Setpoit 35 °C	1_page_H
Climate control_H	Audio_Video_H
Shutters_H	

Graphical User Interface Login

KNX/EIB LogicMachine has IP address 192.168.0.10 set by default to LAN interface. Use this address as www address in the browser's address field.

Note! Make sure that the PC connecting to the LogicMachine has IP set from the same subnet.

After successful login a default page appears.



- ***Logic Machine*** – visualization creator, scripts, object relations, alerts, KNX objects and KNX objects, designing building view and visualization maps

- ***Network configuration*** – IP and KNXnet/IP specific configuration

- ***User mode visualization*** – defined visualization maps with objects

- ***Touch visualization*** – Visualization system for iPhone/iPod/iPad/Android touch screen devices

- ***User mode schedulers*** – User defined schedulers

- ***Trend logs*** – Trends for data logs

1. LogicMachine configuration

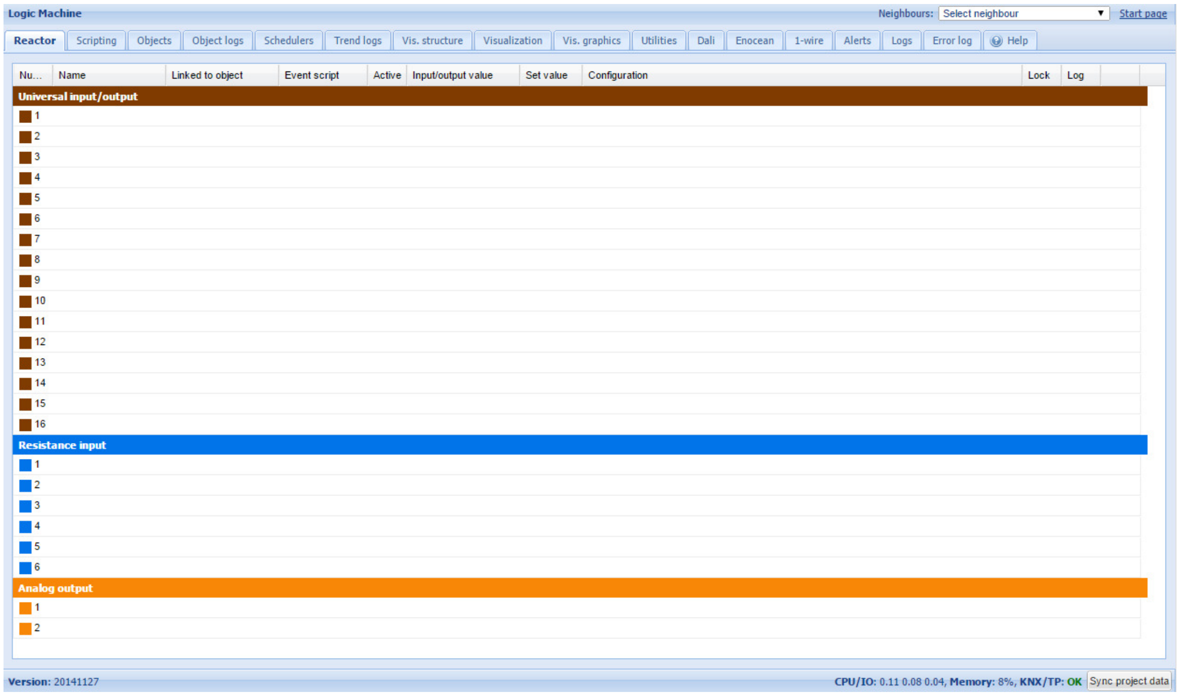
Login	**Password**
admin	admin

This is a home directory for LogicMachine configuration management. The main menu consists of the following menus:

- ***Reactor*** – LM3 Reactor specific IO settings
- ***Scripting*** – scripting repository management
- ***Objects***– KNX bus object management
- ***Object logs***– KNX bus object historical logs
- ***Schedulers***– administrator interface for user mode schedulers
- ***Trend logs*** – administrator interface for trend logs
- ***Vis.structure*** – visualization structure definition
- ***Visualization***– Visualization creation, control and monitoring
- ***Vis.graphics***– icon, background, font management
- ***Utilities*** – utilities including import from ETS, reset object DB, backup, update system installation
- ***BACnet* –** BACnet client with scanner
- ***Modbus*** – Modbus mapper
- ***Enocean*** – Enocean mapper
- ***1*-wire** – 1-wire object mapping to KNX
- ***Alerts*** – alert messages defined with *alert* function
- ***Logs*** – log messages defined with *log* function
- ***Error log*** – error messages in KNX bus
- ***Help*** – documentation for scripting syntaxes

1.1. Reactor

Reactor IO configuration and mapping to KNX is done under *Reactor* tab. No additional software is needed to configure KNX mapping of IO ports.

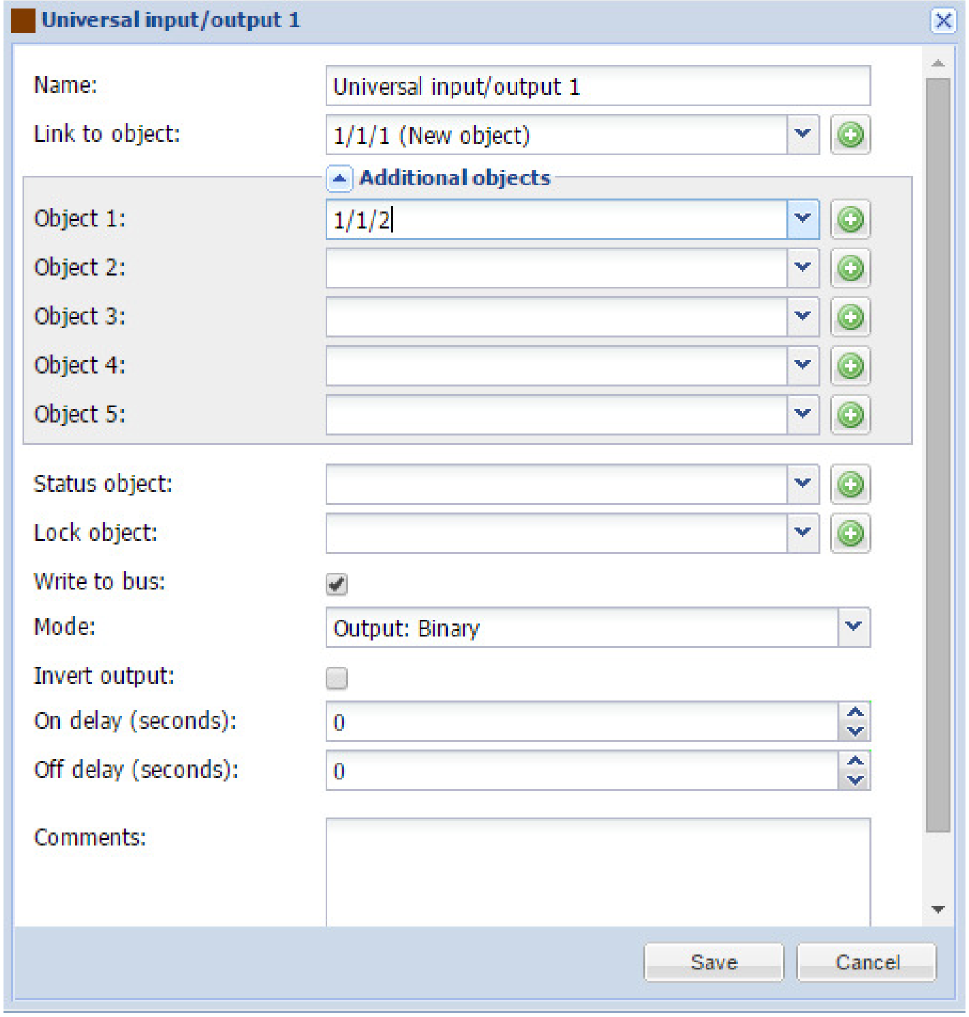


Nu...	Name	Linked to object	Event script	Active	Input/output value	Set value	Configuration	Lock	Log
Universal input/output									
🟫 1									
🟫 2									
🟫 3									
🟫 4									
🟫 5									
🟫 6									
🟫 7									
🟫 8									
🟫 9									
🟫 10									
🟫 11									
🟫 12									
🟫 13									
🟫 14									
🟫 15									
🟫 16									
Resistance input									
🟦 1									
🟦 2									
🟦 3									
🟦 4									
🟦 5									
🟦 6									
Analog output									
🟧 1									
🟧 2									

1.1.1. *Universal input/output*

You can set up Universal I/O port as binary output, voltage input, binary ON/OFF input, binary SHORT/LONG input, Step dimmer 1 byte, Relative dimmer 4 bit.

Output: Binary



Universal input/output 1

Name: Universal input/output 1

Link to object: 1/1/1 (New object)

Additional objects

Object 1: 1/1/2

Object 2:

Object 3:

Object 4:

Object 5:

Status object:

Lock object:

Write to bus: ☑

Mode: Output: Binary

Invert output: □

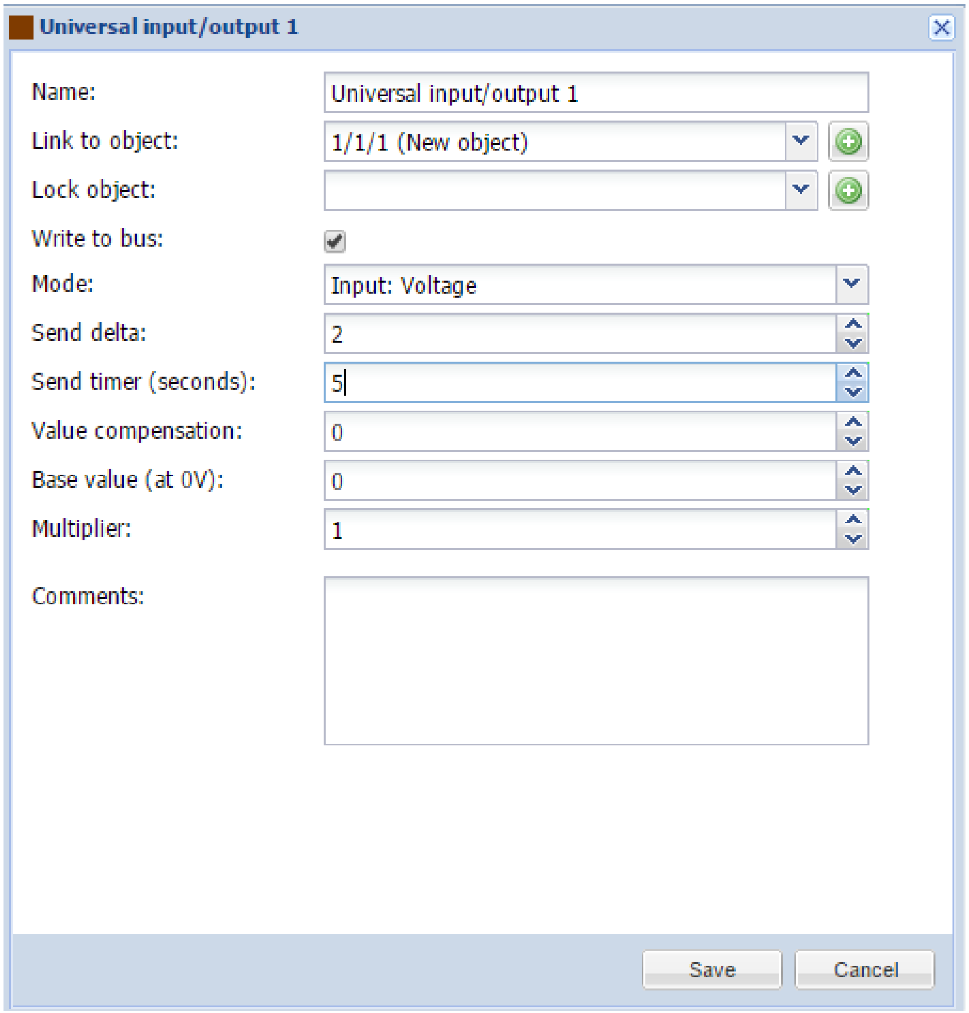
On delay (seconds): 0

Off delay (seconds): 0

Comments:

- ➢ ***Name*** – name of the port
- ➢ ***Link to object*** – group address to link the object with. By pressing the ⊕ button, the field will be automatically filled with the next free group address. You can add up to 6 objects to one output.
- ➢ ***Status object*** – status object group address.
- ➢ ***Lock object*** – lock object group address
- ➢ ***Write to bus*** – defines either to write or not to bus on value change
- ➢ ***Mode*** – port operating mode
- ➢ ***Invert output*** – defines either to invert the output value
- ➢ ***On delay (seconds)*** – delay in seconds when getting in On state
- ➢ ***Off delay (seconds)*** – delay in seconds when getting in Off state
- ➢ ***Comments*** – comment of the object

Input: Voltage



Universal input/output 1

Name: Universal input/output 1

Link to object: 1/1/1 (New object)

Lock object:

Write to bus: ☑

Mode: Input: Voltage

Send delta: 2

Send timer (seconds): 5

Value compensation: 0

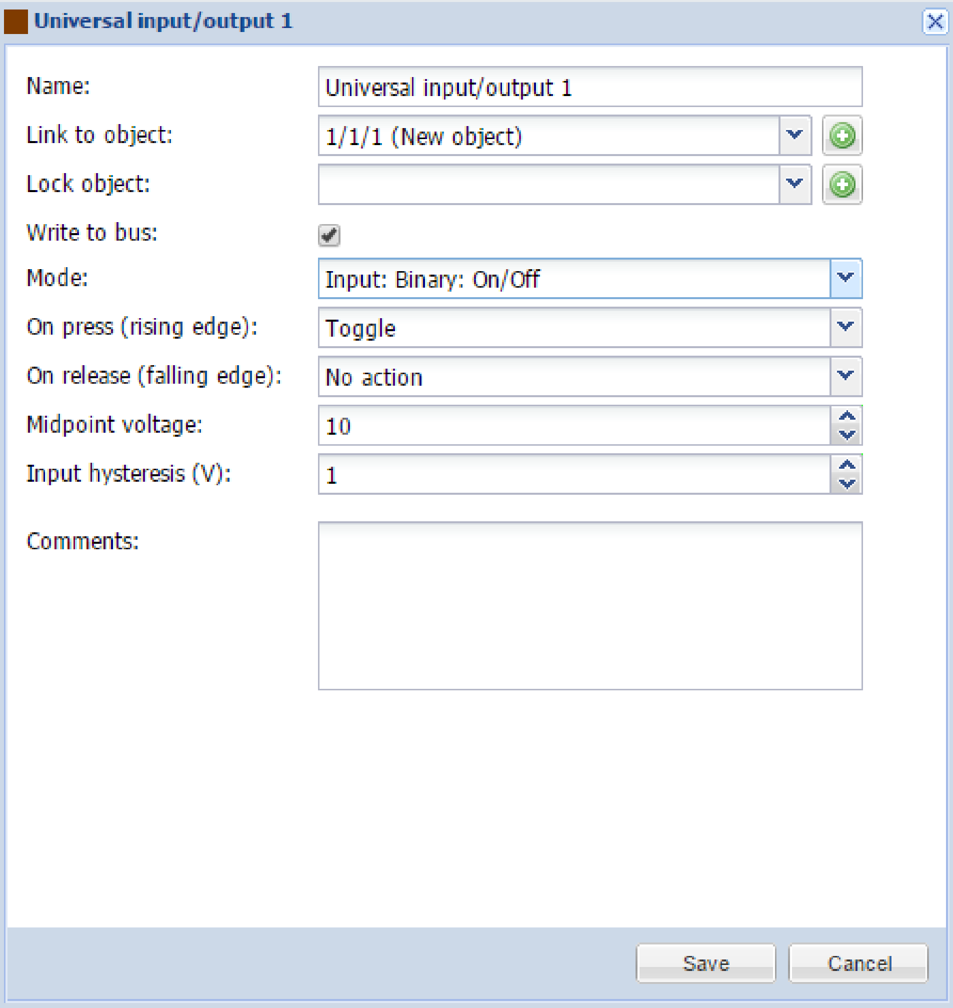
Base value (at 0V): 0

Multiplier: 1

Comments:

- ➢ ***Name*** – name of the port
- ➢ ***Link to object*** – group address to link the object with. By pressing the ⊕ button, the field will be automatically filled with the next free group address
- ➢ ***Lock object*** – lock object group address
- ➢ ***Write to bus*** – defines either to write or not to bus on value change
- ➢ ***Mode*** – port operating mode
- ➢ ***Send delta*** – send the data upon specific delta value has changed
- ➢ ***Send timer (seconds)*** – time interval to send the reading
- ➢ ***Value compensation*** – compensation of the value
- ➢ ***Base value (at 0V)*** – value at 0V
- ➢ ***Multiplier*** – multiplier for the received value

Input: Binary ON/OFF

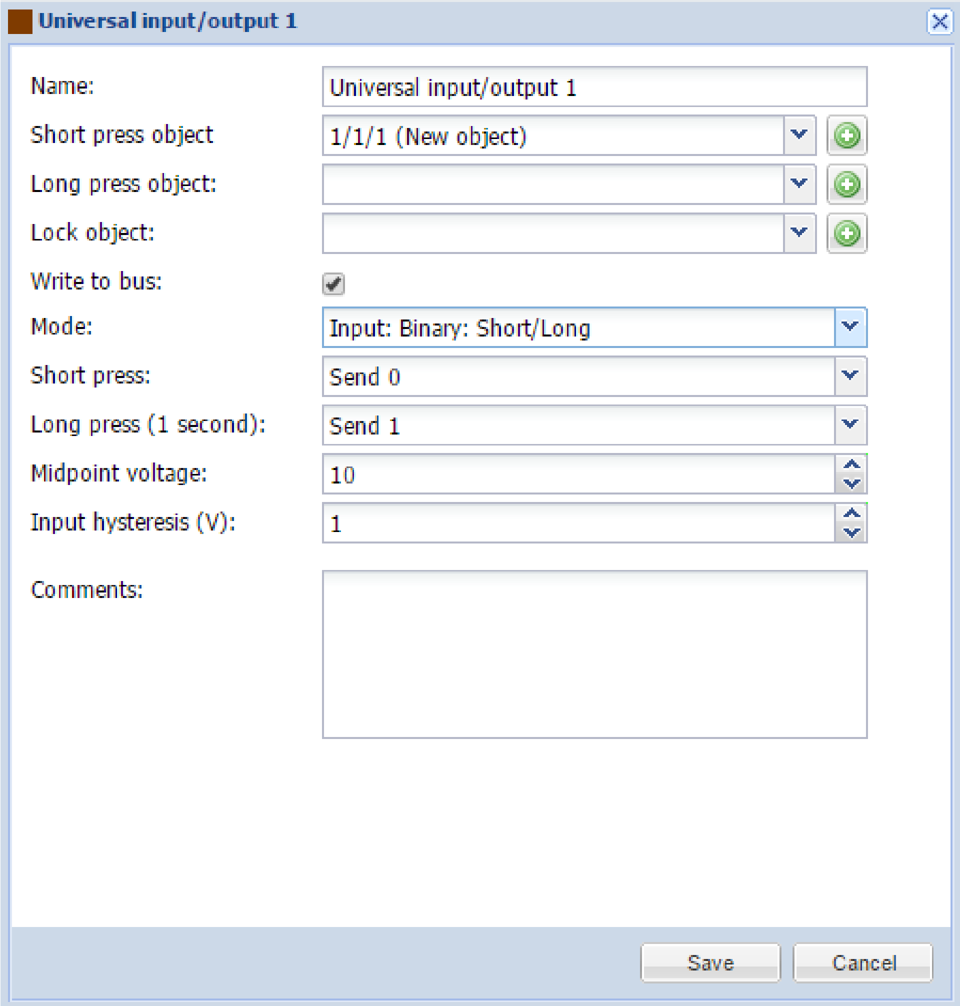


Universal input/output 1

Name:	Universal input/output 1
Link to object:	1/1/1 (New object)
Lock object:	
Write to bus:	☑
Mode:	Input: Binary: On/Off
On press (rising edge):	Toggle
On release (falling edge):	No action
Midpoint voltage:	10
Input hysteresis (V):	1
Comments:	

- ➢ ***Name*** – name of the port
- ➢ ***Link to object*** – group address to link the object with. By pressing the ⊕ button, the field will be automatically filled with the next free group address.
- ➢ ***Lock object*** – lock object group address
- ➢ ***Write to bus*** – defines either to write or not to bus on value change
- ➢ ***Mode*** – port operating mode
- ➢ ***On press (rising edge)[Send 0; Send 1; Toggle]*** – action on rising edge
- ➢ ***On release (falling edge) [Send 0; Send 1; Toggle]*** – action on falling edge
- ➢ ***Midpoint voltage*** – voltage midpoint to determine the On/Off state
- ➢ ***Input hysteresis (V)* –** If binary input is OFF, it will change to ON when voltage level is above MIDPOINT + HYSTERESIS. If binary input is ON, it will change to OFF when voltage level is below MIDPOINT - HYSTERESIS

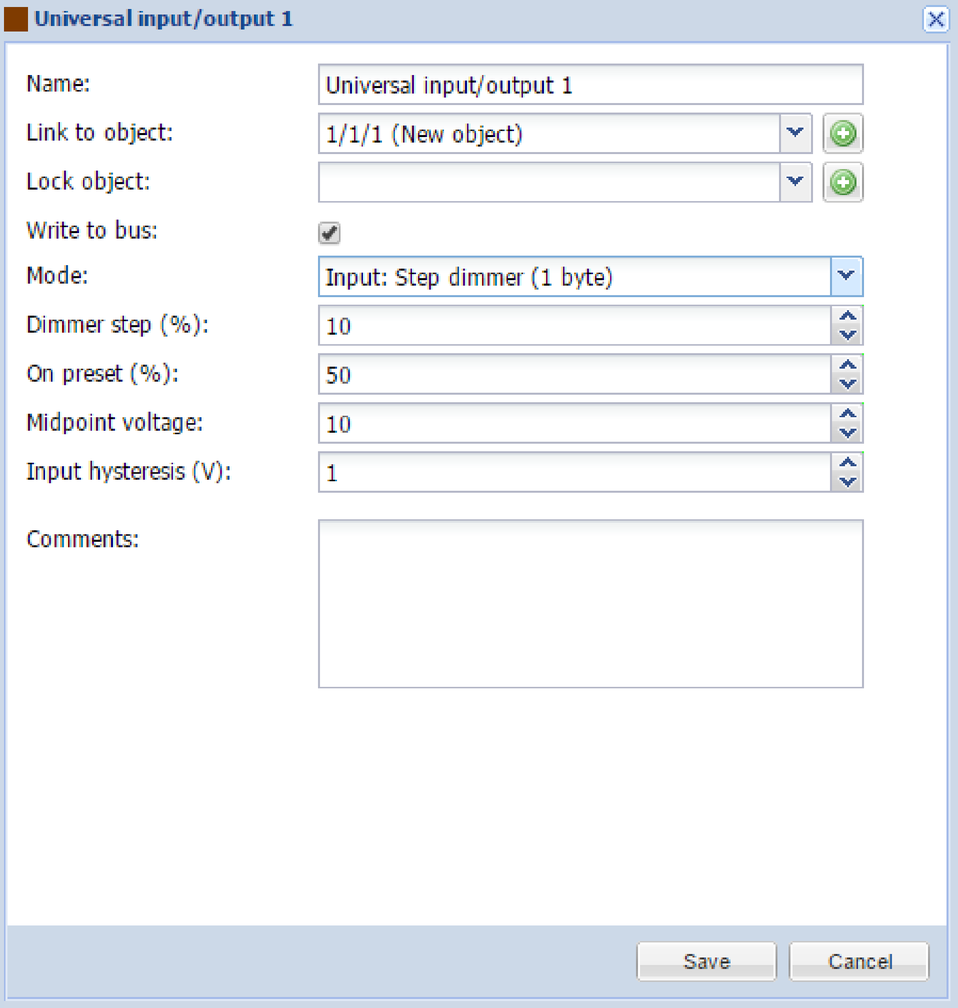
Input: Binary Short/Long



Name: Universal input/output 1
Short press object: 1/1/1 (New object)
Long press object:
Lock object:
Write to bus: ☑
Mode: Input: Binary: Short/Long
Short press: Send 0
Long press (1 second): Send 1
Midpoint voltage: 10
Input hysteresis (V): 1
Comments:

- *Name* – name of the port
- ***Short press object*** – group address to link the object with on short press. By pressing the ⊕ button, the field will be automatically filled with the next free group address.
- ***Long press object*** – group address to link the object with on long press. By pressing the ⊕ button, the field will be automatically filled with the next free group address.
- ***Lock object*** – lock object group address
- ***Write to bus*** – defines either to write or not to bus on value change
- ***Mode*** – port operating mode
- ***Short press [Send 0; Send 1; Toggle]*** – action on short press
- ***Long press (1 second) [Send 0; Send 1; Toggle]*** – action on long press
- ***Midpoint voltage*** – voltage midpoint to determine the Short/Long state
- ***Input hysteresis (V)* –** If binary input is OFF, it will change to ON when voltage level is above MIDPOINT + HYSTERESIS. If binary input is ON, it will change to OFF when voltage level is below MIDPOINT - HYSTERESIS

Input: Step dimmer (1 byte)

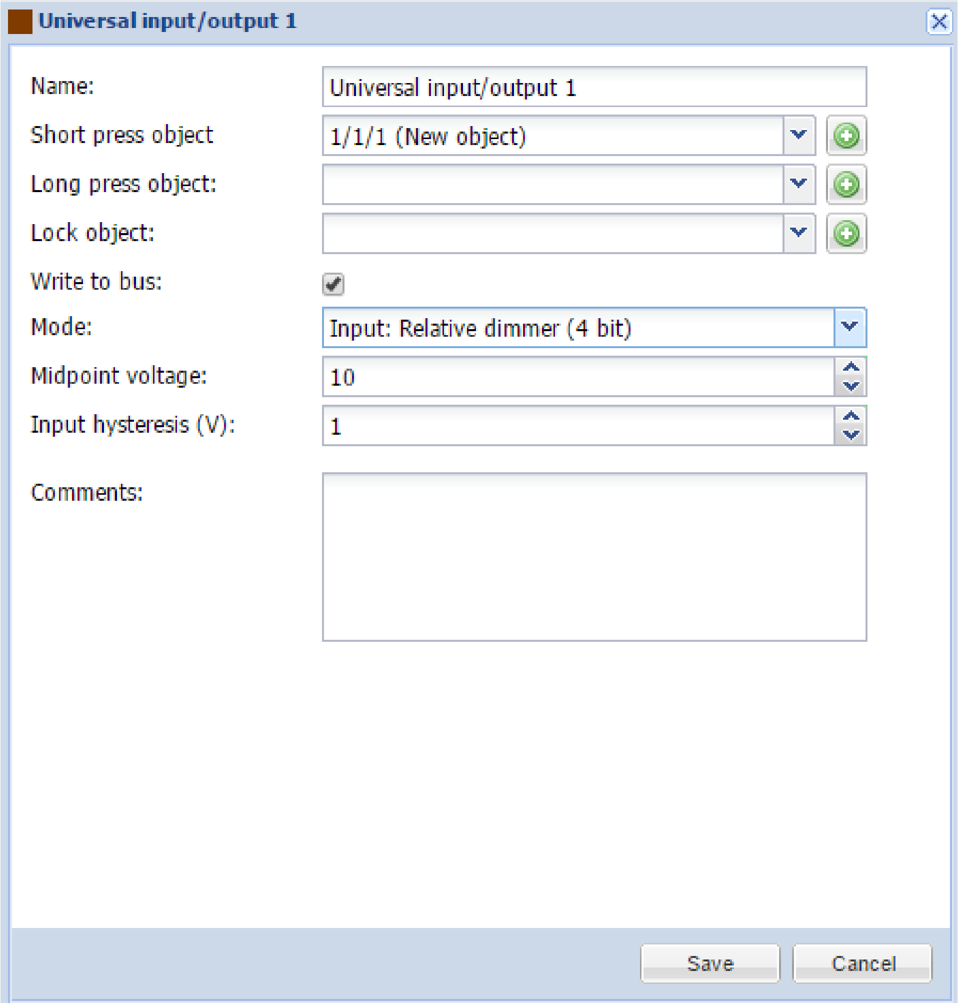


Universal input/output 1

Name:	Universal input/output 1
Link to object:	1/1/1 (New object)
Lock object:	
Write to bus:	☑
Mode:	Input: Step dimmer (1 byte)
Dimmer step (%):	10
On preset (%):	50
Midpoint voltage:	10
Input hysteresis (V):	1
Comments:	

- ➢ ***Name*** – name of the port
- ➢ ***Link to object*** – group address to link the object with. By pressing the ⊕ button, the field will be automatically filled with the next free group address.
- ➢ ***Lock object*** – lock object group address
- ➢ ***Write to bus*** – defines either to write or not to bus on value change
- ➢ ***Mode*** – port operating mode
- ➢ ***Dimmer step (%)***– value on which the brightness value will change by one step
- ➢ ***On preset (%)*** – brightness preset when getting in On state
- ➢ ***Midpoint voltage*** – voltage midpoint
- ➢ ***Input hysteresis (V)*** – If binary input is OFF, it will change to ON when voltage level is above MIDPOINT + HYSTERESIS. If binary input is ON, it will change to OFF when voltage level is below MIDPOINT – HYSTERESIS

Input: Relative dimmer (4 bit)

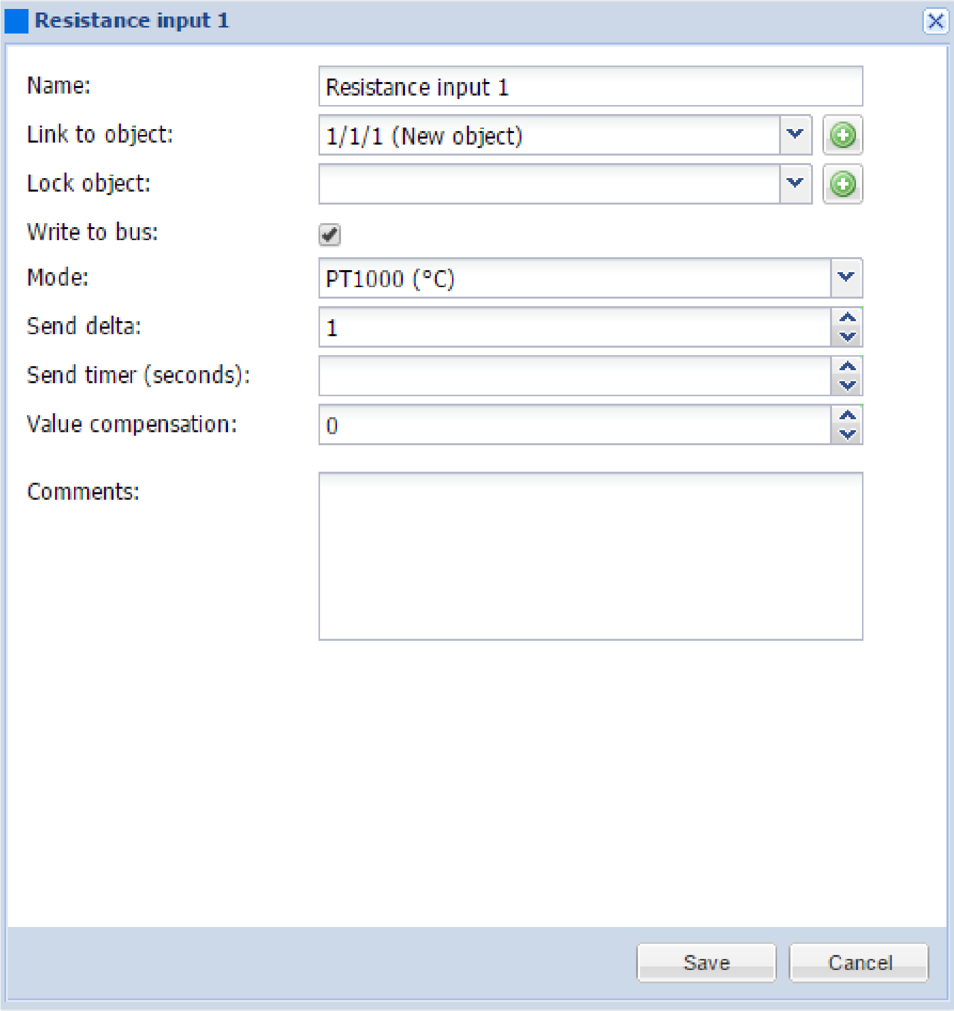

Universal input/output 1

Name:	Universal input/output 1
Short press object	1/1/1 (New object)
Long press object:	
Lock object:	
Write to bus:	☑
Mode:	Input: Relative dimmer (4 bit)
Midpoint voltage:	10
Input hysteresis (V):	1
Comments:	

- ***Name*** – name of the port
- ***Short press object*** – group address to link the object with on short press. By pressing the ⊕ button, the field will be automatically filled with the next free group address.
- ***Long press object –*** group address to link the object with on long press. By pressing the ⊕ button, the field will be automatically filled with the next free group address.
- ***Lock object*** – lock object group address
- ***Write to bus*** – defines either to write or not to bus on value change
- ***Mode*** – port operating mode
- ***Midpoint voltage*** – voltage midpoint
- ***Input hysteresis (V)*** – If binary input is OFF, it will change to ON when voltage level is above MIDPOINT + HYSTERESIS. If binary input is ON, it will change to OFF when voltage level is below MIDPOINT – HYSTERESIS

1.1.2. *Resistance input*

Resistive input can be either PT1000 temperature sensor, PT100 temperature sensor or show the data in raw form.



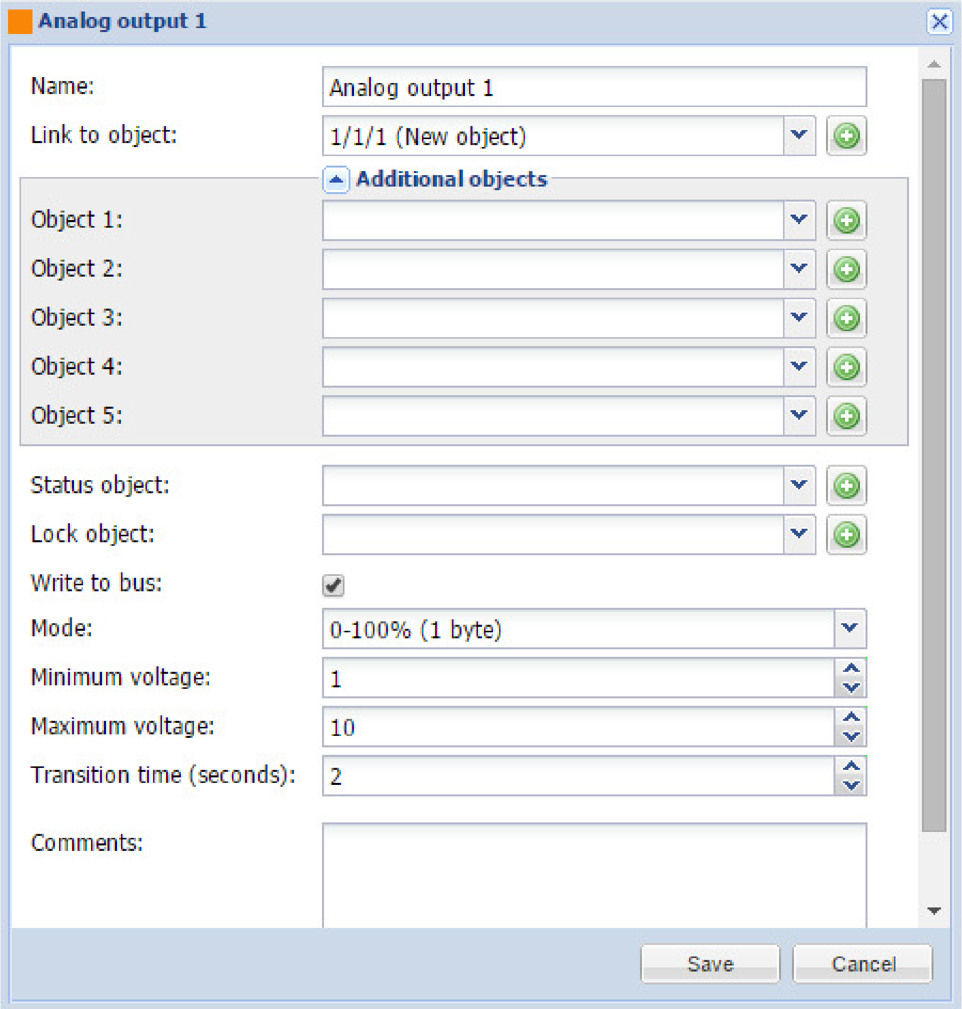
Resistance input 1

Name:	Resistance input 1
Link to object:	1/1/1 (New object)
Lock object:	
Write to bus:	☑
Mode:	PT1000 (°C)
Send delta:	1
Send timer (seconds):	
Value compensation:	0
Comments:	

- ➢ ***Name*** – name of the port
- ➢ ***Link to object*** – group address to link the object with. By pressing the ⊕ button, the field will be automatically filled with the next free group address.
- ➢ ***Lock object*** – lock object group address
- ➢ ***Write to bus*** – defines either to write or not to bus on value change
- ➢ ***Mode [PT1000; PT100, RAW]*** – type of input sensor
- ➢ ***Send delta*** – send the data upon specific delta value has changed
- ➢ ***Send timer (seconds)*** – time interval to send the reading
- ➢ ***Value compensation*** – compensation of the value
- ➢ ***Comments*** – comment of the object

1.1.3. Analog output

Analog output can be used either as 1byte (0-100%) or 2byte (voltage) output.



Analog output 1

Name: Analog output 1

Link to object: 1/1/1 (New object)

Additional objects

Object 1:

Object 2:

Object 3:

Object 4:

Object 5:

Status object:

Lock object:

Write to bus: ☑

Mode: 0-100% (1 byte)

Minimum voltage: 1

Maximum voltage: 10

Transition time (seconds): 2

Comments:

- ➢ ***Name*** – name of the port
- ➢ ***Link to object*** – group address to link the object with. By pressing the ⊕ button, the field will be automatically filled with the next free group address. You can add up to 6 group addresses to one output.
- ➢ ***Status object*** – status object group address.
- ➢ ***Lock object*** – lock object group address
- ➢ ***Write to bus*** – defines either to write or not to bus on value change
- ➢ ***Mode [0-100%(1byte); Voltage(2byte)]*** – output mode
- ➢ ***Minimum voltage*** – minimum voltage value
- ➢ ***Maximum voltage*** – maximum voltage value
- ➢ ***Transition time (seconds)*** – transition time between two values
- ➢ ***Comments*** – comment of the object

1.2. Scripting

Scripting menu allows adding and managing various scripts, depending on the type of the script. There are two ways to program logics – blocks and via Lua programming language. Most of the Lua language aspects are covered in the first edition of "Programming in Lua" which is freely available at http://lua.org/pil/

Note! *Data format — in most cases data is stored and transferred between LogicMachine parts using hex-encoded strings (2 bytes per 1 byte of data).*

There are six main types of scripts:

Event-based – scripts that are executed when a group event occurs on the bus. Usually used when nearly real-time response is required.

Resident– scripts that use polling to check for object state changes. Usually used for heating and ventilation when data is gathered from more than one group address.

Scheduled– scripts that run at the required time and day. Can be used for various security systems and presence simulations.

User libraries – user defined scripts to call from other scripts

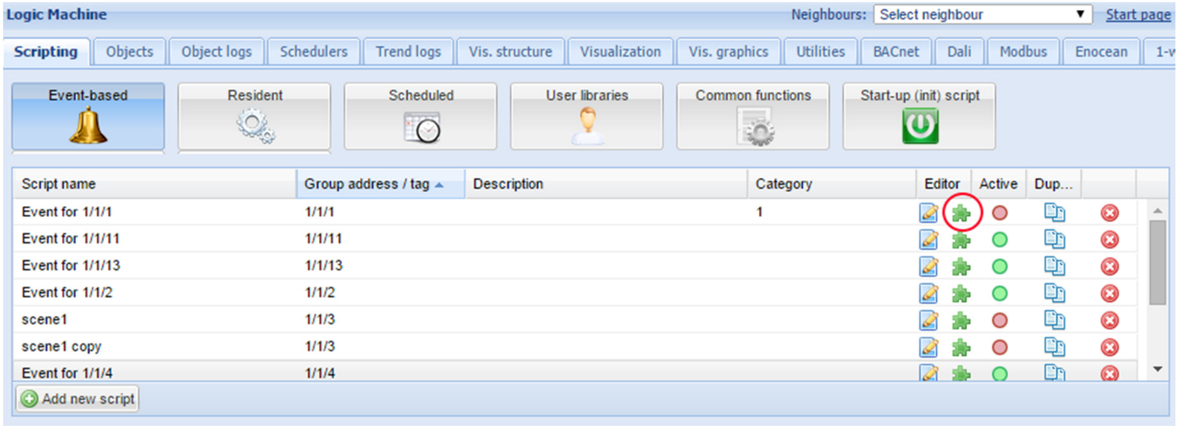
Common functions – common functions to call from other scripts

Start-up (init) script – initialization script that is run upon system starting.

1.2.1. Block programming

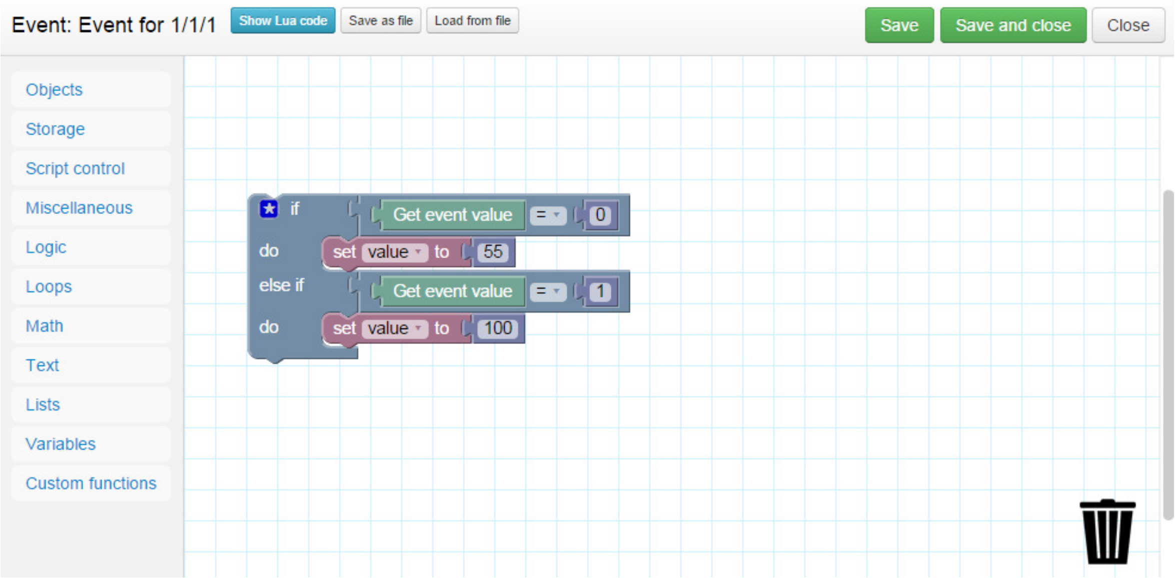
In order to create blocks, enable this functionality in *Utilities* → *General configuration* → *Enable Block Editor*.

Once the script is added, you can see puzzle icon to access Block editor.

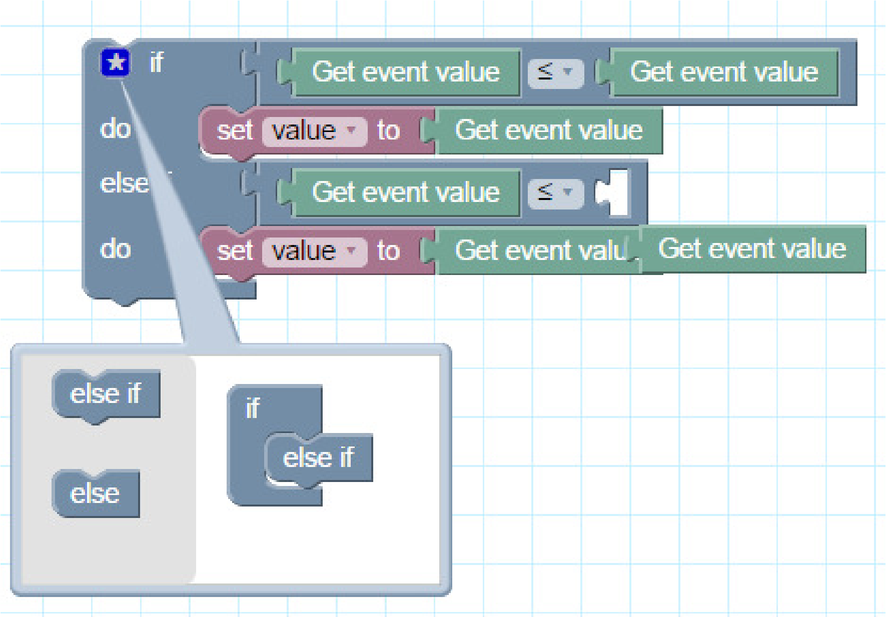


Script name	Group address / tag ▲	Description	Category	Editor	Active	Dup...	
Event for 1/1/1	1/1/1		1	✎ 🧩	🔴	⧉	⊗
Event for 1/1/11	1/1/11			✎ 🧩	🟢	⧉	⊗
Event for 1/1/13	1/1/13			✎ 🧩	🟢	⧉	⊗
Event for 1/1/2	1/1/2			✎ 🧩	🟢	⧉	⊗
scene1	1/1/3			✎ 🧩	🔴	⧉	⊗
scene1 copy	1/1/3			✎ 🧩	🔴	⧉	⊗
Event for 1/1/4	1/1/4			✎ 🧩	🟢	⧉	⊗

Blocks are sorted by categories on the left side. Each block is puzzle based and can be put only in appropriate location / other block.

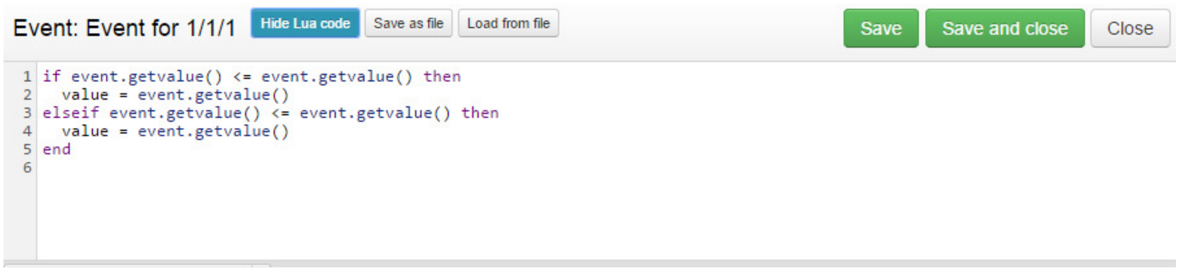


If the block is indicated with the blue label on the top left corner, you can define the structure of the block (e.g. If Else)



Press Delete button or drag the block to the garbage if you want to delete it

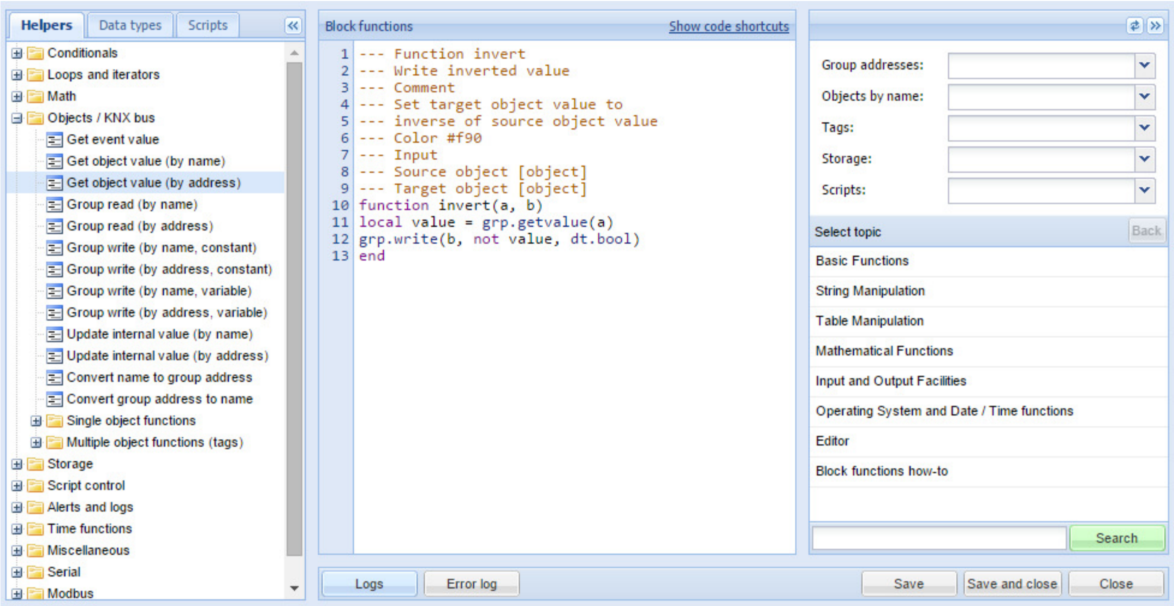
You can always look at the LUA code by clicking on *Show/Hide Lua code* button. This will allow to learn the scripting language.



```
if event.getvalue() <= event.getvalue() then
  value = event.getvalue()
elseif event.getvalue() <= event.getvalue() then
  value = event.getvalue()
end

```

1.2.2. *Block functions*

In Scripting menu there is *Block functions* button. Here you can create custom block functions which can be later used as ready block in Block editor.



```
--- Function invert
--- Write inverted value
--- Comment
--- Set target object value to
--- inverse of source object value
--- Color #f90
--- Input
--- Source object [object]
--- Target object [object]
function invert(a, b)
local value = grp.getvalue(a)
grp.write(b, not value, dt.bool)
end
```

Each function must have a special comment in order to be converted to a block.

- First line must have **Function** keyword followed by the function name
- Second line contains short function description which is shown as block title
- If third line contains **Comment** keyword, all following lines until Input or Output will be added to block comment tooltip
- Optionally, block color may be specified in hexadecimal format (#f00 or #ff9900) or numeric format as hue value between 0 and 359
- Following lines contain input and output lists. Each block can have any number of inputs and outputs:

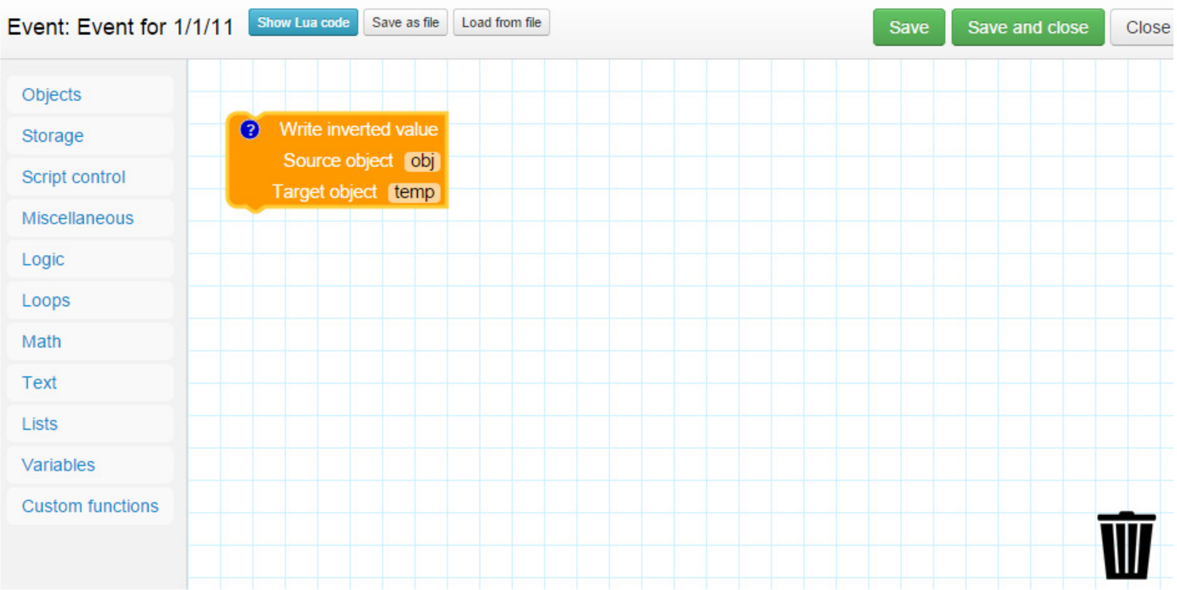
Inputs are a function parameter, other blocks can be connected to inputs by default. If input definition has **[object], [storage]** or **[tag]** in its name then the input is replaced with object, storage or tag selection input.

Each output variable is assigned to the corresponding function return value.

