

# Grenton

## User Manual

*Grenton 2*

Document version: *1.0.8*

Date: *02.11.2021*

### Table of Contents

#### **Important information**

#### **I. System structure**

#### **II. Foundation - Grenton Logical Interface**

1. Introduction
2. Features
  - 2.1. Built-in features
  - 2.2. User features
3. Methods
4. Events
5. Features and methods addresses

#### **III. Project preparation**

1. Electrical system preparation
2. System architecture selection
3. Modules power supply

#### **IV. Components installation**

1. Modules installation in the switching action
2. Flush-mounted wire modules installation
3. Z-Wave flush-mounted modules installation

#### **V. Object Manager**

1. OM installation
  - A. Windows
  - B. macOS
  - C. Linux
2. OM structure
  - 2.1. Object filtering
  - 2.2. Renaming an object
3. Project files
  - 3.1. Saved projects catalogue
  - 3.2. Project backup
4. Basic elements
  - 4.1. Objects configurator
  - 4.2. Script builder

- 4.3. Connections diagram
- 4.4. Visual Builder
- 4.5. myGrenton
- 4.6. Bin

## **VI. Basic system configuration**

- 1. Connecting OM to CLU
- 2. IP addresses
- 3. Opening new project
- 4. CLU Discovery function
- 5. CLU status
  - 5.1. Module diodes
  - 5.2. CLU module icon in OM
- 6. Connecting Z-Wave modules
  - 6.1. Adding Z-Wave modules
  - 6.2. Removal of Z-Wave modules