Example:

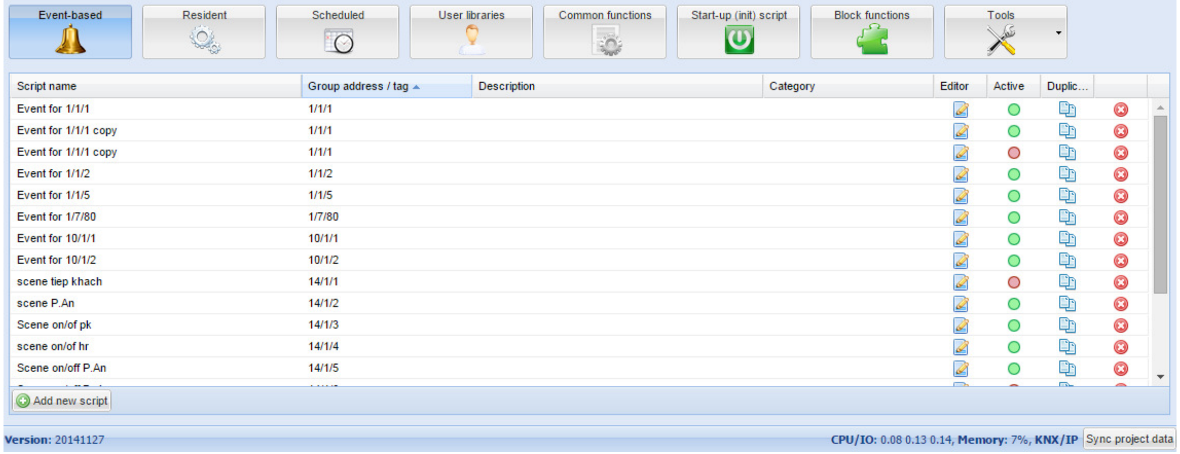
```
--- Function invert
--- Write inverted value
--- Comment
--- Set target object value to
--- inverse of source object value
--- Color #f90
--- Input
--- Source object [object]
--- Target object [object]
function invert(a, b)
local value = grp.getvalue(a)
grp.write(b, not value, dt.bool)
end
```

Once block function is added, it is available as a block in Block editor.



1.2.3. Adding a new script

Click on *Add new script* button on the bottom part of the *Event-based, Resident* or *Scheduled submenus*

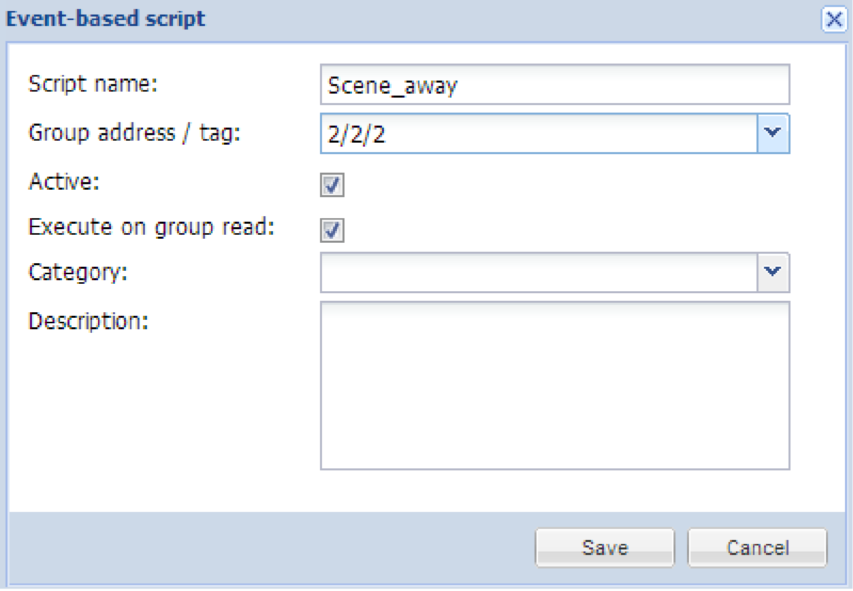


Script name	Group address / tag ▲	Description	Category	Editor	Active	Duplic...	
Event for 1/1/1	1/1/1			✎	🟢	⧉	⊗
Event for 1/1/1 copy	1/1/1			✎	🟢	⧉	⊗
Event for 1/1/1 copy	1/1/1			✎	🔴	⧉	⊗
Event for 1/1/2	1/1/2			✎	🟢	⧉	⊗
Event for 1/1/5	1/1/5			✎	🟢	⧉	⊗
Event for 1/7/80	1/7/80			✎	🟢	⧉	⊗
Event for 10/1/1	10/1/1			✎	🟢	⧉	⊗
Event for 10/1/2	10/1/2			✎	🟢	⧉	⊗
scene tiep khach	14/1/1			✎	🔴	⧉	⊗
scene P.An	14/1/2			✎	🟢	⧉	⊗
Scene on/of pk	14/1/3			✎	🟢	⧉	⊗
scene on/of hr	14/1/4			✎	🟢	⧉	⊗
Scene on/off P.An	14/1/5			✎	🟢	⧉	⊗

Version: 20141127 **CPU/IO:** 0.08 0.13 0.14, **Memory:** 7%, **KNX/IP**

The following fields should be filled when adding a new script:

Event-based



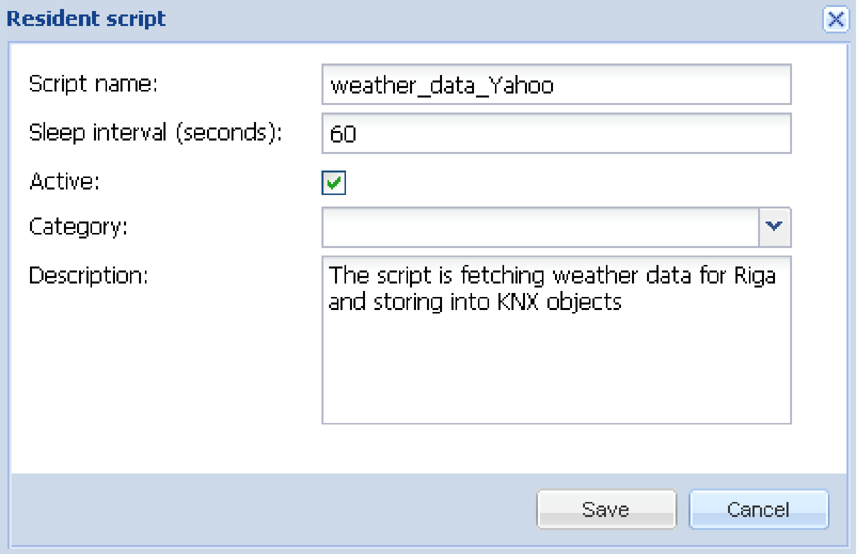
Event-based script

Script name:	Scene_away
Group address / tag:	2/2/2
Active:	☑
Execute on group read:	☑
Category:	
Description:	

- ➢ ***Script name*** – the name of the script
- ➢ ***Group address / Tag*** – specific group address or tag name on which the script will be triggered
- ➢ ***Active***– specifies whether the script is active (green circle) or disabled (red circle)
- ➢ ***Execute on group read***– specifies whether the script is executed on KNX group read telegram
- ➢ ***Category*** – a new or existing name of the category the script will be included. This will not affect on script action, helps only by grouping the scripts and watching by categories in *Tools* → *Print* script listings page

- ***Description***– description of the script

Resident



Resident script

Script name: weather_data_Yahoo

Sleep interval (seconds): 60

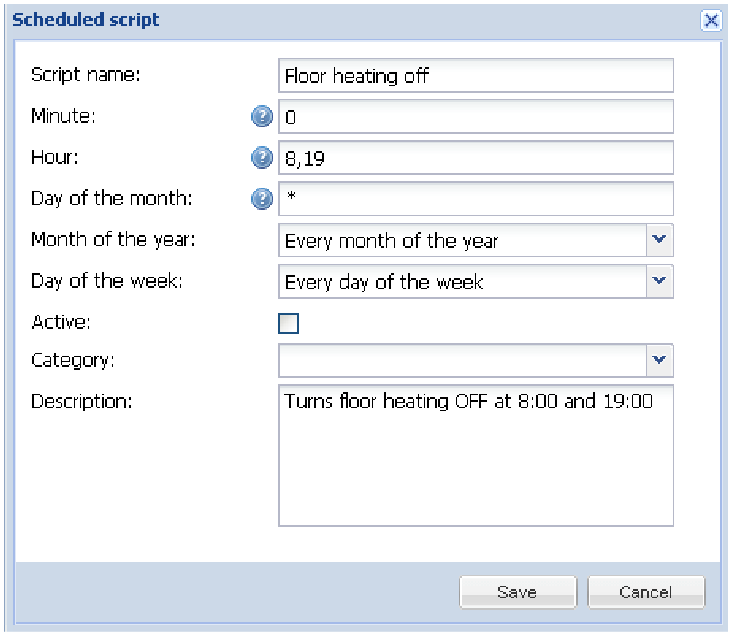
Active: ☑

Category:

Description: The script is fetching weather data for Riga and storing into KNX objects

- ***Script name*** – the name of the script
- ***Sleep interval (seconds)*** – interval after which the script will be executed.
- ***Active***– specifies whether the script is active (green circle) or disabled (red circle)
- ***Category*** – a new or existing name of the category the script will be included. This will not affect on script action, helps only by grouping the scripts and watching by categories in *Tools* → *Print* script listings page
- ***Description***– description of the script

Scheduled



Scheduled script

Script name: Floor heating off

Minute: 0

Hour: 8,19

Day of the month: *

Month of the year: Every month of the year

Day of the week: Every day of the week

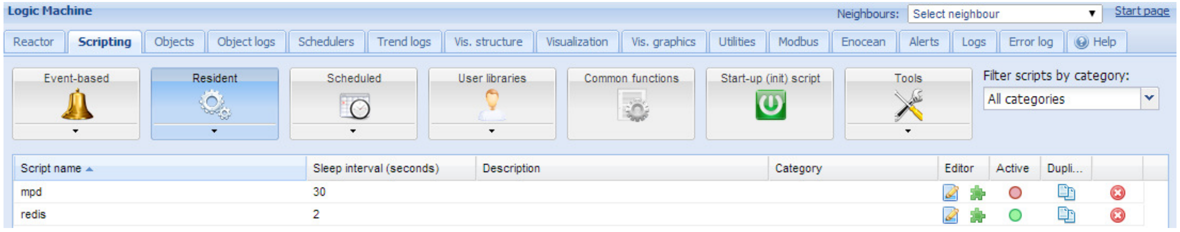
Active: □

Category:

Description: Turns floor heating OFF at 8:00 and 19:00

- ➢ ***Script name*** – the name of the script
- ➢ ***Minute*** – Minute
- ➢ ***Hour*** – Hour
- ➢ ***Day of the month*** – Day of the month
- ➢ ***Month of the year*** – Month of the year
- ➢ ***Day of the week*** – Day of the week
- ➢ ***Active***– specifies whether the script is active (green circle) or disabled (red circle)
- ➢ ***Category*** – a new or existing name of the category the script will be included. This will not affect on script action, helps only by grouping the scripts and watching by categories in *Tools* → *Print* script listings page
- ➢ ***Description***– description of the script

List of scripts



Filter scripts by category: All categories

Script name ▲	Sleep interval (seconds)	Description	Category	Editor	Active	Dupli...	
mpd	30				🔴		⊗
redis	2				🟢		⊗

There are five actions you can do with each of the script:

Duplicate – Duplicate the script with its source code
Editor – Enter scripting editor to write specific code for the particular program. It can be source code editor or block programming
Active – Make script active (green) or deactivate it (red)
Delete – Delete the script. When pressing this icon the confirmation is asked to accept the delete.

1.2.4. Event-based scripting

Event-based scripting can be used to implement custom logic for group address or tag events. User-defined function is executed when a "group write" or "group read" (if checked while adding the script) event occurs for given group address. Event information is stored in global **event** variable.Variable contents:

- dstraw (integer) — raw destination group address
- srcraw (integer) — raw source individual address
- dst (string) — decoded destination group address (for example: 1/1/4)
- src (string) — decoded source individual address (for example: 1.1.2)
- type (string) — type of event, either "groupwrite", "groupread", "groupresponse". Currently user-defined scripts are bound to "group write" events only.
- dataraw (integer/string) — raw binary data
- datahex (string) — data as a hex-encoded string which can be used to convert value to Lua variable

*Note!**event** variable is available only in Event-based functions, not in Resident and Scheduled.*

Note! All event-based scripts are executed in a single queue-like manner. Make sure event scripts do not contain infinite loops, sleep calls or other blocking parts.

Note! To get event value in scripts, use the following command: **a = event.getvalue()**

Note! To get event group address object name, use the following command:
a = grp.alias(event.dst)

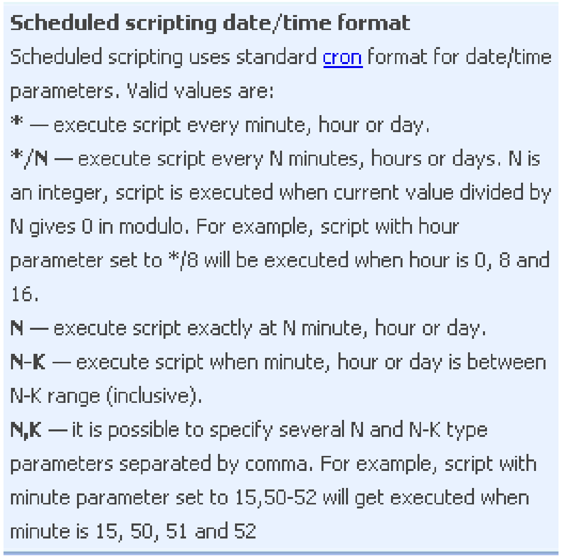
1.2.5. Resident scripting

Resident scripts are executed infinite amount of times. Scripts are put into inactive state after each call and are resumed after delay timer expires.

Note!even though resident scripts are executed in parallel they should not have infinite loops or it will not be possible to reload scripts after editing.

1.2.6. Scheduled scripting

Scheduled scripts are executed when the system time matches the specified script start time. Scheduled script is run only once after each timer call.



Scheduled scripting date/time format

Scheduled scripting uses standard cron format for date/time parameters. Valid values are:

***** — execute script every minute, hour or day.

***/N** — execute script every N minutes, hours or days. N is an integer, script is executed when current value divided by N gives 0 in modulo. For example, script with hour parameter set to */8 will be executed when hour is 0, 8 and 16.

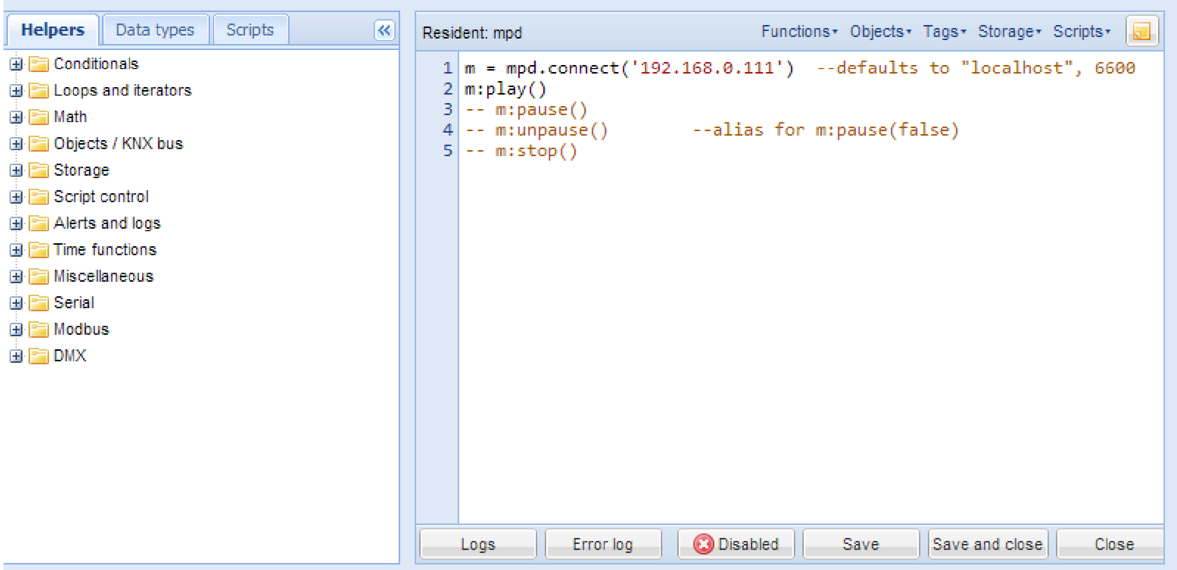
N — execute script exactly at N minute, hour or day.

N-K — execute script when minute, hour or day is between N-K range (inclusive).

N,K — it is possible to specify several N and N-K type parameters separated by comma. For example, script with minute parameter set to 15,50-52 will get executed when minute is 15, 50, 51 and 52

1.2.7. Script editor

When a script is added 📝 icon appears in *Editor*column that allows opening a script in scripting editor and re-working it with built-in code snippets.



```
m = mpd.connect('192.168.0.111')  --defaults to "localhost", 6600
m:play()
-- m:pause()
-- m:unpause()        --alias for m:pause(false)
-- m:stop()
```

The idea is that not knowing the syntaxes you get a helper for writing your own scripts. Code snippets save also a time and make the coding much more convenient. After clicking on appropriate snippet, it automatically adds code to the editor field.

There are three main groups of Script editor:

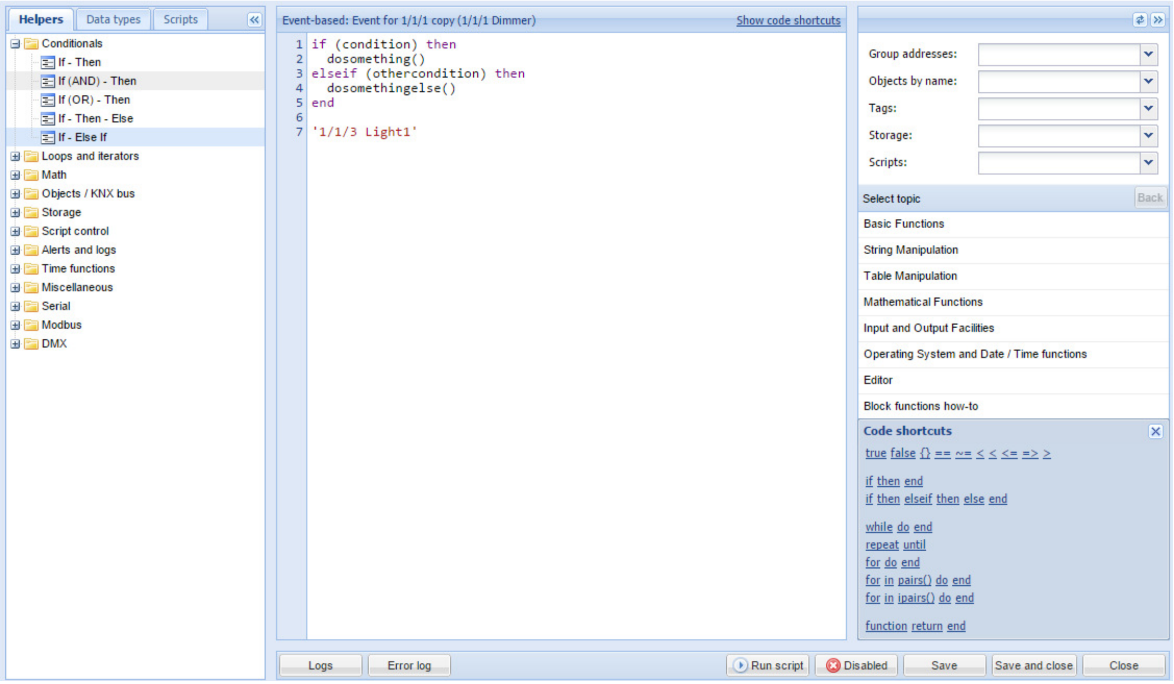
Helpers – predefined code snippets, like if-then statement. Helpers consist of three main sub-groups:

Conditionals – If Else If, If Then etc.
Loops and iterators – Array, Repeat..Untiletc
Math – Random value, Ceiling, Absolute value, Round etc.
Objects/KNX bus – Get object value, Group read, Group write, Update interval etc.
Storage – Get data from storage, Save data to storage
Script control – Get other script status, enable or disable other scripts
Alerts and logs – Alert, Log variables, Formatted alert
Time functions – Delay script execution
Miscellaneous – Sunrise/sunset etc.
Serial – Communication through internal LogicMachine IO ports
Modbus – Create RTU/TCP connection, Write register, Read register etc.
DMX – Communication with DMX devices

Data types – choose object by data type
Scripts – list of all scripts added in the LogicMachine

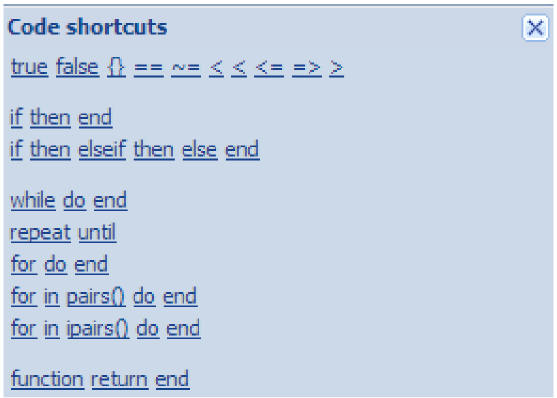
Code helpers on the right side of the editor

There is a special section in scripting editor which allows quickly find functions, objects or tags by name and storage variables.

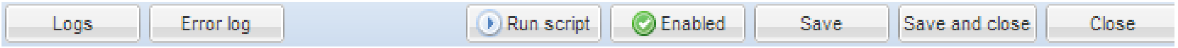


```
if (condition) then
  dosomething()
elseif (othercondition) then
  dosomethingelse()
end

'1/1/3 Light1'
```

There is also a code shortcut button, which helps with most common function structure.



There are also following helpful button in the script editor, which allows quickly access Error Logs, Test the script, Enable or disable it.



1.2.8. Object functions

grp provides simplified access to the objects stored in the database and group address request helpers.

Most functions use alias parameter — object group address or unique object name. (e.g. '1/1/1' or 'My object')

grp.getvalue(alias)
Returns value for the given alias or Lua nil when object cannot be found.

grp.find(alias)
Returns single object for the given alias. Object value will be decoded automatically only if the data type has been specified in the 'Objects' module. Returns Lua nil when object cannot be found, otherwise it returns Lua table with the following items:

- *address* — object group address
- *updatetime* — latest update time in UNIX timestamp format. Use Lua os.date() to convert to readable date formats

When object data type has been specified in the 'Objects' module the following fields are available:

- *name* — unique object name
- *datatype* — object data type as specified by user
- *decoded* — set to true when decoded value is available
- *value* — decoded object value

grp.tag(tags, mode)
Returns Lua table containing objects with the given tag. Tags parameter can be either Lua table or a string. Mode parameter can be either 'all' (return objects that have all of the given tags) or 'any' (default — returns objects that have any of the given tags). You can use*Returned object functions* on the returned table.

grp.alias(alias)
Converts group address to object name or name to address. Returns Lua nil when object cannot be found.

1.2.9. Returned object functions, group communication functions

Objects received by using grp.find(alias) or grp.tag(tags, mode) have the following functions attached to them:

Always check that the returned object was found otherwise calling these functions will result in an error. See the example below.

object:write(value, datatype)
Sends group write request to object's group address. Data type is taken from the database if not specified as second parameter. Returns Lua boolean as the result.

object:response(value, datatype)
Similar to object:write. Sends group response request to object's group address.

object:read()
Sends group read request to object's group address. Note: this function returns immediately and cannot be used to return the result of read request. Use event-based script instead.

object:update(value, datatype)
Similar to object:write, but does not send new value to the bus. Useful for objects that are used only in visualization.

1.2.10.Group communication functions

These functions should only be used if it is required to access objects by group address directly, it is recommended to use single or multiple object functions.

grp.write(alias, value, datatype)
Sends group write request to the given alias. Data type is taken from the database if not specified as third parameter. Returns Lua boolean as the result.

grp.response(alias, value, datatype)
Similar to grp.write. Sends group response request to the given alias.

grp.read(alias)
Sends group read request to the given alias. Note: this function returns immediately and cannot be used to return the result of read request. Use event-based script instead.

grp.update(alias, value, datatype)
Similar to grp.write, but does not send new value to the bus. Useful for objects that are used only in visualization.

1.2.11.Object function examples

<u>Find object by name and write new value.</u>

```
myobject=grp.find('My object')
-- grp.find will return nil if object was not found
ifmyobjectthen
myobject:write(1)-- update object value with 1
end
```

<u>Find object by address and write new value.</u>

```
myobject=grp.find('1/1/15')
-- verify that the requested object was found
ifmyobjectthen
myobject:write(52.12, dt.float16)-- explicitly set data type to dt.float16 (2-byte floating point)
```

```
end
```

Switch all binary objects tagged 'lights' off.

```
lights =grp.tag('lights')
lights:write(false)
```

Group write to the specified group address and data type.

```
grp.write('1/1/1', true, dt.bool)-- write 1-bit 'on' to 1/1/1
grp.write('1/1/2', 50, dt.scale)-- write 1-byte 50% to 1/1/2
```

1.2.12. Data type functions, data types

knxdatatype object provides data encoding and decoding between Lua and KNX data formats.

knxdatatype.decode(value, datatype)
Converts hex-encoded data to Lua variable based on given data type. Data type is specified either as KNX primary data type (integer between 1 and 16) or a secondary data type (integer between 1000 and 16000).Return values:

- success — decoded data as Lua variable (type depends on data type), value length in bytes
- error — nil, error string

1.2.13.Data types

The following data types can be used for encoding and decoding of KNX data. Data representation on Lua level and predefined constants (in bold) is given below:

- *1 bit (boolean) - **dt.bool*** — boolean
- *2 bit (1 bit controlled) - **dt.bit2*** — number
- *4 bit (3 bit controlled) - **dt.bit4*** — number
- *1 byte ASCII character - **dt.char*** — string
- *1 byte unsigned integer - **dt.uint8*** — number
- *1 byte signed integer - **dt.int8*** — number
- *2 byte unsigned integer - **dt.uint16*** — number
- *2 byte signed integer - **dt.int16*** — number
- *2 byte floating point - **dt.float16*** — number
- *3 byte time / day - **dt.time*** — table with the following items:
  - day — number (0-7)
  - hour — number (0-23)
  - minute — number (0-59)
  - second — number (0-59)
- *3 byte date - **dt.date*** — table with the following items:
  - day — number (1-31)
  - month — number (1-12)
  - year — number (1990-2089)

- *4 byte unsigned integer -* ***dt.uint32*** — number
- *4 byte signed integer -* ***dt.int32*** — number
- *4 byte floating point -* ***dt.float32*** — number
- *4 byte access control -* ***dt.access*** — number, currently not fully supported
- *14 byte ASCII string -* ***dt.string*** — string, null characters ('\0') are discarded during decoding

1.2.14. Data storage function

storage object provides persistent key-value data storage for user scripts. Only the following Lua data types are supported:

- boolean
- number
- string
- table

storage.set(key, value)

Sets new value for the given key. Old value is overwritten. Returns boolean as the result and an optional error string.

storage.get(key, default)

Gets value for the given key or returns default value (nil if not specified) if key is not found in the data storage.

Note: all user scripts share the same data storage. Make sure that same keys are not used to store different types of data.

Examples

- The following examples shows the basic syntax of storage.set. Result will return boolean true since the passed parameters are correct

```lua
result=storage.set('my_stored_value_1', 12.21)
```

- This example will return false as the result because we are trying to store a function which is not possible.

```lua
testfn=function(t)
return t * t
end
result =storage.set('my_stored_value_2', testfn)-- this will result in an error
```

- The following examples shows the basic syntax of storage.get. Assuming that key value was not found, first call will return nil while second call will return number 0 which was specified as a default value.

```
result =storage.get('my_stored_value_3')-- returns nil if value is not found
result =storage.get('my_stored_value_3', 0)-- returns 0 if value is not found
```

- When storing tables make sure to check the returned result type. Assume we have created a storage item with key test_object_data.

```
objectdata={}
objectdata.temperature=23.1
objectdata.scene='default'
result =storage.set('test_object_data', objectdata)-- store objectdata variable as 'test_object_data'
```

- Now we are retrieving data from storage. Data type is checked for correctness.

```
objectdata=storage.get('test_object_data')
iftype(objectdata)=='table'then
ifobjectdata.temperature> 24 then
-- do something if temperature level is too high
end
end
```

1.2.15. *Alert function*

alert(message, [var1, [var2, [var3]]])
Stores alert message and current system time in the main database. All alerts are accessible in the "Alerts" module. This function behaves exactly as Lua string.format.

Example

```
temperature = 25.3
if temperature > 24 then
-- resulting message: 'Temperature levels are too high: 25.3'
  alert('Temperature level is too high: %.1f', temperature)
end
```

1.2.16. *Log function*

log(var1, [var2, [var3, ...]])
Converts variables to human-readable form and stores them in the main database. All items are accessible in the "Logs" module.

Example

```
-- log function accepts Lua nil, boolean, number and table (up to 5 nested levels) type variables
a ={ key1 ='value1', key2 =2}
b ='test'
c =123.45
-- logs all passed variables
log(a, b, c)
```

1.2.17.Scheduled scripting date/time format

Scheduled scripting uses standard *cron* format for date/time parameters. Valid values are:

*** — execute script every minute, hour or day.

/N — execute script every N minutes, hours or days. N is an integer, script is executed when current value divided by N gives 0 in modulo. For example, script with hour parameter set to */8 will be executed when hour is 0, 8 and 16.

N — execute script exactly at N minute, hour or day.

N-K — execute script when minute, hour or day is between N-K range (inclusive).

N,K — it is possible to specify several *N* and *N-K* type parameters separated by comma. For example, script with minute parameter set to 15,50-52 will get executed when minute is 15, 50, 51 and 52

1.2.18.Time function

os.sleep(delay)
Delay the next command execution for the delay seconds.

os.microtime ()
Returns two values: current timestamp in seconds and timestamp fraction in nanoseconds

os.udifftime (sec, usec)
Returns time difference as floating point value between now and timestamp components passed to this function (seconds, nanoseconds)

1.2.19.Data Serialization

serialize.encode (value)
Generates a storable representation of a value.

serialize.decode (value)
Creates a Lua value from a stored representation.

1.2.20.String functions

This library provides generic functions for string manipulation, such as finding and extracting substrings, and pattern matching. When indexing a string in Lua, the first character is at position 1 (not at 0, as in C).

Indices are allowed to be negative and are interpreted as indexing backwards, from the end of the string. Thus, the last character is at position -1, and so on.

The string library provides all its functions inside the table string. It also sets a meta table for strings where the __index field points to the string table. Therefore, you can use the string functions in object-oriented style. For instance, *string.byte(s, i)* can be written as *s:byte(i)*.The string library assumes one-byte character encodings.

string.trim (str)

Trims the leading and trailing spaces off a given string.

string.split (str, sep)

Splits string by given separator string. Returns Lua table.

string.byte (s [, i [, j]])

Returns the internal numerical codes of the characters *s[i], s[i+1], ···, s[j]*. The default value for *i* is 1;the default value for *j* is i.Note that numerical codes are not necessarily portable across platforms.

string.char (···)

Receives zero or more integers. Returns a string with length equal to the number of arguments, in which each character has the internal numerical code equal to its corresponding argument. Note that numerical codes are not necessarily portable across platforms.

string.find (s, pattern [, init [, plain]])

Looks for the first match of pattern in the string s. If it finds a match, then find returns the indices of *s* where this occurrence starts and ends; otherwise, it returns *nil*. A third, optional numerical argument init specifies where to start the search; its default value is 1 and can be negative. A value of true as a fourth, optional argument plain turns off the pattern matching facilities, so the function does a plain "find substring" operation, with no characters in pattern being considered "magic". Note that if plain is given, then init must be given as well. If the pattern has captures, then in a successful match the captured values are also returned, after the two indices.

string.format (formatstring, ···)

Returns a formatted version of its variable number of arguments following the description given in its first argument (which must be a string). The format string follows the same rules as the printf family of standard C functions. The only differences are that the options/modifiers *, l, L, n, p, and h are not supported and that there is an extra option, q. The q option formats a string in a form suitable to be safely read back by the Lua interpreter: the string is written between double quotes, and all double quotes, newlines, embedded zeros, and backslashes in the string are correctly escaped when written. For instance, the call

```
string.format('%q', 'a string with "quotes" and \n new line')
```

will produce the string:

```
    "a string with \"quotes\" and \
```

new line"

The options *c, d, E, e, f, g, G, i, o, u, X,* and *x* all expect a number as argument, whereas *q* and *s* expect a string. This function does not accept string values containing embedded zeros, except as arguments to the *q* option.

string.gmatch (s, pattern)

Returns an iterator function that, each time it is called, returns the next captures from pattern over strings. If pattern specifies no captures, then the whole match is produced in each call. As an example, the following loop

```
s ="hello world from Lua"
for w instring.gmatch(s, "%a+")do
print(w)
end
```

will iterate over all the words from string *s*, printing one per line. The next example collects all pairs *key=value* from the given string into a table:

```
t ={}
s ="from=world, to=Lua"
for k, v instring.gmatch(s, "(%w+)=(%w+)")do
   t[k]= v
end
```

For this function, a '^' at the start of a pattern does not work as an anchor, as this would prevent the iteration.

string.gsub (s, pattern, repl [, n])

Returns a copy of s in which all (or the first n, if given) occurrences of the pattern have been replaced by are placement string specified by repl, which can be a string, a table, or a function. gsub also returns, as its second value, the total number of matches that occurred.
If *repl* is a string, then its value is used for replacement. The character % works as an escape character:any sequence in repl of the form *%n*, with *n* between 1 and 9, stands for the value of the n-th capture dsub string (see below). The sequence %0 stands for the whole match. The sequence %% stands for a single %.
If *repl* is a table, then the table is queried for every match, using the first capture as the key; if the pattern specifies no captures, then the whole match is used as the key.
If *repl* is a function, then this function is called every time a match occurs, with all captured substrings passed as arguments, in order; if the pattern specifies no captures, then the whole match is passed as a sole argument.
If the value returned by the table query or by the function call is a string or a number, then it is used as the replacement string; otherwise, if it is *false* or *nil*, then there is no replacement (that is, the original match is kept in the string).

Examples:

```
x =string.gsub("hello world", "(%w+)", "%1 %1")
```

--> x="hello hello world world"

```
x =string.gsub("hello world", "%w+", "%0 %0", 1)
```

--> x="hello hello world"

```
x =string.gsub("hello world from Lua", "(%w+)%s*(%w+)", "%2 %1")
```

--> x="world hello Lua from"

```
x =string.gsub("home = $HOME, user = $USER", "%$(%w+)", os.getenv)
```

--> x="home = /home/roberto, user = roberto"

```
x =string.gsub("4+5 = $return 4+5$", "%$(.-)%$", function(s)
returnloadstring(s)()
end)
```

--> x="4+5 = 9"

```
local t ={name="lua", version="5.1"}
      x =string.gsub("$name-$version.tar.gz", "%$(%w+)", t)
```

--> x="lua-5.1.tar.gz"

string.len (s)

Receives a string and returns its length. The empty string "" has length 0. Embedded zeros are counted, so"a\000bc\000" has length 5.

string.lower (s)

Receives a string and returns a copy of this string with all uppercase letters changed to lowercase. All other characters are left unchanged. The definition of what an uppercase letter is depends on the current locale.

string.match (s, pattern [, init])

Looks for the first match of pattern in the string s. If it finds one, then match returns the captures from the pattern; otherwise it returns *nil*. If pattern specifies no captures, then the whole match is returned. A third, optional numerical argument init specifies where to start the search; its default value is 1 and can be negative.

string.rep (s, n)

Returns a string that is the concatenation of n copies of the string s.

string.reverse (s)

Returns a string that is the string s reversed.

string.sub (s, i [, j])

Returns the substring of s that starts at i and continues until j; i and j can be negative. If j is absent, then it is assumed to be equal to -1 (which is the same as the string length). In particular, the call*string.sub(s,1,j)* returns a prefix of s with length j, and *string.sub(s, -i)* returns a suffix of *s* with length *i*.

string.upper (s)

Receives a string and returns a copy of this string with all lowercase letters changed to uppercase. All other characters are left unchanged. The definition of what a lowercase letter is depends on the current locale.

<u>Patterns</u>

Character Class:

A character class is used to represent a set of characters. The following combinations are allowed in describing a character class:

- **x**: (where x is not one of the magic characters ^$()%.[]*+-?) represents the character x itself.
- **.**: (a dot) represents all characters.
- **%a**: represents all letters.
- **%c**: represents all control characters.
- **%d**: represents all digits.
- **%l**: represents all lowercase letters.
- **%p**: represents all punctuation characters.

- **%s**: represents all space characters.
- **%u**: represents all uppercase letters.
- **%w**: represents all alphanumeric characters.
- **%x**: represents all hexadecimal digits.
- **%z**: represents the character with representation 0.
- **%x**: (where x is any non-alphanumeric character) represents the character x. This is the standard way to escape the magic characters. Any punctuation character (even the non magic) can be preceded by a '%' when used to represent itself in a pattern.
- **[set]**: represents the class which is the union of all characters in set. A range of characters can be specified by separating the end characters of the range with a '-'. All classes %x described above can also be used as components in set. All other characters in set represent themselves. For example,[%w_] (or [_%w]) represents all alphanumeric characters plus the underscore, [0-7] represents the octal digits, and [0-7%l%-] represents the octal digits plus the lowercase letters plus the '-'character.
- The interaction between ranges and classes is not defined. Therefore, patterns like [%a-z]or [a-%%] have no meaning.
- **[^set]**: represents the complement of set, where set is interpreted as above.

For all classes represented by single letters (%a, %c, etc.), the corresponding uppercase letter represents the complement of the class. For instance, %S represents all non-space characters.
The definitions of letter, space, and other character groups depend on the current locale. In particular, the class [a-z] may not be equivalent to %l.

Pattern Item:
A pattern item can be:

- a single character class, which matches any single character in the class;
- a single character class followed by '*', which matches 0 or more repetitions of characters in the class. These repetition items will always match the longest possible sequence;
- a single character class followed by '+', which matches 1 or more repetitions of characters in the class. These repetition items will always match the longest possible sequence;
- a single character class followed by '-', which also matches 0 or more repetitions of characters in the class. Unlike '*', these repetition items will always match the shortest possible sequence;
- a single character class followed by '?', which matches 0 or 1 occurrence of a character in the class;
- %n, for n between 1 and 9; such item matches a substring equal to the n-th captured string (see below);
- %bxy, where x and y are two distinct characters; such item matches strings that start with x, end with y, and where the x and y are balanced. This means that, if one reads the string from left to right, counting +1 for an x and -1 for a y, the ending y is the first y where the count reaches 0. For instance, the item %b() matches expressions with balanced parentheses.

Pattern:
A pattern is a sequence of pattern items. A '^' at the beginning of a pattern anchors the match at the beginning of the subject string. A '$' at the end of a pattern anchors the match at the end of the subject string. At other positions, '^' and '$' have no special meaning and represent themselves.

Captures:
A pattern can contain sub-patterns enclosed in parentheses; they describe captures. When a match succeeds, the substrings of the subject string that match captures are stored (captured) for future use. Captures are numbered according to their left parentheses. For instance, in the pattern

"(a*(.)%w(%s*))",the part of the string matching "a*(.)%w(%s*)" is stored as the first capture (and therefore has number 1);the character matching "." is captured with number 2, and the part matching "%s*" has number 3.
As a special case, the empty capture () captures the current string position (a number). For instance, if we apply the pattern "()aa()" on the string "flaaap", there will be two captures: 3 and 5.A pattern cannot contain embedded zeros. Use %z instead.

1.2.21.Input and output functions

io.exists (path)

Checks if given path (file or directory) exists. Return boolean.

io.readfile (file)

Reads whole file at once. Return file contents as a string on success or nil on error.

io.writefile (file, data)

Writes given data to a file. Data can be either a value convertible to string or a table of such values. When data is a table then each table item is terminated by a new line character. Return boolean as write result when file can be open for writing or nil when file cannot be accessed.

Example: Write event status to log file located on plugged USB flash drive:

```
value = knxdatatype.decode(event.datahex, dt.bool)
data =string.format('%s value is %s', os.date('%c'), tostring(value))
-- write to the end of log file preserving all previous data
file =io.open('/mnt/usb/log.txt', 'a+')
file:write(data .. '\r\n')
file:close()
```

Output:

```
Mon Jan 3 05:25:13 2011 value is false
Mon Jan 3 05:25:14 2011 value is true
Mon Jan 3 05:25:32 2011 value is false
Mon Jan 3 05:25:33 2011 value is true
```

Example: Read data from file (config in format key=value)

```
for line inio.lines('/mnt/usb/config.txt')do
-- split line by '=' sing
  items = line:split('=')
-- two items, line seems to be valid
if #items == 2 then
    key = items[1]:trim()
    value = items[2]:trim()
    alert('[config] %s = %s', key, value)
end
end
```

1.2.22.Script control functions

script.enable('scriptname')
Enable the script with the name scriptname.

script.disable('scriptname')
Disable the script with the name scriptname.

status = script.status('scriptname')
Returns true/false if script is found, nil otherwise

1.2.23.JSON library

Note: json is not loaded by default, use *require('json')* before calling any functions from this library.

json.encode (value)
Converts Lua variable to JSON string. Script execution is stopped in case of an error.

json.pencode (value)
Converts Lua variable to JSON string in protected mode, returns nil on error.

json.decode (value)
Converts JSON string to Lua variable. Script execution is stopped in case of an error.

json.pdecode (value)
Converts JSON string to Lua variable in protected mode, returns nil on error.

1.2.24.Conversion

Compatibility layer:*lmcore* is an alias of *cnv*.

cnv.strtohex (str)
Converts given binary string to a hex-encoded string.

cnv.hextostr (hex [, keepnulls])
Converts given hex-encoded string to a binary string. NULL characters are ignored by default, but can be included by setting second parameter to true.

cnv.tonumber (value)
Converts the given value to number using following rules: numbers and valid numeric strings are treated as is, boolean *true* is 1, boolean *false* is 0, everything else is *nil*.

cnv.hextoint(hexvalue, bytes)
Converts the given hex string to and integer of a given length in bytes.

cnv.inttohex(intvalue, bytes)
Converts the given integer to a hex string of given bytes.

cnv.strtohex(str)
Converts the given binary string to a hex-encoded string.

cnv.hextostr(hexstr)
Converts the given hex-encoded string to a binary string.

1.2.25.Bit operators

bit.bnot (value)
Binary not

bit.band (x1 [, x2...])
Binary and between any number of variables

bit.bor (x1 [, x2...])
Binary and between any number of variables

bit.bxor (x1 [, x2...])
Binary and between any number of variables

bit.lshift (value, shift)
Left binary shift

bit.rshift (value, shift)
Right binary shift

1.2.26.Input and Output Facilities

The I/O library provides two different styles for file manipulation. The first one uses implicit file descriptors; that is, there are operations to set a default input file and a default output file, and all input/output operations are over these default files. The second style uses explicit file descriptors. When using implicit file descriptors, all operations are supplied by table *io*. When using explicit file descriptors, the operation *io.open* returns a file descriptor and then all operations are supplied as methods of the file descriptor.
The table *io* also provides three predefined file descriptors with their usual meanings from C: *io.stdin, io.stdout*, and *io.stderr*. The I/O library never closes these files.
Unless otherwise stated, all I/O functions return *nil* on failure (plus an error message as a second result and a system-dependent error code as a third result) and some value different from *nil* on success.

io.close ([file])
Equivalent to *file:close()*. Without a file, closes the default output file.

io.flush ()
Equivalent to file:flush over the default output file.

io.input ([file])
When called with a file name, it opens the named file (in text mode), and sets its handle as the default input file. When called with a file handle, it simply sets this file handle as the default input file. When called without parameters, it returns the current default input file. In case of errors this function raises the error, instead of returning an error code.

io.lines ([filename])
Opens the given file name in read mode and returns an iterator function that, each time it is called, returns a new line from the file. Therefore, the construction

for line in io.lines(filename) do body end

will iterate over all lines of the file. When the iterator function detects the end of file, it returns nil (to finishthe loop) and automatically closes the file.
The call *io.lines()* (with no file name) is equivalent to *io.input():lines()*; that is, it iterates over the lines of the default input file. In this case it does not close the file when the loop ends.

io.open (filename [, mode])
This function opens a file, in the mode specified in the string mode. It returns a new file handle, or, in case of errors, nil plus an error message. The mode string can be any of the following:

- "r": read mode (the default);
- "w": write mode;
- "a": append mode;
- "r+": update mode, all previous data is preserved;
- "w+": update mode, all previous data is erased;
- "a+": append update mode, previous data is preserved, writing is only allowed at the end of file.

The mode string can also have a 'b' at the end, which is needed in some systems to open the file in binary mode. This string is exactly what is used in the standard C function *fopen*.

io.output ([file])
Similar to io.input, but operates over the default output file.

1.2.27.Mathematical functions

This library is an interface to the standard C math library. It provides all its functions inside the table math.

math.abs (x)
Returns the absolute value of x.

math.acos (x)
Returns the arc cosine of x (in radians).

math.asin (x)
Returns the arc sine of x (in radians).

math.atan (x)
Returns the arc tangent of x (in radians).

math.atan2 (y, x)
Returns the arc tangent of y/x (in radians), but uses the signs of both parameters to find the quadrant of the result. (It also handles correctly the case of x being zero.)

math.ceil (x)
Returns the smallest integer larger than or equal to x.

math.cos (x)
Returns the cosine of x (assumed to be in radians).

math.cosh (x)
Returns the hyperbolic cosine of x.

math.deg (x)
Returns the angle x (given in radians) in degrees.

math.exp (x)
Returns the value e^x.

math.floor (x)
Returns the largest integer smaller than or equal to x.

math.fmod (x, y)
Returns the remainder of the division of x by y that rounds the quotient towards zero.

math.frexp (x)
Returns m and e such that x = $m2^e$, e is an integer and the absolute value of m is in the range [0.5, 1) (or zero when x is zero).

math.huge
The value HUGE_VAL, a value larger than or equal to any other numerical value.

math.ldexp (m, e)
Returns $m2^e$,(e should be an integer).

math.log (x)
Returns the natural logarithm of x.

math.log10 (x)
Returns the base-10 logarithm of x.

math.max (x, ···)
Returns the maximum value among its arguments.

math.min (x, ···)
Returns the minimum value among its arguments.

math.modf (x)
Returns two numbers, the integral part of x and the fractional part of x.

math.pi
The value of pi.

math.pow (x, y)
Returns x^y. (You can also use the expression x^y to compute this value.)

math.rad (x)
Returns the angle x (given in degrees) in radians.

math.random ([m [, n]])
This function is an interface to the simple pseudo-random generator function rand provided by ANSI C. (No guarantees can be given for its statistical properties.)
When called without arguments, returns a uniform pseudo-random real number in the range [0,1). When called with an integer number m, math. random returns a uniform pseudo-random integer in the range [1,m]. When called with two integer numbers m and n, math. random returns a uniform pseudo-random integer in the range [m, n].

math.randomseed (x)
Sets x as the "seed" for the pseudo-random generator: equal seeds produce equal sequences of numbers.

math.sin (x)
Returns the sine of x (assumed to be in radians).

math.sinh (x)
Returns the hyperbolic sine of x.

math.sqrt (x)
Returns the square root of x. (You can also use the expression x^0.5 to compute this value.)

math.tan (x)
Returns the tangent of x (assumed to be in radians).

math.tanh (x)

Returns the hyperbolic tangent of x.

1.2.28.Table manipulations

This library provides generic functions for table manipulation. It provides all its functions inside the table. Most functions in the table library assume that the table represents an array or a list. For these functions, when we talk about the "length" of a table we mean the result of the length operator.

table.concat (table [, sep [, i [, j]]])
Given an array where all elements are strings or numbers, returns *table[i]..sep..table[i+1]* ··· *sep..table[j]*. The default value for sep is the empty string, the default for *i* is 1, and the default for *j* is the length of the table. If *i* is greater than *j*, returns the empty string.

table.insert (table, [pos,] value)
Inserts element value at position pos in table, shifting up other elements to open space, if necessary. The default value for *pos* is *n+1*, where n is the length of the table, so that a call*table.insert(t,x)* inserts x at the end of table t.

table.maxn (table)
Returns the largest positive numerical index of the given table, or zero if the table has no positive numerical indices. (To do its job this function does a linear traversal of the whole table.)

table.remove (table [, pos])
Removes from table the element at position pos, shifting down other elements to close the space, if necessary. Returns the value of the removed element. The default value for pos is n, where n is the length of the table, so that a call *table.remove(t)* removes the last element of table t.

table.sort (table [, comp])
Sorts table elements in a given order, in-place, from *table[1]* to *table[n]*, where n is the length of the table. If comp is given, then it must be a function that receives two table elements, and returns true when the first is less than the second (so that not *comp(a[i+1],a[i]*) will be true after the sort). If comp is not given, then the standard Lua operator < is used instead.
The sort algorithm is not stable; that is, elements considered equal by the given order may have their relative positions changed by the sort.

1.2.29.Operating system facilities

os.date ([format [, time]])
Returns a string or a table containing date and time, formatted according to the given string format. If the time argument is present, this is the time to be formatted (see the *os.time* function for a description of this value). Otherwise, date formats the current time.
If format starts with '!', then the date is formatted in Coordinated Universal Time. After this optional character, if format is the string "*t", then date returns a table with the following fields: year (four digits),month (1--12), day (1--31), hour (0--23), min (0--59), sec (0--61), wday (weekday, Sunday is 1), yday (dayof the year), and isdst (daylight saving flag, a boolean).
If format is not "*t", then date returns the date as a string, formatted according to the same rules as the C function strftime.

When called without arguments, date returns a reasonable date and time representation that depends on the host system and on the current locale (that is, *os.date()* is equivalent to os.date("%c")).

os.difftime (t2, t1)

Returns the number of seconds from time t1 to time t2. In POSIX, Windows, and some other systems, this value is exactly t2-t1.

os.execute ([command])

This function is equivalent to the C function system. It passes command to be executed by an operating system shell. It returns a status code, which is system-dependent. If command is absent, then it returns nonzero if a shell is available and zero otherwise.

os.exit ([code])

Calls the C function exit, with an optional code, to terminate the host program. The default value for code is the success code.

os.getenv (varname)

Returns the value of the process environment variable varname, or *nil* if the variable is not defined.

os.remove (filename)

Deletes the file or directory with the given name. Directories must be empty to be removed. If this function fails, it returns nil, plus a string describing the error.

os.rename (oldname, newname)

Renames file or directory named oldname to newname. If this function fails, it returns *nil*, plus a string describing the error.

os.time ([table])

Returns the current time when called without arguments, or a time representing the date and time specified by the given table. This table must have fields year, month, and day, and may have fields hour, min, sec,and *isdst* (for a description of these fields, see the *os.date* function).
The returned value is a number, whose meaning depends on your system. In POSIX, Windows, and some other systems, this number counts the number of seconds since some given start time (the "epoch"). In other systems, the meaning is not specified, and the number returned by time can be used only as an argument to date and *difftime*.

os.tmpname ()

Returns a string with a file name that can be used for a temporary file. The file must be explicitly opened before its use and explicitly removed when no longer needed. On some systems (POSIX), this function also creates a file with that name, to avoid security risks. (Someone
else might create the file with wrong permissions in the time between getting the name and creating the file.) You still have to open the file to use it and to remove it (even if you do not use it).
When possible, you may prefer to use *io.tmpfile*, which automatically removes the file when the program
ends.

1.2.30.Extended function library

toboolean(value)

Converts the given value to boolean using following rules: *nil*, boolean *false*, *0*, *empty* string, *'0'* string are treated as *false*, everything else as *true*

string.split(str, sep)

Splits the given string into chunks by the given separator. Returns Lua table.

knxlib.decodeia(indaddressa, indaddressb)

Converts binary-encoded individual address to Lua string. This function accepts either one or two arguments (interpreted as two single bytes).

knxlib.decodega(groupaddressa, groupaddressb)

Converts binary-encoded group address to Lua string. This function accepts either one or two arguments (interpreted as two single bytes).

knxlib.encodega(groupaddress, separate)

Converts Lua string to binary-encoded group address. Returns group address a single Lua number when second argument is *nil* or *false* and two separate bytes otherwise.

ipairs (t)

Returns three values: an iterator function, the table t, and 0, so that the construction

```
      for i,v inipairs(t)dobodyend
```

will iterate over the pairs (1,t[1]), (2,t[2]), ···, up to the first integer key absent from the table.

next (table [, index])

Allows a program to traverse all fields of a table. Its first argument is a table and its second argument is an index in this table. next returns the next index of the table and its associated value. When called with *nil* as its second argument, next returns an initial index and its associated value. When called with the last index, or with *nil* in an empty table, next returns *nil*. If the second argument is absent, then it is interpreted as*nil*. In particular, you can use *next(t*) to check whether a table is empty. The order in which the indices are enumerated is not specified, even for numeric indices. (To traverse a table in numeric order, use a numerical for or the *ipairs* function.)The behavior of next is undefined if, during the traversal, you assign any value to a non-existent field in the table. You may however modify existing fields. In particular, you may clear existing fields.

pairs (t)

Returns three values: the *next* function, the table *t*, and *nil*, so that the construction

```
for k,v inpairs(t)do body end
```

will iterate over all key–value pairs of table t.

tonumber (e [, base])

Tries to convert its argument to a number. If the argument is already a number or a string convertible to a number, then tonumber returns this number; otherwise, it returns *nil*.

An optional argument specifies the base to interpret the numeral. The base may be any integer between 2and 36, inclusive. In bases above 10, the letter 'A' (in either upper or lower case) represents 10, 'B'represents 11, and so forth, with 'Z' representing 35. In base 10 (the default), the number can have a decimal part, as well as an optional exponent part. In other bases, only unsigned integers are accepted.

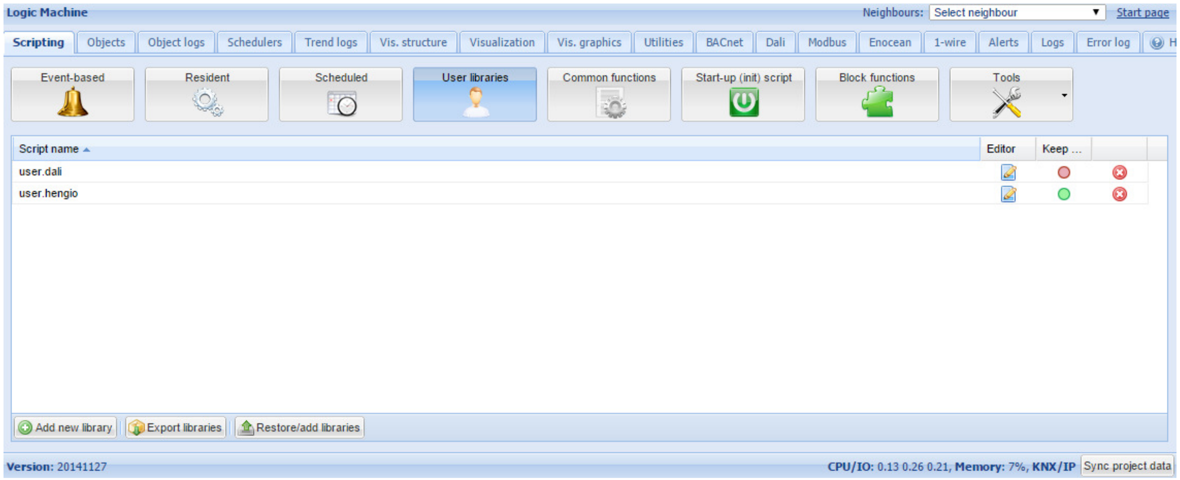
tostring (e)

Receives an argument of any type and converts it to a string in a reasonable format. For complete control of how numbers are converted, use *string.format*.
If the meta table of e has a "__tostring" field, then *tostring* calls the corresponding value with e as argument, and uses the result of the call as its result.

type (v)

Returns the type of its only argument, coded as a string. The possible results of this function are "nil" (astring, not the value *nil*), "number", "string", "boolean", "table", "function", "thread", and "userdata".

1.2.31.User libraries



Script name ▲	Editor	Keep ...	
user.dali	✎	🔴	⊗
user.hengio	✎	🟢	⊗

Version: 20141127

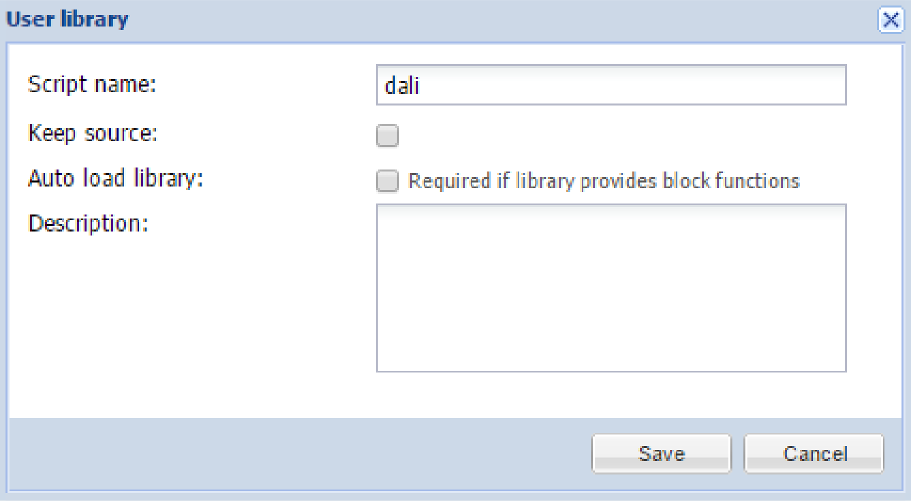
CPU/IO: 0.13 0.26 0.21, Memory: 7%, KNX/IP

User libraries usually contain user defined functions which are later called from other scripts.

You have to include your library in the script with the following command:
require('user.test')unless you have enabled *Auto load library***.**

<u>Secure the code</u>

There is an option *keep source* available for user libraries. Once disabled, the code is compiled in the binary form and can't be seen for further editing. If this option is enabled, the source code is seen in the editor.



User library

Script name: dali

Keep source: □

Auto load library: □ Required if library provides block functions

Description:

Auto load library means that the library will be automatically loaded so you don't have to use **require** when writing scripts. Also this have to be checked if Block programming is used.

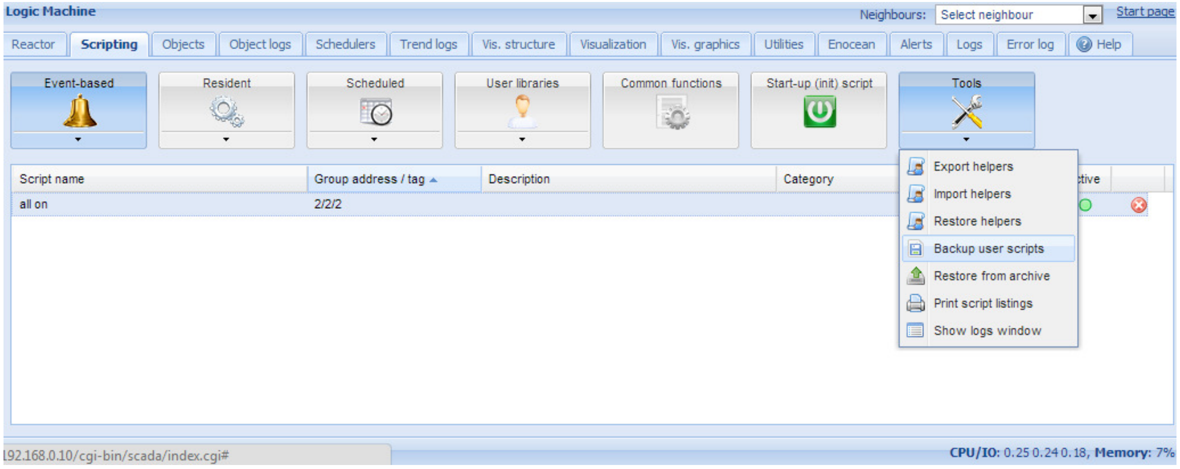
1.2.32.Common functions

Common functions contains library of globally used functions. They can be called from any script, any time, without special including like with *user libraries*. Functions like *sunrise/sunset*, *Email* are included by default in C*ommon function*s.

1.2.33.Start-up (init) script

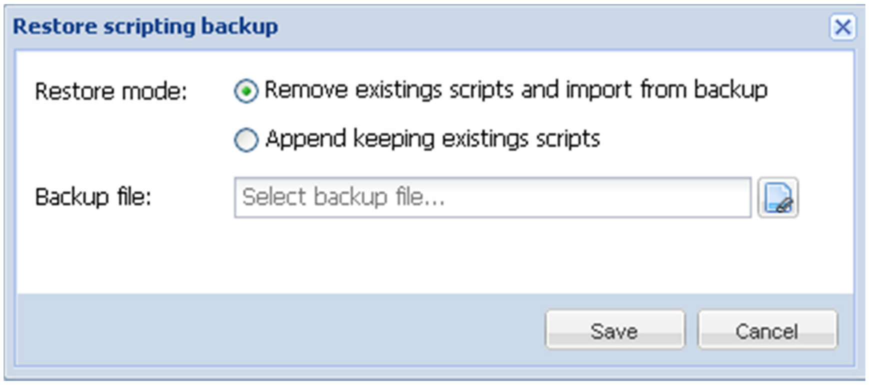
Init script is used for initialization on specific system or bus values on system start. Init script is run each time after system is restarted for some reason.

1.2.34.Tools



Script name	Group address / tag	Description	Category	...tive
all on	2/2/2			

- ➢ ***Export helpers*** – export scripting helpers
- ➢ ***Import helpers*** – import scripting helpers
- ➢ ***Restore helpers*** – restore default scripting helpers
- ➢ ***Backup user scripts*** – backup all scripts in *.gz file
- ➢ ***Restore from archive*** – restore script from archive (*.gz) file with two possibilities:
  - o Remove existing scripts and import from backup
  - o Append keeping existing (s) scripts

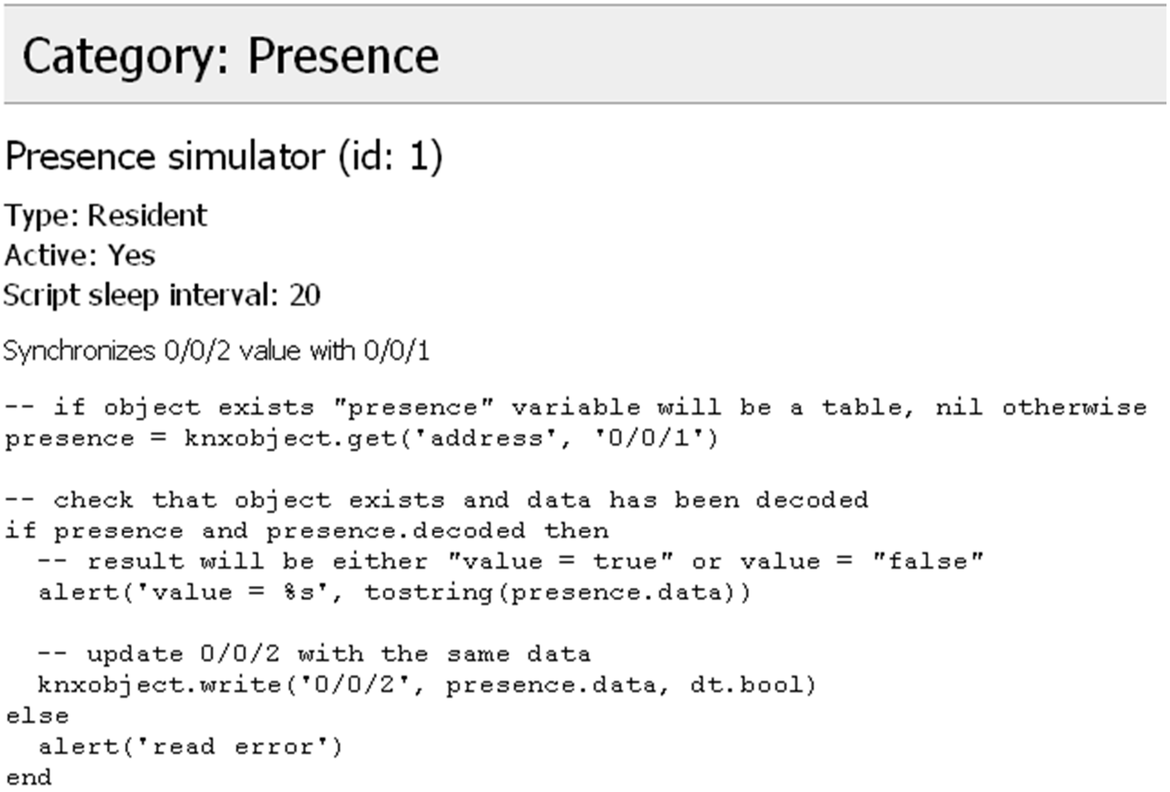


Restore mode: ● Remove existings scripts and import from backup

○ Append keeping existings scripts

Backup file: Select backup file...

- ➢ ***Print script listings*** – shows all scripts with codes in list format sorted by Categories.



Category: Presence

Presence simulator (id: 1)

Type: Resident
Active: Yes
Script sleep interval: 20

Synchronizes 0/0/2 value with 0/0/1

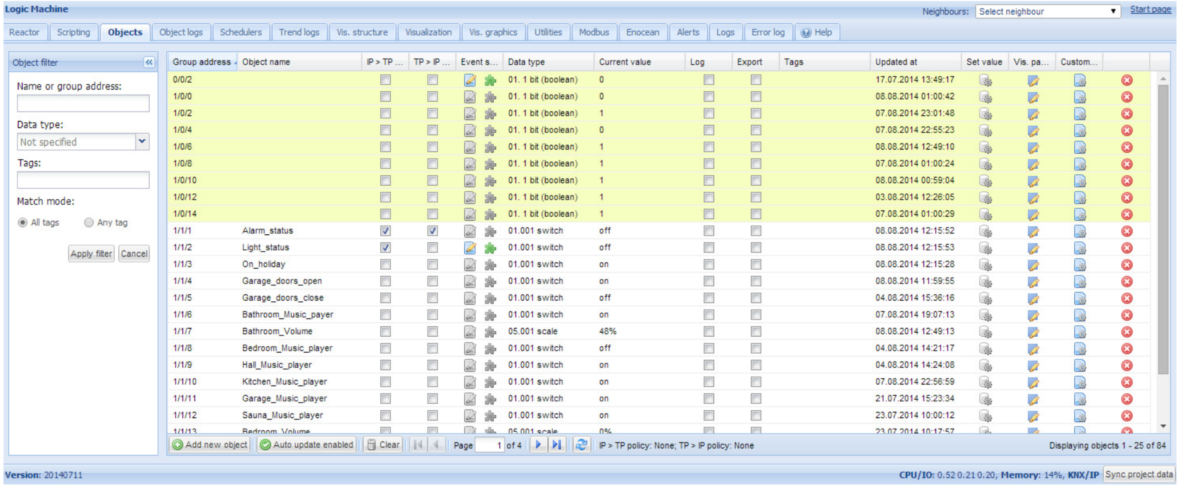
```
-- if object exists "presence" variable will be a table, nil otherwise
presence = knxobject.get('address', '0/0/1')

-- check that object exists and data has been decoded
if presence and presence.decoded then
  -- result will be either "value = true" or value = "false"
  alert('value = %s', tostring(presence.data))

  -- update 0/0/2 with the same data
  knxobject.write('0/0/2', presence.data, dt.bool)
else
  alert('read error')
end
```

- ➢ ***Show logs window*** – show logs in separate window

1.3. Objects

List of KNX network objects appears in *Objects* menu. The object appears in the list by way of:

- sniffing the bus for telegrams from unknown group addresses (if enabled in *Utilities*)
- adding manually
- importing ESF file (in *Utilities*)



Name or group address:

Data type: Not specified

Tags:

Match mode: ● All tags ○ Any tag

Group address	Object name	IP > TP ...	TP > IP ...	Event s...	Data type	Current value	Log	Export	Tags	Updated at	Set value	Vis. pa...	Custom...
0/0/2		☐	☐		01. 1 bit (boolean)	0	☐	☐		17.07.2014 13:49:17			
1/0/0		☐	☐		01. 1 bit (boolean)	0	☐	☐		08.08.2014 01:00:42			
1/0/2		☐	☐		01. 1 bit (boolean)	1	☐	☐		07.08.2014 23:01:48			
1/0/4		☐	☐		01. 1 bit (boolean)	0	☐	☐		07.08.2014 22:55:23			
1/0/6		☐	☐		01. 1 bit (boolean)	1	☐	☐		08.08.2014 12:49:10			
1/0/8		☐	☐		01. 1 bit (boolean)	1	☐	☐		07.08.2014 01:00:24			
1/0/10		☐	☐		01. 1 bit (boolean)	1	☐	☐		08.08.2014 00:59:04			
1/0/12		☐	☐		01. 1 bit (boolean)	1	☐	☐		03.08.2014 12:26:05			
1/0/14		☐	☐		01. 1 bit (boolean)	1	☐	☐		07.08.2014 01:00:29			
1/1/1	Alarm_status	☑	☑		01.001 switch	off	☐	☐		08.08.2014 12:15:52			
1/1/2	Light_status	☑	☐		01.001 switch	off	☐	☐		08.08.2014 12:15:53			
1/1/3	On_holiday	☐	☐		01.001 switch	on	☐	☐		08.08.2014 12:15:28			
1/1/4	Garage_doors_open	☐	☐		01.001 switch	on	☐	☐		08.08.2014 11:59:55			
1/1/5	Garage_doors_close	☐	☐		01.001 switch	off	☐	☐		04.08.2014 15:36:16			
1/1/6	Bathroom_Music_payer	☐	☐		01.001 switch	on	☐	☐		07.08.2014 19:07:13			
1/1/7	Bathroom_Volume	☐	☐		05.001 scale	48%	☐	☐		08.08.2014 12:49:13			
1/1/8	Bedroom_Music_player	☐	☐		01.001 switch	off	☐	☐		04.08.2014 14:21:17			
1/1/9	Hall_Music_player	☐	☐		01.001 switch	on	☐	☐		04.08.2014 14:24:08			
1/1/10	Kitchen_Music_player	☐	☐		01.001 switch	on	☐	☐		07.08.2014 22:56:59			
1/1/11	Garage_Music_player	☐	☐		01.001 switch	on	☐	☐		21.07.2014 15:23:34			
1/1/12	Sauna_Music_player	☐	☐		01.001 switch	on	☐	☐		23.07.2014 10:00:12			
1/1/13	Bedroom_Volume	☐	☐		05.001 scale	0%	☐	☐		23.07.2014 10:17:57			

Page 1 of 4

IP > TP policy: None; TP > IP policy: None

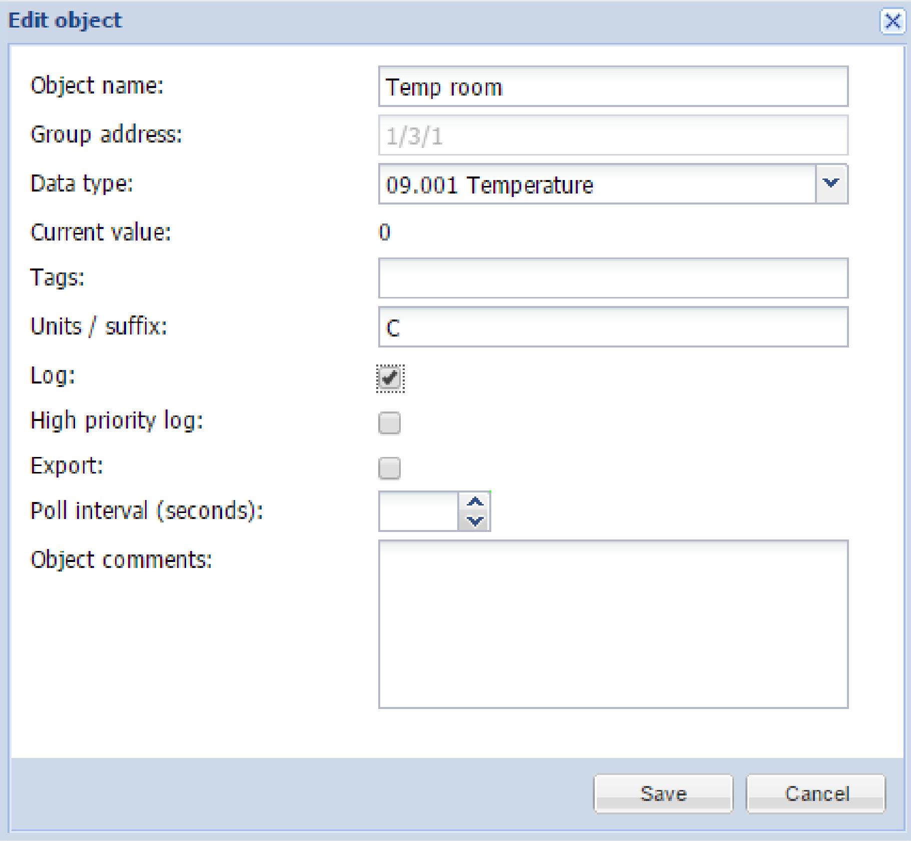
Displaying objects 1 - 25 of 84

Version: 20140711

CPU/IO: 0.52 0.21 0.20, **Memory:** 14%, **KNX/IP**

1.3.1. Object parameters

To change the settings for existing or new objects, press on the specific list entry.



Edit object

Object name: Temp room

Group address: 1/3/1

Data type: 09.001 Temperature

Current value: 0

Tags:

Units / suffix: C

Log: ☑

High priority log: □

Export: □

Poll interval (seconds):

Object comments:

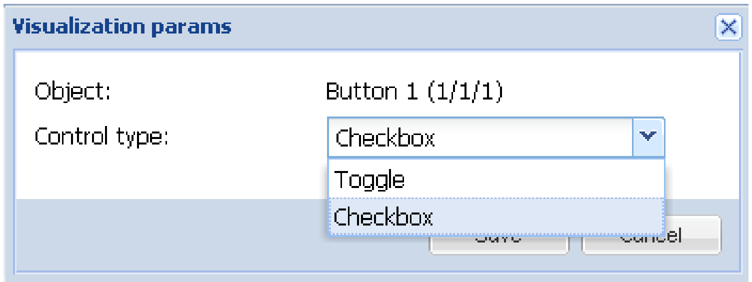
- ***Object name*** – Name for the object
- ***Group address*** – Group address of this object
- ***Data type*** – KNX data type for the object. This has to be set once the LM sniffs the new object for proper work.
- ***Units / suffix*** – units for the object which will appear on the visualization along with the value
- ***Log*** – enable logging for this object. Logs will appear in *Objects logs* menu.
- ***High priority log*** – mark the object for high priority logging; when the log database is cleared, first standard logs are cleared, only then high priority
- ***Export*** – Make object visible by remote XML requests and in BACnet network (if KNX – BACnet gateway functionality is used)
- ***Poll interval (seconds)*** – perform automatic object read after some time interval
- ***Tags*** – assign this object to some tag which can be later used in writing scripts, for example, *All_lights_first_floor*.
- ***Current value***– Current value of the object
- ***Object comments*** – Comment for the object

There is a possibility to sort the objects by one of the following – Name, Group address, Data type, Current value, Tags, Comments

1.3.2. *Object visualization parameters*

By pressing on the button of the corresponding object you can set specific visualization parameters for this type of object.

1 bit

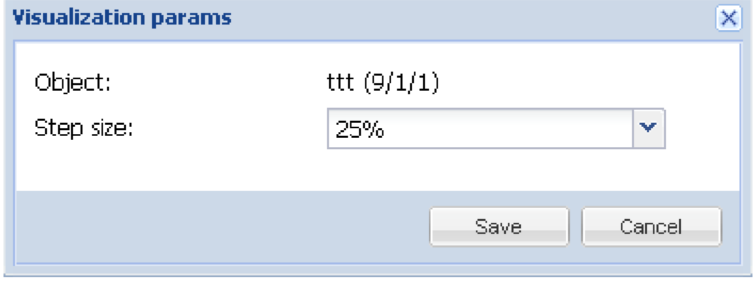

Visualization params

Object: Button 1 (1/1/1)

Control type: Checkbox

- ***Control type*** – type of the visual control element
  - Toggle
  - Checkbox

4 bit (3 bit controlled)


Visualization params

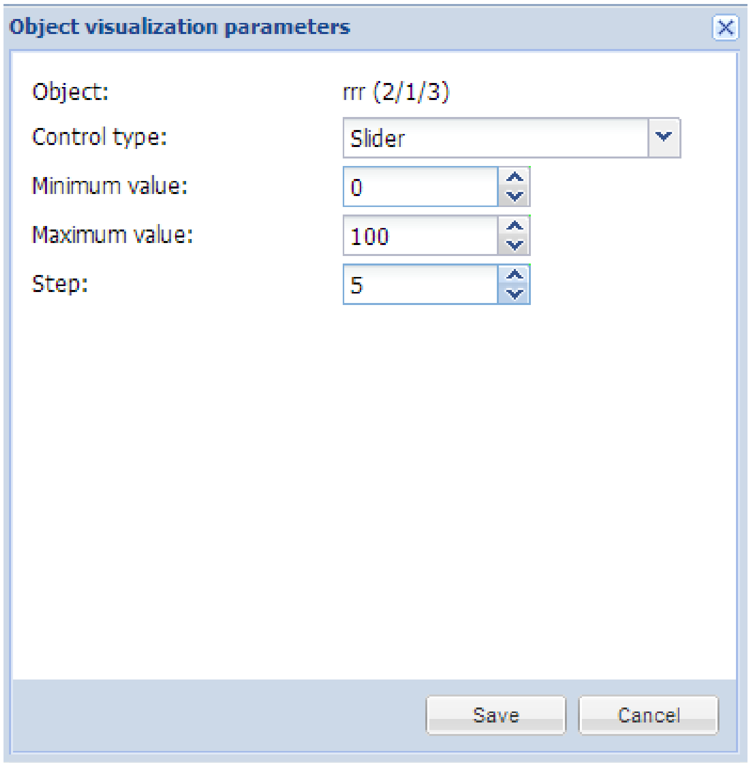
Object: ttt (9/1/1)

Step size: 25%

- ***Step size*** – step size for example for blinds control

2 bit (1 bit controlled), 1 byte unsigned integer (scale), 1 byte signed integer, 2 byte unsigned integer, 2 byte signed integer, 2 byte floating point (temperature), 4 byte unsigned integer, 4 byte signed integer, 4 byte floating point

1byte



Object visualization parameters

Object: rrr (2/1/3)

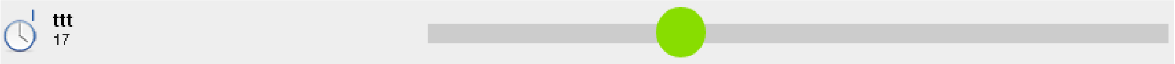
Control type: Slider

Minimum value: 0

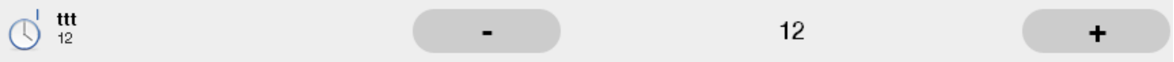
Maximum value: 100

Step: 5

- Control type – type of the visual control element
    - Slider



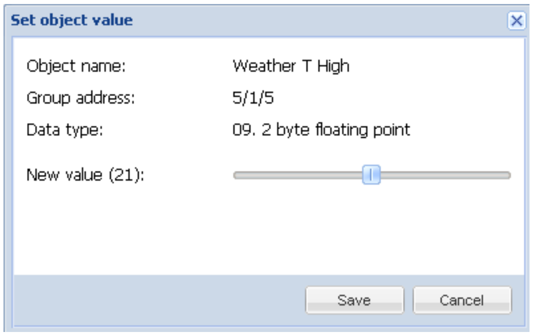
    - Direct input / Step +/-



- Minimum value
- Maximum value

1.3.3. Change the object state

In the object list, by pressing on the button, you can change the state of the object. The appearance of the *New value* depends on what visualization parameters are set for specific object.



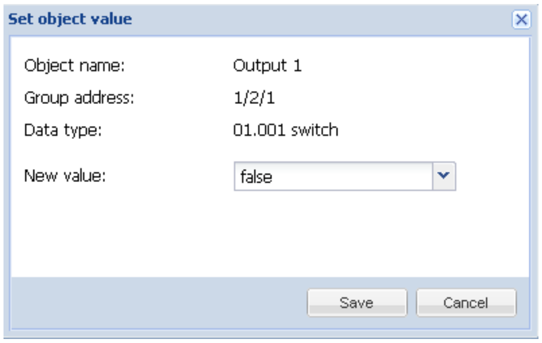
Set object value

Object name: Weather T High

Group address: 5/1/5

Data type: 09. 2 byte floating point

New value (21):



Set object value

Object name: Output 1

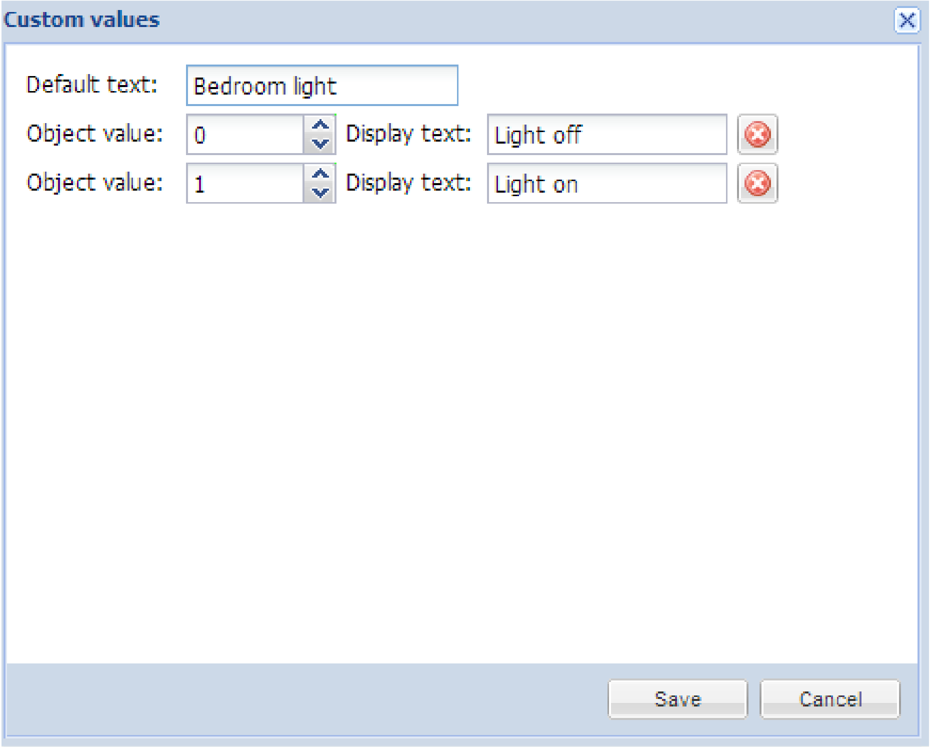
Group address: 1/2/1

Data type: 01.001 switch

New value: false

1.3.4. Custom values

If special value naming is necessary, use this icon to set it up (only for Boolean and Integer data types)



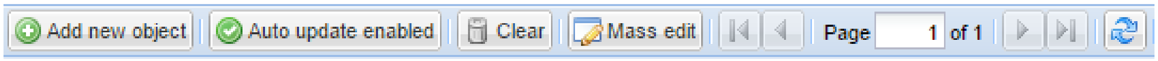
Custom values

Default text: Bedroom light

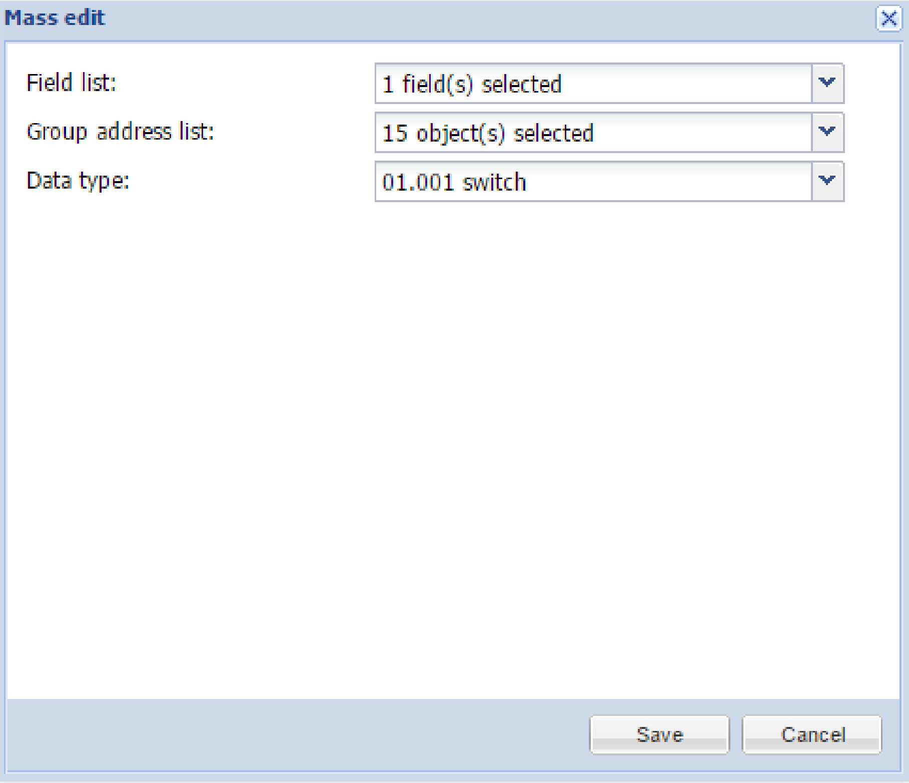
Object value: 0 Display text: Light off

Object value: 1 Display text: Light on

1.3.5. Object control bar



- ➢ ***Add new object*** – Manually add new object to the list
- ➢ ***Auto update enabled*** –Specifies either the object list is updated automatically or not
- ➢ ***Clear*** – Clear the list of group addresses
- ➢ ***Next/Previous page*** – move to next or previous page
- ➢ ***Refresh*** – refresh the object list
- ➢ ***Mass edit*** – mass edit objects by a specific criteria

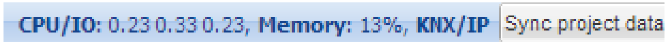


Field list: 1 field(s) selected

Group address list: 15 object(s) selected

Data type: 01.001 switch

There is also the following bar on the bottom of the configuration screen:

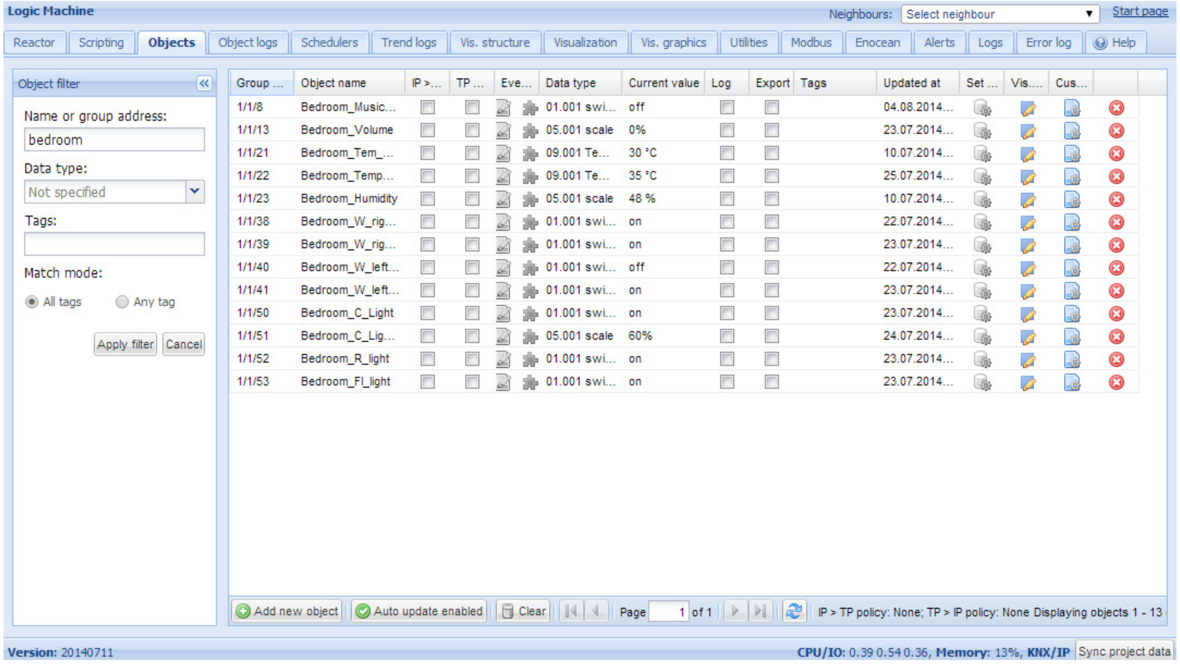


CPU/IO: 0.23 0.33 0.23, Memory: 13%, KNX/IP

- ***CPU/IO*** –Load average. The load average represents the average system load over a period of time. It conventionally appears in the form of three numbers which represent the system load during the last one-, five-, and fifteen-minute periods. More on UNIX style load calculation can be found here: http://en.wikipedia.org/wiki/Load_(computing)#Unix-style_load_calculation
- ***Memory*** – memory usage in %
- ***KNX/IP / KNX/TP*** – type of connection to KNX bus. If KNX/TP is set and it is not available, there will be error notification
- ***Sync project data*** – save all project data to internal flash by pressing this button. Otherwise the data is saved once in 30 minutes from RAM to Flash, or when Reboot or Shutdown commands are sent

1.3.6. Filter objects

On the left side of the object list there is filtering possible. To perform the filtering type the name, group address, tag or specify the data type of the object and press on *Filter* button.



Name or group address: bedroom

Data type: Not specified

Tags:

Match mode: ● All tags ○ Any tag

Group ...	Object name	IP >...	TP ...	Eve...	Data type	Current value	Log	Export	Tags	Updated at
1/1/8	Bedroom_Music...	□	□		01.001 swi...	off	□	□		04.08.2014...
1/1/13	Bedroom_Volume	□	□		05.001 scale	0%	□	□		23.07.2014...
1/1/21	Bedroom_Tem_...	□	□		09.001 Te...	30 °C	□	□		10.07.2014...
1/1/22	Bedroom_Temp...	□	□		09.001 Te...	35 °C	□	□		25.07.2014...
1/1/23	Bedroom_Humidity	□	□		05.001 scale	48 %	□	□		10.07.2014...
1/1/38	Bedroom_W_rig...	□	□		01.001 swi...	on	□	□		22.07.2014...
1/1/39	Bedroom_W_rig...	□	□		01.001 swi...	on	□	□		23.07.2014...
1/1/40	Bedroom_W_left...	□	□		01.001 swi...	off	□	□		22.07.2014...
1/1/41	Bedroom_W_left...	□	□		01.001 swi...	on	□	□		23.07.2014...
1/1/50	Bedroom_C_Light	□	□		01.001 swi...	on	□	□		23.07.2014...
1/1/51	Bedroom_C_Lig...	□	□		05.001 scale	60%	□	□		24.07.2014...
1/1/52	Bedroom_R_light	□	□		01.001 swi...	on	□	□		23.07.2014...
1/1/53	Bedroom_Fl_light	□	□		01.001 swi...	on	□	□		23.07.2014...

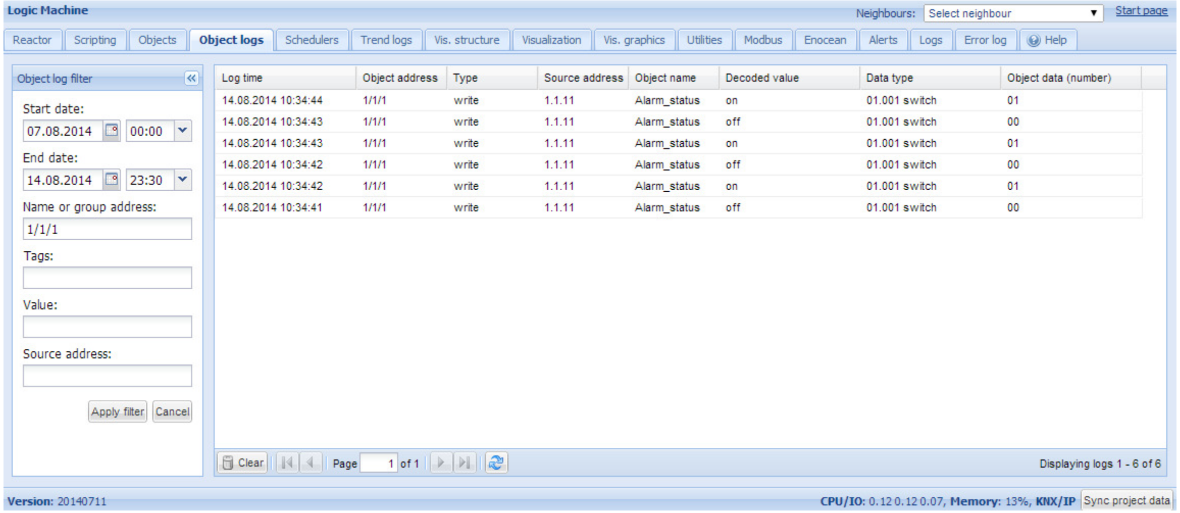
IP > TP policy: None; TP > IP policy: None Displaying objects 1 - 13

Match mode:

All tags – represents AND function when all tags should match
Any tag – represents OR function when any one of listed should match

1.4. Object logs

Object historical telegrams are available in *Object logs*. Once logging is enabled for object, all it's further history will be logged.



Start date: 07.08.2014 00:00

End date: 14.08.2014 23:30

Name or group address: 1/1/1

Tags:

Value:

Source address:

Log time	Object address	Type	Source address	Object name	Decoded value	Data type	Object data (number)
14.08.2014 10:34:44	1/1/1	write	1.1.11	Alarm_status	on	01.001 switch	01
14.08.2014 10:34:43	1/1/1	write	1.1.11	Alarm_status	off	01.001 switch	00
14.08.2014 10:34:43	1/1/1	write	1.1.11	Alarm_status	on	01.001 switch	01
14.08.2014 10:34:42	1/1/1	write	1.1.11	Alarm_status	off	01.001 switch	00
14.08.2014 10:34:42	1/1/1	write	1.1.11	Alarm_status	on	01.001 switch	01
14.08.2014 10:34:41	1/1/1	write	1.1.11	Alarm_status	off	01.001 switch	00

Displaying logs 1 - 6 of 6

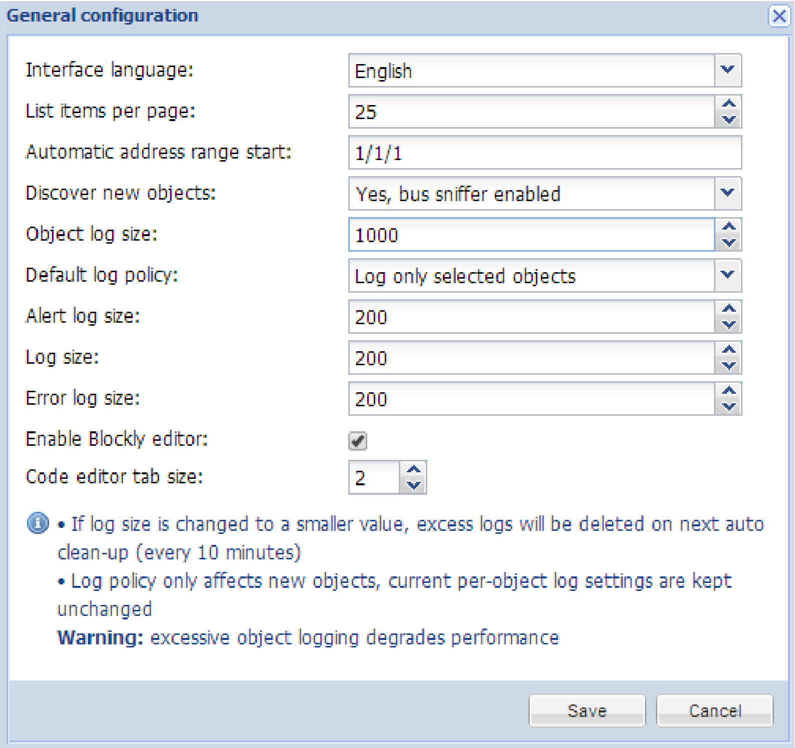
Filtering is available when there is a need to find specific period information

- ***Start date*** – start date and time for log filtering
- ***End date*** – start date and time for log filtering
- ***Name or group address*** – specific name or group address of object

- ***Tags*** – tag names
- ***Value*** – specific object value
- ***Source address*** – specific source address

You can clear all logs by pressing on *Clear* button.

Size of log is defined *in Utilities →General Configuration*



General configuration

Interface language:	English
List items per page:	25
Automatic address range start:	1/1/1
Discover new objects:	Yes, bus sniffer enabled
Object log size:	1000
Default log policy:	Log only selected objects
Alert log size:	200
Log size:	200
Error log size:	200
Enable Blockly editor:	☑
Code editor tab size:	2

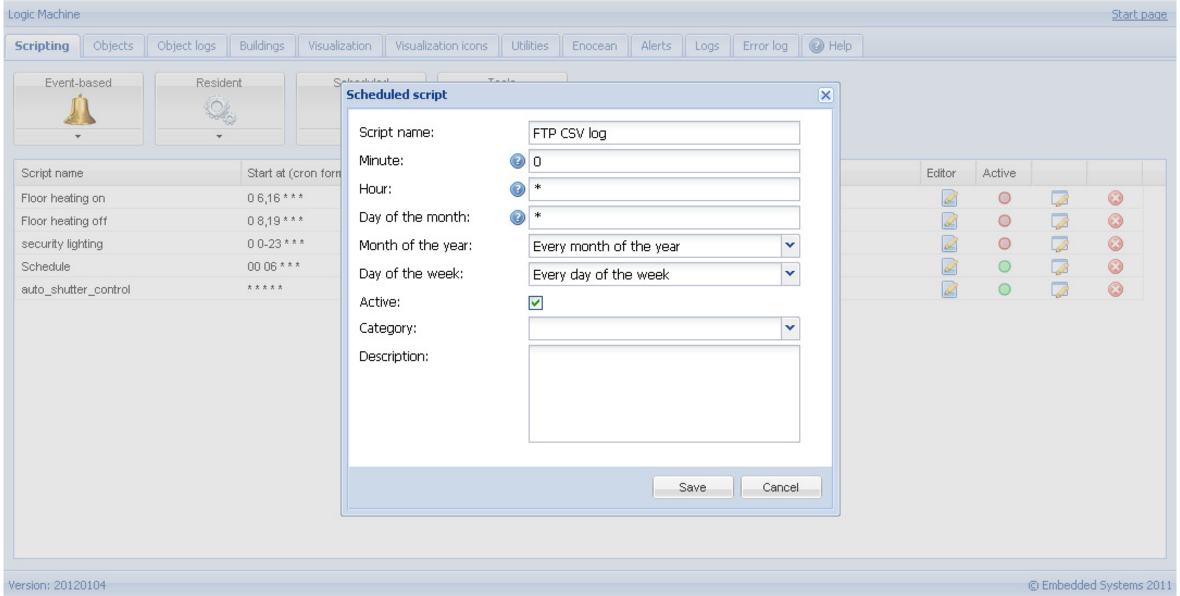
• If log size is changed to a smaller value, excess logs will be deleted on next auto clean-up (every 10 minutes)
• Log policy only affects new objects, current per-object log settings are kept unchanged
Warning: excessive object logging degrades performance

1.4.1. Export logs

<u>Example</u>

Once an hour, make CSV file with all objects logs and send to external FTP server with IP 192.168.1.11, login ‘ftplogin‘, password ‘ftppassword‘.

- In *Scripting* -> *Scheduled* add the script which will run once an hour



Script name	Start at (cron form
Floor heating on	0 6,16 * * *
Floor heating off	0 8,19 * * *
security lighting	0 0-23 * * *
Schedule	00 06 * * *
auto_shutter_control	* * * * *

Scheduled script

Script name:	FTP CSV log
Minute:	0
Hour:	*
Day of the month:	*
Month of the year:	Every month of the year
Day of the week:	Every day of the week
Active:	☑
Category:	
Description:	

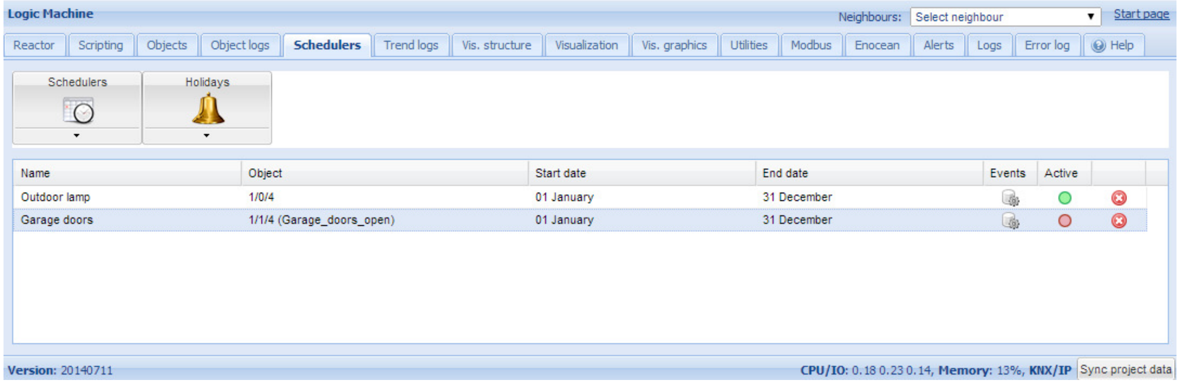
- Add the following code in Script editor for this particular script.

```
require('socket.ftp')

-- ftp file
ftpfile=string.format('ftp://ftplogin:ftppassword@192.168.1.11/%s.csv', os.date('%Y-%m-%d_%H-%M'))
-- get past hour data (3600 seconds)
logtime=os.time() - 60*60

-- list of objects by id
objects ={}

-- objects with logging enabled
query ='SELECT address, datatype, name FROM objects WHERE disablelog=0'
for _, object inipairs(db:getall(query))do
  objects[tonumber(object.address)]={
datatype=tonumber(object.datatype),
    name =tostring(object.name or''),
}
end

-- csv buffer
buffer ={'"date","address","name","value"'}

-- get object logs
query='SELECT src, address, datahex, logtime, eventtype FROM objectlog WHERE logtime>= ? ORDER BY id DESC'
for _, row inipairs(db:getall(query, logtime))do
  object = objects[tonumber(row.address)]

```

```
-- found matching object and event type is group write
if object androw.eventtype=='write'then
datatype=object.datatype

-- check that object datatype is set
ifdatatypethen
-- decode data
      data =knxdatatype.decode(row.datahex, datatype)

-- remove null chars from char/string datatype
ifdatatype==dt.charordatatype==dt.stringthen
        data =data:gsub('%z+', '')
-- date to DD.MM.YYYY
elseifdatatype==dt.datethen
        data =string.format('%.2d.%.2d.%.2d', data.day, data.month, data.year)
-- time to HH:MM:SS
elseifdatatype==dt.timethen
        data =string.format('%.2d:%.2d:%.2d', data.hour, data.minute,
data.second)
end
else
      data =''
end

-- format csv row
logdate=os.date('%Y.%m.%d %H:%M:%S', row.logtime)
csv=string.format('%q,%q,%q,%q', logdate, knxlib.decodega(row.address),
object.name, tostring(data))

-- add to buffer
table.insert(buffer, csv)
end
end

-- upload to ftp only when there's data in buffer
if #buffer > 1 then
  result, err =socket.ftp.put(ftpfile, table.concat(buffer, '\r\n'))
end

-- error while uploading
if err then
  alert('FTP upload failed: %s', err)
end
```

1.5. Schedulers

Schedulers contain administration of user mode schedulers. Schedulers allow for end user to control KNX group address values based on the date or day of the week.

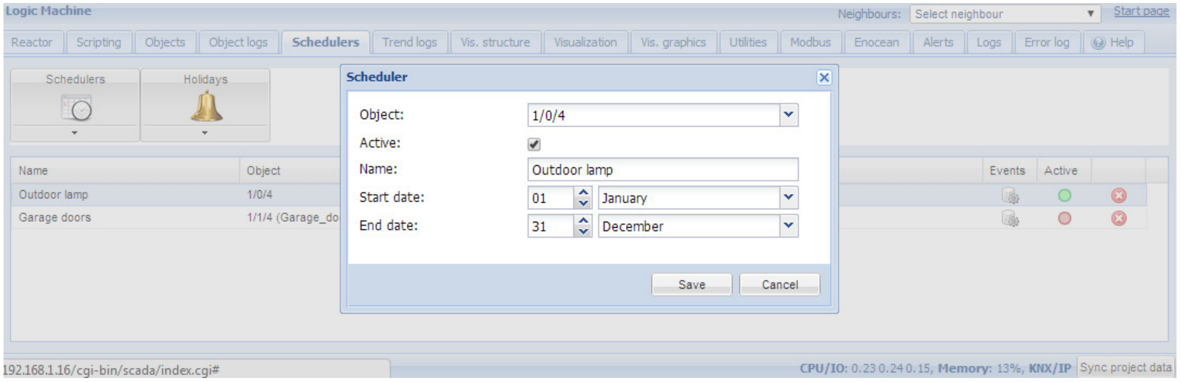


Name	Object	Start date	End date	Events	Active
Outdoor lamp	1/0/4	01 January	31 December		●
Garage doors	1/1/4 (Garage_doors_open)	01 January	31 December		●

Version: 20140711
CPU/IO: 0.18 0.23 0.14, Memory: 13%, KNX/IP

1.5.1. Add new scheduler

By clicking on the Schedulers → Add new scheduler you will see such parameter window:



Scheduler

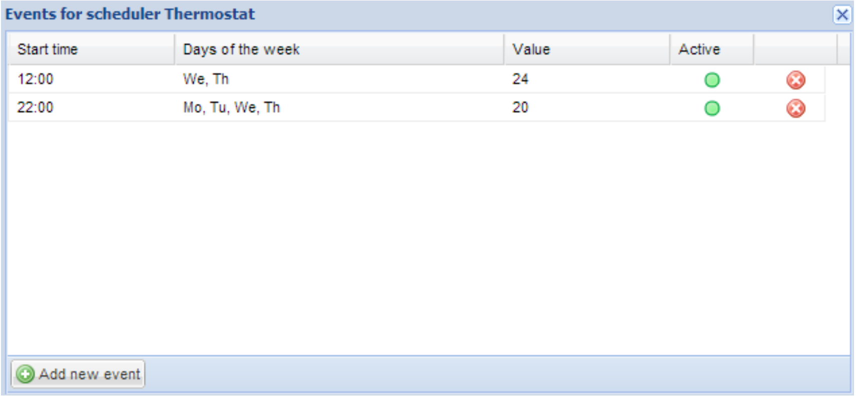
Object: 1/0/4
Active: ☑
Name: Outdoor lamp
Start date: 01 January
End date: 31 December

192.168.1.16/cgi-bin/scada/index.cgi#
CPU/IO: 0.23 0.24 0.15, Memory: 13%, KNX/IP

- ➢ ***Object*** – the object group address which will be controlled by scheduler
- ➢ ***Active*** – define this scheduler as active or not
- ➢ ***Name*** – name of the scheduler
- ➢ ***Start date*** – start date of the scheduler
- ➢ ***End date*** – end date of the scheduler

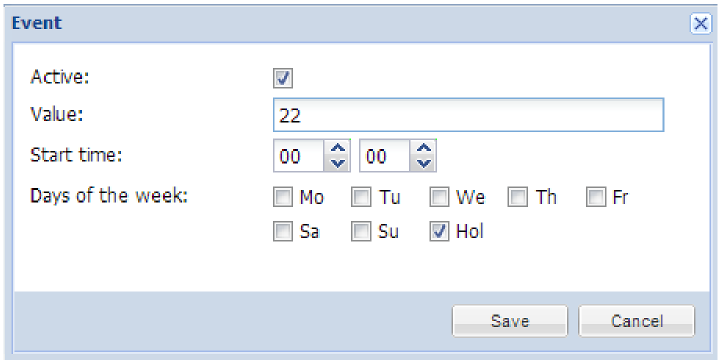
1.5.2. Scheduler events

Event can be added both in administrator interface as well as by end user in the special *User mode schedulers* interface.



Events for scheduler Thermostat

Start time	Days of the week	Value	Active	
12:00	We, Th	24	●	⊗
22:00	Mo, Tu, We, Th	20	●	⊗



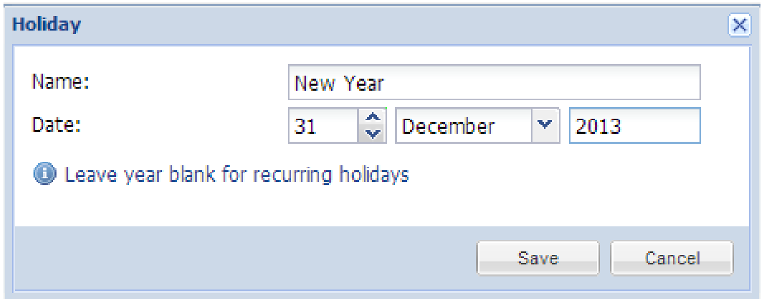
Event

Active:	☑
Value:	22
Start time:	00 : 00
Days of the week:	□ Mo □ Tu □ We □ Th □ Fr □ Sa □ Su ☑ Hol

Active – define the event active or not
Value – value to send to the group address when the event will be triggered
Start time – start time for the event
Days of the week – days of the week when the event will be triggered.
Hol– holidays which are defined in *Holidays* tab

1.5.3. Scheduler holidays

Once the event will be marked to run in *Hol*, Holiday entries will be activated.



Holiday

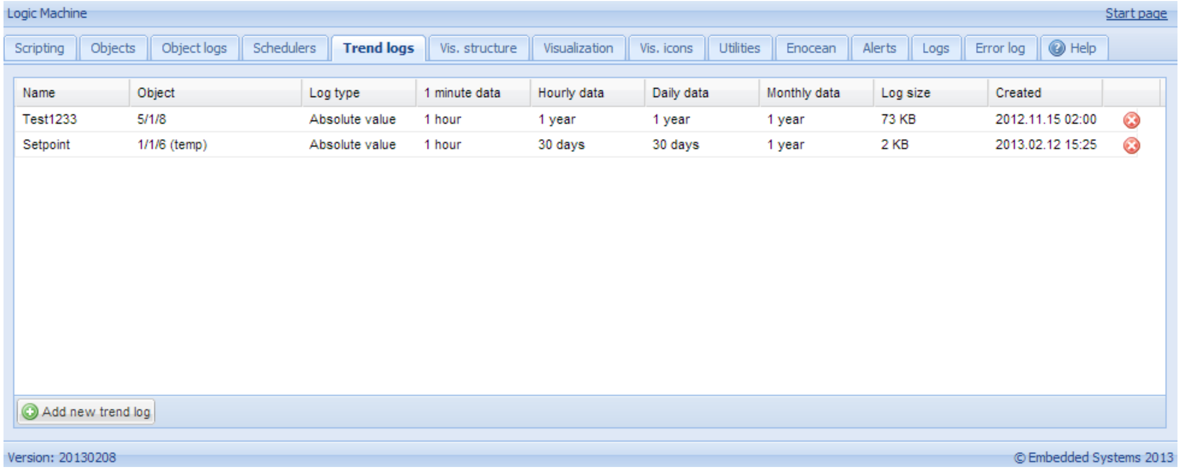
Name:	New Year
Date:	31 December 2013

Leave year blank for recurring holidays

Name – the name of the holiday entry
Date – date of the holiday

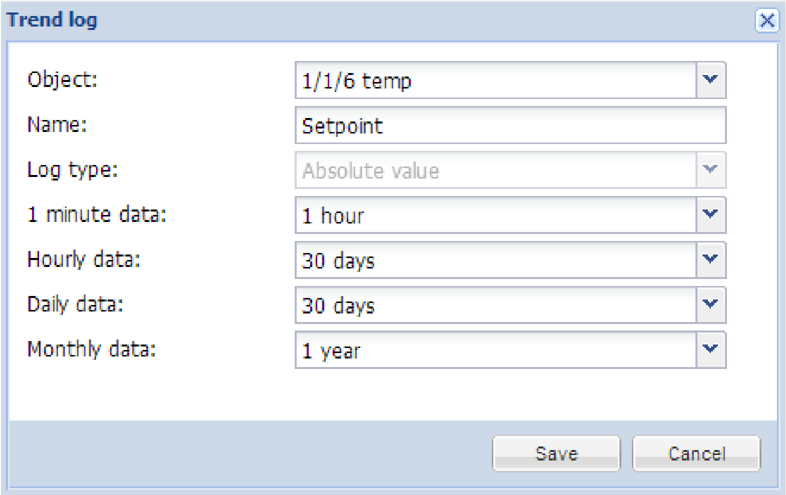
1.6. Trend logs

Trends logs are administration of user mode trends, used to see historical object graphical values, compare with other period values.



Name	Object	Log type	1 minute data	Hourly data	Daily data	Monthly data	Log size	Created
Test1233	5/1/8	Absolute value	1 hour	1 year	1 year	1 year	73 KB	2012.11.15 02:00
Setpoint	1/1/6 (temp)	Absolute value	1 hour	30 days	30 days	1 year	2 KB	2013.02.12 15:25

1.6.1. Add new trend log



Object:	1/1/6 temp
Name:	Setpoint
Log type:	Absolute value
1 minute data:	1 hour
Hourly data:	30 days
Daily data:	30 days
Monthly data:	1 year

Object – choose from list of object the one to make trends for
Name – name of the trend
Log type [Counter, Absolute value] – type of the log. *Counter* type is used to count the date, *Absolute value* – saves the actual readings
1 minute data – average value of 1 minute for specific time interval data will be shown on the trend. E.g. if 1 hour – trend step will be 1 hour with average 60 readings data
Hourly data – average value of hourly data for specific time interval

Daily data –average value of daily data for specific time interval
Monthly data – average value of monthly data for specific time interval

Note! One trend data point reading takes *8bytes* of flash memory. E.g. reading some value once in every 10 minutes, will consume ~0.4MB of flash each year.

1.6.2. Trend logs functions

To process logged information in trends, you can use built in trend log functions from scripting.

Include library before calling trend log functions:

require('genohm-scada.trends')

Fetch one or many values for the given period:

trends.fetch(name, mode, period)
trends.fetchone(name, mode, period)

Parameters:

- ***name*** – trend log name, required
- ***mode*** – either 'day', 'month' or 'year', required
- ***period*** – optional, will use current date if not specified
  If specified, must be a Lua table with the following fields:
  - *day* – required for day mode only
  - *month* – required for day and month modes
  - *year* – required for all modes

Return values:

fetch returns Lua table with values for the given period or nil on error. Number of values in the table depends on period and log retention settings. For example, in month mode this function can return values for each day or only a single value for the whole month

fetchone returns single value for the given period or nil on error

Example:

```
require('genohm-scada.trends')


-- fetch current value

today = trends.fetchone('Gas', 'day')


-- get current date as table and set day to yesterday
```

```
date = os.date('*t')
date.day = date.day - 1

-- fetch previous value
yesterday = trends.fetchone('Gas', 'day', date)
```

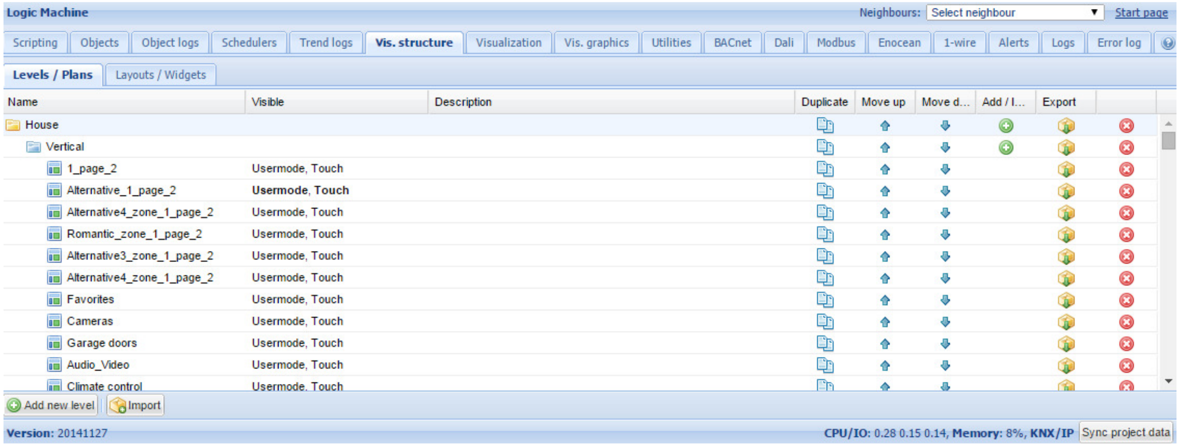
trends.NaN value is used for points which contain invalid values or cannot be found. The default value is 0, but it can also be set to 0 / 0 (NaN - not a number).

Example:

```
require('genohm-scada.trends')

-- use not a number for invalid value
trends.NaN = 0 / 0

-- get total hot water usage for year 2011
value = trends.fetchone('Hot Water', 'year', { year = 2011 })

-- NaN ~= NaN, means value was not found
if value ~= value then
  return
end
```

1.7. Visualization structure

In *Vis.structure* menu the structure of the visualization is defined and visualization backgrounds are uploaded.



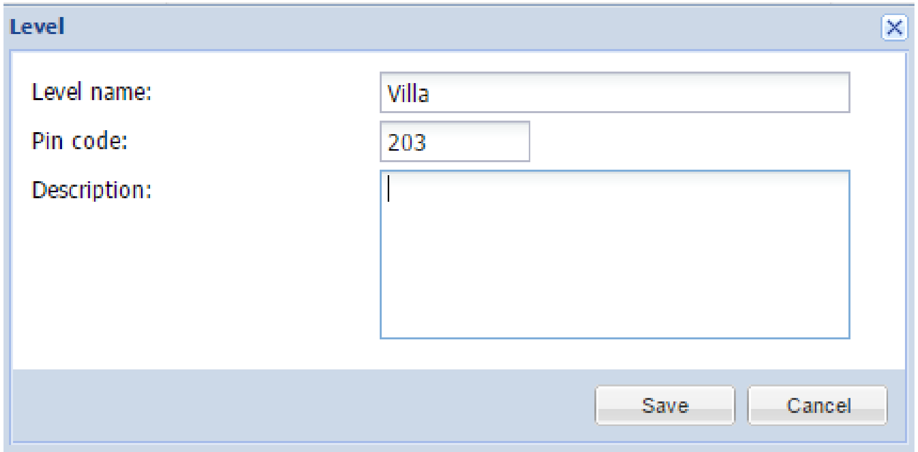
Name	Visible	Description	Duplicate	Move up	Move d...	Add / I...	Export	
House			⧉	↑	↓	⊕	▣	⊗
Vertical			⧉	↑	↓	⊕	▣	⊗
1_page_2	Usermode, Touch		⧉	↑	↓		▣	⊗
Alternative_1_page_2	**Usermode, Touch**		⧉	↑	↓		▣	⊗
Alternative4_zone_1_page_2	Usermode, Touch		⧉	↑	↓		▣	⊗
Romantic_zone_1_page_2	Usermode, Touch		⧉	↑	↓		▣	⊗
Alternative3_zone_1_page_2	Usermode, Touch		⧉	↑	↓		▣	⊗
Alternative4_zone_1_page_2	Usermode, Touch		⧉	↑	↓		▣	⊗
Favorites	Usermode, Touch		⧉	↑	↓		▣	⊗
Cameras	Usermode, Touch		⧉	↑	↓		▣	⊗
Garage doors	Usermode, Touch		⧉	↑	↓		▣	⊗
Audio_Video	Usermode, Touch		⧉	↑	↓		▣	⊗
Climate control	Usermode, Touch		⧉	↑	↓		▣	⊗

Version: 20141127

CPU/IO: 0.28 0.15 0.14, **Memory:** 8%, **KNX/IP**

1.7.1. Levels / Plans

By default there is *Main* level added. To add a new level/building, press "*Add new level*" button. Please note that you can limit access to this specific level by adding PIN code.

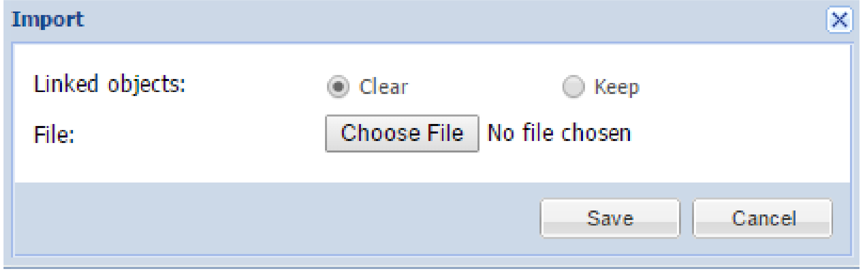


Level name: Villa

Pin code: 203

Description:

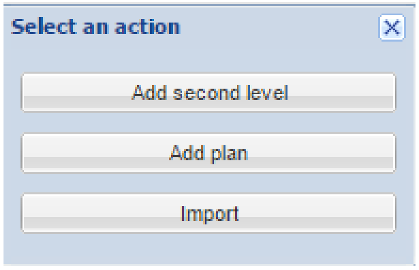
You can also add a new level by importing it from the file (which is exported on other LM for example). Press *Import* button for this purpose. Object linkage can be either cleared or imported as-is.


Import

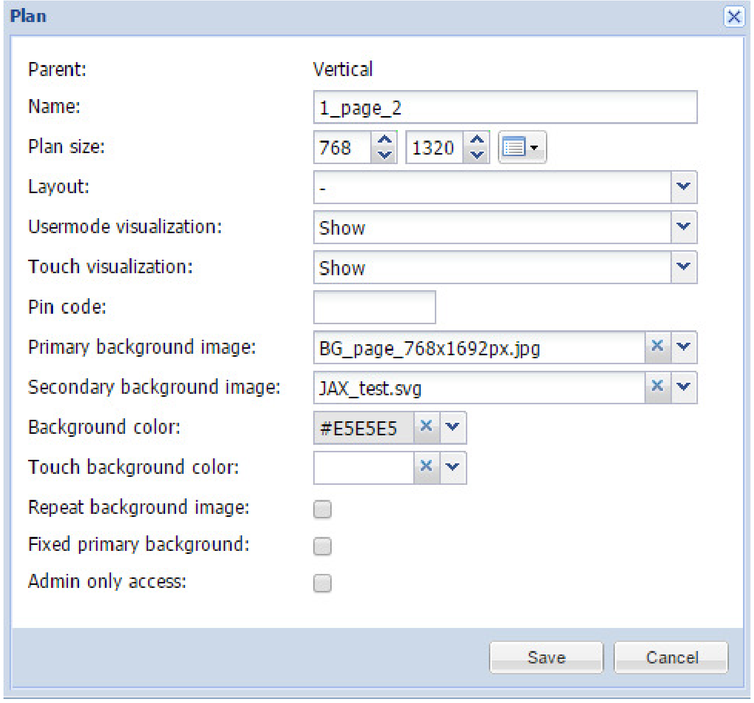
Linked objects: ● Clear ○ Keep

File: Choose File No file chosen

Once a new level is added, you can add second level or upload floor pictures related to this particular building. To add a new entry, click on the green icon ⊕, to delete a specific entry press on the red icon ⊗.



When adding new plan, the following parameters should be defined:


Plan

Parent: Vertical

Name: 1_page_2

Plan size: 768 1320

Layout: -

Usermode visualization: Show

Touch visualization: Show

Pin code:

Primary background image: BG_page_768x1692px.jpg

Secondary background image: JAX_test.svg

Background color: #E5E5E5

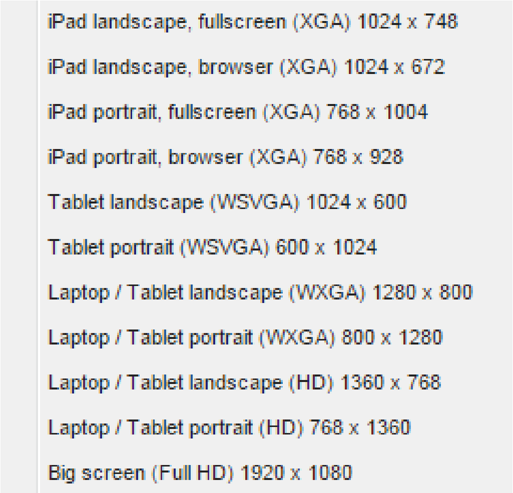
Touch background color:

Repeat background image: □

Fixed primary background: □

Admin only access: □

- ***Parent*** – name of parent level
- ***Name*** – name for the plan
- ***Plan size*** – plan size in pixels. There are predefined resolutions available when clicking on the icon on the right size of this parameter:



iPad landscape, fullscreen (XGA) 1024 x 748

iPad landscape, browser (XGA) 1024 x 672

iPad portrait, fullscreen (XGA) 768 x 1004

iPad portrait, browser (XGA) 768 x 928

Tablet landscape (WSVGA) 1024 x 600

Tablet portrait (WSVGA) 600 x 1024

Laptop / Tablet landscape (WXGA) 1280 x 800

Laptop / Tablet portrait (WXGA) 800 x 1280

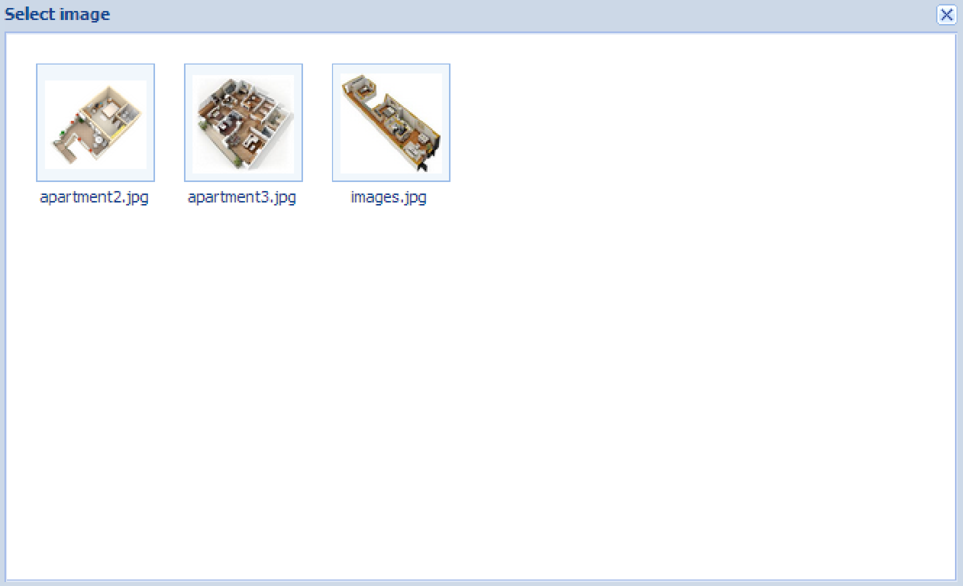
Laptop / Tablet landscape (HD) 1360 x 768

Laptop / Tablet portrait (HD) 768 x 1360

Big screen (Full HD) 1920 x 1080

- ***Layout*** – layout for this specific plan. All object from Layout will be duplicated on this particular plan including background color and plan image if they are not defined separately for this specific plan
- ***Usermode visualization [Show, Show and make default, Hide]*** – visibility for this particular plan in Usermode visualization
- ***Touch visualization [Show, Show and make default, Hide]***– visibility for this particular plan in Touch visualization
- ***PIN code*** – specify PIN code to access the plan
- ***Primary background image*** – choose primary background image from the list added in *Vis.graphics* → *Images/Backgrounds*
- ***Secondary background image*** – choose secondary background image from the list added in *Vis.graphics* → *Images/Backgrounds*
- ***Background color*** – choose background color of the plan
- ***Touch background color –*** define a color for touch visualization
- ***Repeat background image*** – either to show the image once or repeat it and fill the whole plan
- ***Fixed primary background*** – specify if first background image should be fixed. By enabling this, you can enable Parallax effect for your visualization
- ***Admin only access*** – enable admin only access for this floor

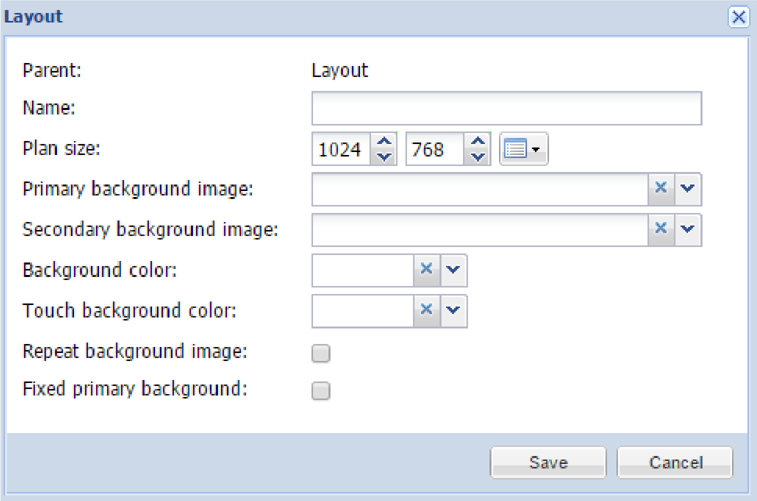
When clicking on Background image, the following window appears with background images which has to be added in *Vis.graphics* → *Images/Backgrounds*in advance:



You can duplicate the plan with all its objects and settings by pressing on icon.
Levels can be sorted by pressing ⬆ and ⬇ icons. You can export the plan structure by clicking in this icon

1.7.2. Layouts / Widgets

Layouts are used as templates for further use when adding *Levels* in *Levels/Plans* tab.
Layouts will not be visible from the Usermode/Touch visualizations. When you add any background, objects to layouts level in *Visualization*, they will automatically appear on all linked Levels.

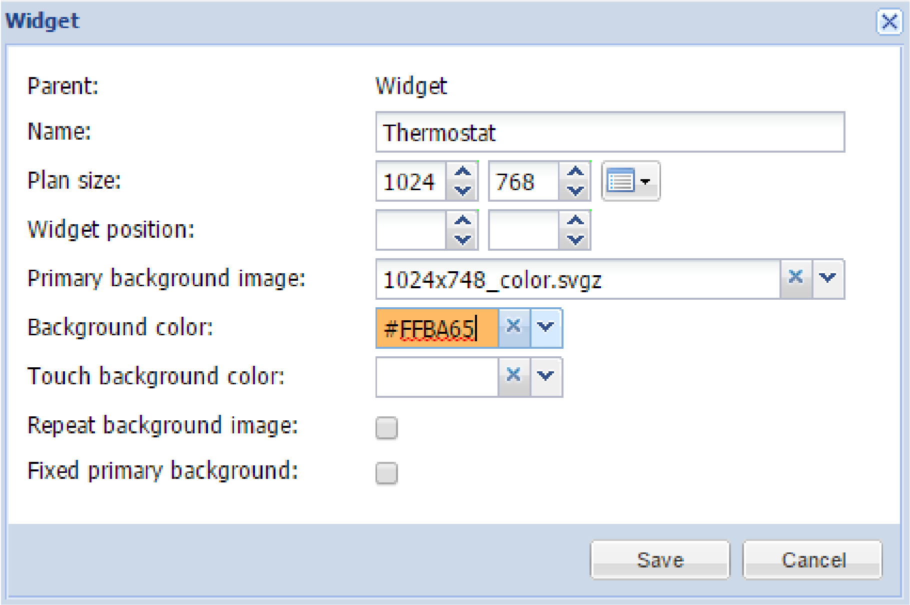


Layout

Parent:	Layout
Name:	
Plan size:	1024 768
Primary background image:	
Secondary background image:	
Background color:	
Touch background color:	
Repeat background image:	□
Fixed primary background:	□

- ***Parent*** – name of parent layout
- ***Name*** – name for the layout
- ***Plan size*** – plan size in pixels. There are predefined resolutions available when clicking on the icon on the right size of this parameter
- ***Primary background image*** – choose primary background image from the list added in *Vis.graphics* → *Images/Backgrounds*
- ***Secondary background image*** – choose secondary background image from the list added in *Vis.graphics* → *Images/Backgrounds*
- ***Background color*** – choose background color of the plan
- ***Touch background color –*** define a color for touch visualization
- ***Repeat background image*** – either to show the image once or repeat it and fill the whole plan
- ***Fixed primary background*** – specify if first background image should be fixed. By enabling this, you can enable Parallax effect for your visualization

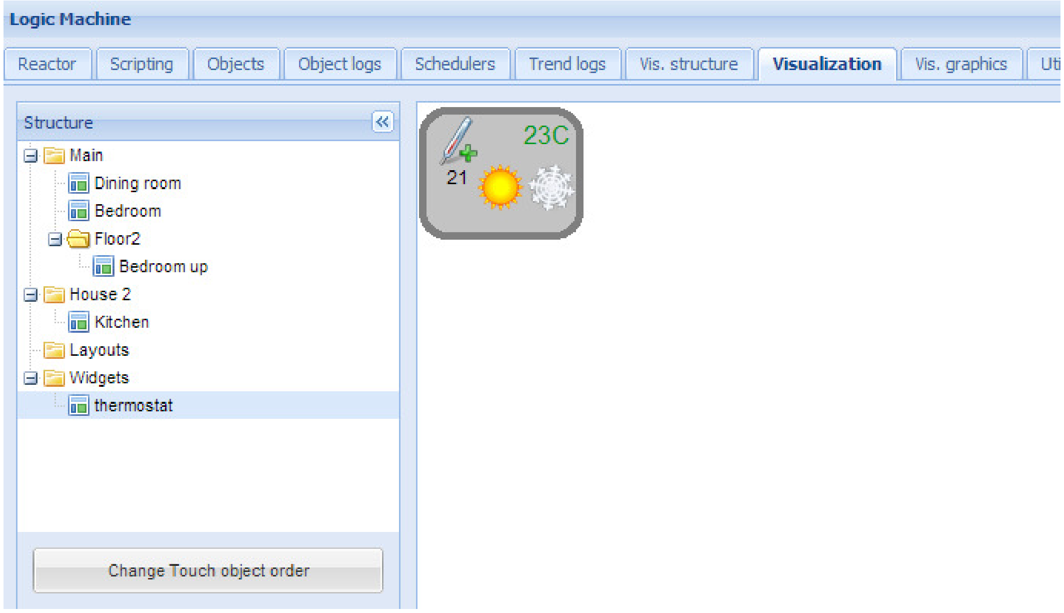
Widgets are used to combine several objects under one object in visualization.
Background image for the widget should be added in *Vis.graphics* → *Images/Backgrounds* in advance.



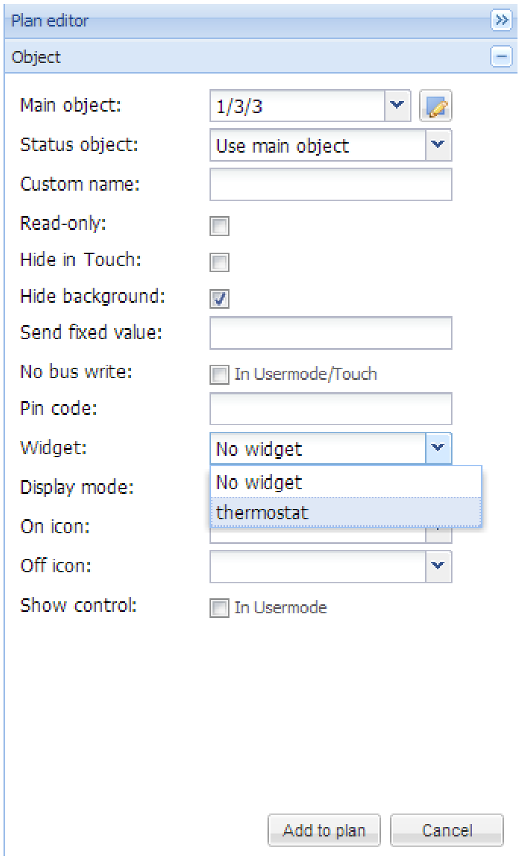
Parent:	Widget
Name:	Thermostat
Plan size:	1024 768
Widget position:	
Primary background image:	1024x748_color.svgz
Background color:	#FFBA65
Touch background color:	
Repeat background image:	☐
Fixed primary background:	☐

- ***Parent*** – name of parent widget
- ***Name*** – name for the widget
- ***Plan size*** – plan size in pixels. There are predefined resolutions available when clicking on the icon on the right size of this parameter
- ***Widget position*** – default position of the widget on the screen
- ***Primary background image*** – choose primary background image from the list added in *Vis.graphics* → *Images/Backgrounds*
- ***Background color*** – choose background color of the widget
- ***Touch background color –*** define a color for touch visualization
- ***Repeat background image*** – either to show the image once or repeat it and fill the whole plan
- ***Fixed primary background*** – specify if first background image should be fixed. By enabling this, you can enable Parallax effect for your visualization

When you have defined the widget in *Layouts/Widgets* tab, you can add objects to it in *Visualization* tab.

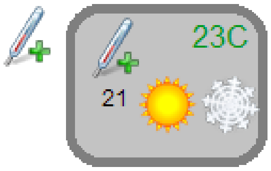


When you have added necessary objects to the widget, you can choose it when adding objects for main Levels e.g. Bedroom in Main level.



Main object:	1/3/3
Status object:	Use main object
Custom name:	
Read-only:	□
Hide in Touch:	□
Hide background:	☑
Send fixed value:	
No bus write:	□ In Usermode/Touch
Pin code:	
Widget:	No widget
Display mode:	
On icon:	
Off icon:	
Show control:	□ In Usermode

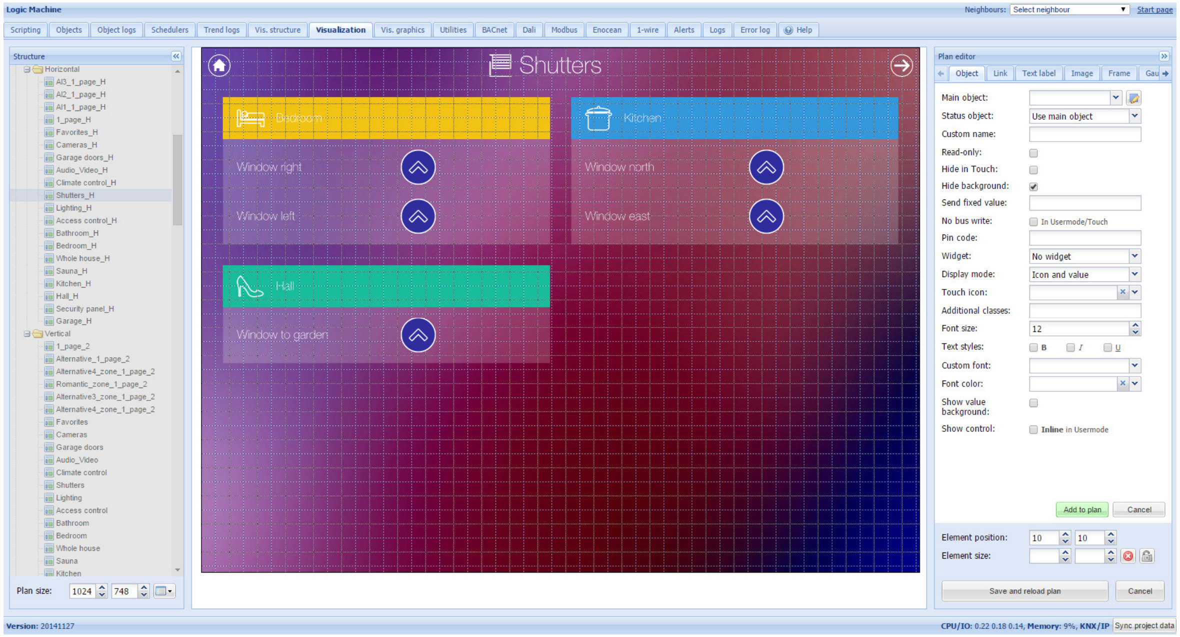
Once added, you can try out the widget in *Usermode visualization* by clicking on added object (temperature sensor icon on the left), the widget appears on click.



1.8. Visualization

After the building and floor structure is defined in Vis.structure tab, it is visualized in *Visualization* tab. Controlled and monitored objects can be added and managed in this section.

Both side bars can be minimized by pressing on « icon making the map more visible especially on small displays.



Plan size: 1024 748

Main object:
Status object: Use main object
Custom name:
Read-only: □
Hide in Touch: □
Hide background: ☑
Send fixed value:
No bus write: □ In Usermode/Touch
Pin code:
Widget: No widget
Display mode: Icon and value
Touch icon:
Additional classes:
Font size: 12
Text styles: □ B □ *I* □ U
Custom font:
Font color:
Show value background: □
Show control: □ **Inline** in Usermode

Element position: 10 10
Element size:

Version: 20141127

CPU/IO: 0.22 0.18 0.14, **Memory:** 9%, **KNX/IP**

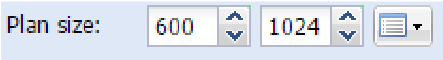
1.8.1. Plan editor

Plan editor is located on the right side of the visualization map. By clicking on *Unlock current plan for editing* button, the following main menus appear for configuration:

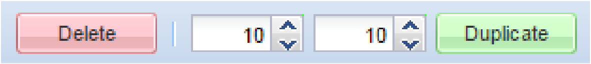
- *Object* – new object to be added to the map
- *Link* – linking several floors with special icons
- *Text Label* – text label to put on visualization
- *Image* – Add specific image on the visualization
- *Frame* – add frame object to the visualization

- *Gauge* – Metering gauge
- *Camera* – IP web camera integration into visualization
- *Graph* – Real-time graph to monitor value of scale-type objects

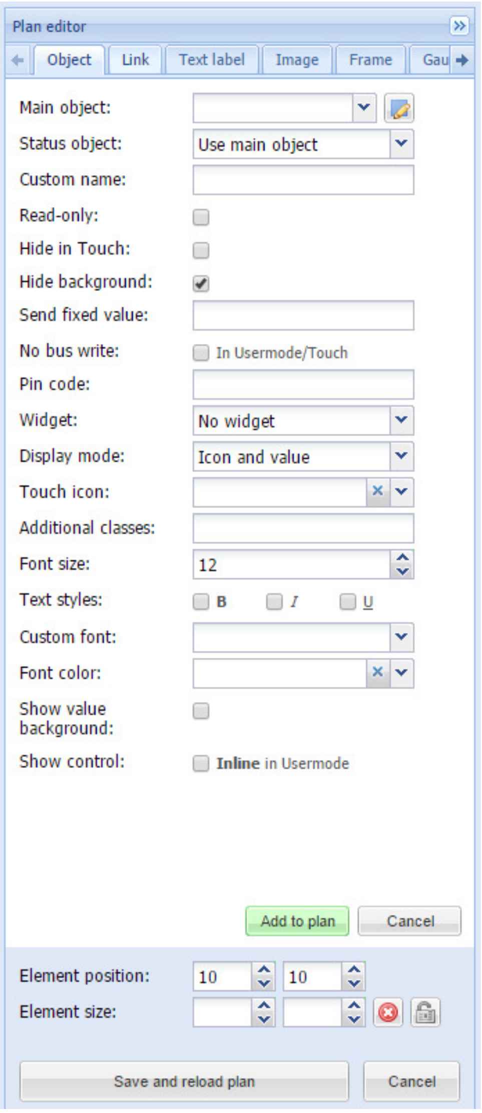
While in editing mode, on the left side you can change plan resolution on the fly


Plan size: 600 1024

When some object is selected and in the editing mode, there appears Delete / Duplicate buttons so you can either delete or copy the object



1.8.2. Object



Main object:

Status object: Use main object

Custom name:

Read-only: □

Hide in Touch: □

Hide background: ☑

Send fixed value:

No bus write: □ In Usermode/Touch

Pin code:

Widget: No widget

Display mode: Icon and value

Touch icon:

Additional classes:

Font size: 12

Text styles: □ **B** □ *I* □ U

Custom font:

Font color:

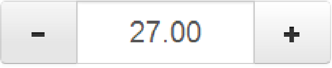
Show value background: □

Show control: □ **Inline** in Usermode

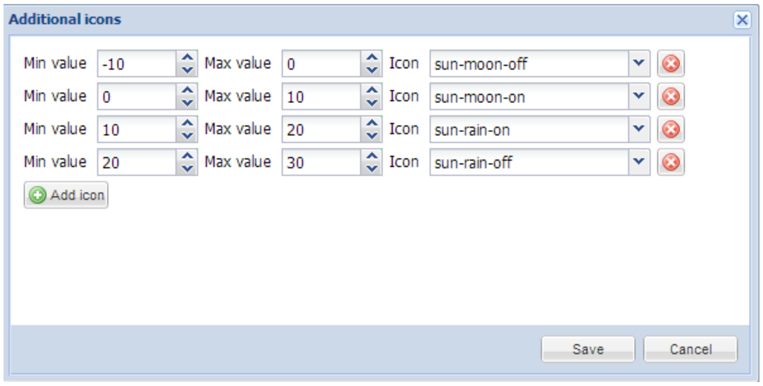
Element position: 10 10

Element size:

- ***Main object*** – list of existing group addresses on KNX/EIB bus, the ones available for configuration in *Objects* tab
- ***Status object*** – list of status objects on KNX/EIB bus
- ***Custom name*** – Name for the object
- ***Read-only*** – the object is read-only, no write permission
- ***Hide in touch***– do not show this object in *Touch Visualization*
- ***Hide background***– Hide icon background
- ***Send fixed value***– Allows to send specific value to the bus each time the object is pressed
- ***No bus write*** – do not send telegram into the bus once clicked on this object in Usermode/Touch visualizations
- ***PIN code*** – PIN code which will be asked to provide when click on this object to perform group write
- ***Widget*** – specify widget which will be launched when click on this object
- ***Display mode [icon and value; icon; value]*** – how to display the object
- ***Touch icon*** – icon for Touch visualization
- ***On icon*** – On state icon for binary-type objects. Icons library is located in *Vis.graphics → Icons tab*
- ***Off icon*** –Off state icon for binary-type objects. Icons library is located in *Vis.graphics → Icons tab*
- ***Additional classes*** – additional CSS classes for the element
- ***Show control*** –scale-type object specific setting defining either to show the control in Usermode visualization without icon

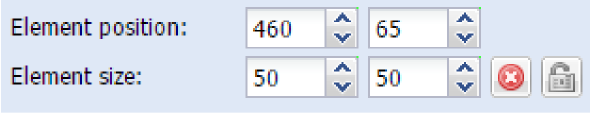


For scale-type objects additional button appears while specifying parameters – *Additional icons.* It's possible to define different icons for different object values in the window.


Additional icons

Min value	-10	Max value	0	Icon	sun-moon-off
Min value	0	Max value	10	Icon	sun-moon-on
Min value	10	Max value	20	Icon	sun-rain-on
Min value	20	Max value	30	Icon	sun-rain-off

On the bottom of setting you can see element position and size parameters, which you can freely change. By pressing ⊗ you will reset size. By pressing 🔒 you can lock aspect ratio.

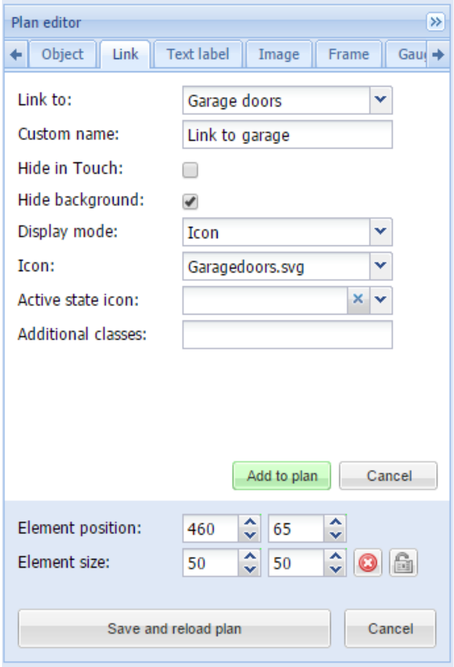

Element position:	460	65
Element size:	50	50

Once the object parameters are defined, press *Add to plan* button and newly created object will appear. You can move the object to the location it will be located. Note that while being in editing mode, the object will not work. When all necessary objects are added, press *Save and reload plan* button so the objects starts functioning.

You can edit each added object when clicking on it while in Editing mode.

1.8.3. Link

In order to make visualization more convenient, there are floor links integrated. You can add icons or text on the map, which links to other floors.



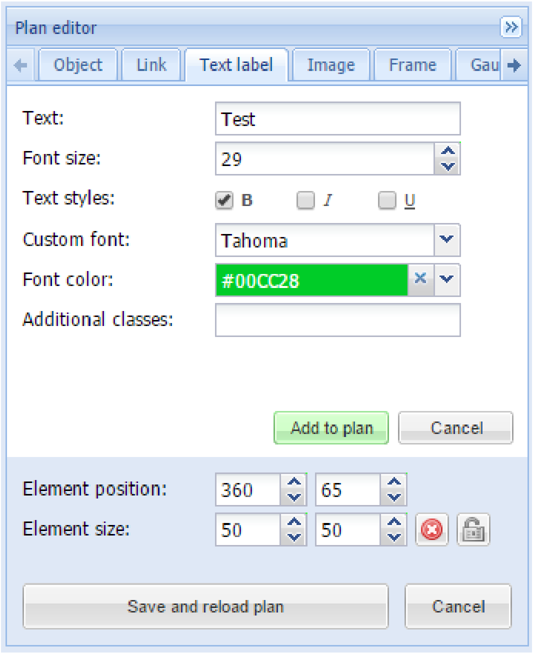
Link to:	Garage doors
Custom name:	Link to garage
Hide in Touch:	□
Hide background:	☑
Display mode:	Icon
Icon:	Garagedoors.svg
Active state icon:	
Additional classes:	
Element position:	460 65
Element size:	50 50

- ***Link to*** – Linked plan name or link to Schedulers / Trends or External Link (use the link in form http://www.openrb.com)
- ***Custom name*** – name for the link
- ***Hide in touch*** – do not show this object in *Touch Visualization*
- ***Hide background***– Hide icon background
- ***Display mode [Icon; Value]*** – either to show icon or its value
- ***Icon*** – Icon which will be showed in visualization (if chosen, no further parameters are available)
- ***Active state icon*** – active state icon if the link is to current plan (in case you have several smaller plans on one visualization and want to display the current one)
- ***Additional classes*** – additional CSS classes for the element

Once the floor link parameters are defined, press *Add to plan* button and newly created object will appear. You can move the object to the location it will be located. Note that while being in editing mode, the object will not work. Press on *Save and reload plan* button so the objects starts functioning.

1.8.4. *Text Label*

Text labels can be added and moved across the visualization map.



Text: Test
Font size: 29
Text styles: ☑ B ☐ *I* ☐ U
Custom font: Tahoma
Font color: #00CC28
Additional classes:
Element position: 360 65
Element size: 50 50

- ***Text*** – label text
- ***Font size*** – label font size
- ***Text style*** – style of the text – bold, italic, underscored
- ***Custom font*** – font name
- ***Font color***– label font color
- ***Additional classes*** – additional CSS classes for the element

Once the label parameters are defined, press *Add to plan* button and newly created object will appear on the map. You can move the object to the location it will be located. Press on *Save and reload plan* button so the objects starts functioning.

1.8.5. *Image*

Image section allows adding images from the internet into the visualization map. Useful for example, to grab dynamic weather cast images.

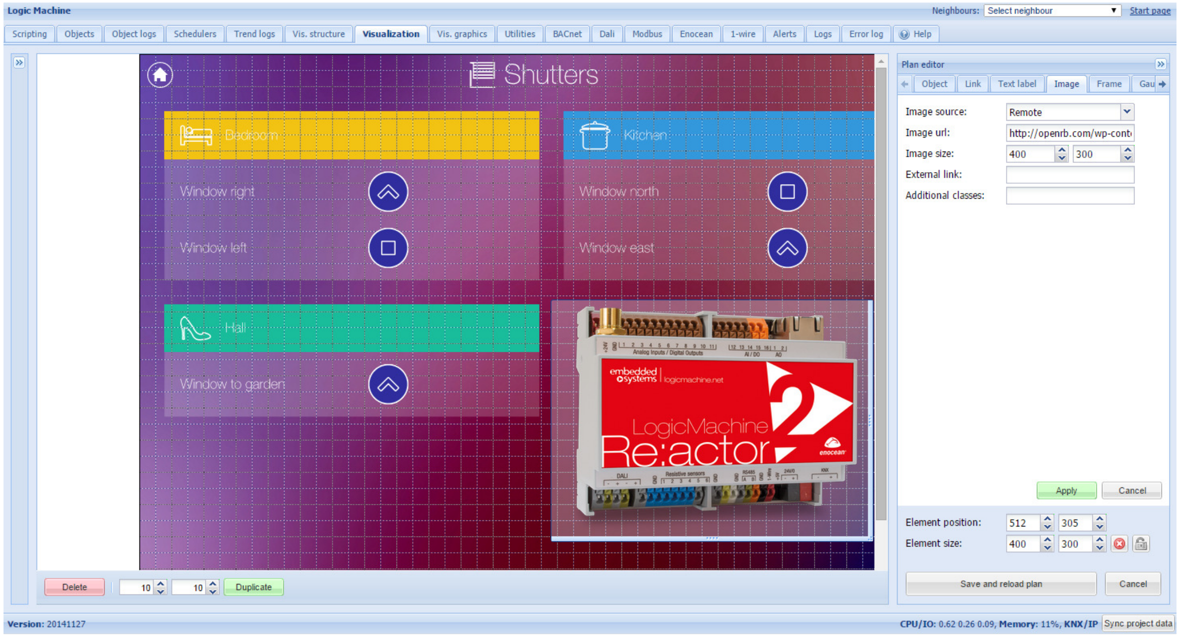


Image source: Remote
Image url: http://openrb.com/wp-cont
Image size: 400 300
External link:
Additional classes:
Element position: 512 305
Element size: 400 300

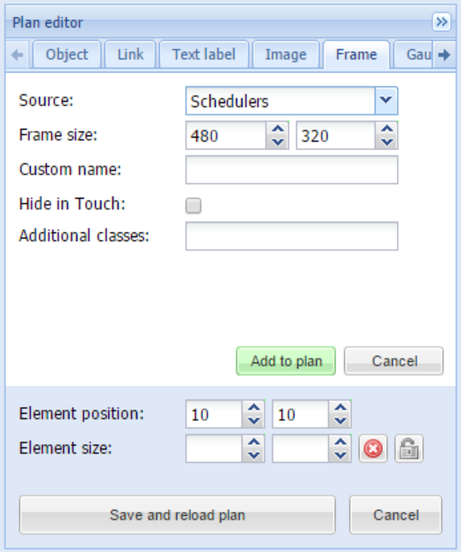
Version: 20141127
CPU/IO: 0.62 0.26 0.09, Memory: 11%, KNX/IP

- ***Image source [Local; Remote]*** – image source location
- ***Source url / Select image*** – Source URL of the image or image from local database
- ***Image size*** – width and height of the image
- ***External link*** – external link URL when pressing on the image
- ***Additional classes*** – additional CSS classes for the element

Once the image parameters are defined, press *Add to plan* button and newly created object will appear on the map. You can move the object to the location it will be located. Press on *Save and reload plan* button so the objects starts functioning.

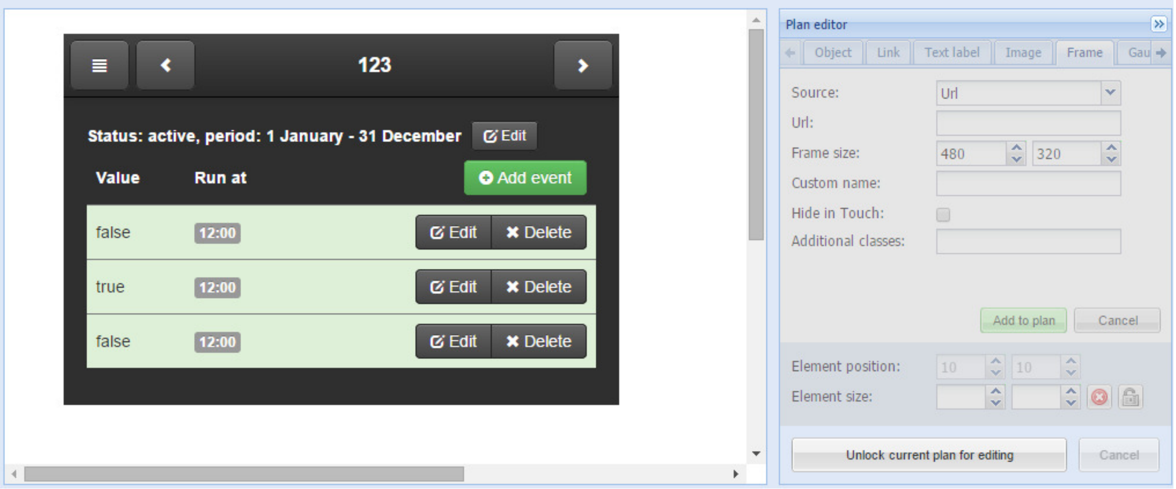
1.8.6. Frame

With Frame functionality you can integrate 3rd party applications, we resources or local Trends/Schedulers into one common visualization.



Source: Schedulers
Frame size: 480 320
Custom name:
Hide in Touch: □
Additional classes:
Element position: 10 10
Element size:

- ***Source [Url, Schedulers; Trend logs]*** – frame source
- ***Url*** – Source URL of the page to integrate
- ***Frame size*** – width and height of the frame
- ***Custom name*** – custom name of the frame object
- ***External link*** – external link URL when pressing on the image
- ***Hide in Touch*** – defines either to hide frame in Touch visualization
- ***Additional classes*** – additional CSS classes for the element



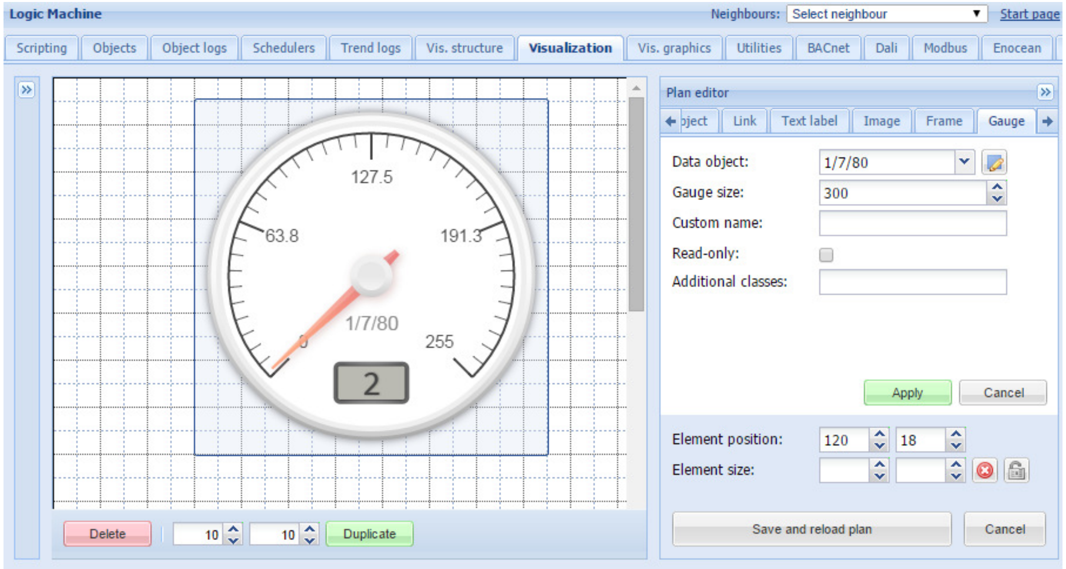
Status: active, period: 1 January - 31 December

Value	Run at	
false	12:00	Edit ✖ Delete
true	12:00	Edit ✖ Delete
false	12:00	Edit ✖ Delete

Source: Url
Url:
Frame size: 480 320
Custom name:
Hide in Touch: □
Additional classes:
Element position: 10 10
Element size:

1.8.7. *Gauge*

Gauge allows visualizing and changing object value in the gauge.



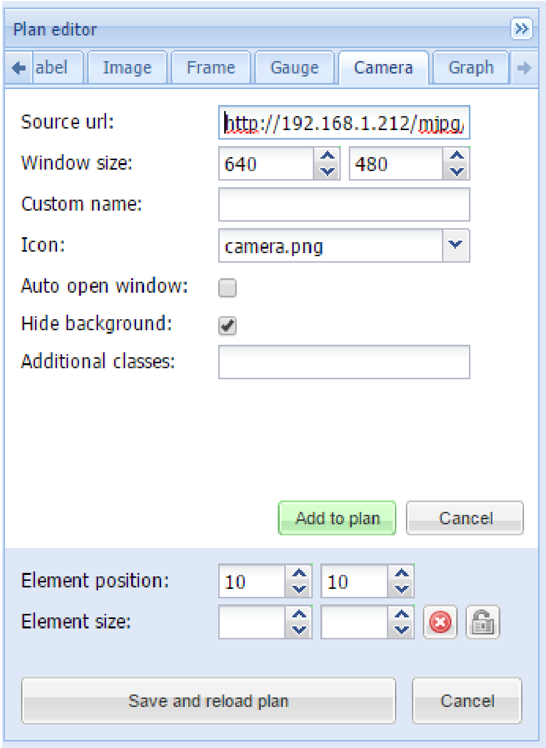
Data object: 1/7/80
Gauge size: 300
Custom name:
Read-only: □
Additional classes:
Element position: 120 18
Element size:

- ***Data object*** – KNX group address
- ***Gauge size*** – size of the gauge
- ***Custom name*** – custom name for the object
- ***Read only*** – make the gauge read only
- ***Additional classes*** – additional CSS classes for the element

Once the gauge parameters are defined, press *Add to plan* button and newly created object will appear on the map. You can move the object to the location it will be located. Press on *Save and reload plan* button so the objects starts functioning.

1.8.8. Camera

LogicMachine supports third party IP web camera integration into its visualization.



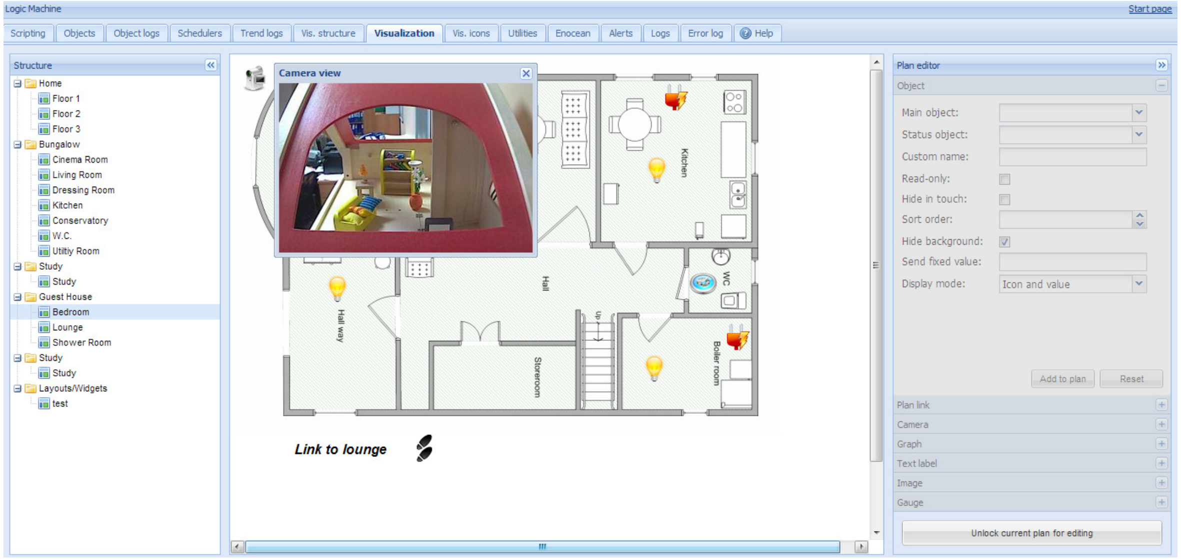
Source url: http://192.168.1.212/mjpg
Window size: 640 480
Custom name:
Icon: camera.png
Auto open window: □
Hide background: ☑
Additional classes:

Element position: 10 10
Element size:

- ***Source url*** – source address of the video stream
- ***Window size*** – size of the window of camera picture
- ***Custom name*** – name for the object
- ***Icon*** – icon for the object
- ***Auto open window*** – automatically open video window, otherwise it is launched by click on the icon
- ***Hide background***– hide icon background
- ***Additional classes*** – additional CSS classes for the element

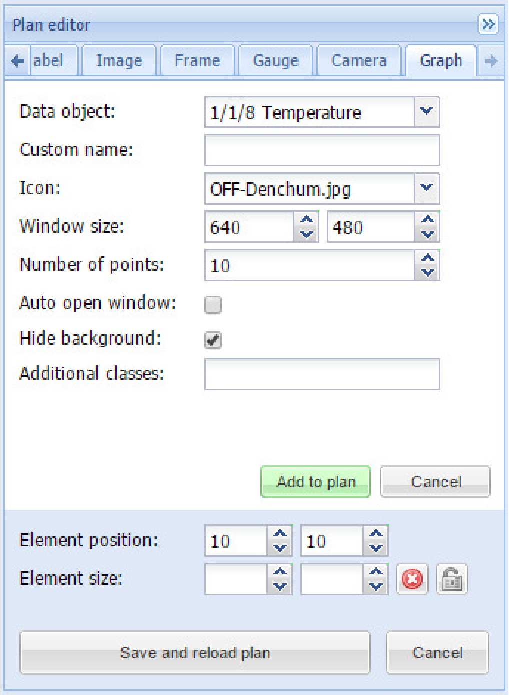
Note! If IP camera requires user name and password, enter the url in form ***http://USER:PASSWORD@IP***

Once the camera parameters are defined, press *Add to plan* button and newly created object will appear in look of video camera. You can move the object to the location it will be located. Note that while being in editing mode, the object will not work. Press on *Save and reload plan* button so the objects starts functioning. By pressing on video camera, a new sub-window appears with a picture from your IP web camera. The window can be freely moved to other location so not to cover other visualization objects.



1.8.9. *Graph*

Real-time graphs can be integrated into visualization system to monitor the current and old value of scale-type objects. Make sure logging is enabled for the object in *Object* tab which values is planned to be shown in the graph.



Data object: 1/1/8 Temperature

Custom name:

Icon: OFF-Denchum.jpg

Window size: 640 480

Number of points: 10

Auto open window: □

Hide background: ☑

Additional classes:

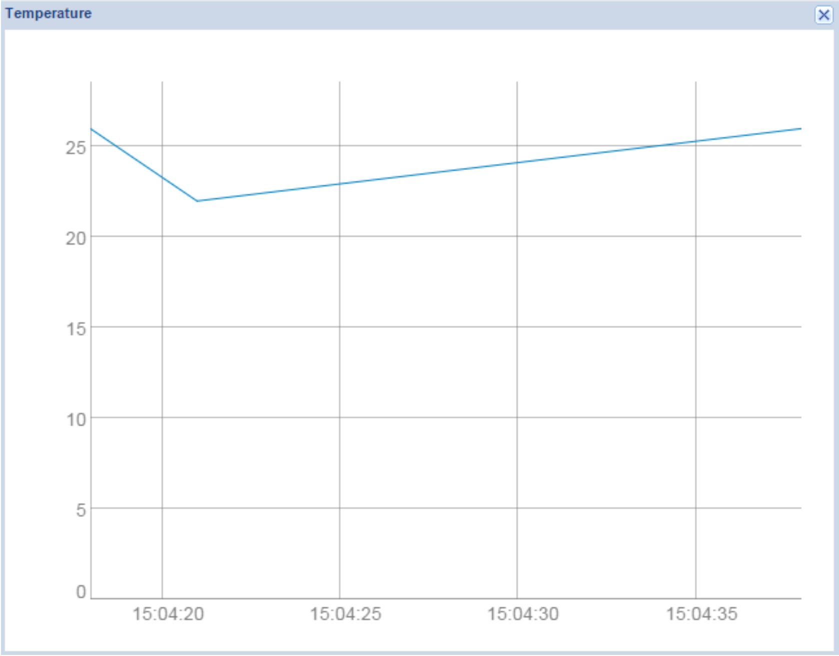
Element position: 10 10

Element size:

- ➢ ***Data object*** – group address of the object
- ➢ ***Custom name*** – name of the object
- ➢ ***Icon***– icon to launch the graph
- ➢ ***Windows size*** – size of the graph window
- ➢ ***Number of points*** – number of data points to show in the graph

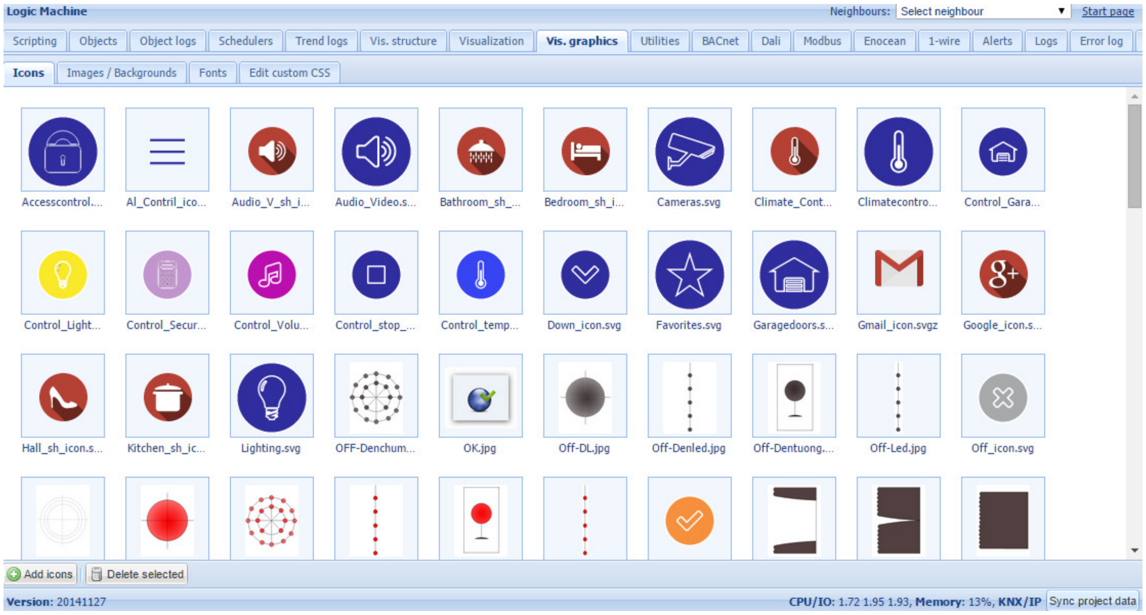
- ***Auto open window*** – graph window is automatically opened
- ***Hide background*** – hide icon background
- ***Additional classes*** – additional CSS classes for the element

Once the graph parameters are defined, press *Add to plan* button and newly created object will appear. You can move the object to the location it will be located. Note that while being in editing mode, the object will not work. Press on *Save and reload plan* button so the objects starts functioning.

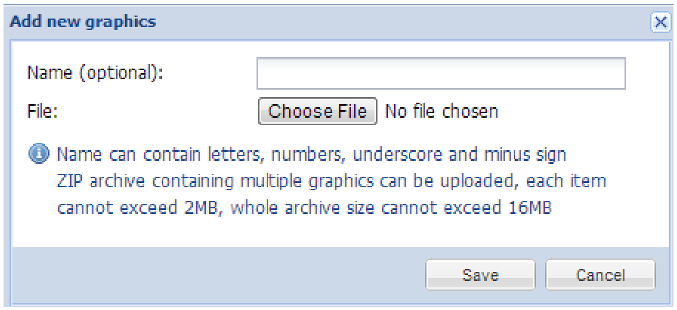


1.9. Vis.graphics

The list of predefined icons, list of images and backgrounds is available in *Vis.graphics* tab.



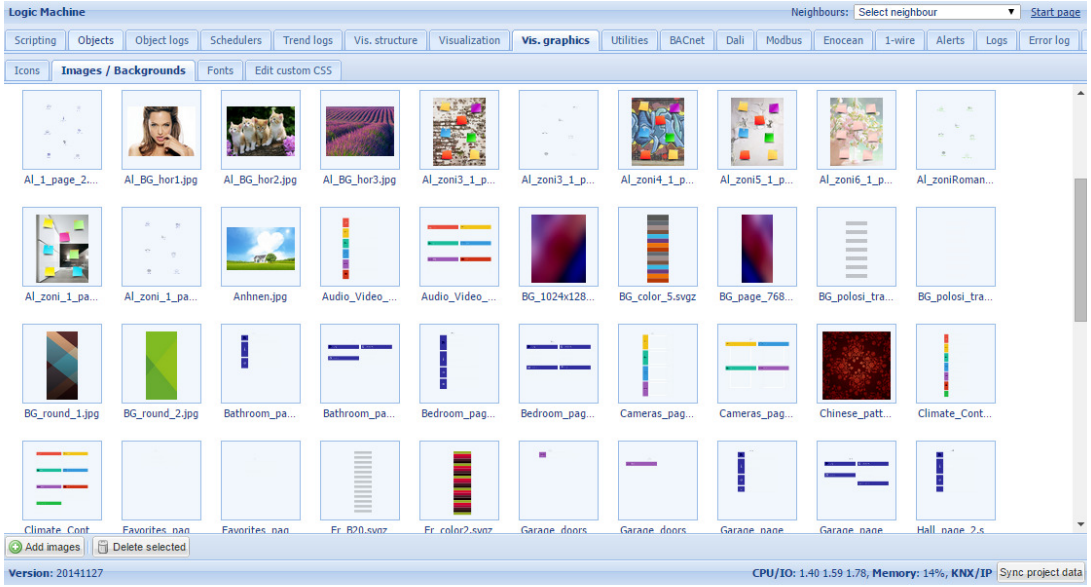
Press on *Add icons* button to add a new entry. The system accepts any size icons. GIF is also supported.



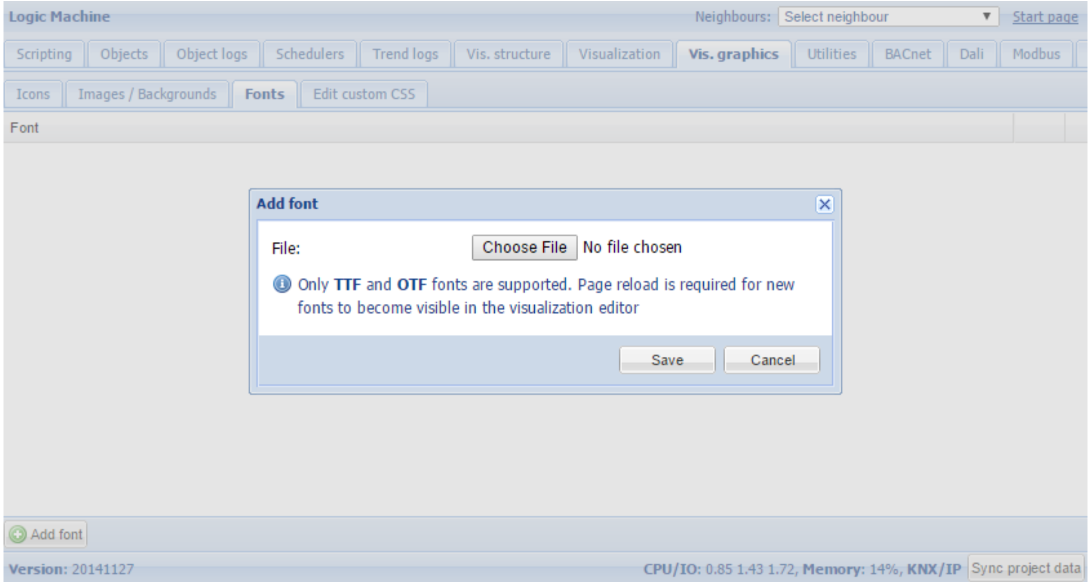
Name can contain letters, numbers, underscore and minus sign
ZIP archive containing multiple graphics can be uploaded, each item cannot exceed 2MB, whole archive size cannot exceed 16MB

Name (optional) – the name of the icon
File – Icon file location

Images/Backgrounds tab is used to upload image files for visualization purposes



In *Fonts* tab you can add custom fonts



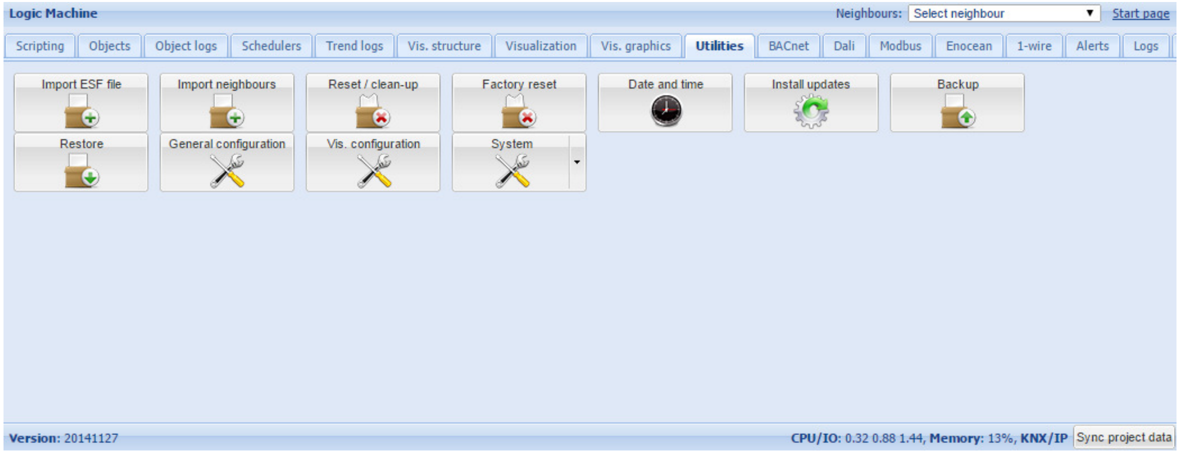
File: No file chosen

Only **TTF** and **OTF** fonts are supported. Page reload is required for new fonts to become visible in the visualization editor

In *Custom CSS* tab you can add your CSS style for the visualization which you can use when adding elements into visualization, so any elements of Look and Feel is customizable with this solution.

1.10. Utilities

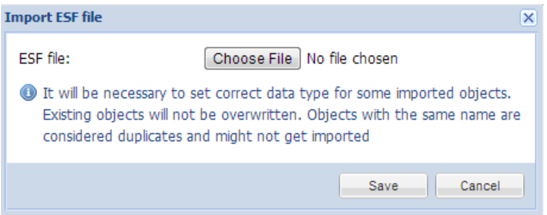
There are following utilities in the tab available:



Version: 20141127

CPU/IO: 0.32 0.88 1.44, **Memory:** 13%, **KNX/IP**

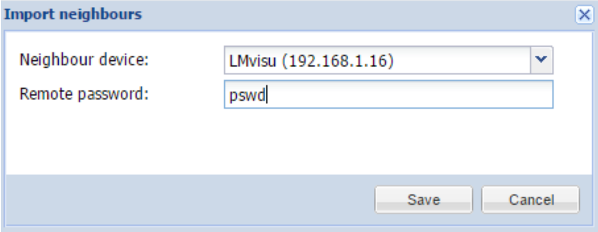
Import ESF file– imports ETS object file. It will be necessary to set correct data types for some imported objects. Existing objects will not be overwritten. Objects with the same name are considered duplicates and might not be imported



ESF file: No file chosen

It will be necessary to set correct data type for some imported objects. Existing objects will not be overwritten. Objects with the same name are considered duplicates and might not get imported

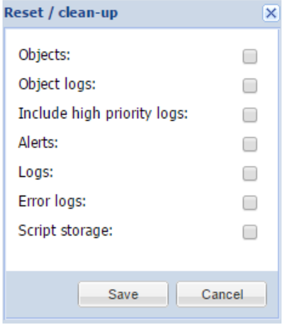
Import neighbours – import list of objects from network LM devices



Neighbour device: LMvisu (192.168.1.16)

Remote password: pswd

Reset / clean-up – delete all objects from the Logic Machine, they disappear from visualization aswell



Objects: ☐

Object logs: ☐

Include high priority logs: ☐

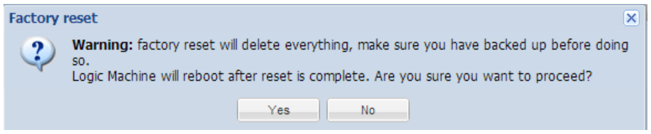
Alerts: ☐

Logs: ☐

Error logs: ☐

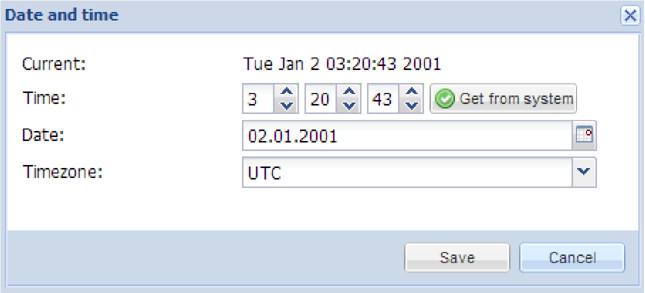
Script storage: ☐

Factory reset– delete all configuration and return to factory defaults


Factory reset

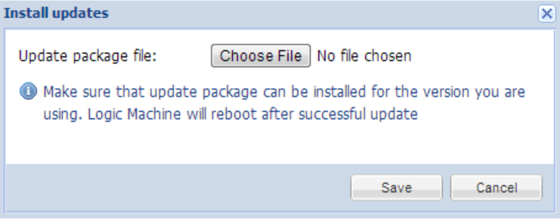
Warning: factory reset will delete everything, make sure you have backed up before doing so.
Logic Machine will reboot after reset is complete. Are you sure you want to proceed?

Date and time – data and time settings


Date and time

Current: Tue Jan 2 03:20:43 2001
Time: 3 20 43
Date: 02.01.2001
Timezone: UTC

Install updates – install LogicMachine update file *.lmu. LogicMachine will reboot after successful update

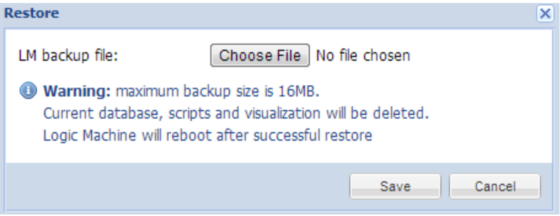

Install updates

Update package file: No file chosen

Make sure that update package can be installed for the version you are using. Logic Machine will reboot after successful update

Backup – backup all objects, logs, scripts, visualization.

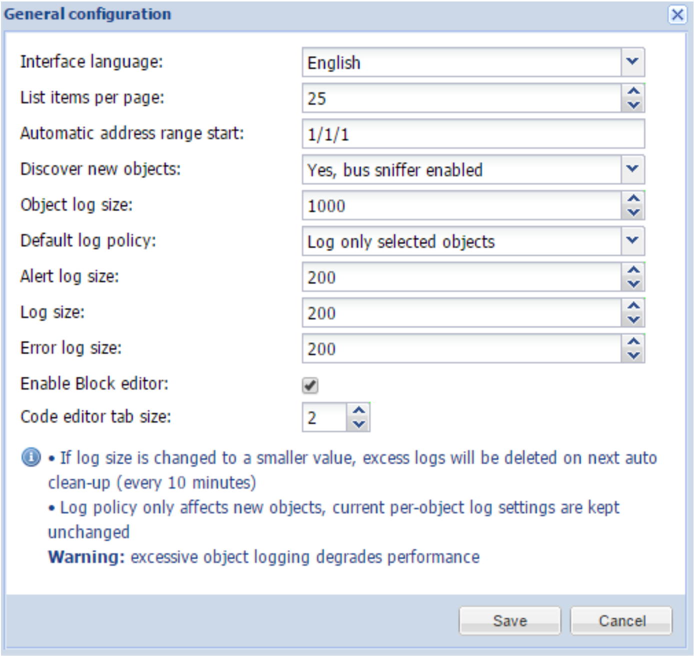
Restore– restore configuration from backup


Restore

LM backup file: No file chosen

Warning: maximum backup size is 16MB.
Current database, scripts and visualization will be deleted.
Logic Machine will reboot after successful restore

General Configuration – system general settings



Interface language:	English
List items per page:	25
Automatic address range start:	1/1/1
Discover new objects:	Yes, bus sniffer enabled
Object log size:	1000
Default log policy:	Log only selected objects
Alert log size:	200
Log size:	200
Error log size:	200
Enable Block editor:	☑
Code editor tab size:	2

- If log size is changed to a smaller value, excess logs will be deleted on next auto clean-up (every 10 minutes)
- Log policy only affects new objects, current per-object log settings are kept unchanged

Warning: excessive object logging degrades performance

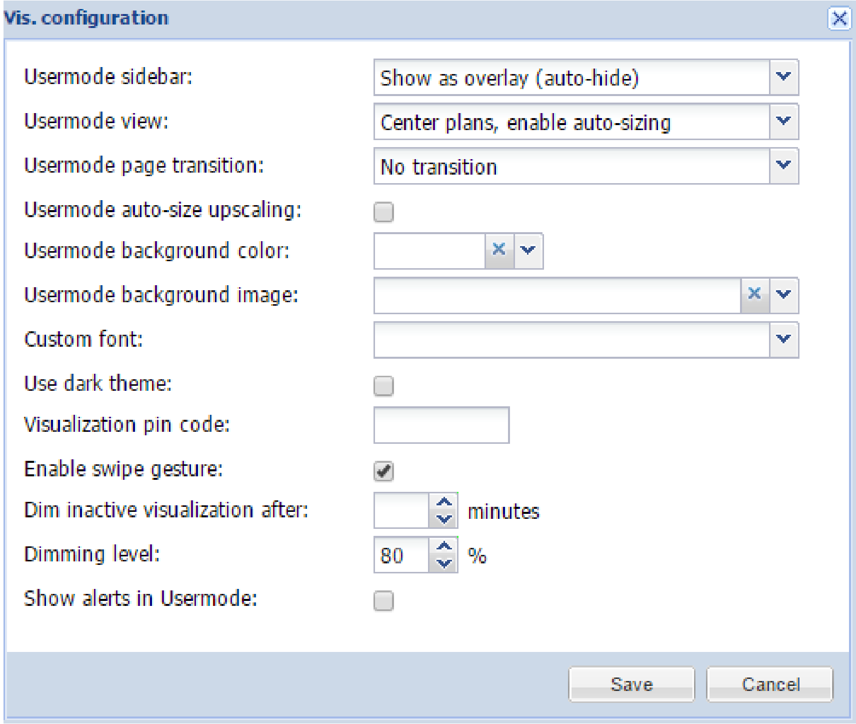
Interface language – interface language
List items per page –count of lines per page e.g. *Objects, Object logs, Alerts etc.*
Automatic address range start – start group address when using automatic addressing in scripts, IO settings and other
Discover new objects– either KNX object sniffer is enabled. If yes, once triggered all new objects will appear automatically in the Objects list
Object log size – max count of object logs
Default log policy– either to log status change for all objects or only for checked objects
Alert log size – max count of alerts logged
Log size – max count of logs
Error log size – max count of errors logged
Enable block editor – either to enable scripting block editor
Code editor tab size – specify tab size to be used in the scripting editor

Note! If log size is changed to a smaller value, excess logs will be deleted on next auto clean-up (every 10 minutes)

Note! Log policy only affects new objects, current per-object log settings are kept unchanged

Warning! Excessive object logging degrades LogicMachine performance

Vis. Configuration – visualization specific settings



Usermode sidebar: Show as overlay (auto-hide)

Usermode view: Center plans, enable auto-sizing

Usermode page transition: No transition

Usermode auto-size upscaling: ☐

Usermode background color:

Usermode background image:

Custom font:

Use dark theme: ☐

Visualization pin code:

Enable swipe gesture: ☑

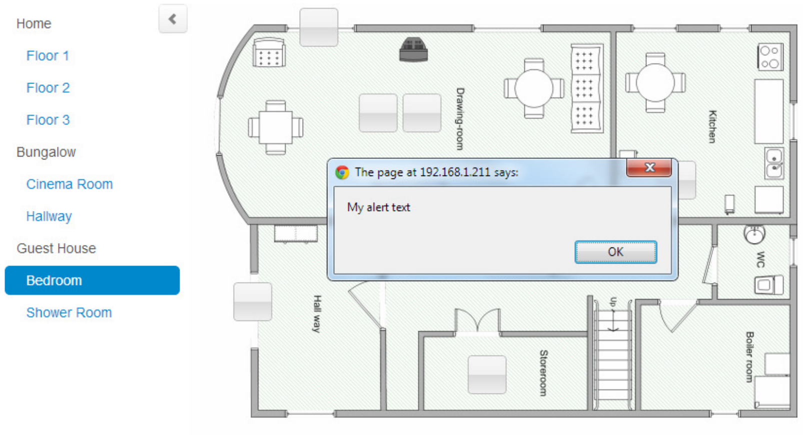
Dim inactive visualization after: minutes

Dimming level: 80 %

Show alerts in Usermode: ☐

Usermode sidebar [Show docked, Show as overlay (auto-hide), Hide (fullscreen mode] – visibility of sidebar when in Usermode Visualization
Usermode view [Align plans to top left, no size limit; Center plans, limit size; Center plans, enable auto-sizing; Center horizontally, auto-size width] – defines the look of Usermode visualization
Usermode page transition [Flip X; Flip Y; Shrink; Expand; Slide up; Slide down, Slide left; Slide right; Slide up big; Slide down big; Slide left big; Slide right big] – transition when changing plans in visualization
Usermode auto-size upscaling – enable this to scale the visualization automatically on each display device. Please note to use SVG format images and icons so the quality is not affected by upscaling
Usermode background color – background color in usermode visualization
Usermode background image – specific image for usermode visualization
Custom font – select custom font to use in visualization
Use dark theme – check to enable dark theme in both usermode and touch visualizations
Visualization pin code – PIN code to access visualization
Enable swipe gesture – check to enable swipe gesture to move across plans from your touch device
Dim inactive visualization after – define time in minutes after which the screen will be dimmed where visualization is opened
Dim level – dim level for the display

Show alerts in Usermode – once new Alerts is triggered it will pop-up in User mode visualization

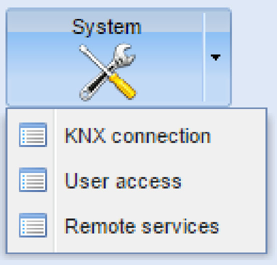


The page at 192.168.1.211 says:

My alert text

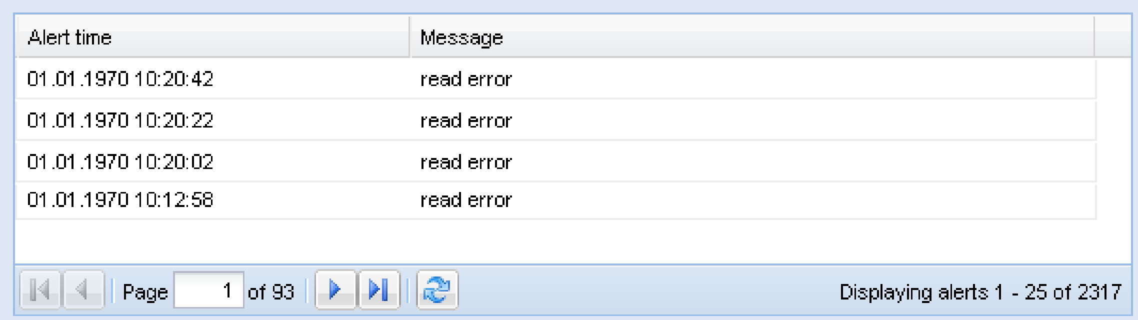
OK

System – by clicking on the arrow near System button, *KNX Connection, User Access, Remote Services* settings can be access. By clicking on the *System* button, network configuration window opens in new browser's tab.



1.11. Alerts

In *Alert* tab a list of alert messages defined with ***alert*** function in scripts is located. The messages are stored on the compact flash. Information on system start and KNX connection status messages are also automatically displayed in this window.

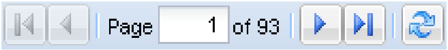


Alert time	Message
01.01.1970 10:20:42	read error
01.01.1970 10:20:22	read error
01.01.1970 10:20:02	read error
01.01.1970 10:12:58	read error

Page 1 of 93

Displaying alerts 1 - 25 of 2317

On the communication panel you can jump by pages and reload the page.

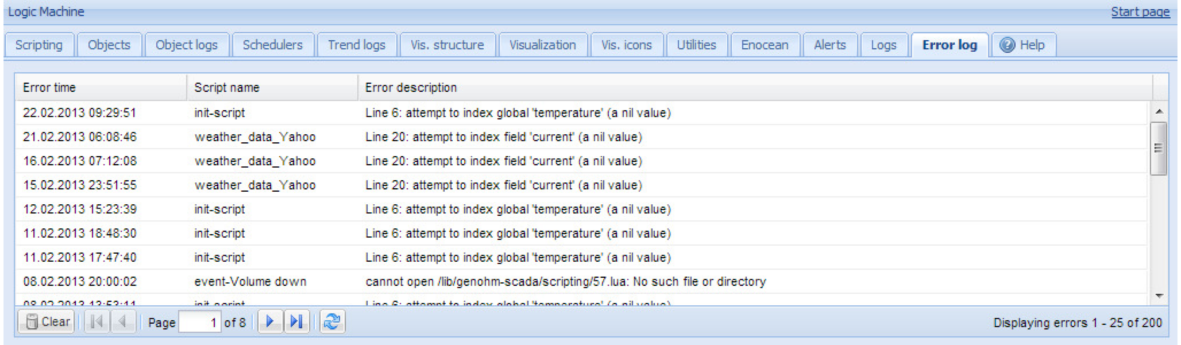


<u>Example</u>

```
temperature = 25.3

if temperature > 24 then
-- resulting message: 'Temperature levels are too high: 25.3'
alert('Temperature level is too high: %.1f', temperature)
end
```

1.12. Error log

Error messages from scripts are displayed in *Error log* tab.

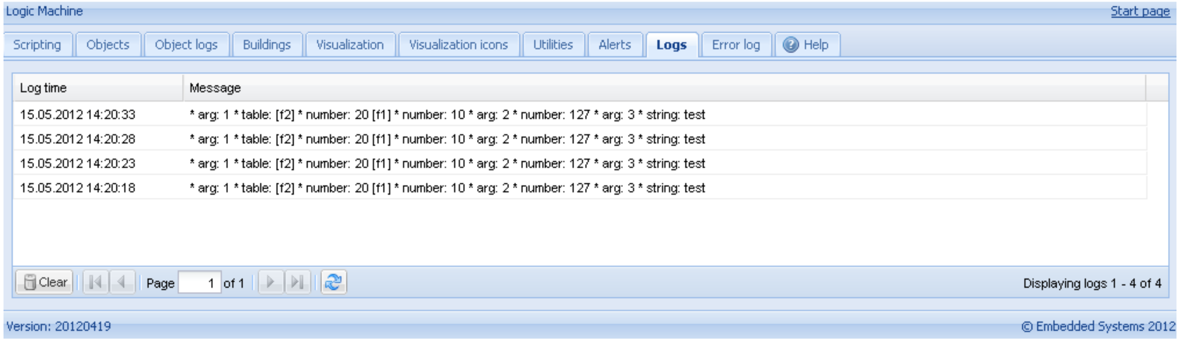


Error time	Script name	Error description
22.02.2013 09:29:51	init-script	Line 6: attempt to index global 'temperature' (a nil value)
21.02.2013 06:08:46	weather_data_Yahoo	Line 20: attempt to index field 'current' (a nil value)
16.02.2013 07:12:08	weather_data_Yahoo	Line 20: attempt to index field 'current' (a nil value)
15.02.2013 23:51:55	weather_data_Yahoo	Line 20: attempt to index field 'current' (a nil value)
12.02.2013 15:23:39	init-script	Line 6: attempt to index global 'temperature' (a nil value)
11.02.2013 18:48:30	init-script	Line 6: attempt to index global 'temperature' (a nil value)
11.02.2013 17:47:40	init-script	Line 6: attempt to index global 'temperature' (a nil value)
08.02.2013 20:00:02	event-Volume down	cannot open /lib/genohm-scada/scripting/57.lua: No such file or directory
08.02.2013 13:53:11	init-script	Line 6: attempt to index global 'temperature' (a nil value)

Displaying errors 1 - 25 of 200

1.13. Logs

Logs can be used for scripting code debugging. The log messages appear defined by *log* function.



Log time	Message
15.05.2012 14:20:33	* arg: 1 * table: [f2] * number: 20 [f1] * number: 10 * arg: 2 * number: 127 * arg: 3 * string: test
15.05.2012 14:20:28	* arg: 1 * table: [f2] * number: 20 [f1] * number: 10 * arg: 2 * number: 127 * arg: 3 * string: test
15.05.2012 14:20:23	* arg: 1 * table: [f2] * number: 20 [f1] * number: 10 * arg: 2 * number: 127 * arg: 3 * string: test
15.05.2012 14:20:18	* arg: 1 * table: [f2] * number: 20 [f1] * number: 10 * arg: 2 * number: 127 * arg: 3 * string: test

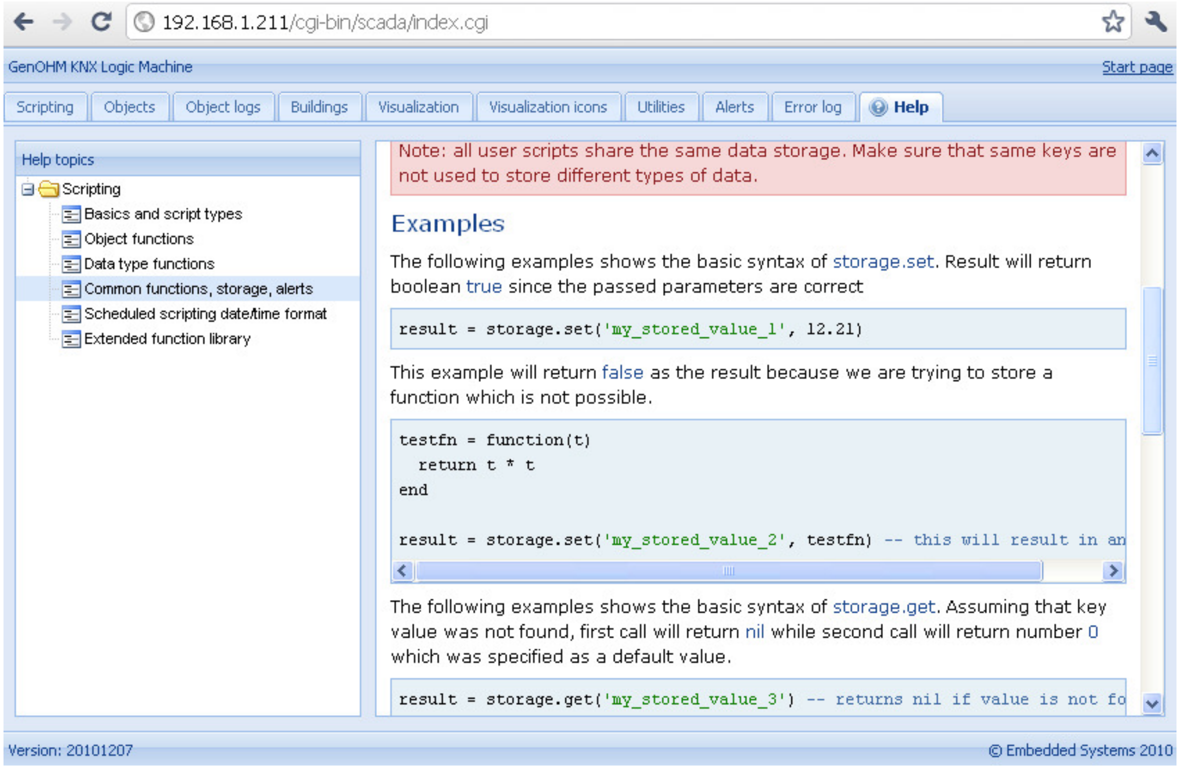
Displaying logs 1 - 4 of 4

Version: 20120419

© Embedded Systems 2012

1.14. Help

Documentation for scripting syntaxes is displayed in *Help* tab.



Note: all user scripts share the same data storage. Make sure that same keys are not used to store different types of data.

Examples

The following examples shows the basic syntax of storage.set. Result will return boolean true since the passed parameters are correct

```
result = storage.set('my_stored_value_1', 12.21)
```

This example will return false as the result because we are trying to store a function which is not possible.

```
testfn = function(t)
  return t * t
end

result = storage.set('my_stored_value_2', testfn) -- this will result in an
```

The following examples shows the basic syntax of storage.get. Assuming that key value was not found, first call will return nil while second call will return number 0 which was specified as a default value.

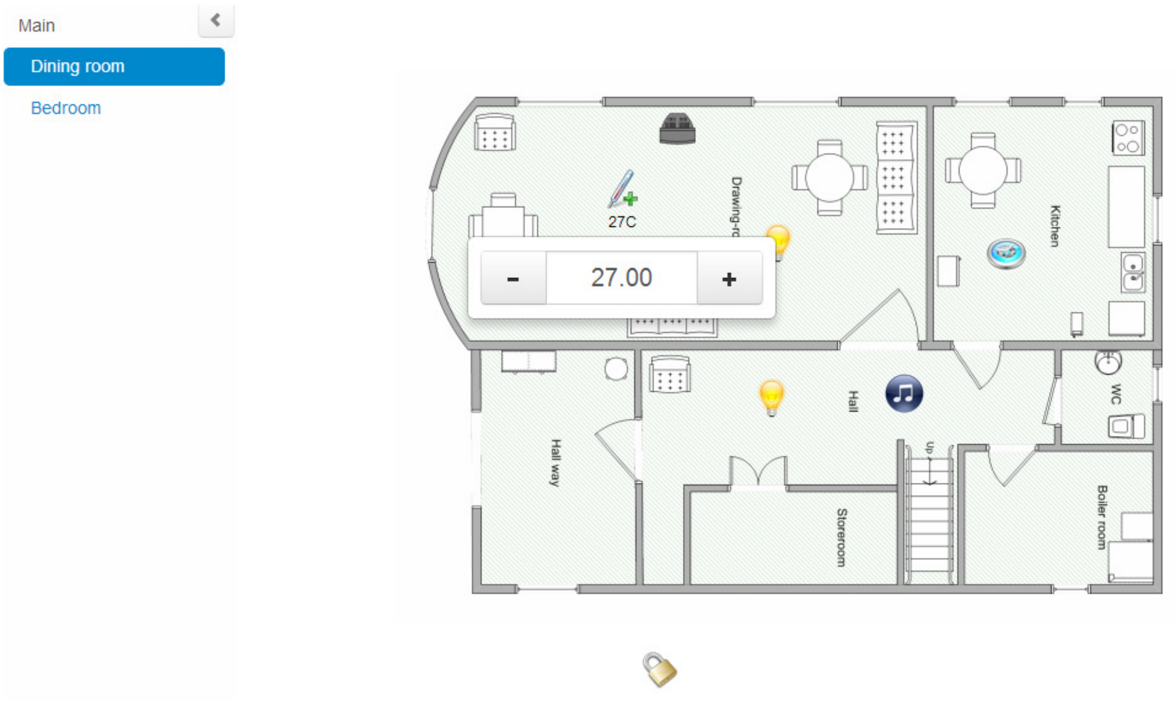
```
result = storage.get('my_stored_value_3') -- returns nil if value is not fo
```

2. User mode visualization

User mode visualization contains created visualization maps.

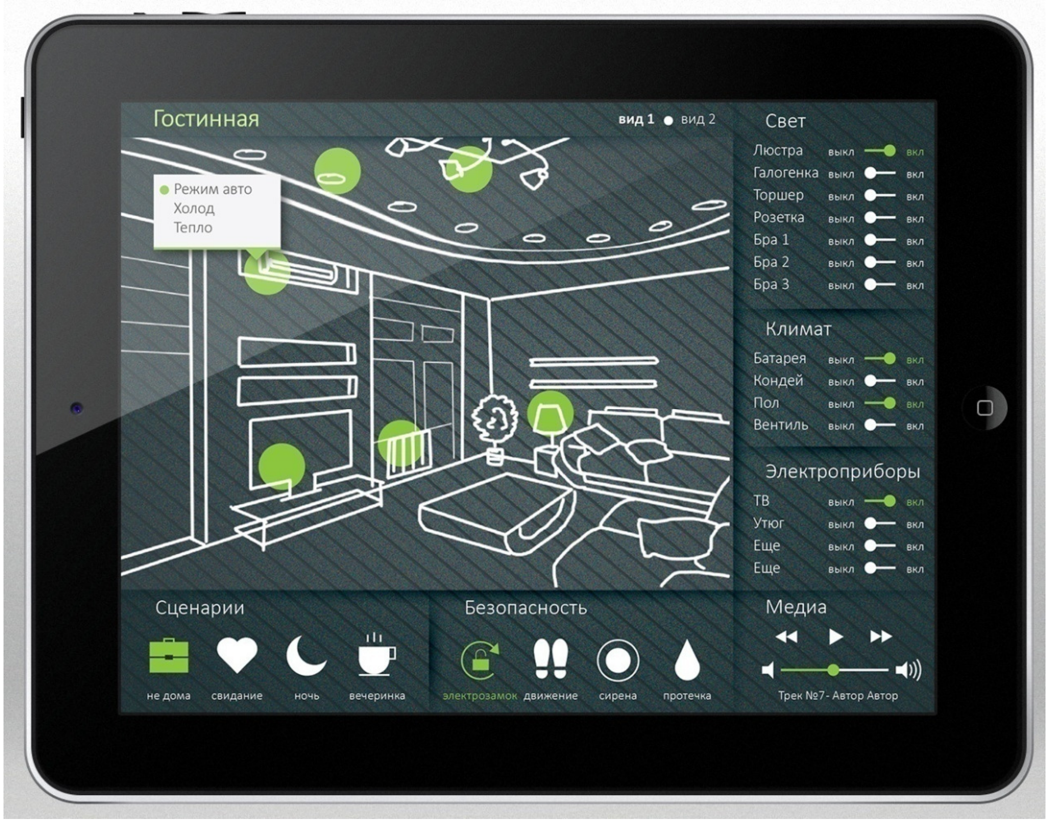
There are three access levels: read, write, admin (password access can be also disabled)

Access level	Login	Password
Read-only	Visview	visview
Write	viscontrol	viscontrol
Write+admin level	visadmin	visadmin



2.1. Custom design Usermode visualization

Through Custom CSS styles it is possible to create different type of visualization maps.



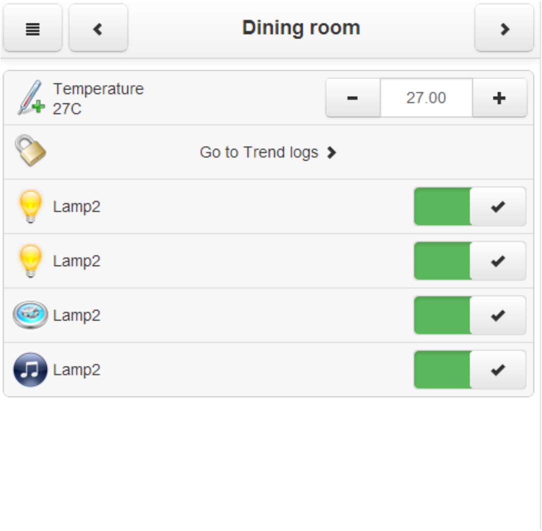
3. Touch visualization

Touch visualization is designed for iPhone/iPod/iPad/Android touch screen devices. All objects which are added in *Logic Machine* configuration by default are visible in touch visualization (if there is no *Hide in touch* option enabled).

There are three access levels: read, write, admin

Access level	Login	Password
Read-only	visview	visview
Write	viscontrol	viscontrol
Write+admin level	visadmin	visadmin

The main window is Building view where you can choose which Floor from which Building to control. Once you choose the floor, all objects which are assigned to it, are listed and can be controlled.



Dining room

Temperature
27C – 27.00 +

Go to Trend logs ›

Lamp2 ✓

Lamp2 ✓

Lamp2 ✓

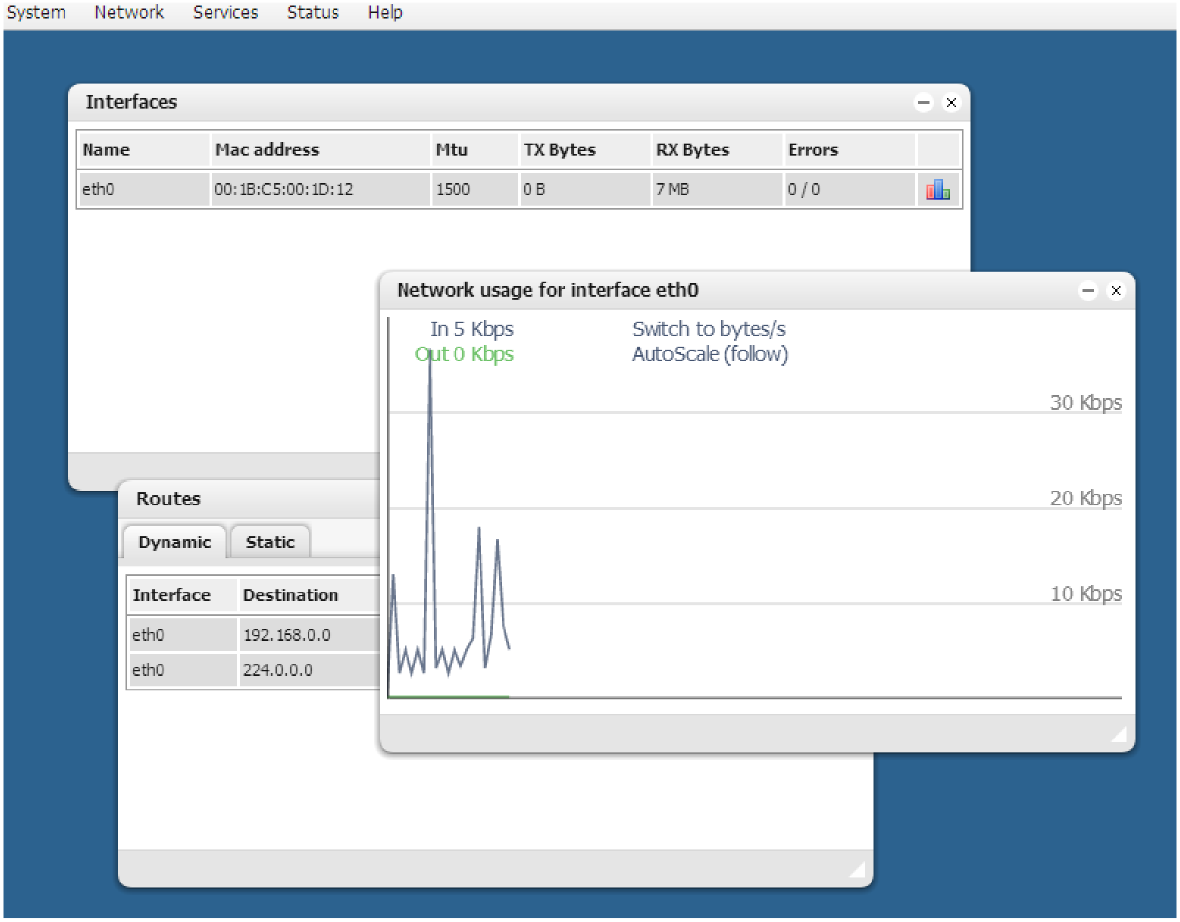
Lamp2 ✓

<u>Launching visualization on touch device (iPad in this case)</u>

- Make sure your iPad is connected wirelessly to the LogicMachine (either through separate access point or directly to Logic Machine’s USB WiFi adapter).
- In the browser enter Logic Machine’s IP (default 192.168.0.10).
- Click on the Touch Visualization icon.
- Save the application as permanent/shortcut in your iPad

4. System configuration

System configuration allows managing router functionality on KNX/EIB LogicMachine as well as do access control management, upgrade firmware, see network and system status and others.



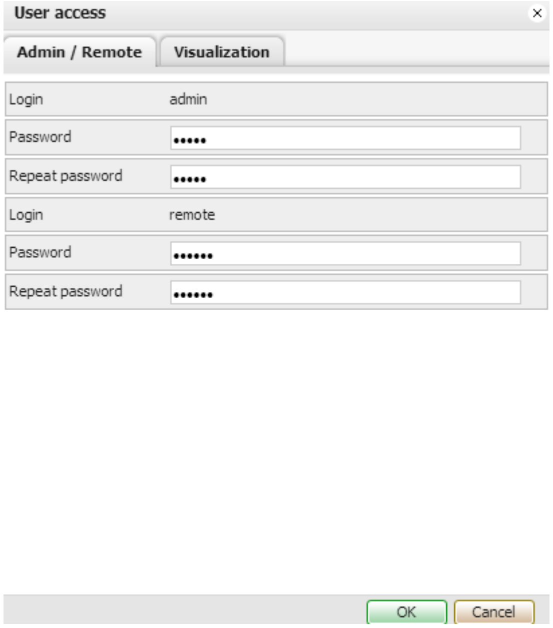
Name	Mac address	Mtu	TX Bytes	RX Bytes	Errors	
eth0	00:1B:C5:00:1D:12	1500	0 B	7 MB	0 / 0	📊

Interface	Destination
eth0	192.168.0.0
eth0	224.0.0.0

Login	**Password**
admin	admin

4.1. Changing password

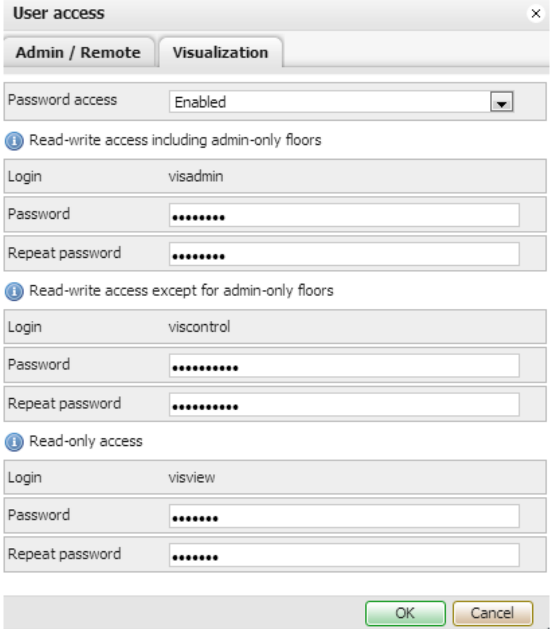
The login and password configuration window is located in *System* → *User access*.



User access

Admin / Remote | Visualization

Login	admin
Password	•••••
Repeat password	•••••
Login	remote
Password	••••••
Repeat password	••••••



User access

Admin / Remote | Visualization

Password access	Enabled

Read-write access including admin-only floors

Login	visadmin
Password	••••••••
Repeat password	••••••••

Read-write access except for admin-only floors

Login	viscontrol
Password	••••••••••
Repeat password	••••••••••

Read-only access

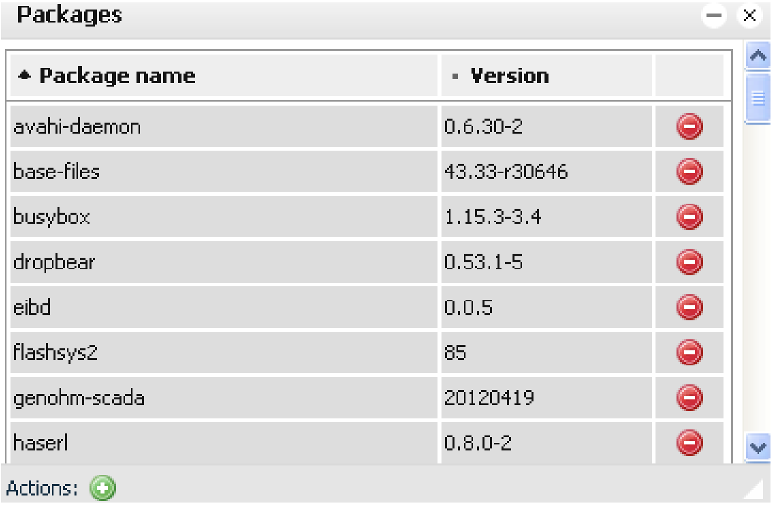
Login	visview
Password	•••••••
Repeat password	•••••••

Access control is separated in 3 tabs:

Admin/Remote – access parameters for *Logic Machine, Network Configuration, RSS* and *XML*
Visualization – access parameters for *Touch* and *User mode visualization*

4.2. Packages

System → *Packages* shows the packages installed in the system. You can add new packaged by pressing on +

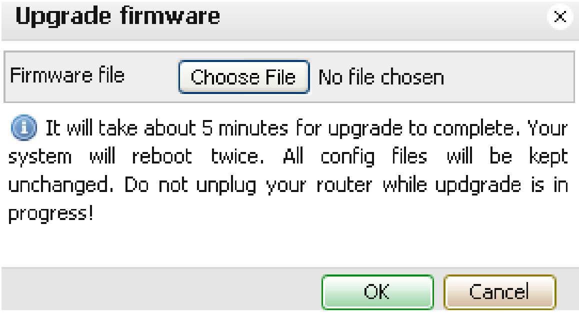


Packages

Package name	Version	
avahi-daemon	0.6.30-2	⊖
base-files	43.33-r30646	⊖
busybox	1.15.3-3.4	⊖
dropbear	0.53.1-5	⊖
eibd	0.0.5	⊖
flashsys2	85	⊖
genohm-scada	20120419	⊖
haserl	0.8.0-2	⊖

4.3. Upgrade firmware

System → *Upgrade* firmware is used to do a full upgrade of the system (both OS part as well as LogicMachine part).



Firmware file No file chosen

It will take about 5 minutes for upgrade to complete. Your system will reboot twice. All config files will be kept unchanged. Do not unplug your router while updgrade is in progress!

4.4. Reboot Logic Machine

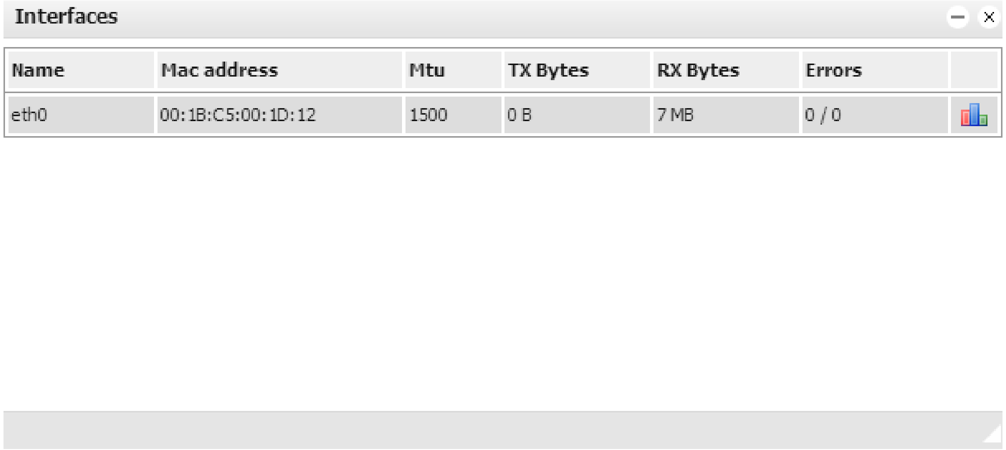
You can restart the LogicMachine by executing *System* →*Reboot* command.

4.5. Shutdown Logic Machine

You can shutdown the LogicMachine by executing *System* →*Shutdown* command. It is advisable to shutdown the system before plug out the power, because the database is saved safely.

4.6. Interface configuration

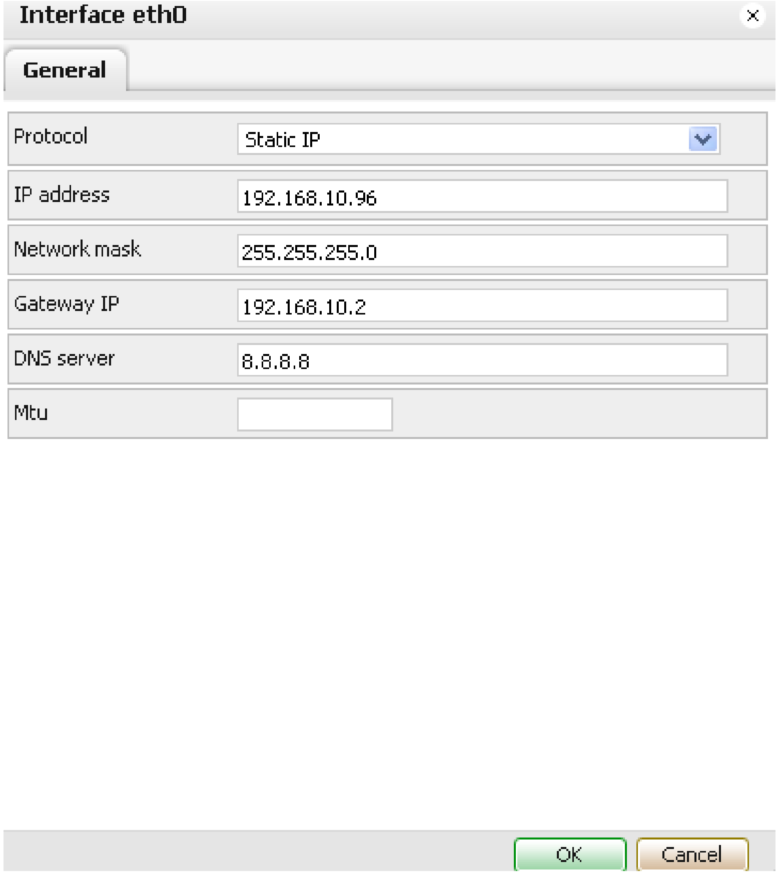
Ethernet interface is listed in the first tab. There are possibilities to disable/enable or to take a look at the traffic flow graph using special icons on the right side.



Interfaces

Name	Mac address	Mtu	TX Bytes	RX Bytes	Errors	
eth0	00:1B:C5:00:1D:12	1500	0 B	7 MB	0 / 0	📊

By clicking on the interface you get to the configuration.



Interface eth0

Protocol	Static IP
IP address	192.168.10.96
Network mask	255.255.255.0
Gateway IP	192.168.10.2
DNS server	8.8.8.8
Mtu	

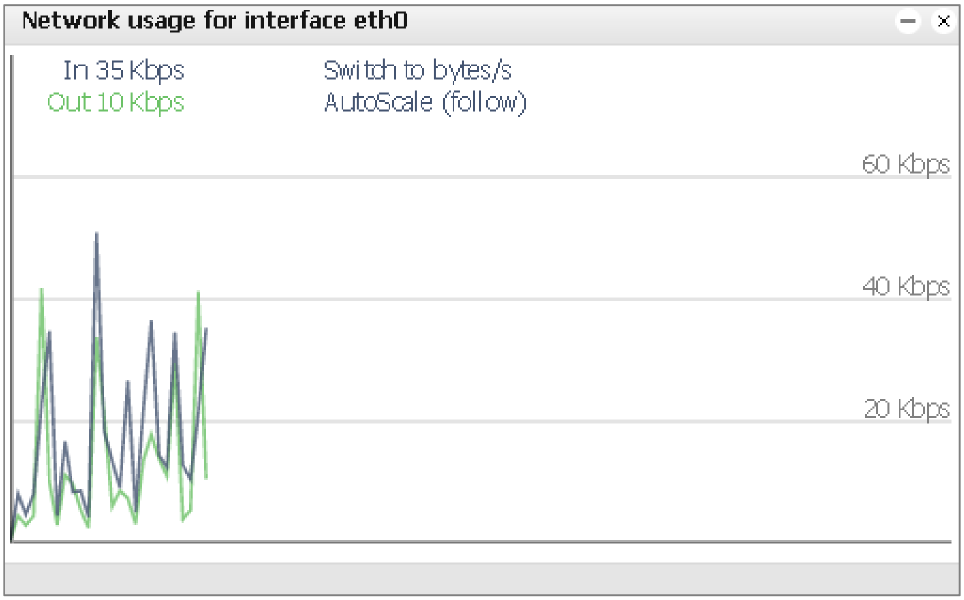
- ***Protocol***– specific protocol used for addressing
  Static IP – static IP address. By default 192.168.0.10
  DHCP – use DHCP protocol to get IP configuration.

Current IP– the IP address got from DHCP server. This field appears only if the IP address is given otherwise it's hidden.

- ***Network mask*** – network mask. By default 255.255.255.0 (/24)
- ***Gateway IP*** – gateway IP address
- ***DNS server*** – DNS server IP address
- ***MTU***– maximum transmission unit, the largest size of the packet which could be passed in the communication protocol. By default 1500

4.6.1. Ethernet interface data throughput graph

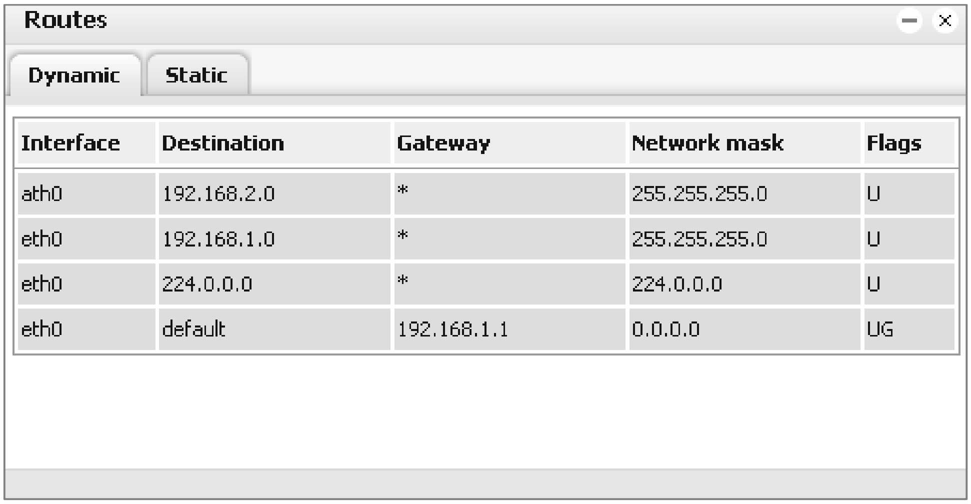
On the main window of the Ethernets tab, if you click on the 📊 button, a new window is opened. It draws a real-time graph of the traffic flow passing the interface (both In and Out). There is a possibility to switch the units of measurement – bytes/s or bytes/s.



4.7. Routing Table

System routing table is located in *Network→Routes* menu. The window is divided in two parts – Static routes and Dynamic routes.

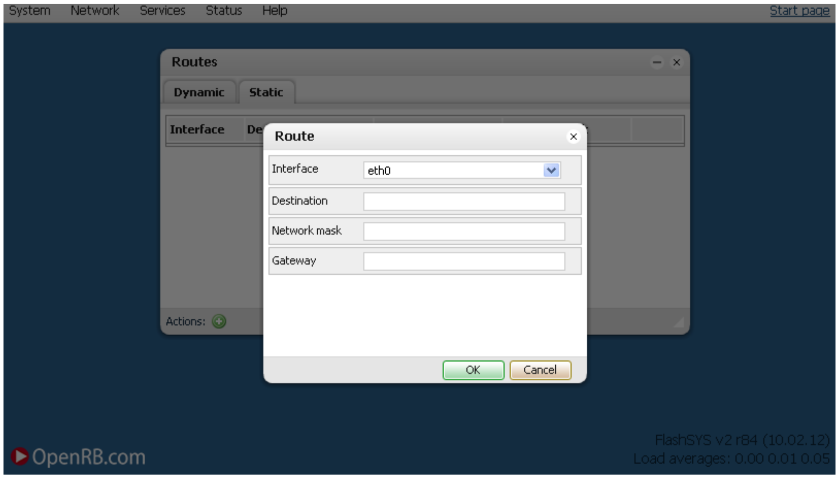
4.7.1. Dynamic routes



Interface	Destination	Gateway	Network mask	Flags
ath0	192.168.2.0	*	255.255.255.0	U
eth0	192.168.1.0	*	255.255.255.0	U
eth0	224.0.0.0	*	224.0.0.0	U
eth0	default	192.168.1.1	0.0.0.0	UG

- ➢ ***Interface*** – interface name
- ➢ ***Destination***– destination IP address
- ➢ ***Network mask*** – network mask
- ➢ ***Gateway*** – gateway IP address

4.7.2. Static routes



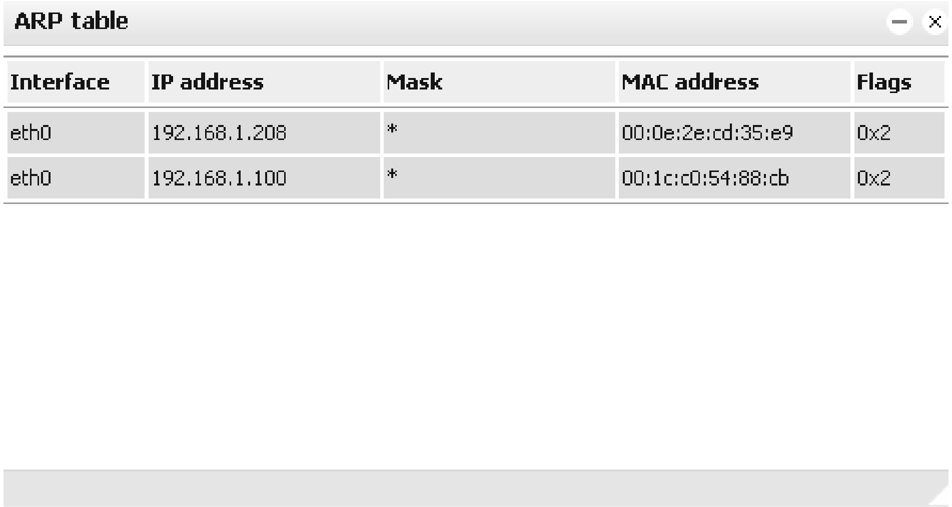
Interface	eth0
Destination	
Network mask	
Gateway	

FlashSYS v2 r84 (10.02.12)
Load averages: 0.00 0.01 0.05

- ➢ ***Interface*** – interface name
- ➢ ***Destination***– destination IP address
- ➢ ***Network mask*** – network mask
- ➢ ***Gateway*** – gateway IP address

4.8. ARP table

Address Resolution Protocol table is listed in *Network →ARP table*.

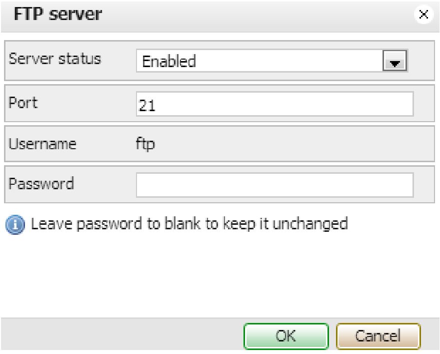


ARP table

Interface	IP address	Mask	MAC address	Flags
eth0	192.168.1.208	*	00:0e:2e:cd:35:e9	0x2
eth0	192.168.1.100	*	00:1c:c0:54:88:cb	0x2

4.9. FTP server

You can enable access to FTP server of LogicMachine by enabling this service in *Service →FTP Server*.



FTP server

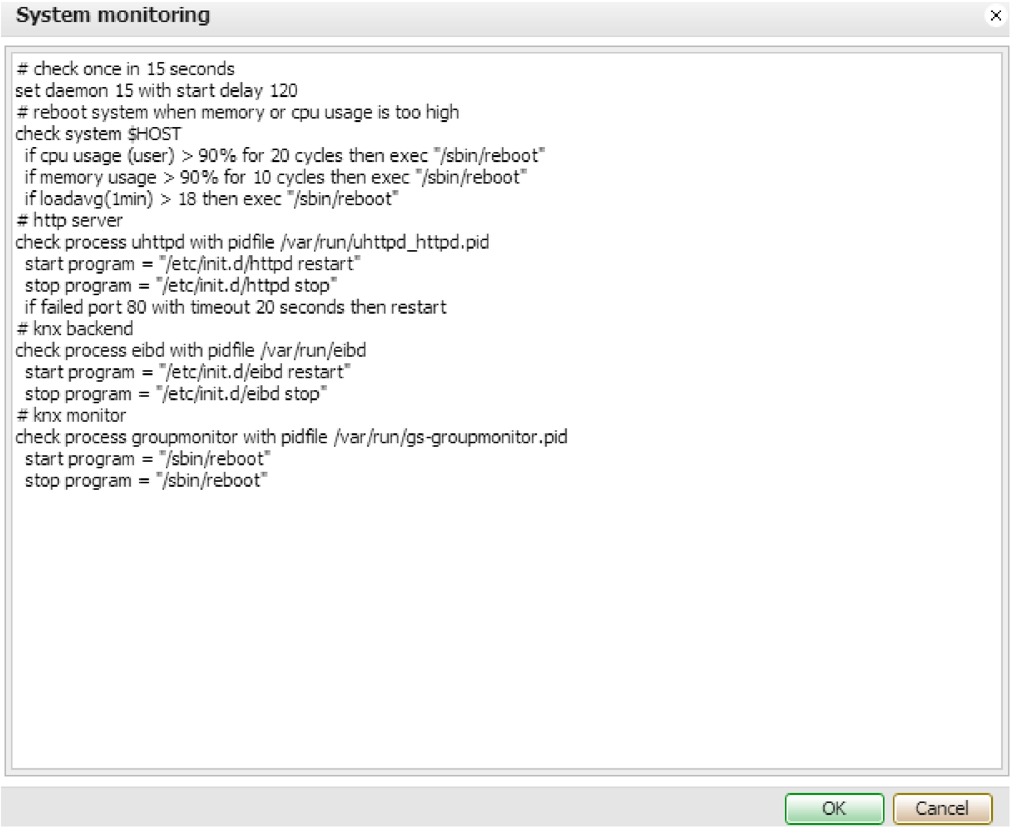
Server status	Enabled
Port	21
Username	ftp
Password	

Leave password to blank to keep it unchanged

- ➢ ***Server status*** – secure tunnel mode
- ➢ ***Port*** – port of the service
- ➢ ***Username –*** login name, *ftp*
- ➢ ***Password*** – password, length 4-20 symbols

4.10. System monitoring

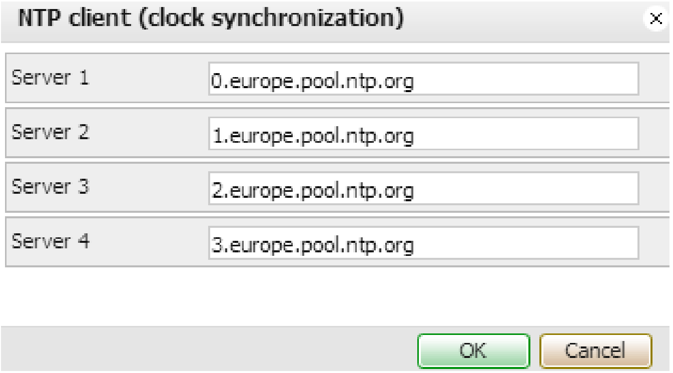
System monitoring is used to monitor system processes, hardware. In case of failure, the system will be rebooted or specific task restarted.



```
# check once in 15 seconds
set daemon 15 with start delay 120
# reboot system when memory or cpu usage is too high
check system $HOST
  if cpu usage (user) > 90% for 20 cycles then exec "/sbin/reboot"
  if memory usage > 90% for 10 cycles then exec "/sbin/reboot"
  if loadavg(1min) > 18 then exec "/sbin/reboot"
# http server
check process uhttpd with pidfile /var/run/uhttpd_httpd.pid
  start program = "/etc/init.d/httpd restart"
  stop program = "/etc/init.d/httpd stop"
  if failed port 80 with timeout 20 seconds then restart
# knx backend
check process eibd with pidfile /var/run/eibd
  start program = "/etc/init.d/eibd restart"
  stop program = "/etc/init.d/eibd stop"
# knx monitor
check process groupmonitor with pidfile /var/run/gs-groupmonitor.pid
  start program = "/sbin/reboot"
  stop program = "/sbin/reboot"
```

4.11. NTP client

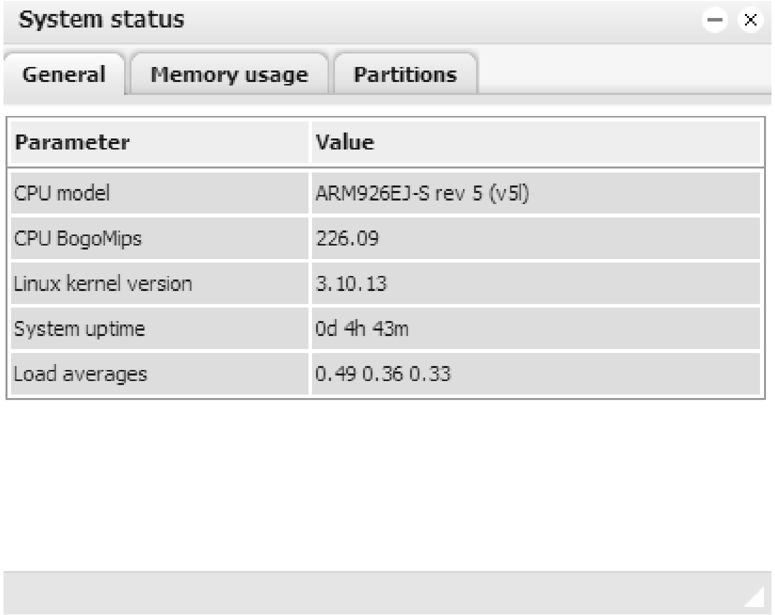
NTP servers can be specified in Service → NTP client window.



Server 1	0.europe.pool.ntp.org
Server 2	1.europe.pool.ntp.org
Server 3	2.europe.pool.ntp.org
Server 4	3.europe.pool.ntp.org

4.12. System status

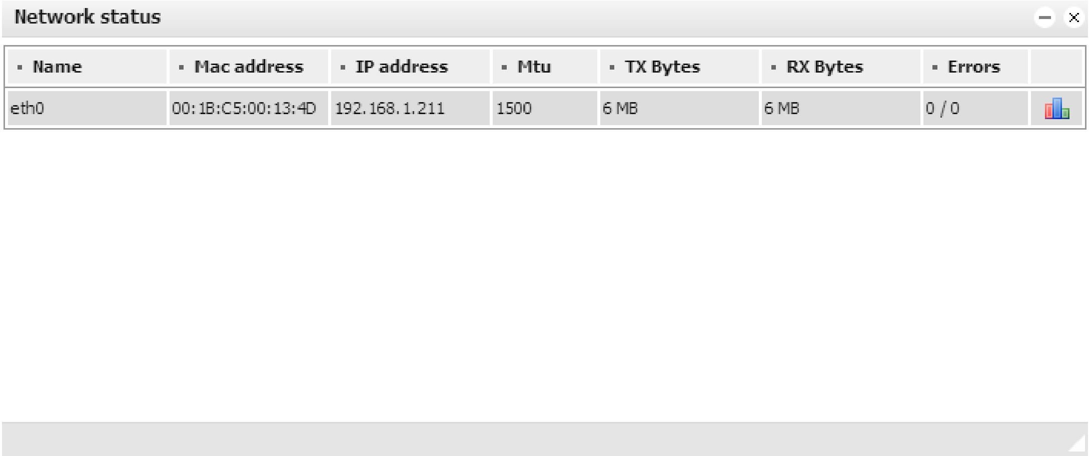
General system status with CPU usage, Memory usage, Partition information can be seen in *Status →System status* window.



Parameter	Value
CPU model	ARM926EJ-S rev 5 (v5l)
CPU BogoMips	226.09
Linux kernel version	3.10.13
System uptime	0d 4h 43m
Load averages	0.49 0.36 0.33

4.13. Network status

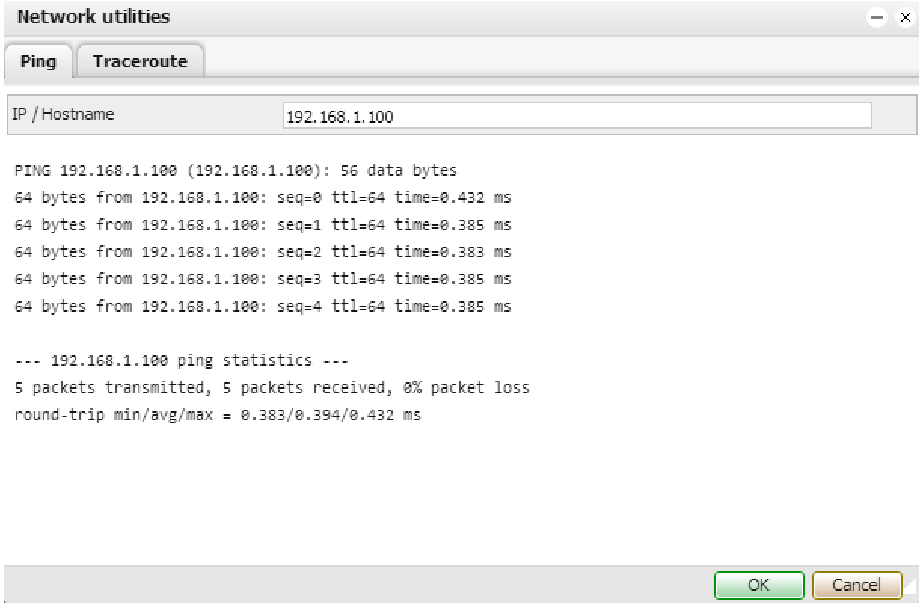
Network overview of IP setting and transferred data can be seen in *Status →Network status* window.



Name	Mac address	IP address	Mtu	TX Bytes	RX Bytes	Errors	
eth0	00:1B:C5:00:13:4D	192.168.1.211	1500	6 MB	6 MB	0 / 0	📊

4.14. Network utilities

Ping and *Traceroute* utilities are located in *Status* →*Network utilities* window. Both IP address and DNS names are accepted.

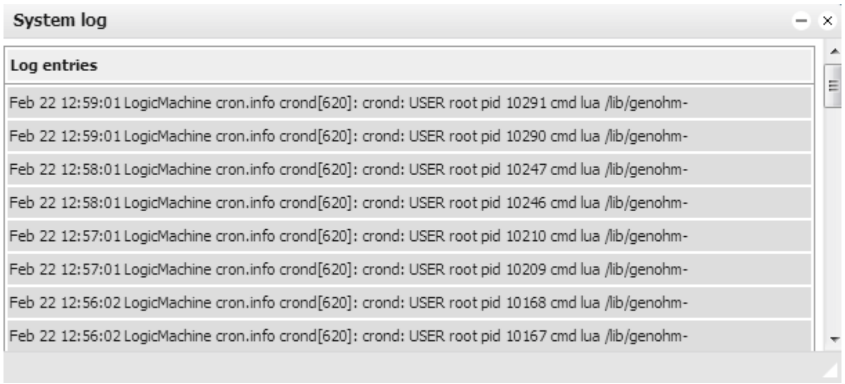


IP / Hostname 192.168.1.100

```
PING 192.168.1.100 (192.168.1.100): 56 data bytes
64 bytes from 192.168.1.100: seq=0 ttl=64 time=0.432 ms
64 bytes from 192.168.1.100: seq=1 ttl=64 time=0.385 ms
64 bytes from 192.168.1.100: seq=2 ttl=64 time=0.383 ms
64 bytes from 192.168.1.100: seq=3 ttl=64 time=0.385 ms
64 bytes from 192.168.1.100: seq=4 ttl=64 time=0.385 ms

--- 192.168.1.100 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.383/0.394/0.432 ms
```

4.15. System log

Operating system log is available in *Status* → *System log*.

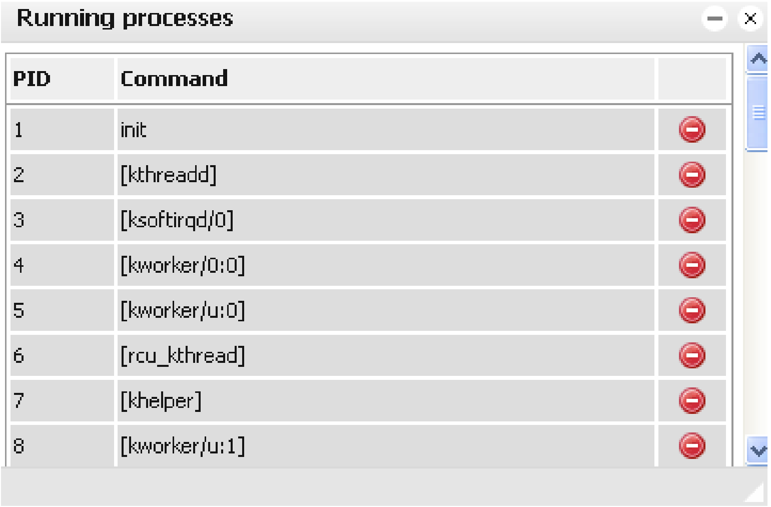


Log entries

```
Feb 22 12:59:01 LogicMachine cron.info crond[620]: crond: USER root pid 10291 cmd lua /lib/genohm-
Feb 22 12:59:01 LogicMachine cron.info crond[620]: crond: USER root pid 10290 cmd lua /lib/genohm-
Feb 22 12:58:01 LogicMachine cron.info crond[620]: crond: USER root pid 10247 cmd lua /lib/genohm-
Feb 22 12:58:01 LogicMachine cron.info crond[620]: crond: USER root pid 10246 cmd lua /lib/genohm-
Feb 22 12:57:01 LogicMachine cron.info crond[620]: crond: USER root pid 10210 cmd lua /lib/genohm-
Feb 22 12:57:01 LogicMachine cron.info crond[620]: crond: USER root pid 10209 cmd lua /lib/genohm-
Feb 22 12:56:02 LogicMachine cron.info crond[620]: crond: USER root pid 10168 cmd lua /lib/genohm-
Feb 22 12:56:02 LogicMachine cron.info crond[620]: crond: USER root pid 10167 cmd lua /lib/genohm-
```

4.16. Running processes

System running processes can be seen in *Status* →*Running processes* window.



Running processes

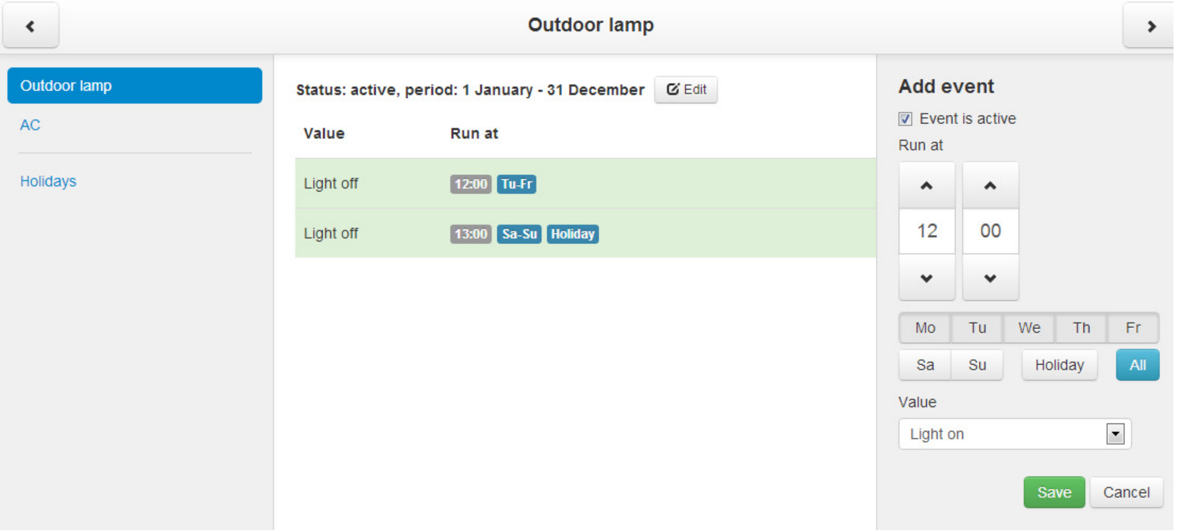
PID	Command	
1	init	⛔
2	[kthreadd]	⛔
3	[ksoftirqd/0]	⛔
4	[kworker/0:0]	⛔
5	[kworker/u:0]	⛔
6	[rcu_kthread]	⛔
7	[khelper]	⛔
8	[kworker/u:1]	⛔

5. User mode schedulers

User mode schedulers contains user-friendly interface for end-user to manage scheduler tasks, for example, specify thermostat values depending of the day of the week, time and holidays.

5.1. Events

Each scheduler is mapped to specific group address in administration panel (*see section 1.4 of this manual*).



Status: active, period: 1 January - 31 December

Value	Run at
Light off	12:00 Tu-Fr
Light off	13:00 Sa-Su Holiday

☑ Event is active

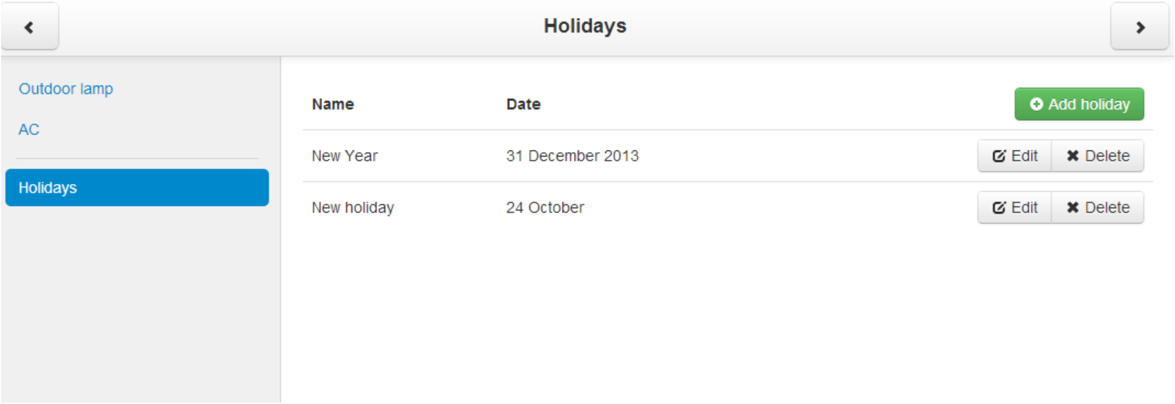
Run at: 12 00

Value: Light on

When adding the new task for specific scheduler you can specify day of the week, start time, value to send to the object.

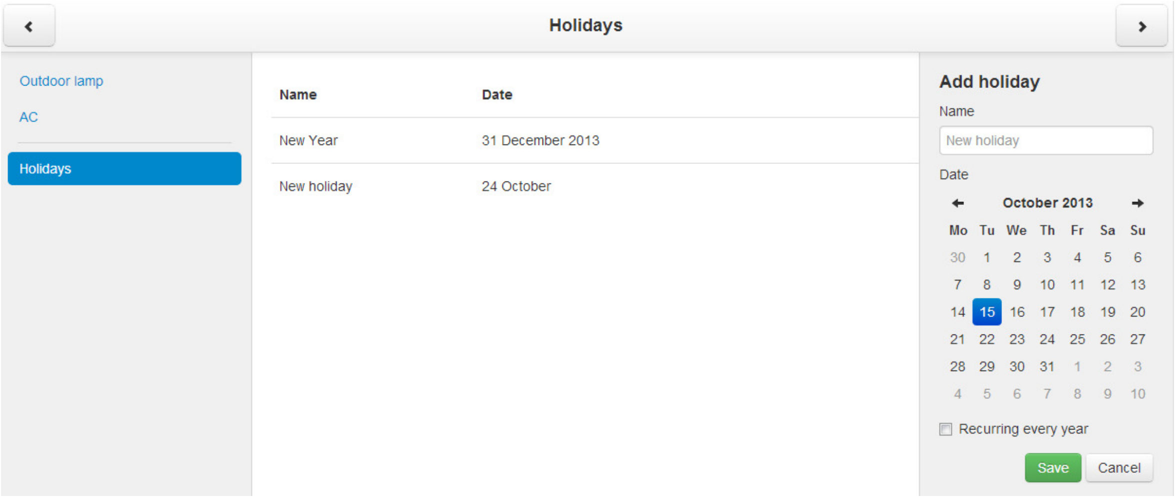
5.2. Holidays

In *Holidays* special days are specified which are then used adding new events.



Name	Date	
New Year	31 December 2013	Edit Delete
New holiday	24 October	Edit Delete

Click on *Add new holiday* button to specify a holiday.



Name	Date
New Year	31 December 2013
New holiday	24 October

Add holiday

Name

New holiday

Date

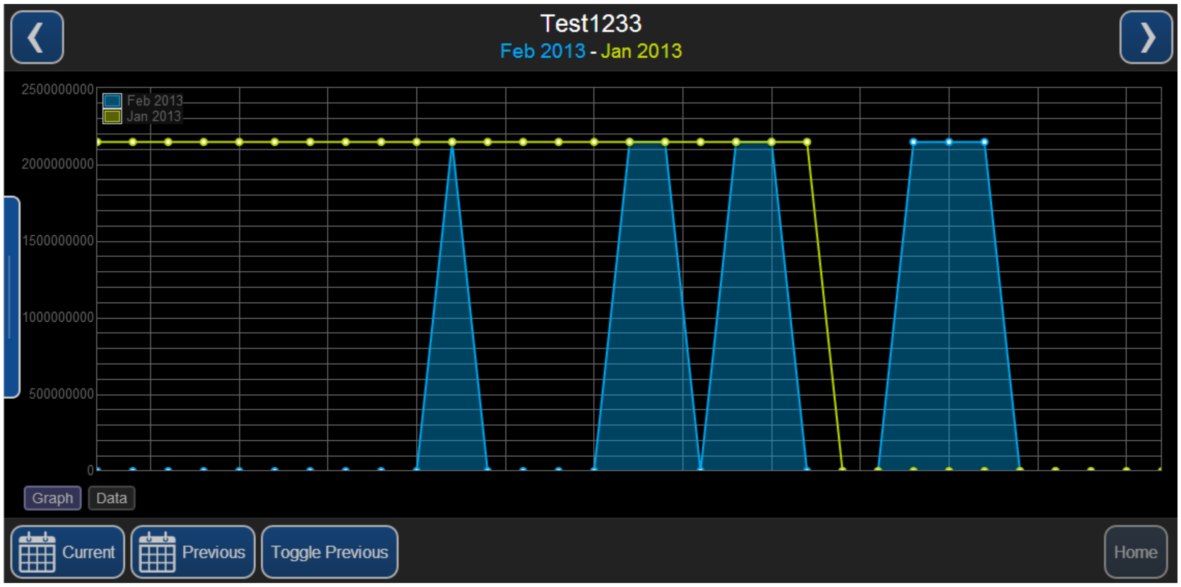
October 2013

Mo	Tu	We	Th	Fr	Sa	Su
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

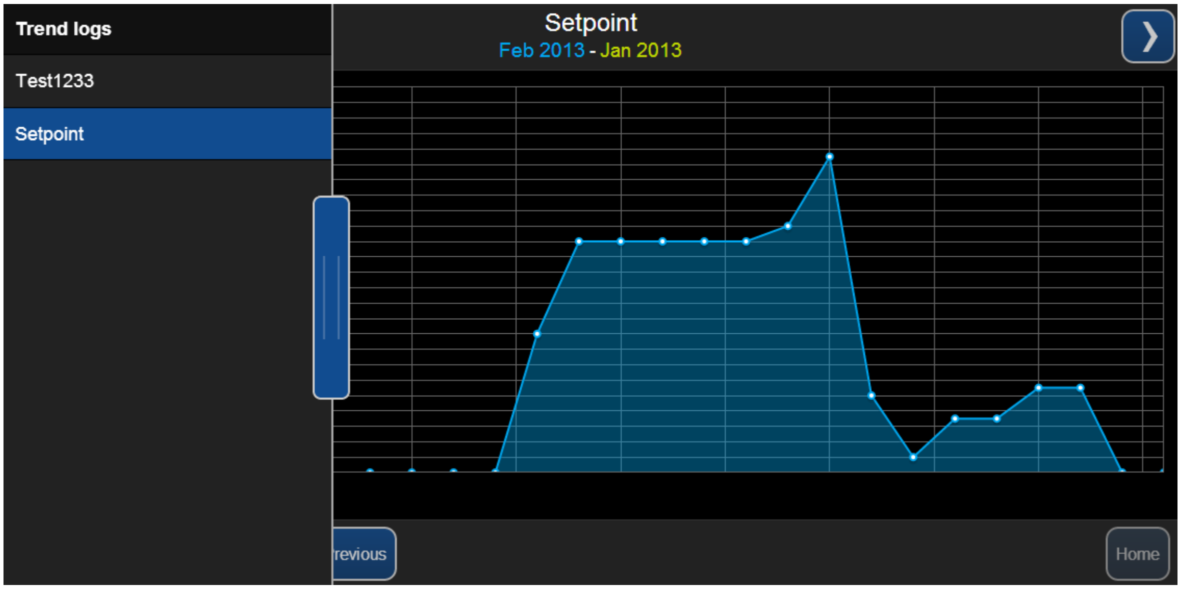
□ Recurring every year

6. Trend logs

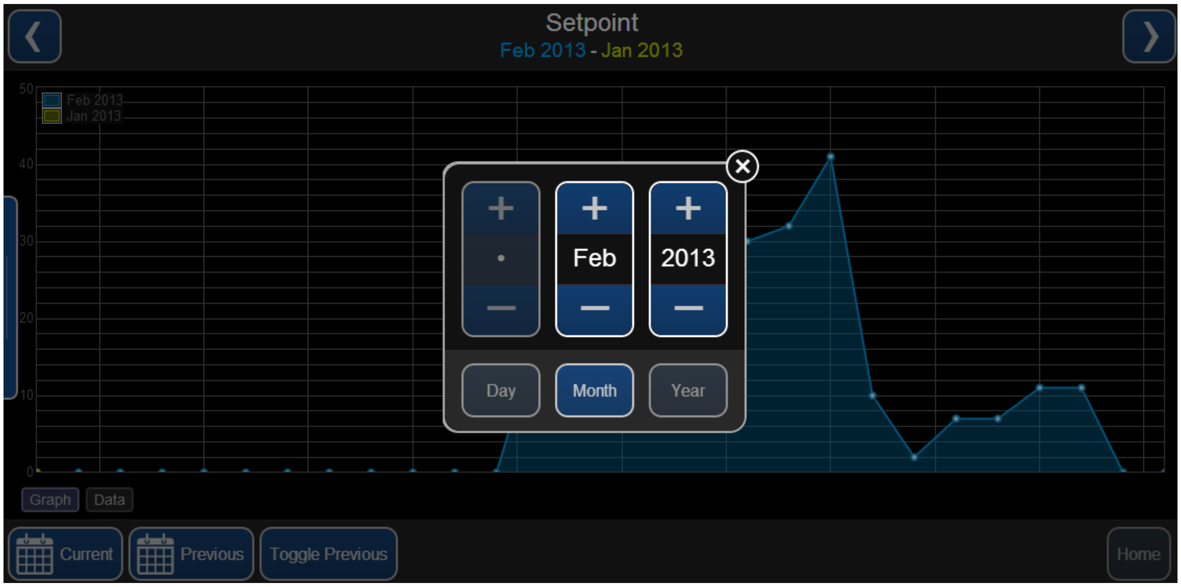
Trend logs are end user interface for trends (*defined in administrator interface in section 1.5*).



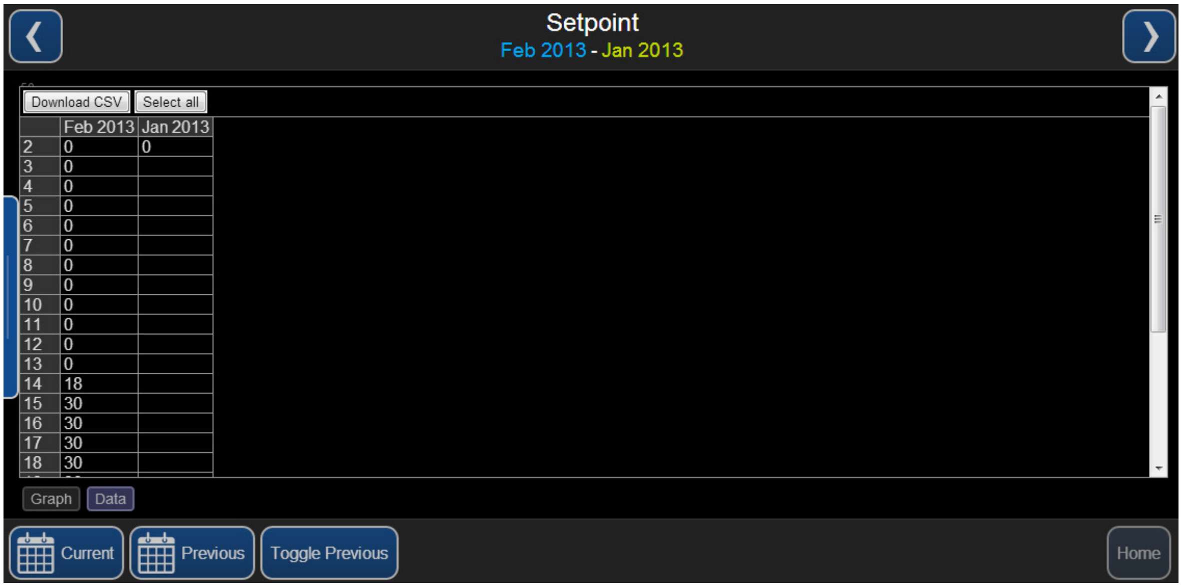
By clicking on the hidden blue menu you can change to different trends where each is mapped to a specific KNX group address.



Current – Current trend is drawn in blue, you can choose either to show Day, Month or Year view
Previous – previous time period, you can choose either to show Day, Month or Year view
Toggle previous – when enabled a yellow trend line appears showing *Previous* trend above *Current* trend
Home – LogicMachine home screen.



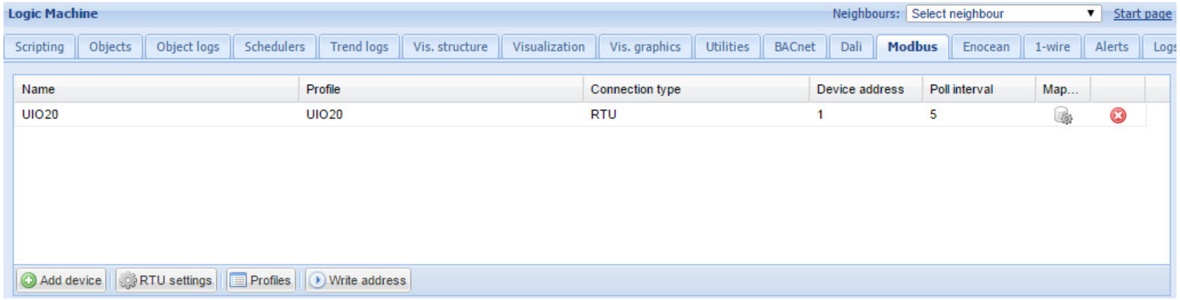
Data points can be shown also in a way of table which can be later exported as CSV file.



	Feb 2013	Jan 2013
2	0	0
3	0	
4	0	
5	0	
6	0	
7	0	
8	0	
9	0	
10	0	
11	0	
12	0	
13	0	
14	18	
15	30	
16	30	
17	30	
18	30	

7. Modbus RTU/TCP interconnection with LM

Modbus RTU is supported over RS485 interface. Modbus TCP is supported over Ethernet port. Modbus communication is done either from visual Modbus mapper for Modbus Master or through scripts for Modbus Slave.



Name	Profile	Connection type	Device address	Poll interval	Map...
UIO20	UIO20	RTU	1	5	

Modbus Master – user graphical mapper interface in Modbus tab
Modbus Slave – to use LM as Modbus Slave, disable Modbus RTU in Modbus→RTU settings, and use scripts for the communication

7.1. Modbus device profile

First thing you should do is to define Modbus device profile – it is a *.json file with the following structure e.g. a fragment from UIO20 device by Embedded Systems:

```
{
"manufacturer": "Embedded Systems",
"description": "Universal 16+4 I/O module",
"mapping": [
{ "name": "Output 1", "bus_datatype": "bool", "type": "coil", "address": 0, "writable": 1 },
{ "name": "Input 1", "bus_datatype": "float16", "type": "inputregister", "address": 0,
"value_multiplier": 0.001, "units": "V" }
]
}
```

Name – Object name, e.g. Output 2 (String, Required)
Bus_datatype - KNX object data type, key from dt table, e.g. float32 *(String/Number, Required)*
Type – Modbus register type, possible values: coil discreteinput register inputregister *(String, Required)*
Address – Register address (0-based*) (Number, Required)*
Writable - Set to true to enable writing to register if type is either coil or discreteinput *(Boolean)*
Datatype – Modbus value data type. If set, conversion will be done automatically. Possible values: uint16 int16 float16 uint32 int32 float32 uint64 int64 quad10k s10k *(String)*

Value_delta – New value is sent when the difference between previously sent value and current value is larger than delta. Defaults to 0 (send after each read) *(Number)*

Value_multiplier – Multiply resulting value by the specified number, value = value_base + value * value_multiplier *(Number)*

Value_bitmask – Bit mask to apply, shifting is done automatically based on least significant 1 found in the mask *(Number)*

Value_nan – Array of 16-bit integers. If specified and read operation returns the same array no further processing of value is done *(Array)*

Value_conv – Apply one of built-in conversion functions *(String, Internal)*

Value_custom – Name of a built-in enumeration or a list of key -> value mapping, resulting value will be 0 if key is not found *(String/Object)*

Internal – Not visible to user when set to true, should be used for scale registers *(Boolean)*

Units – KNX object units/suffix *(String)*

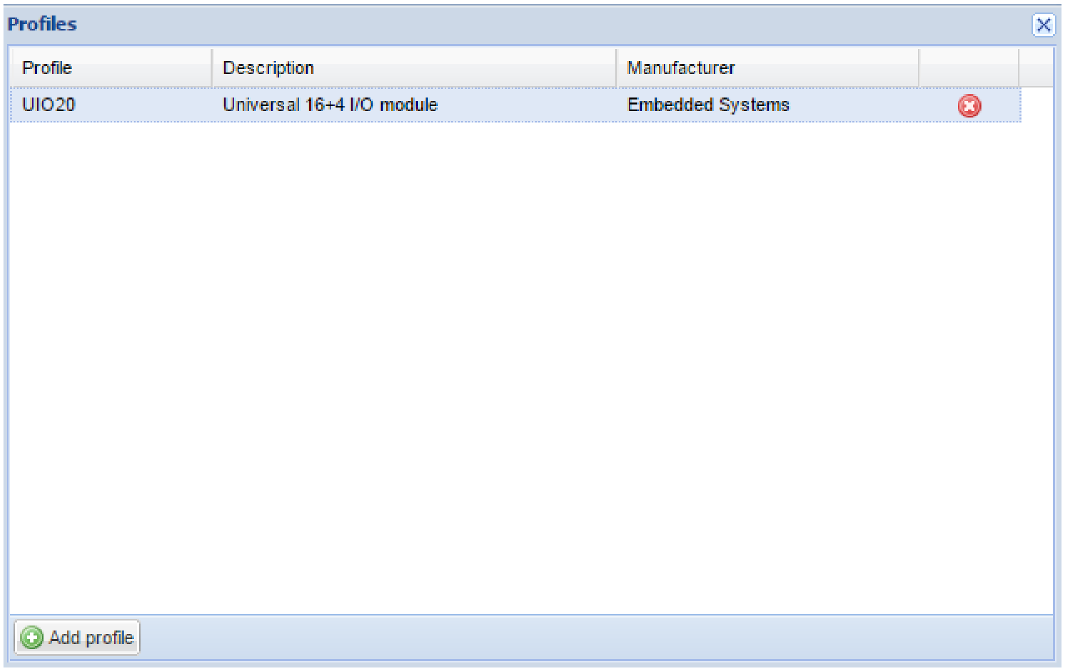
Address_scale – Address of register containing value scale, value = value * 10 ^ scale *(Number)*

Read_count – Number of register to read at once (for devices that only support reading of a specific block of registers) (*Number)*

Read_swap – Swap register order during conversion (endianness) *(Boolean)*

Read_offset – Position of first register of data from the block of registers (0-based) *(Number)*

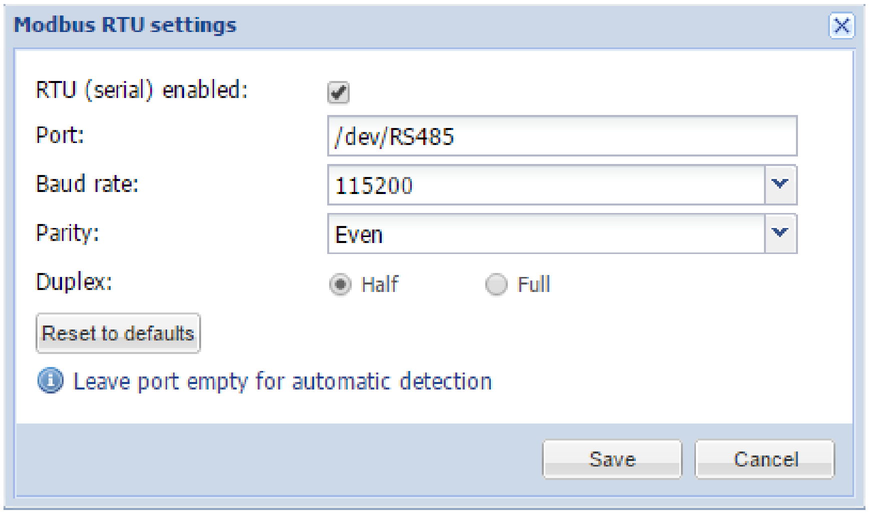
When the Modbus device profile file is created, upload it by clicking on *Profiles* button.



Profile	Description	Manufacturer	
UIO20	Universal 16+4 I/O module	Embedded Systems	⊗

7.2. *Modbus RTU settings*

If the communication is over Modbus RTU protocol (over RS-485 serial port), you should do base serial port settings by clicking on *RTU settings* button.



Modbus RTU settings

RTU (serial) enabled: ☑

Port: /dev/RS485

Baud rate: 115200

Parity: Even

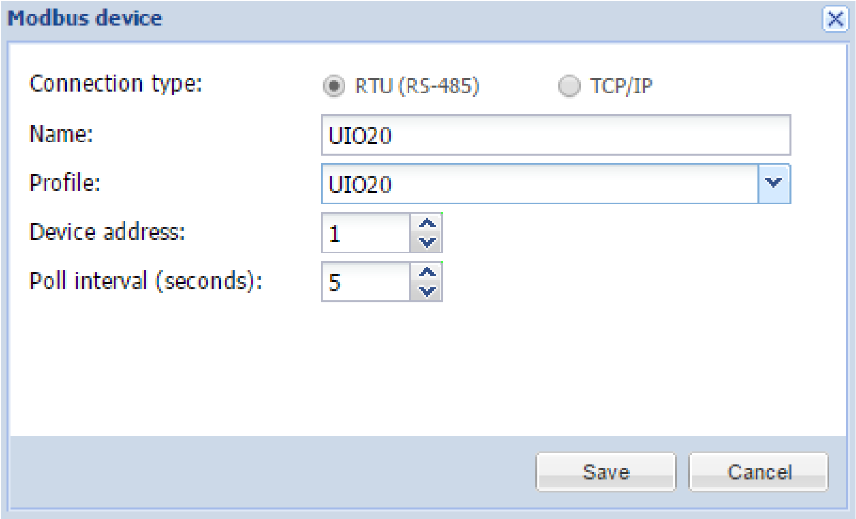
Duplex: ● Half ○ Full

Leave port empty for automatic detection

RTU (serial) enabled – define either RTU is enabled or not
Port – port name. In case of several RS-485 ports on the device, the name of the port is incremented by one, e.g. RS485-1, RS485-2, RS485-3 etc.
Baud rate – baud rate for the connection
Parity – parity for the connection
Duplex – specify either it is *half* or *full* duplex
Reset to defaults – reset RTU settings to defaults

7.3. Adding Modbus device

Once profiles are defined and RTU settings set, add Modbus device by clicking *Add device* button.



Modbus device

Connection type: ● RTU (RS-485) ○ TCP/IP

Name: UIO20

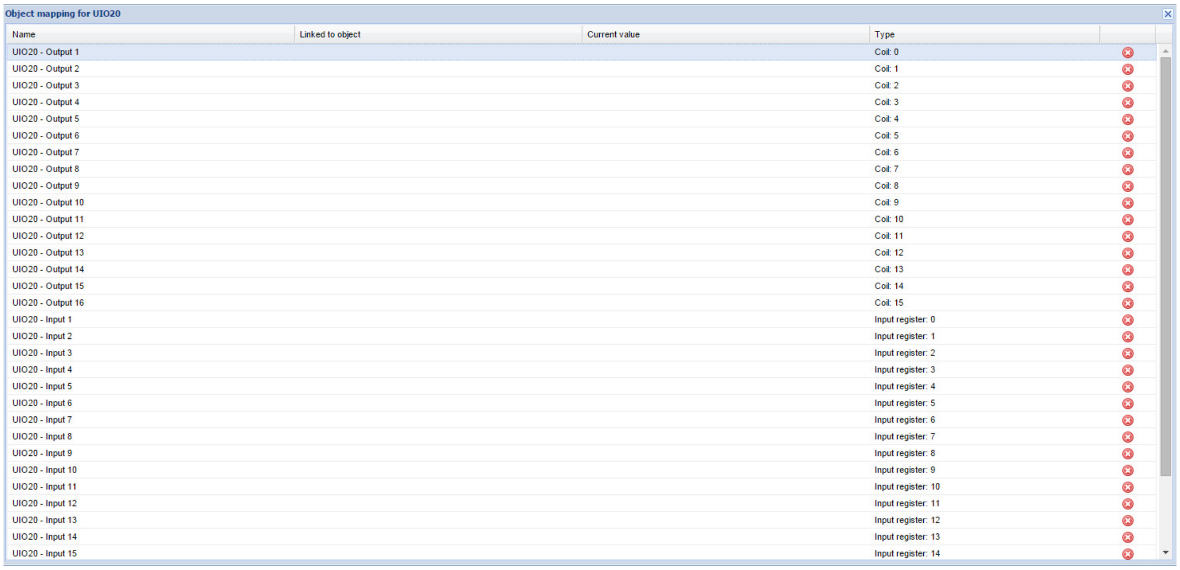
Profile: UIO20

Device address: 1

Poll interval (seconds): 5

Connection type – define either it is Modbus RTU or Modbus TCP connection
Name – name of the device
Profile – profile of the device
Device address – device address
Poll interval (seconds) – interval to poll the device
IP – IP address of the device in case Modbus TCP is used
Port – Communication port of the device in case Modbus TCP is used

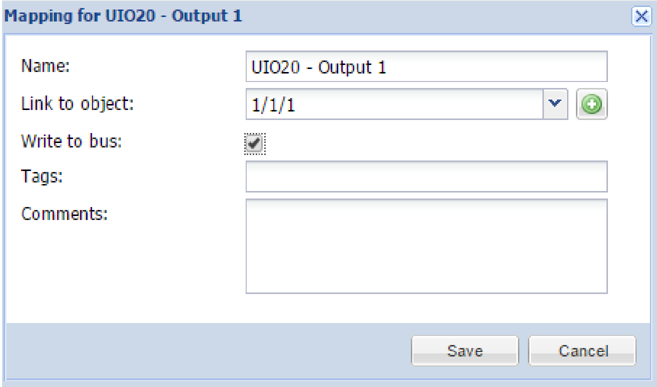
Once the device is added, you can do mapping to KNX addresses by clicking on icon. First, you see a list of all objects on the Modbus device.



Object mapping for UIO20

Name	Linked to object	Current value	Type	
UIO20 - Output 1			Coil: 0	⊗
UIO20 - Output 2			Coil: 1	⊗
UIO20 - Output 3			Coil: 2	⊗
UIO20 - Output 4			Coil: 3	⊗
UIO20 - Output 5			Coil: 4	⊗
UIO20 - Output 6			Coil: 5	⊗
UIO20 - Output 7			Coil: 6	⊗
UIO20 - Output 8			Coil: 7	⊗
UIO20 - Output 9			Coil: 8	⊗
UIO20 - Output 10			Coil: 9	⊗
UIO20 - Output 11			Coil: 10	⊗
UIO20 - Output 12			Coil: 11	⊗
UIO20 - Output 13			Coil: 12	⊗
UIO20 - Output 14			Coil: 13	⊗
UIO20 - Output 15			Coil: 14	⊗
UIO20 - Output 16			Coil: 15	⊗
UIO20 - Input 1			Input register: 0	⊗
UIO20 - Input 2			Input register: 1	⊗
UIO20 - Input 3			Input register: 2	⊗
UIO20 - Input 4			Input register: 3	⊗
UIO20 - Input 5			Input register: 4	⊗
UIO20 - Input 6			Input register: 5	⊗
UIO20 - Input 7			Input register: 6	⊗
UIO20 - Input 8			Input register: 7	⊗
UIO20 - Input 9			Input register: 8	⊗
UIO20 - Input 10			Input register: 9	⊗
UIO20 - Input 11			Input register: 10	⊗
UIO20 - Input 12			Input register: 11	⊗
UIO20 - Input 13			Input register: 12	⊗
UIO20 - Input 14			Input register: 13	⊗
UIO20 - Input 15			Input register: 14	⊗

Click on specific object to do mapping.



Mapping for UIO20 - Output 1

Name: UIO20 - Output 1

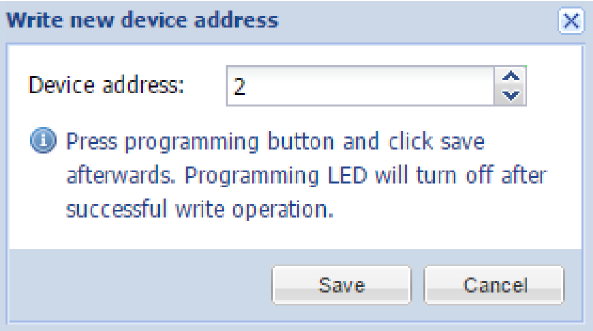
Link to object: 1/1/1

Write to bus: ☑

Tags:

Comments:

7.4. Program address for UIO20 Modbus device



Write new device address

Device address: 2

Press programming button and click save afterwards. Programming LED will turn off after successful write operation.

There is a separate Write address button to program address for UIO20 device. Press programming button and click save afterwards. Programming LED will turn off after successful write operation.

Once script is added, you can add the code in the Script Editor. There are lots of predefined code blocks in the Helpers.

7.5. Modbus Slave examples

Add the following code to *Common functions*

```
-- modbus proxy
mbproxy ={
-- supported function list
  functions ={
'readdo',
'readcoils',
'readdi',
'readdiscreteinputs',
'readao',
'readregisters',
'readai',
'readinputregisters',
'writebits',
'writemultiplebits',
'writeregisters',
'writemultipleregisters',
'reportslaveid',
'getcoils',
'getdiscreteinputs',
'getinputregisters',
'getregisters',
'setcoils',
'setdiscreteinputs',
'setinputregisters',
'setregisters',
},
-- new connecton init
  new =function()
require('rpc')
local mb =setmetatable({}, { __index = mbproxy })

    mb.slaveid =0
    mb.rpc = rpc.client('127.0.0.1', 28002, 'mbproxy')

for _, fn inipairs(mbproxy.functions)do
      mb[ fn ]=function(self, ...)
return mb:request(fn, ...)
end
end

return mb
end
}

-- set local slave id
function mbproxy:setslave(slaveid)
  self.slaveid = slaveid
end

-- send rpc request for a spefic function
function mbproxy:request(fn, ...)
local res, err = self.rpc:request({
    fn = fn,
    params ={ ... },
    slaveid = self.slaveid or0,
})

-- request error
if err then
returnnil, err
-- request ok
else
-- reply with an error
if res[ 1 ]==nilthen
returnnil, res[2]
-- normal reply
else
returnunpack(res)
end
end
end
```

<u>Handler (resident script with 0 delay) configuration</u>

1. *mb:open('/dev/RS485', 38400, 'E', 8, 1, 'H')*
set baudrate and other serial port parameters

2. *mb:setslave(10)*
set slave device id

3. *mb:setmapping(10, 10, 10, 10)*
set number coils, discrete inputs, holding registers and input registers

4.*mb:setwritecoilcb(function(coil, value)...*
callback function which is executed for each coil write

5. *mb:setwriteregistercb(function(coil, value)...*
callback function which is executed for each register write

Handler script example

```lua
-- modbus init
ifnot mb then
require('luamodbus')
  mb = luamodbus.rtu()
  mb:open('/dev/ttyS2', 38400, 'E', 8, 1, 'H')
  mb:connect()

-- slave id
  mb:setslave(10)

-- init slave storage for coils, discrete inputs, holding registers and input registers
  mb:setmapping(10, 10, 10, 10)

-- coil write callback
  mb:setwritecoilcb(function(coil, value)
if coil == 0 then
      grp.write('1/1/1', value, dt.bool)
else
      alert('coil: %d = %s', coil, tostring(value))
end
end)

-- register write callback
  mb:setwriteregistercb(function(register, value)
if register == 0 then
-- send value limited to 0..100
      grp.write('4/1/5', math.min(100, value), dt.scale)
else
      alert('register: %d = %d', register, value)
end
end)
end

-- server part init
ifnot server then
require('rpc')

-- incoming data handler
local handler =function(request)
local fn, res

    fn =tostring(request.fn)

ifnot mb[ fn ]then
```

```
return{nil, 'unknown function ' .. fn }
end

iftype(request.params)=='table'then
table.insert(request.params, 1, mb)
      res ={ mb[ fn ](unpack(request.params))}
else
      res ={ mb[ fn ](mb)}
end

return res
end

  server = rpc.server('127.0.0.1', 28002, 'mbproxy', handler, 0.01)
end

mb:handleslave()
server:step()
```

Example: event script which changes modbus slave coil (address 0)

Must be mapped to a group address with binary value.

```
value = event.getvalue()
mb = mbproxy.new()
mb:setcoils(0, value)
```

Example: event script which changes modbus slave register (address 5)

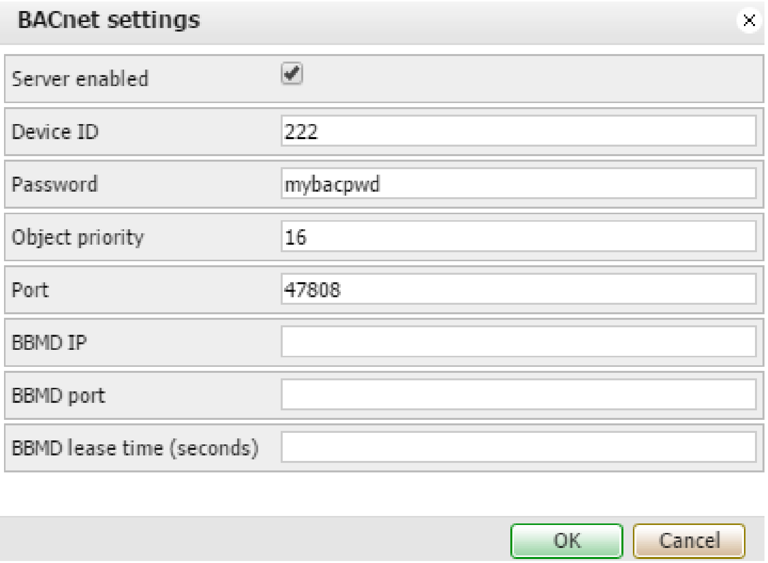
Must be mapped to a group address with scaling (0..100) value

```
value = event.getvalue()
mb = mbproxy.new()
mb:setregisters(5, value)
```

6. BACnet IP interconnection with LM

6.1. BACnet server mode: transparent data transfer to BACnet network

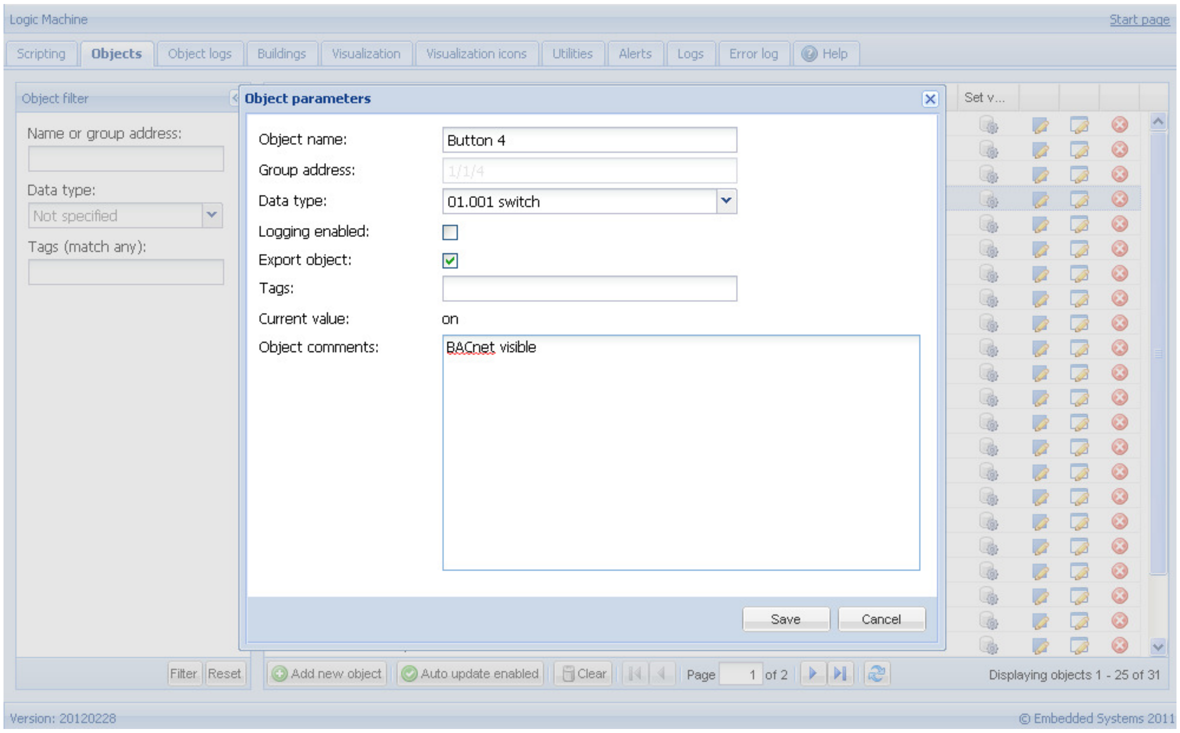
BACnet server specific configuration can be done in ***System Configuration* → *Network* →*BACnet Settings*.**



Server enabled	☑
Device ID	222
Password	mybacpwd
Object priority	16
Port	47808
BBMD IP	
BBMD port	
BBMD lease time (seconds)	

Server enabled – specify if BACnet server is enabled or not
Device ID – device ID in BACnet network
Password – device password
Object priority – object priority
Port – port number
BBMD IP – BACnet router IP. When router IP and port are set, LM will act as a foreign device and will attempt to register with BACnet router.
***BBMD port* –** BACnet router port. When router IP and port are set, LM will act as a foreign device and will attempt to register with BACnett router
BBMD lease time (seconds) – registration resend interval

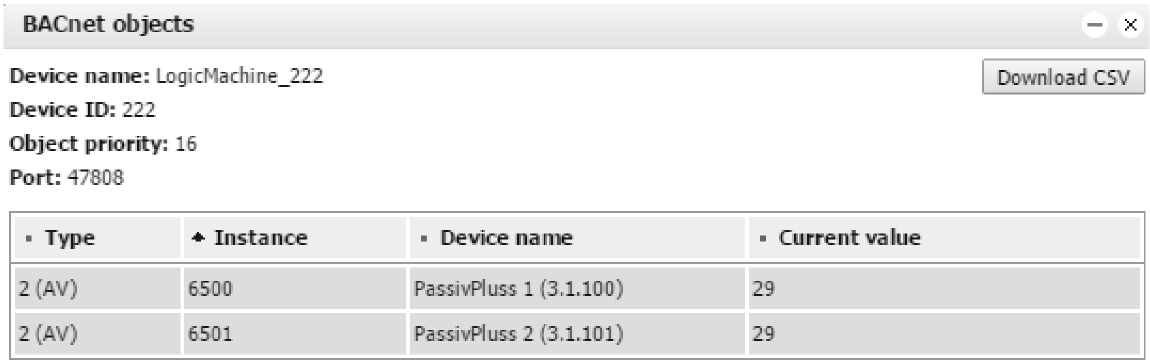
To make KNX/EIB objects BACnet readable/writable, mark necessary objects in LogicMachine as “Export object”. Binary objects will appear as Binary Values, other numeric values will appear as Analog Values. Other types are not currently supported. KNX bus write changes priority array value at configured object priority index



Object parameters

Object name:	Button 4
Group address:	1/1/4
Data type:	01.001 switch
Logging enabled:	□
Export object:	☑
Tags:	
Current value:	on
Object comments:	BACnet visible

In ***System Configuration → Network →BACnet objects*** you can see marked objects on LogicMachine which are sent to BACnet network.



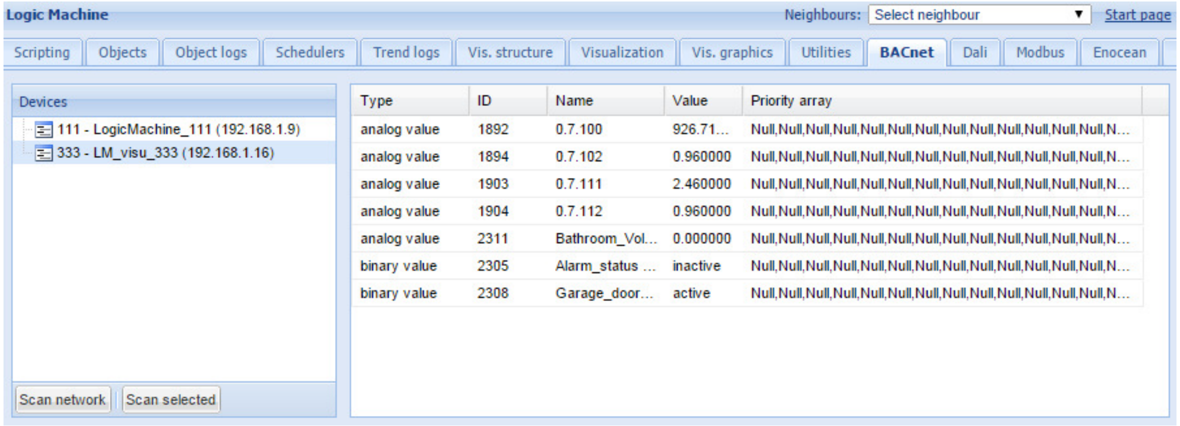
BACnet objects

Device name: LogicMachine_222
Device ID: 222
Object priority: 16
Port: 47808

Type	Instance	Device name	Current value
2 (AV)	6500	PassivPluss 1 (3.1.100)	29
2 (AV)	6501	PassivPluss 2 (3.1.101)	29

6.2. BACnet client mode

Normally this mode is used to interconnect LogicMachine, for example, with VRV systems over BACnet IP protocol. The settings are available in BACnet tab.



111 - LogicMachine_111 (192.168.1.9)
333 - LM_visu_333 (192.168.1.16)

Type	ID	Name	Value	Priority array
analog value	1892	0.7.100	926.71...	Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,N...
analog value	1894	0.7.102	0.960000	Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,N...
analog value	1903	0.7.111	2.460000	Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,N...
analog value	1904	0.7.112	0.960000	Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,N...
analog value	2311	Bathroom_Vol...	0.000000	Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,N...
binary value	2305	Alarm_status ...	inactive	Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,N...
binary value	2308	Garage_door...	active	Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,Null,N...

By clicking on *Scan Network* button you can see a list of BACnet server devices on the network. With *Scan Selected* you can rescan specific BACnet server for respective objects.

Mapping to KNX objects currently is done over scripting.

Before using any BACnet function, you must include the library:
require('bacnet')

Read current value of binary or analog object:
bacnet.readvalue(device_id, object_type, object_id)

Read binary object:
value = bacnet.readvalue(127001, 'binary value', 2305)

Read analog object:
value = bacnet.readvalue(127001, 'analog value', 2306)

Write new value to binary or analog object priority array:
bacnet.write = function(device_id, object_type, object_id, value, priority)
Value can be nil, boolean, number or a numeric string
Priority parameter is optional, lowest priority is used by default

Set binary object value:
bacnet.write(127001, 'binary value', 2305, true)

Set analog object value:
bacnet.write(127001, 'analog value', 2306, 22.5)

Set binary object value at priority 12:
bacnet.write(127001, 'binary value', 2305, true, 12)

Set analog object value at priority 10:
bacnet.write(127001, 'analog value', 2306, 22.5, 10)

Clear binary object value at priority 12:
bacnet.write(127001, 'binary value', 2305, nil, 12)

7. 1-wire configuration

1-wire is a bus technology which is built based on client-server topology and allowing to connect up to 300 devices to one controller. It is either 2-wire or 3-wire bus installation. In case of 2-wire system, a parasitic powering is used directly from the bus, normally up to 20 devices can work in this way. In case of bigger amount of 1-wire sensors, you can use LogicMachine 5V DC output to power 1-wire devices.

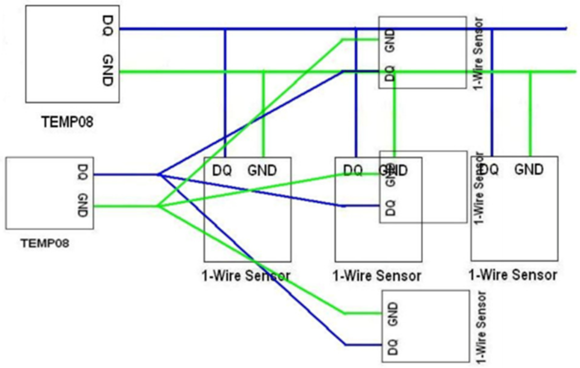
Advantages of 1-wire over KNX:

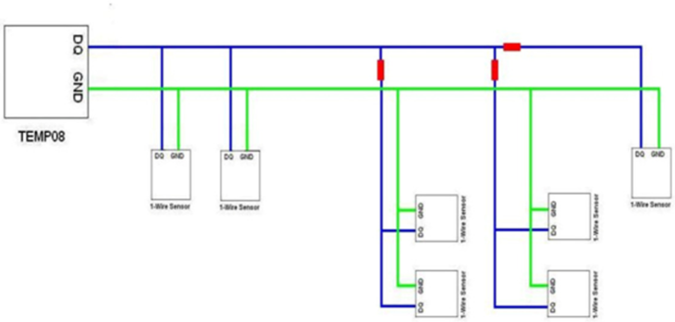
- No need in ETS
- Very cost-effective
- You can use the same wiring as KNX does and connect all standard sensors

Advantages of 1-wire over resistive sensors:

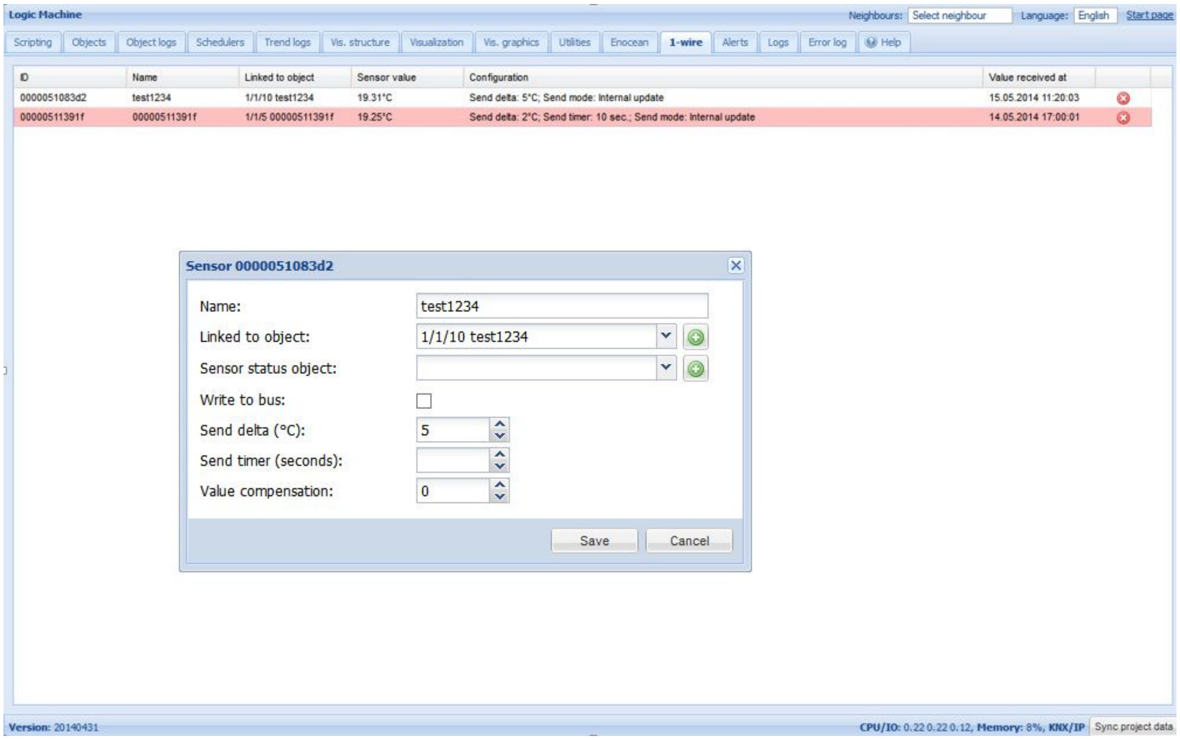
- Substantial savings on equipment
- Easier connection diagram allows to reduce the complexity of laying wiring
- Extension possibility: connection of additional sensors without changing basic wiring
- Ability of remote monitoring of sensors (open circuit, short circuit etc.)
- No need to take into account the resistance of conductors like in the circuit with resistive sensors

1-wire connection diagrams:





Once 1-wire sensors are connected to the 1-wire interface of LogicMachine



ID	Name	Linked to object	Sensor value	Configuration	Value received at
0000051083d2	test1234	1/1/10 test1234	19.31°C	Send delta: 5°C; Send mode: Internal update	15.05.2014 11:20:03
00000511391f	00000511391f	1/1/5 00000511391f	19.25°C	Send delta: 2°C; Send timer: 10 sec.; Send mode: Internal update	14.05.2014 17:00:01

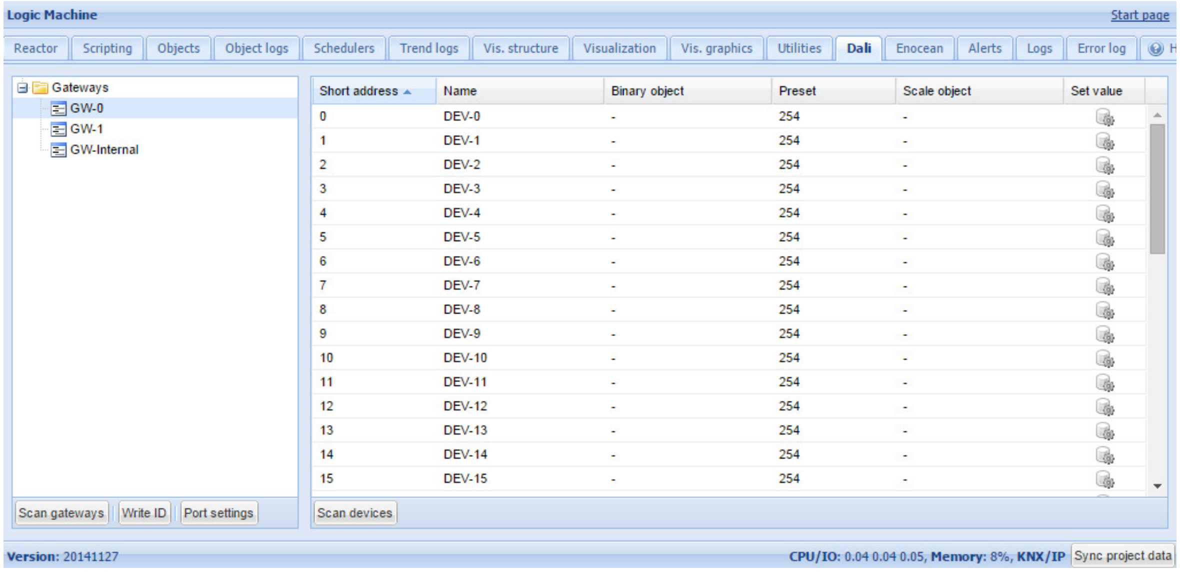
Sensor 0000051083d2

Name:	test1234
Linked to object:	1/1/10 test1234
Sensor status object:	
Write to bus:	□
Send delta (°C):	5
Send timer (seconds):	
Value compensation:	0

Name – name of the 1wire device
Linked to object – mapped KNX object
Sensor status object – mapped KNX status object
Write to bus – define either to write telegram in KNX bus on read value
Send delta – define either to send delta of temperature sensor
Send timer (seconds) – define interval in which send the measurement
Value compensation – compensate value of the reading of temperature

8. DALI configuration

LogicMachine4 and Reactor V2 have DALI Master built-in. We recommends to connect no more than 32 ballasts to one DALI line. If more ballasts are necessary to connect, you can use external DALI-RS-485 interfaces and connect to RS-485 port.



Short address ▲	Name	Binary object	Preset	Scale object	Set value
0	DEV-0	-	254	-	
1	DEV-1	-	254	-	
2	DEV-2	-	254	-	
3	DEV-3	-	254	-	
4	DEV-4	-	254	-	
5	DEV-5	-	254	-	
6	DEV-6	-	254	-	
7	DEV-7	-	254	-	
8	DEV-8	-	254	-	
9	DEV-9	-	254	-	
10	DEV-10	-	254	-	
11	DEV-11	-	254	-	
12	DEV-12	-	254	-	
13	DEV-13	-	254	-	
14	DEV-14	-	254	-	
15	DEV-15	-	254	-	

Version: 20141127

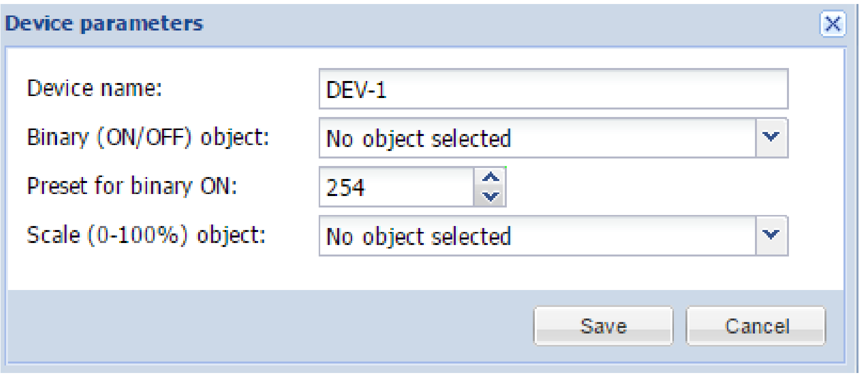
CPU/IO: 0.04 0.04 0.05, **Memory:** 8%, **KNX/IP**

- ***Scan gateways*** - scans for currently connected gateways, address mapping for missing devices is deleted automatically
- ***Write ID*** - allows setting a unique address for each gateway
- ***Scan devices*** - scans for currently connected DALI devices to the selected gateway, assigns short address automatically. You can also set not to overwrite existing addresses during scan
- ***Port settings*** – serial port name if there are external DALI-RS-485 interfaces connected

For each DALI device, you can set a custom name and map to binary on/off and scale object. This allows communication with DALI devices from KNX bus and visualization without any additional scripts.

8.1. DALI object mapping

Once DALI objects are scanned, you can click on corresponding object and perform the configuration.



Device name: DEV-1
Binary (ON/OFF) object: No object selected
Preset for binary ON: 254
Scale (0-100%) object: No object selected

Device name – name of the DALI device
Binary (ON/OFF) object – map to KNX binary object
Preset for binary ON – preset on binary ON
Scale (0-100%) object – map to KNX scale object

You can set up specific value by clicking on this icon ⚙

8.2. Access DALI bus from scripts

If you want to access DALI devices from other scripts, you can use **dalicmd** function.

dalicmd(gateway, command, parameters)

Parameters:
gateway - gateway id (0..63)
command - DALI command to execute
parameters - Lua table:
addrtype - address type, only required for addressable commands, possible values: *short group broadcast*
address - short or group address
value - additional value to send

Example:

Use gateway with id 1, switch all ballasts off, set ballast with short address 5 to full on

```
require('user.dali')


dalicmd(1, 'arc', { addrtype = 'broadcast', value = 0 })

dalicmd(1, 'arc', { addrtype = 'short', address = 5, value = 254 })
```

DALI commands

In the list below please see description of parameters of function **dalicmd**.

Command – dalicmd() parameter command
Description – description of command
Addressable – + means that this is addressable command, dalicmd() requires existence of parameter cmddats and addrtype_V’broadcast’. Empty field means that the command is non-addressable and parameter cmddats may be unused
Value – interval of values of parameter value_V.

Command	**Description**	**Addressable**	**Reply**	**Value**
arc	direct arc power control	+		0..254
off	turn off	+		
up	turn on	+		
down	down	+		
stepup	step up	+		
stepdown	step down	+		
recallmin	recall max level	+		
recallmax	recall min level	+		
stepdownoff	step down and off	+		
stepupon	on and step up	+		
gotoscene	go to scene			0..15
reset	reset	+		
storeactual	store actual level in the dtr	+		
storemax	store the dtr as max level	+		
storemin	store the dtr as min level	+		
storesystemfailure	store the dtr as system failure level	+		
storepoweron	store the dtr as power on level	+		
storefadetime	store the dtr as fade time	+		
storefaderate	store the dtr as fade rate	+		
storescene	store the dtr as scene	+		0..15
removescene	remove from scene	+		0..15
addtogroup	add to group	+		0..15
removefromgroup	remove from group	+		0..15
storeshortaddress	store dtr as short address	+		
querystatus	query status	+	+	
queryballast	query ballast	+	+	
querylampfailure	query lamp failure	+	+	
querylamppoweron	query lamp power on	+	+	
querylimiterror	query limit error	+	+	
queryresetstate	query reset state	+	+	
querymissingshort	query missing short address	+	+	

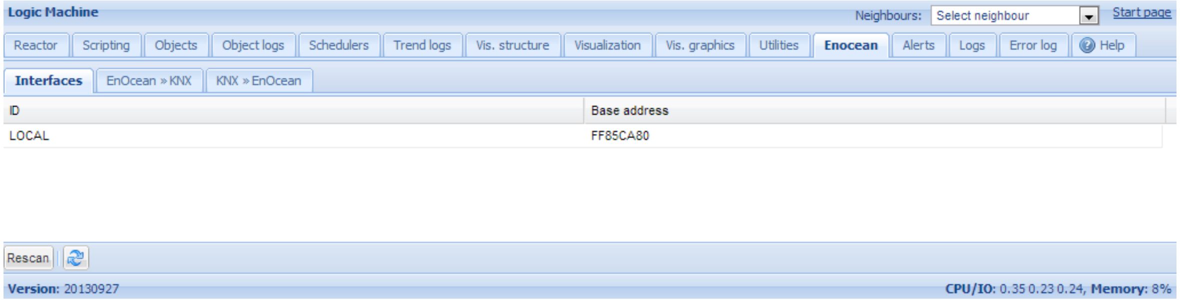
queryversion	query version number	+	+	
querydtr	query content dtr	+	+	
querydevicetype	query device type	+	+	
queryphysicalmin	query physical minimum level	+	+	
querypowerfailure	query power failure	+	+	
queryactual	query actual level	+	+	
querymax	query max level	+	+	
querymin	query min level	+	+	
querypoweron	query power on level	+	+	
querysystemfailure	query system failure level	+	+	
queryfadetimerate	query fade time / fade rate	+	+	
queryscene	query scene level (scenes 0-15)	+	+	0..15
querygroupslow	query groups 0-7	+	+	
querygroupshigh	query groups 8-15	+	+	
queryrandomaddrh	query random address (h)	+	+	
queryrandomaddrm	query random address (m)	+	+	
queryrandomaddrl	query random address (l)	+	+	
terminate	terminate			
setdtr	set data transfer register (dtr)			0..255
initialise	initialise			
randomise	randomise			
compare	compare		+	
withdraw	withdraw			
searchaddrh	set search address (h)			0..255
searchaddrm	set search address (m)			0..255
searchaddrl	set search address (l)			0..255
programshortaddr	program short address			0..63
verifyshortaddr	verify short address		+	0..63
queryshortaddr	query short address		+	
physicalselection	physical selection			
enabledevicetype	enable device type x			0..255

9. EnOcean interconnection with LogicMachine

Logic Machine3 Reactor and Reactor V2 have EnOcean transceiver built-in with no limitation on supported count of devices.

9.1. EnOcean interfaces

EnOcean interface Base address can be found in *Enocean→Interfaces* tab.

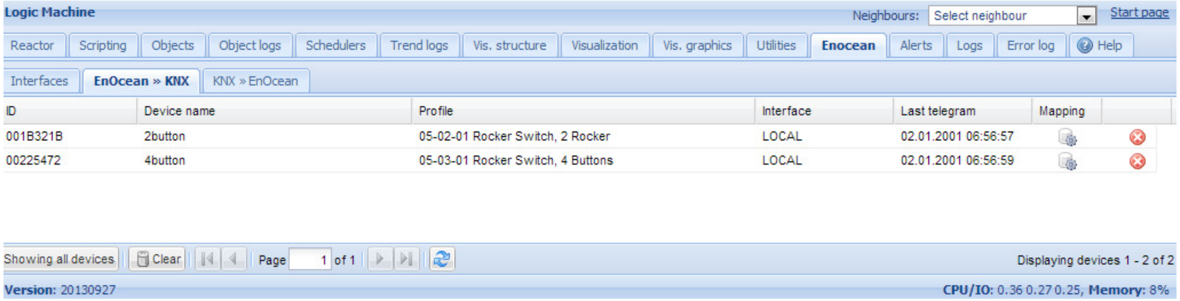


ID	Base address
LOCAL	FF85CA80

Version: 20130927 **CPU/IO:** 0.35 0.23 0.24, **Memory:** 8%

9.2. EnOcean to KNX mapping

All telegrams received from EnOcean devices appears in *Enocean→KNX* section.

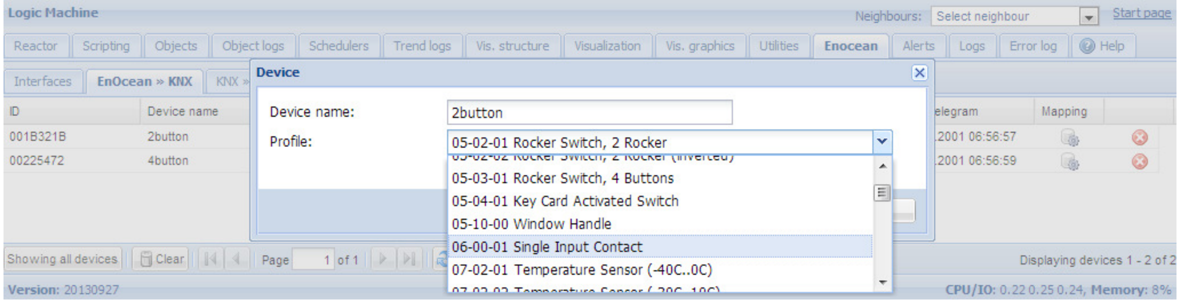


ID	Device name	Profile	Interface	Last telegram	Mapping
001B321B	2button	05-02-01 Rocker Switch, 2 Rocker	LOCAL	02.01.2001 06:56:57	
00225472	4button	05-03-01 Rocker Switch, 4 Buttons	LOCAL	02.01.2001 06:56:59	

Displaying devices 1 - 2 of 2

Version: 20130927 **CPU/IO:** 0.36 0.27 0.25, **Memory:** 8%

Once some specific device has to be mapped to KNX, the corresponding row has to be clicked and profile has to be chosen. There are all main profiles predefined in the list.



Device name: 2button

Profile: 05-02-01 Rocker Switch, 2 Rocker

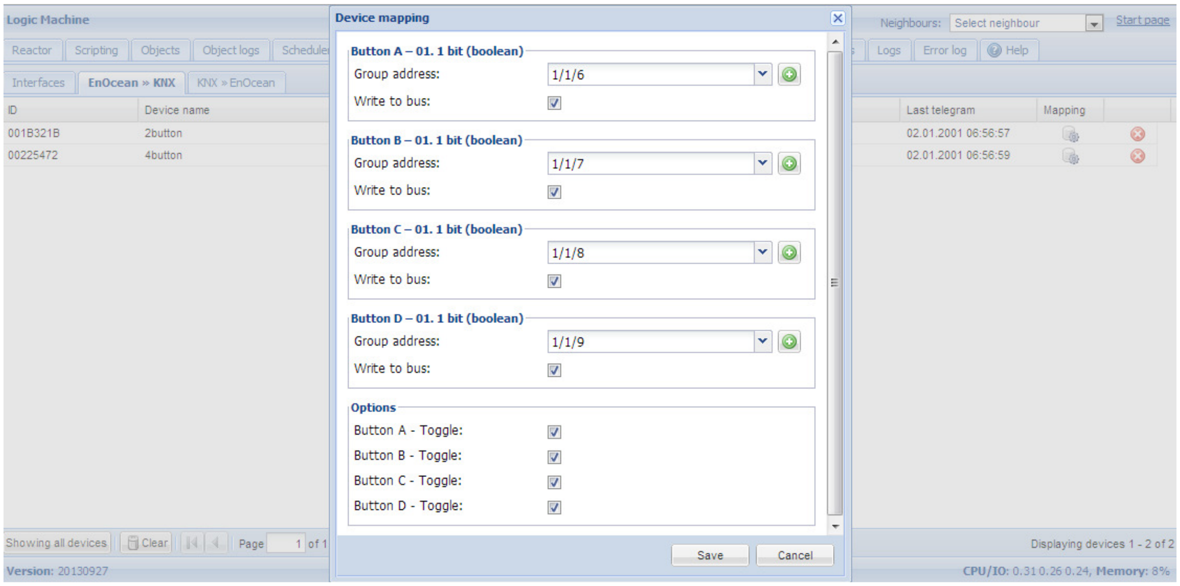
- 05-02-02 Rocker Switch, 2 Rocker (inverted)
- 05-03-01 Rocker Switch, 4 Buttons
- 05-04-01 Key Card Activated Switch
- 05-10-00 Window Handle
- 06-00-01 Single Input Contact
- 07-02-01 Temperature Sensor (-40C..0C)
- 07-02-02 Temperature Sensor (-30C..10C)

ID	Device name	elegram	Mapping
001B321B	2button	2001 06:56:57	
00225472	4button	2001 06:56:59	

Displaying devices 1 - 2 of 2

Version: 20130927 **CPU/IO:** 0.22 0.25 0.24, **Memory:** 8%

Once the device profile is set, you can map functionality of the specific device to KNX group addresses by clicking on *Mapping* icon.



ID	Device name	Last telegram	Mapping
001B321B	2button	02.01.2001 06:56:57	
00225472	4button	02.01.2001 06:56:59	

Device mapping

Button A – 01. 1 bit (boolean)
Group address: 1/1/6
Write to bus: ☑

Button B – 01. 1 bit (boolean)
Group address: 1/1/7
Write to bus: ☑

Button C – 01. 1 bit (boolean)
Group address: 1/1/8
Write to bus: ☑

Button D – 01. 1 bit (boolean)
Group address: 1/1/9
Write to bus: ☑

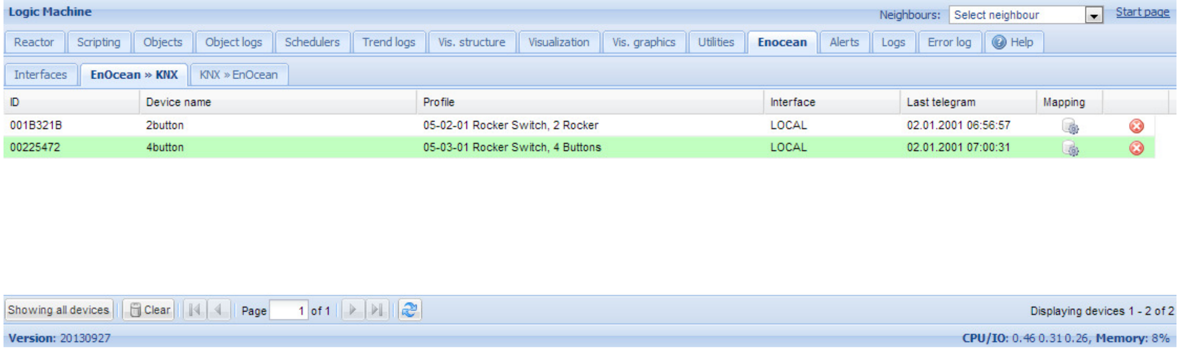
Options
Button A - Toggle: ☑
Button B - Toggle: ☑
Button C - Toggle: ☑
Button D - Toggle: ☑

Displaying devices 1 - 2 of 2

Version: 20130927

CPU/IO: 0.31 0.26 0.24, Memory: 8%

When EnOcean gateway received telegram from specific device, the respective row gets light green.



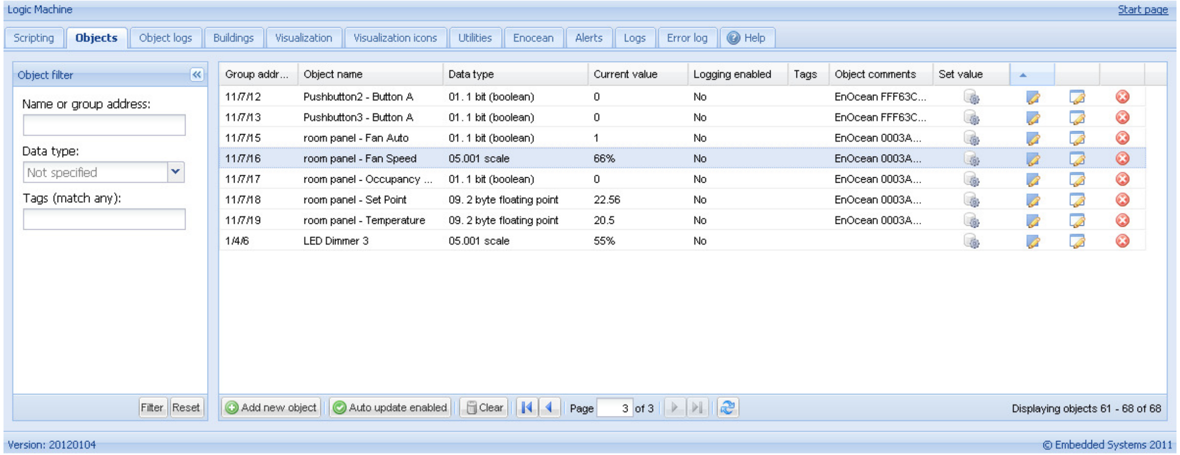
ID	Device name	Profile	Interface	Last telegram	Mapping
001B321B	2button	05-02-01 Rocker Switch, 2 Rocker	LOCAL	02.01.2001 06:56:57	
00225472	4button	05-03-01 Rocker Switch, 4 Buttons	LOCAL	02.01.2001 07:00:31	

Displaying devices 1 - 2 of 2

Version: 20130927

CPU/IO: 0.46 0.31 0.26, Memory: 8%

Respective KNX group addresses get updated with the new values.



Name or group address:

Data type: Not specified

Tags (match any):

Group addr...	Object name	Data type	Current value	Logging enabled	Tags	Object comments	Set value
11/7/12	Pushbutton2 - Button A	01. 1 bit (boolean)	0	No		EnOcean FFF63C...	
11/7/13	Pushbutton3 - Button A	01. 1 bit (boolean)	0	No		EnOcean FFF63C...	
11/7/15	room panel - Fan Auto	01. 1 bit (boolean)	1	No		EnOcean 0003A...	
11/7/16	room panel - Fan Speed	05.001 scale	66%	No		EnOcean 0003A...	
11/7/17	room panel - Occupancy ...	01. 1 bit (boolean)	0	No		EnOcean 0003A...	
11/7/18	room panel - Set Point	09. 2 byte floating point	22.56	No		EnOcean 0003A...	
11/7/19	room panel - Temperature	09. 2 byte floating point	20.5	No		EnOcean 0003A...	
1/4/6	LED Dimmer 3	05.001 scale	55%	No			

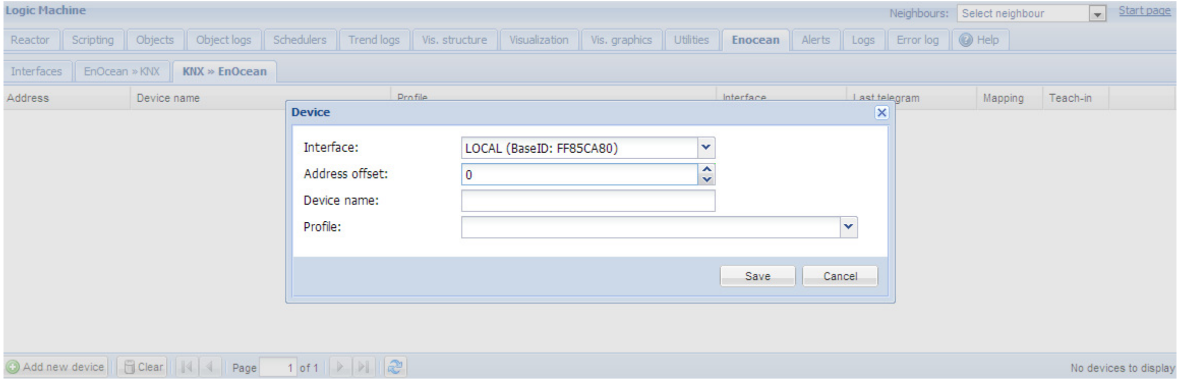
Displaying objects 61 - 68 of 68

Version: 20120104

© Embedded Systems 2011

9.3. KNX to EnOcean mapping

You should click on Add new device button to add EnOcean device which will be communicated from specific KNX object.



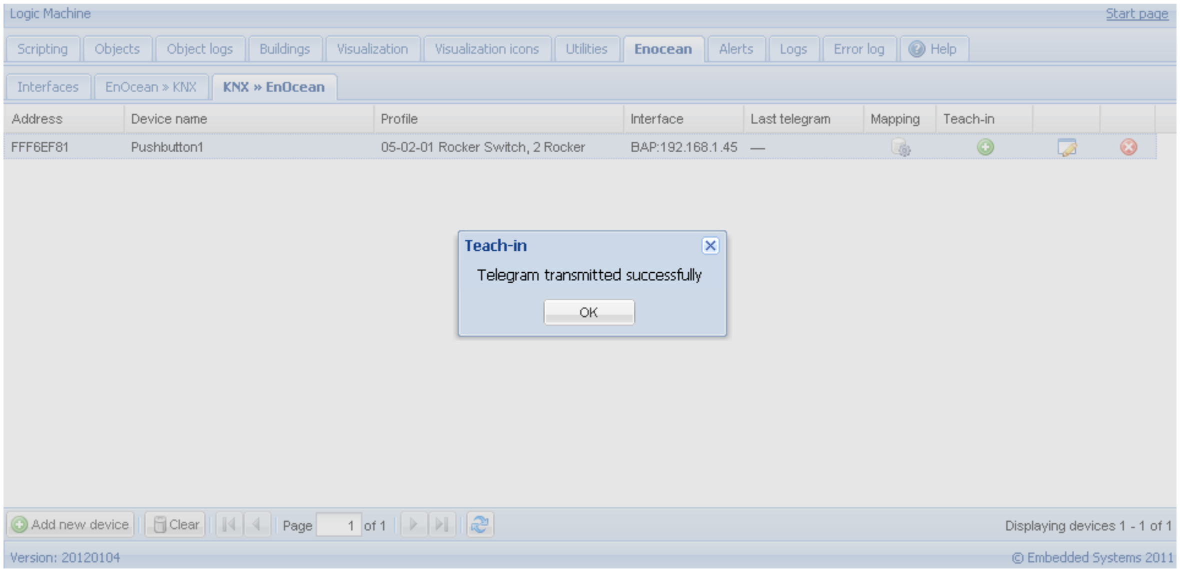
Device

Interface: LOCAL (BaseID: FF85CA80)
Address offset: 0
Device name:
Profile:

No devices to display

Once the device is added, you should pair it with specific device in EnOcean network, press Tech-in button.

Note! EnOcean device should be set in learning mode in order to pair it successfully.



Address	Device name	Profile	Interface	Last telegram	Mapping	Teach-in
FFF6EF81	Pushbutton1	05-02-01 Rocker Switch, 2 Rocker	BAP:192.168.1.45	—		

Teach-in

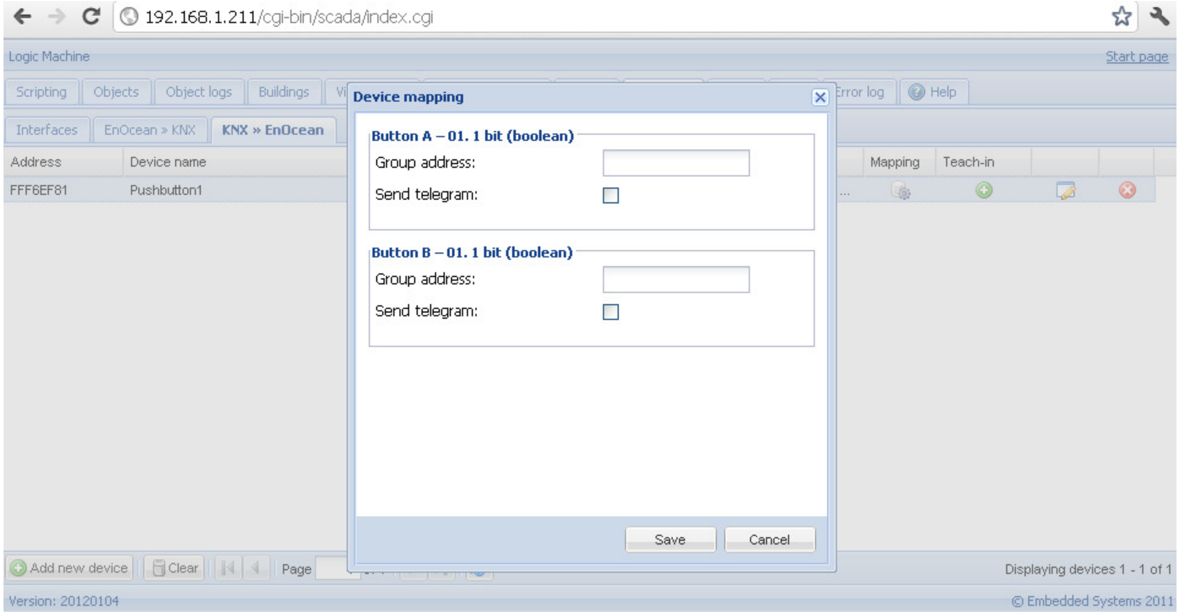
Telegram transmitted successfully

Displaying devices 1 - 1 of 1

Version: 20120104

© Embedded Systems 2011

Further this device can be mapped with specific KNX addresses.
When KNX object value will be updated, the telegram will be sent to respective EnOcean device.



Address	Device name	Mapping	Teach-in
FFF6EF81	Pushbutton1		

Device mapping

Button A – 01. 1 bit (boolean)

Group address:

Send telegram: □

Button B – 01. 1 bit (boolean)

Group address:

Send telegram: □

10. DMX interconnection with LM

DMX protocol support is realized upon RS485 serial port.

Usage

```
d =DMX:init(parameters)
d:run()
```

Parameters

- ***channels*** – (optional, defaults to 3) number of DMX channels to use
- ***resolution*** – (optional, defaults to 20) number of DMX updates per second. Larger value gives smoother transitions, but increases CPU usage
- ***transition*** – (optional, defaults to 2) soft transition time in seconds
- ***port***– (optional) RS-485 port name, usually you don't have to change this value

Common function

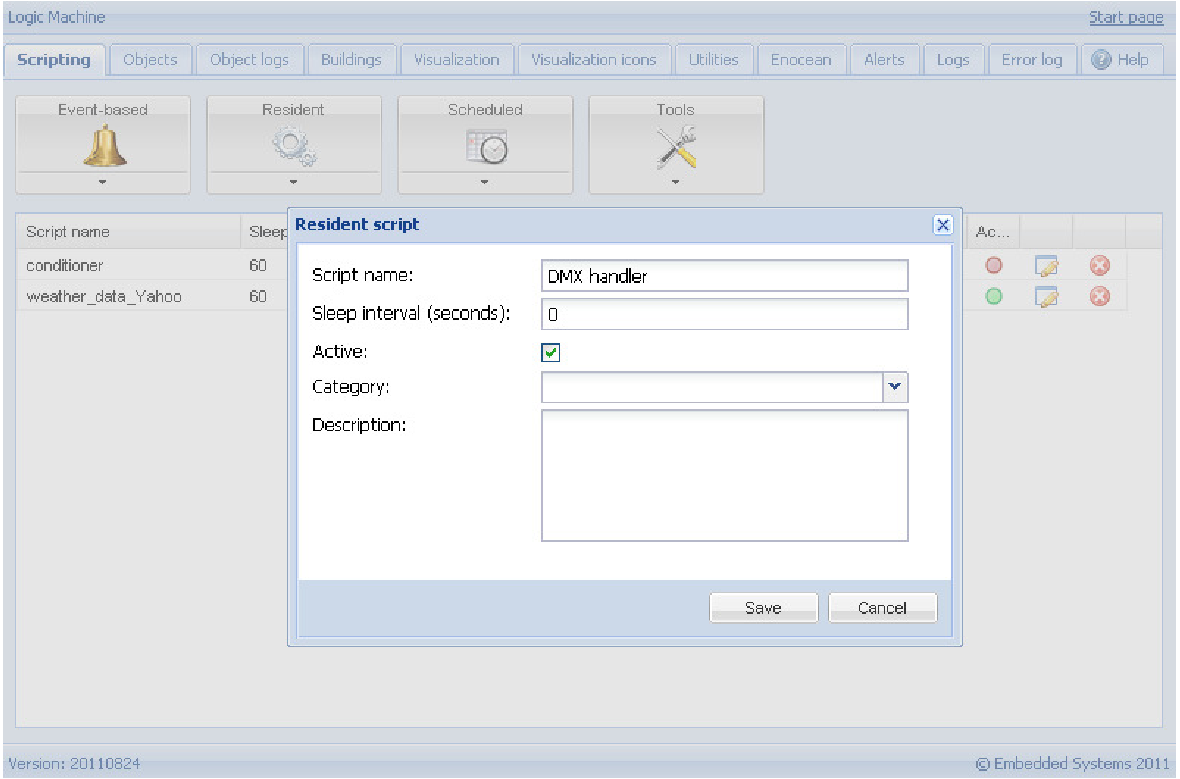
The following program has to be added in Common functions library.

```
DMX = {
  -- default params
defaults = {
    -- storage key
skey = 'dmx_chan_',
    -- RS-485 port
port = '/dev/ttyS2',
    -- number of calls per second
resolution = 20,
    -- total number of channels to use
channels = 3,
    -- transition time in seconds, does not include DMX transfer time
transition = 2,
  },
  -- value setter
set = function(i, v)
    -- validate channel number
if type(i) == 'number' and i >= 1 and i <= 512 then
      -- validate channel value
if type(v) == 'number' and v >= 0 and v <= 255 then
storage.set(DMX.defaults.skey .. i, v)
end
end
end
}

-- DMX init, returns new DMX object
function DMX:init(params)
require('luadmx')

local n = setmetatable({}, { __index = DMX })
local k, v

  -- set user parameters
  n.params = params

  -- copy parameters that are set by user
for k, v in pairs(DMX.defaults) do
if n.params[ k ] == nil then
      n.params[ k ] = v
end
end

n:reset()

return n
end

function DMX:reset()
local err, chan

  self.dm, err = luadmx.open(self.params.port)

    -- error while opening
if err then
os.sleep(1)
error(err)
end
```

```
  -- set channel count
  self.dm:setcount(self.params.channels)

  -- number of transaction ticks
  self.ticks = math.max(1, self.params.transition * self.params.resolution)

  -- calculate sleep time
  self.sleep = 1 / self.params.resolution

  -- reset channel map
  self.channels = {}

  -- fill channel map
for chan = 1, self.params.channels do
self.channels[ chan ] = { current = 0, target = 0, ticks = 0 }

    -- turn off by default
storage.set(self.params.skey .. chan, 0)
    self.dm:setchannel(chan, 0)
end
end

-- get new values
function DMX:getvalues()
local chan, val

  -- check for new values for each channel
for chan = 1, self.params.channels do
val = storage.get(self.params.skey .. chan)

    -- target value differs, set transcation
if val ~= self.channels[ chan ].target then
self.channels[ chan ].target = val
self.channels[ chan ].delta = (self.channels[ chan ].target - self.channels[ chan ].current) / self.ticks
self.channels[ chan ].ticks = self.ticks
end
end
end

-- main loop handler
function DMX:run()
local i, bs, bm, as, am, delta
local res = self.params.resolution

if not self.calibrated then
bs, bm = os.microtime()
end

self:getvalues()

  -- transition loop
for i = 1, res do
self:step()
    self.dm:send()

    -- wait until next step
os.sleep(self.sleep)
end

  -- calibrate delay loop to match 1 second
if not self.calibrated then
as, am = os.microtime()
delta = (as - bs) + (am - bm) / 1000000

if delta > 1.05 then
      self.sleep = self.sleep - math.max(10, self.sleep / res)
else
      self.calibrated = true
end
end
end

-- single transition step
function DMX:step()
local chan, t

  -- transition for each channel
for chan = 1, self.params.channels do
    t = self.channels[ chan ].ticks

    -- transition is active
if t > 0 then
      t = t - 1

self.channels[ chan ].current = self.channels[ chan ].target - self.channels[ chan ].delta * t
self.channels[ chan ].ticks = t

      self.dm:setchannel(chan, self.channels[ chan ].current)
end
end
end
```

DMX handler programs

DMX handler should be placed inside a resident script. Sleep time interval must be set to 0.



Script name	Sleep
conditioner	60
weather_data_Yahoo	60

Resident script

Script name: DMX handler

Sleep interval (seconds): 0

Active: ☑

Category:

Description:

Once the resident script is added we can add the program source in Script Editor

```
ifnot d then
  d =DMX:init({
    channels = 3,
    transition = 2,
})
end

d:run()
```

<u>Setter (used in other scripts)</u>

```
DMX.set(channel, value)
```

- ***channel***– DMX channel number [1..512]
- ***value*** – DMX channel value [0..255]

10.1. Examples

<u>Predefined scene example</u>: The following example should be placed inside a resident script. Sleep time defines scene keep time (at least 1 second).

```
ifnot scenes then
```

```
-- 3 channel scene
  scenes ={
{ 255, 0, 0 },
{ 0, 255, 0 },
{ 0, 0, 255 },
{ 255, 255, 0 },
{ 0, 255, 255 },
{ 255, 0, 255 },
{ 255, 255, 255 },
}

  current = 1
end

-- set current scene values
scene = scenes[ current ]
fori, v inipairs(scene)do
DMX.set(i, v)
end

-- switch to next scene
current = current + 1
if current > #scenes then
  current = 1
end
```

Random scene example: The following example should be placed inside a resident script. Sleep time defines scene keep time (at least 1 second).

```
-- number of steps to use, e.g. 3 steps = { 0, 127, 255 }
steps =5
-- number of channels to set
channels =3
-- first channel number
offset = 1

fori= offset, channels do
  v =math.random(0, (steps - 1))* 255 /(steps - 1)
DMX.set(i, math.floor(v))
end
```

11. 3G modem connection with LM

LogicMachine has standard 3G modem driver built-in (Huawei and other vendor support). Currently this can be used for SMS notifications only – receiving and sending commands. The modem has to be plugged into any of USB ports of LM and it starts operating immediately. We suggest to use external 5V powering for the modem because by USB2.0 standard the output current on USB is 0.75A, but some modems requires up to 2A which is out of standard so the modem can lack the power and get disconnected.

First thing is to lower the modem speed by adding the following code in *Start-up / Init* script:

```
os.execute('echo 1 > /sys/bus/platform/devices/ci_hdrc.0/force_full_speed')
os.execute('echo 1 > /sys/bus/platform/devices/ci_hdrc.1/force_full_speed')
os.execute('usbreset /dev/bus/usb/001/001')
```

After you need to add SMS handler program – a resident script with sleep interval 0.

Note! Change white list telephone numbers and SIM card's PIN code in the below script.

```
-- init
ifnot modem then
-- allowed numbers, SMS message from other number will be ignored
  numbers ={'1234567890', '0123456789'}
-- replace 0000 with SIM pin number, or remove the line below if PIN check is disabled
pincode='0000'
-- modem communication port, ttyUSB2 for Huawei E173
  comport ='ttyUSB2'
-- open serial port
  modem =AT:init('/dev/' .. comport)
-- command parser
  parser =function(cmd, sender)
local find, pos, name, mode, offset, value, jvalue, obj
cmd=cmd:trim()
    mode =cmd:sub(1, 1):upper()
if mode =='W'or mode =='R'then
cmd=cmd:sub(3):trim()
-- parse object name/address
      find =cmd:sub(1, 1)=='"'and'"'or' '
      offset = find =='"'and 1 or0
-- pad with space when in read mode
if mode =='R'and find ==' 'then
cmd=cmd .. ' '
end
-- find name
pos=cmd:find(find, 1 + offset, true)
-- name end not found, stop
ifnotposthen
returnfalse
```

```
end
-- get name part
      name =cmd:sub(1 + offset, pos - offset):trim()
if mode =='W'then
        value =cmd:sub(pos + offset):trim()
ifnot value then
returnfalse
end
-- try decoding value
jvalue=json.pdecode(value)
        value =jvalue ~=nilandjvalueor value
-- send to bus
grp.write(name, value)
-- read request
else
obj=grp.find(name)
-- send read request and wait for update
ifobjthen
obj:read()
os.sleep(1)
-- read new value
          value =grp.getvalue(name)
-- got value, send response
if value ~=nilthen
jvalue=json.pencode(value)
if obj.name then
              name =string.format('%s (%s)', obj.name, obj.address)
end
cmd=string.format('Value of %s is %s', name, jvalue)
modem:sendsms(sender, cmd)
end
end
end
end
end
-- incoming sms handler
  handler =function(sms)
    alert('incoming sms from %s (%s)', sms.sender, sms.data)
-- sms from known number, call parser
iftable.contains(numbers, sms.sender)then
      parser(sms.data, sms.sender)
end
end
-- set sms handler
modem:setsmshandler(handler)
-- send pin if set
ifpincodethen
modem:send('AT+CPIN=' .. pincode)
end
-- set to pdu mode
```

```
modem:send('AT+CMGF=0')
-- enable sms notifications
modem:send('AT+CNMI=1,1,0,0,0')
  alert('SMS handler started')
end
modem:run()
```

Command syntax:

a. Write to bus:

W ALIAS VALUE

b. Read from bus:

R ALIAS

On read request, script will reply with SMS message containing current value of selected object.

ALIAS can be:

a. Group address (e.g. 1/1/1)

b. Name (e.g. Obj1). If name contains spaces then it must be escaped usign double quotes (e.g. "Room Temperature")

NOTE:

a. Object data type and name must be set in Objects tab. Otherwise script won't be able to read and write to object.

b. Only ASCII symbols are accepted in the message.

11.1. Examples

Binary write (send the following SMS to switch kitchen lights on):

W 1/1/1 true

Scaling write (send the following SMS to set value 67% for red LED):

W LED1Red 67

Temperature (floating point) write (send the following SMS to make setpoint in the living room to 22.5 degrees):

W “Room Setpoint” 22.5

Read (send the following SMS to read the security panel value:

R 2/1/1

11.2. Send SMS messages to specific SIM numbers after group-read or group-write is triggered

Task: Assume we have an Event-based script which triggers a program once group-read or group-write is triggered for address 1/1/1. We want to send SMS to numbers 23335555 and 23335556 with 1/1/1 actual status.

```
require('socket')

client =socket.udp()

-- in the message field the number where SMS has to be send should be specified at the beginning
localmsg='23335555 1/1/1 changes its value to: ' .. tonumber(event.datahex)
client:sendto(msg, '127.0.0.1', 12535)

msg='23335556 1/1/1 changes its value to: ' .. tonumber(event.datahex)
client:sendto(msg, '127.0.0.1', 12535)
```

11.3. Send SMS messages without 3G modem

How to send event SMS to mobile phone from LogicMachine through Twilio service, without external 3G adapter?

You can use Twilio service which offers free of charge SMS in the test period and messaging at $0.01 for regular usage. The only disadvantage is it will use your standard Internet connection to send messages to Twilio servers (not via GSM as with 3G adapters).

Twilio account

You can get ID and Token needed for the below example by registering on Twilio. Make sure you enter a verified SIM number list / recipients in your account. Or please contact us for ready example with our account data.

Function

Add the following function in *Scripting –> Common functions*

```
function sms(id, token, from, to, body)
  local escape = require('socket.url').escape
  local request = require('ssl.https').request
  local url = string.format('https://%s:%s@api.twilio.com/2010-04-01/Accounts/%s/Messages.json', id, token, id)
  local body = string.format('From=%s&To=%s&Body=%s', escape(from), escape(to), escape(body))

```

```
   return request(url, body)
end
```

Event-based script

Add event-based program for specific object, like 1/1/2 in this example

```
value = event.getvalue()

from_nr = '+37112345679' -- put sender SIM nr here
to_nr = '+37112345678' -- put recepient SIM nr here
id_nr = 'ACe56f5' -- put your ID here
token_nr = '598c6ff' -- put your token here

sms(id_nr, token_nr, from_nr, to_nr, 'The value for 1/1/2 has changed to'..tostring(value))
```

11.4.

12. HDL protocol integration in LogicMachine

Note! Please contact Embedded Systems team to receive a special package to integrate HDL support into your LM. Once you have the file, add it in *Network configuration -> System -> Packages*.

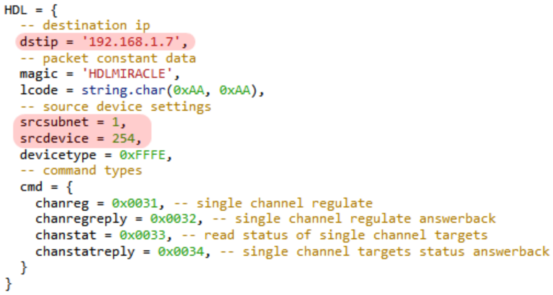
12.1. HDL function

Add HDL script in *Scripting -> Tools -> User function library*

```
HDL ={
-- destination ip
dstip='192.168.1.7',
-- packet constant data
  magic ='HDLMIRACLE',
lcode=string.char(0xAA, 0xAA),
-- source device settings
srcsubnet=1,
srcdevice=254,
devicetype= 0xFFFE,
-- command types
cmd={
chanreg= 0x0031, -- single channel regulate
chanregreply= 0x0032, -- single channel regulate answerback
chanstat= 0x0033, -- read status of single channel targets
chanstatreply= 0x0034, -- single channel targets status answerback
}
}

HDL.init=function()
require('json')
require('crc16')
require('socket')

localip, chunk, chunks, data
-- read interface data
  data =json.pdecode(io.readproc('if-json'))

ifnot data ornot data.eth0 then
error('cannot get interface data')
end

-- ip header
HDL.iphdr=''
```

```lua
-- broadcast address
HDL.bcast= data.eth0.bcast

-- split ip address into chunks
chunks= data.eth0.inetaddr:split('.')

-- add ip address chunks
fori= 1, 4 do
    chunk =tonumber(chunks[i])
HDL.iphdr=HDL.iphdr ..string.char(chunk)
end
end

HDL.decode=function(packet)
locallen, data, src, crc

-- primary header
ifpacket:sub(5, 14) ~=HDL.magicthen
returnnil, 'magic'
end

-- leading code
ifpacket:sub(15, 16) ~=HDL.lcodethen
returnnil, 'lcode'
end

-- get data length and check against
len=packet:byte(17)
iflenandlen + 16 ~=packet:len()then
returnnil, 'len'
end

-- get packet data and check crc
  data =packet:sub(17, len + 14)
crc=packet:byte(len + 15)* 0x100 + packet:byte(len + 16)
if crc16(data) ~=crcthen
returnnil, 'crc'
end

-- return parsed packet
```

Change HDL parameters in the function to correct ones



```
HDL = {
  -- destination ip
  dstip = '192.168.1.7',
  -- packet constant data
  magic = 'HDLMIRACLE',
  lcode = string.char(0xAA, 0xAA),
  -- source device settings
  srcsubnet = 1,
  srcdevice = 254,
  devicetype = 0xFFFE,
  -- command types
  cmd = {
    chanreg = 0x0031, -- single channel regulate
    chanregreply = 0x0032, -- single channel regulate answerback
    chanstat = 0x0033, -- read status of single channel targets
    chanstatreply = 0x0034, -- single channel targets status answerback
  }
}
```

12.2. Usage example – HDL dimmer control

Task of this example is to change HDL dimmer value on specific KNX group address change.

- Add new object in Objects tab
- Add Event-based script which will monitor newly created object
- In Scripting Editor specify the following code for this script

```
local value =dpt.decode(event.datahex, dt.scale)
HDL.chanreg(1, 12, 1, value, 1)
```

<u>*HDL.chanreg* function description</u>

```
HDL.chanreg(dstsubnet, dstdevice, chan, value, delay)
```

Parameters:

- *dstsubnet* – device subnet
- *dstdevice* – device address
- *chan* – channel number (1..n)
- *value* – value (0..100, or true / false)
- *delay* – transition time or delay in seconds (0..65535), by default is 0

<u>Test the program</u>

If you change the value for object 4/1/1 in Objects menu with Set Value, it will automatically change dimmer state in HDL network.

12.3. Usage example – HDL relay control

Task of this example is to change HDL dimmer value on specific KNX group address change.

- Add new object in Objects tab
- Add Event-based script which will monitor newly created object
- In Scripting Editor specify the following code for this script

```
local value =dpt.decode(event.datahex, dt.bool)
HDL.chanreg(1, 11, 1, value))
```

Test the program

If you change the value for object 4/1/2 in Objects menu with Set Value, it will automatically change the relay state in HDL network.

13. Communication with RS232/RS485 serial ports

The following are the naming of Serial ports for different versions of Logic Machine.

LM4

GND	
RS485 A	RS485-1
RS485 B	
GND	
RS485 A	RS485-2
RS485 B	
GND	
RS485 A	RS485-3
RS485 B	

Reactor

GND	
RS485 A	RS485-1
RS485 B	
GND	
RS485 A	RS485-2
RS485 B	

Reactor V2

GND	
RS485 A	RS485
RS485 B	

Functions

Include library before calling serial functions:

```
require('serial')
```

Opens given port, returns: port handle, or, in case of error, nil plus error message

```
port, err = serial.open(device, params)
```

Parameters:

- **device** port device name, required
- **params** parameters table, optional, (defaults are in bold):
  - **baudrate** 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400
  - **parity** "none", "even", "odd"
  - **databits** 5, 6, 7, 8
  - **stopbits** 1, 2
  - **duplex** "full", "half" (Note: "half" is required for RS-485)

Reads the specified number of bytes, execution is blocked until read is complete

```
res, err = port:read(bytes)
```

Parameters:

- **bytes** number of bytes to read

Reads until timeout occurs or the specified number of bytes is received, whichever happens first. Returns data plus number of bytes read, or, in case of error, nil plus error message.

```
res, err = port:read(bytes, timeout)
```

Parameters:

- **bytes** number of bytes to read
- **timeout** maximum time to wait for read to complete, minimum value and timer resolution is 0.1 seconds

Flushes any read/unsent bytes

```
port:flush()
```

Closes serial port, no other port functions may be called afterwards

```
port:close()
```

Examples

Write to port

```
port:write('test data')
```

Blocking read (script will block until 10 characters are read)

```
data=port:read(10)
```

Timeout read (script will wait for 10 characters for 20 seconds)

```
data=port:read(10, 20)
```

Close serial port

```
port:close()
```

Resident script, RS-485 echo test

```
-- open port on first call
if not port then
require('serial')
port = serial.open('/dev/ttyS2', { baudrate = 9600, parity = 'even', duplex =
'half' })
port:flush()
end

-- port ready
if port then
  -- read one byte
char = port:read(1, 1)
  -- send back if read succeeded
if char then
port:write(char)
end
end
```

14. Bluetooth 4.0 integration

Task:

Interconnect LogicMachine with Mio Alpha watch and map heart-rate measurement to KNX group address. Upon excessing specific heart-rate measurement, switch on ventilation on group address 2/2/2. In same way any other Bluetooth LE 4.0 sensor with open API or iBeacon can be integrated with any supported standard by LogicMachine.

Some of supported Bluetooth 4.0 USB adapters:

- Broadcom BCM20702A0
- Trust 18187
- Belkin F8T065bf
- Plugable USB Bluetooth 4.0
- Laird BT820

Steps:

- Add 1byte object 1/1/1 in Objects menu
- Add the following code to Resident script with interval = 0 seconds

```
if proc then
  line = proc:read()
  parseline(line)
else
  mac = 'D7:2D:DA:DF:E4:34' -- MAC of AlphaMio watch

  -- bring bt interface up
  os.execute('hciconfig hci0 up')
  os.sleep(2)

    -- read heart rate data
    proc = io.popen('gatttool -b ' .. mac .. ' -t random --char-write-req -a 0x0025 -n 0100 --listen')
    count = 0

    function parseline(line)
      local pos, rate

      -- invalid data
      if not line then
        return
      end

      -- find value marker
      pos = line:find('value: ', 1, true)
      if not pos then
        return
      end

      -- get current heart rate
      rate = tonumber(line:sub(pos + 10, pos + 11), 16)

      -- send each 5 reads
      count = count + 1
      if count == 5 then
        grp.update('1/1/1', rate)
        count = 0
      end
    end
  end
```

- Add event-based script heart-rate object 1/1/1. This script will switch on ventilation if the heart-rate is >80 and switch off if its lower

```
value = event.getvalue()
if value > 80 then
  grp.write('2/2/2', true)
else
  grp.write('2/2/2', false)
end
```

15. SIP server on LogicMachine

Task: How to pair SIP door entry systems with building automation project? In LogicMachine we have built SIP registrar which can send SIP requests to final SIP clients. For example, one can install Linphone SIP client app on touch devices which are used for visualization control. Upon SIP request from door entry system, LogicMachine will forward the request to the respective SIP client / recipient. On this client's device a new window will appear with options to answer or reject the call. When the call is answered, you will see video and audio from the door entry system. When the call is finished, Linphone app will go to the background.

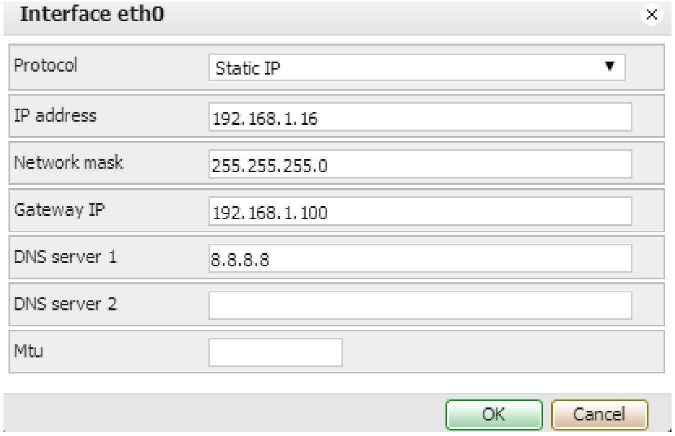
SIP package installation on LM:

Add the following Resident script, 60 sec sleep time, run once:

```
os.execute('opkg --force-depends install
http://dl.openrb.com/pkg/kamailio/terminfo_5.7-5_mxs.ipk')
os.execute('opkg --force-depends install
http://dl.openrb.com/pkg/kamailio/libncurses_5.7-5_mxs.ipk')
os.execute('opkg --force-depends install
http://dl.openrb.com/pkg/kamailio/libreadline_5.2-2_mxs.ipk')
os.execute('opkg --force-depends install
http://dl.openrb.com/pkg/kamailio/kamailio3_3.3.7-1_mxs.ipk')
os.execute('opkg --force-depends install
http://dl.openrb.com/pkg/kamailio/kamailio3-mod-maxfwd_3.3.7-
1_mxs.ipk')
os.execute('opkg --force-depends install
http://dl.openrb.com/pkg/kamailio/kamailio3-mod-registrar_3.3.7-
1_mxs.ipk')
os.execute('opkg --force-depends install
http://dl.openrb.com/pkg/kamailio/kamailio3-mod-rr_3.3.7-1_mxs.ipk')
os.execute('opkg --force-depends install
http://dl.openrb.com/pkg/kamailio/kamailio3-mod-sl_3.3.7-1_mxs.ipk')
os.execute('opkg --force-depends install
http://dl.openrb.com/pkg/kamailio/kamailio3-mod-tm_3.3.7-1_mxs.ipk')
os.execute('opkg --force-depends install
http://dl.openrb.com/pkg/kamailio/kamailio3-mod-usrloc_3.3.7-
1_mxs.ipk')

os.execute('/etc/init.d/kamailio enable')
os.execute('/etc/init.d/kamailio start')
```

Check if LM has Internet access

Check that IP, gateway, subnet, DNS are set correctly set.

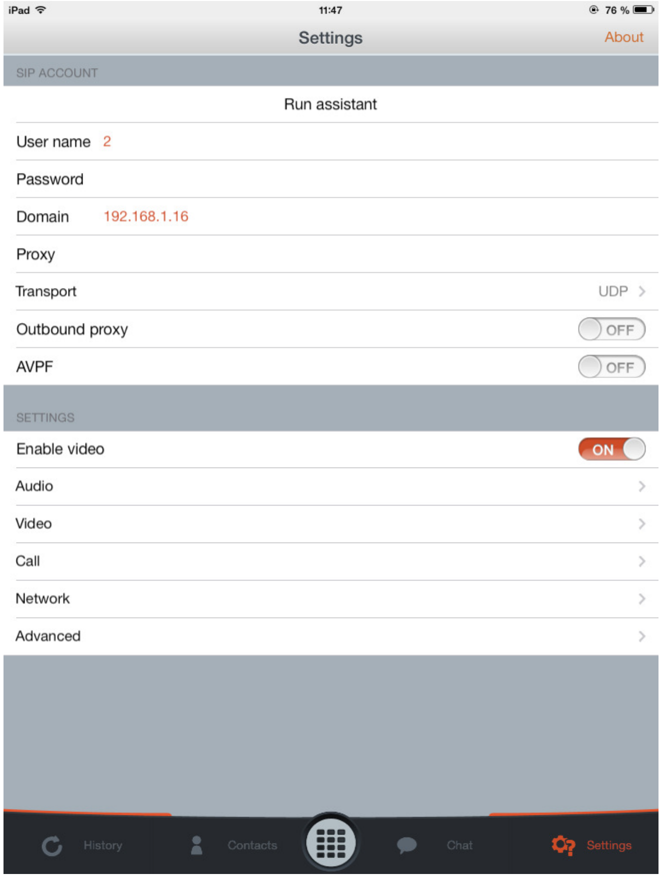


Interface eth0

Protocol	Static IP
IP address	192.168.1.16
Network mask	255.255.255.0
Gateway IP	192.168.1.100
DNS server 1	8.8.8.8
DNS server 2	
Mtu	

SIP client application

You can use for example Linphone as your SIP client. You have to enter IP of LogicMachine in its settings.



Settings

SIP ACCOUNT

Run assistant

User name	2
Password	
Domain	192.168.1.16
Proxy	
Transport	UDP
Outbound proxy	OFF
AVPF	OFF

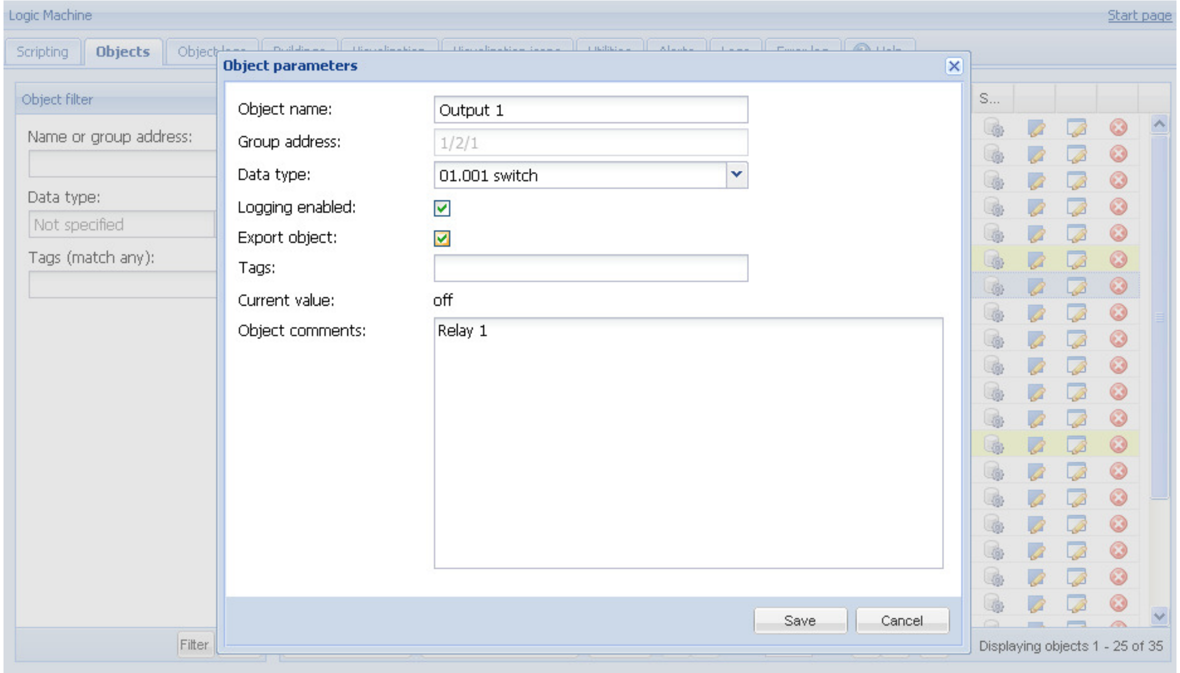
SETTINGS

Enable video	ON
Audio	
Video	
Call	
Network	
Advanced	

16. Object value export via XML

Make KNX objects XML readable

In the *Objects* tab click on the objects which you want to receive the current value by XML request. Check the Export object



Name or group address:

Data type: Not specified

Tags (match any):

Object name: Output 1

Group address: 1/2/1

Data type: 01.001 switch

Logging enabled: ☑

Export object: ☑

Tags:

Current value: off

Object comments: Relay 1

Displaying objects 1 - 25 of 35

XML request from external PC

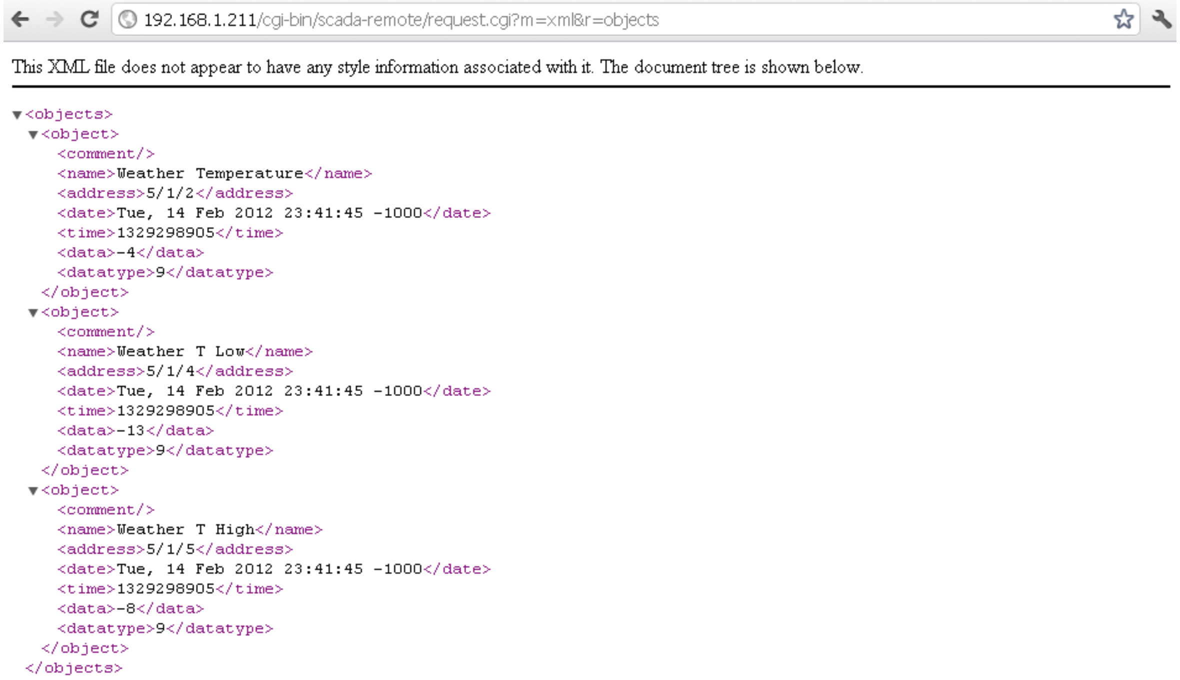
The XML request looks like this:

http://remote:remote@192.168.1.211/cgi-bin/scada-remote/request.cgi?m=xml&r=objects

Parameters:

- ***address*** – object address (e.g. "1/1/1″)
- ***name*** – object name (e.g. "My object")
- ***data*** – decoded object value (e.g 42 or "01.01.2012″)
- ***datatype*** – object datatype (e.g. 1 or 5.001) – standard KNX data types
- ***time*** – object update time (UNIX timestamp)
- ***date*** – object update time (RFC date)
- ***comment*** – object comment (e.g. "Second floor entry lights")
- ***tags*** – optional array of object tags (e.g. "Light", "Second floor")

Note! To get list of objects that have been updated after specific time you can pass an optional "updatetime" parameter (UNIX timestamp format)



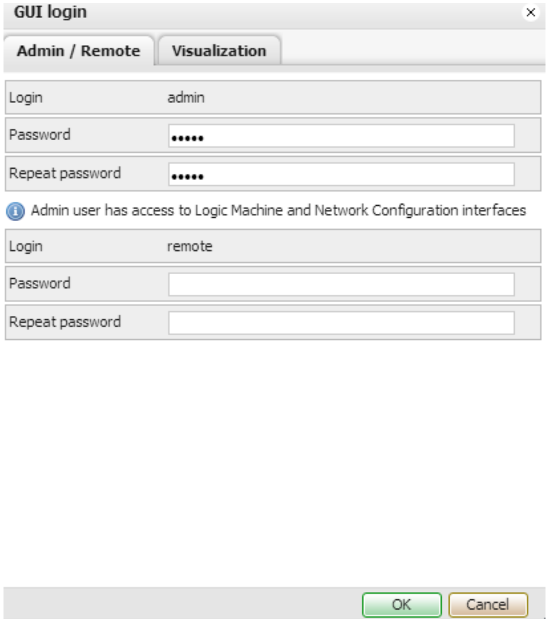
192.168.1.211/cgi-bin/scada-remote/request.cgi?m=xml&r=objects

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<objects>
  ▼<object>
     <comment/>
     <name>Weather Temperature</name>
     <address>5/1/2</address>
     <date>Tue, 14 Feb 2012 23:41:45 -1000</date>
     <time>1329298905</time>
     <data>-4</data>
     <datatype>9</datatype>
   </object>
  ▼<object>
     <comment/>
     <name>Weather T Low</name>
     <address>5/1/4</address>
     <date>Tue, 14 Feb 2012 23:41:45 -1000</date>
     <time>1329298905</time>
     <data>-13</data>
     <datatype>9</datatype>
   </object>
  ▼<object>
     <comment/>
     <name>Weather T High</name>
     <address>5/1/5</address>
     <date>Tue, 14 Feb 2012 23:41:45 -1000</date>
     <time>1329298905</time>
     <data>-8</data>
     <datatype>9</datatype>
   </object>
 </objects>
```

Login, Password for remote XML request

Login and password can be changed in *Network Configuration* → *System* → *GUI Login* →*Admin/Remote* tab.



GUI login

Login	admin
Password	•••••
Repeat password	•••••

Admin user has access to Logic Machine and Network Configuration interfaces

Login	remote
Password	
Repeat password	

16.1. Alerts, Errors values

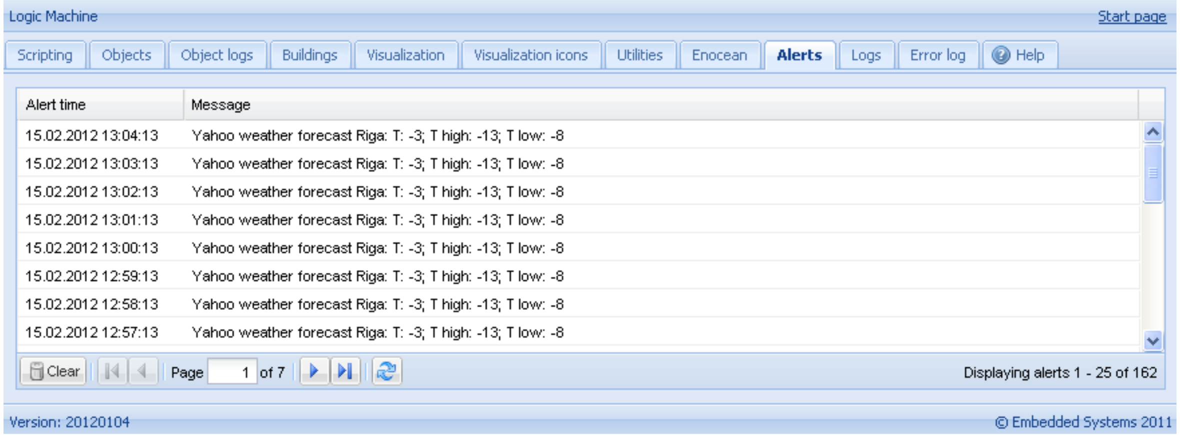
In similar way also Alerts and Errors can be read by XML requests.

Alerts XML request:
http://remote:remote@192.168.0.10/cgi-bin/scada-remote/request.cgi?m=xml&r=alerts

Errors XML request:
http://remote:remote@192.168.0.10/cgi-bin/scada-remote/request.cgi?m=xml&r=errors

17. Read Alerts RSS feeds from LogicMachine

It is possible to read Alerts and Errors messages by remote RSS readers.



Alert time	Message
15.02.2012 13:04:13	Yahoo weather forecast Riga: T: -3; T high: -13; T low: -8
15.02.2012 13:03:13	Yahoo weather forecast Riga: T: -3; T high: -13; T low: -8
15.02.2012 13:02:13	Yahoo weather forecast Riga: T: -3; T high: -13; T low: -8
15.02.2012 13:01:13	Yahoo weather forecast Riga: T: -3; T high: -13; T low: -8
15.02.2012 13:00:13	Yahoo weather forecast Riga: T: -3; T high: -13; T low: -8
15.02.2012 12:59:13	Yahoo weather forecast Riga: T: -3; T high: -13; T low: -8
15.02.2012 12:58:13	Yahoo weather forecast Riga: T: -3; T high: -13; T low: -8
15.02.2012 12:57:13	Yahoo weather forecast Riga: T: -3; T high: -13; T low: -8

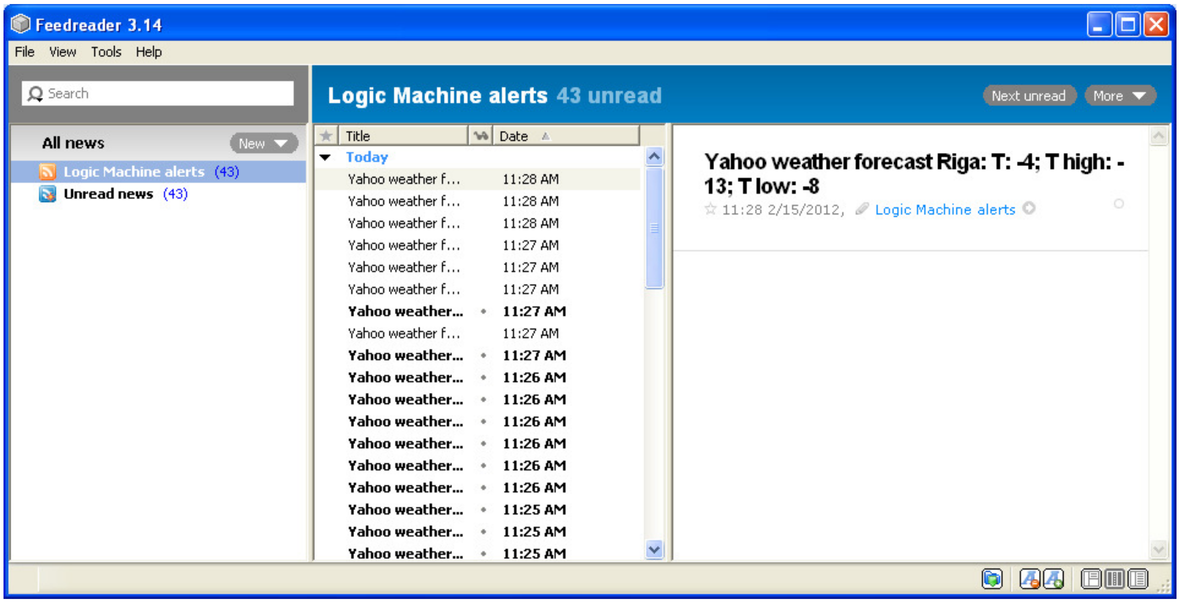
Displaying alerts 1 - 25 of 162

Version: 20120104

© Embedded Systems 2011

Add new RSS feed in the RSS reader

- Use the following URL:
- http://remote:remote@192.168.1.211/cgi-bin/scada-remote/request.cgi?m=rss&r=alerts
- 50 latest alerts will be shown
- *alert time* will be shown in *UNIX timestamp*, *alert date* will be shown as *RFC date*



Logic Machine alerts 43 unread

Title	Date
Today	
Yahoo weather f...	11:28 AM
Yahoo weather f...	11:28 AM
Yahoo weather f...	11:28 AM
Yahoo weather f...	11:27 AM
Yahoo weather f...	11:27 AM
Yahoo weather f...	11:27 AM
Yahoo weather...	**11:27 AM**
Yahoo weather f...	11:27 AM
Yahoo weather...	**11:27 AM**
Yahoo weather...	**11:26 AM**
Yahoo weather...	**11:26 AM**
Yahoo weather...	**11:26 AM**
Yahoo weather...	**11:26 AM**
Yahoo weather...	**11:26 AM**
Yahoo weather...	**11:26 AM**
Yahoo weather...	**11:25 AM**
Yahoo weather...	**11:25 AM**
Yahoo weather...	**11:25 AM**

Yahoo weather forecast Riga: T: -4; T high: -13; T low: -8

☆ 11:28 2/15/2012, Logic Machine alerts

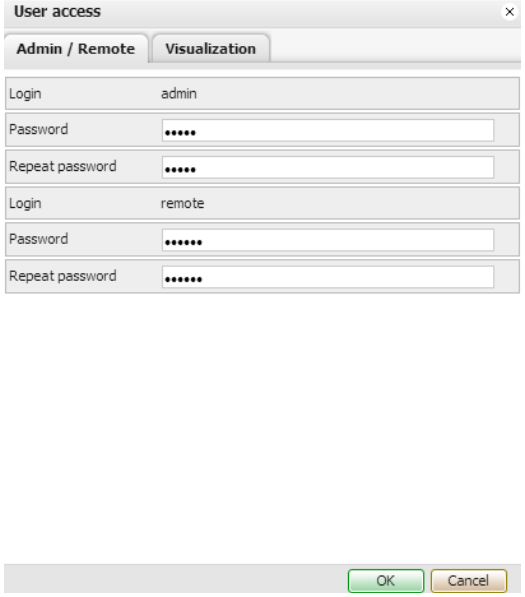
Error tab content by RSS

RSS can be used to read Error tab content as well. In this case the URL would look like:

http://remote:remote@192.168.1.211/cgi-bin/scada-remote/request.cgi?m=rss&r=errors

Login, Password for remote RSS requests

Login and password can be changed in *System Configuration* → *System* →*User access*→*Admin/Remote* tab.



Login	admin
Password	•••••
Repeat password	•••••
Login	remote
Password	••••••
Repeat password	••••••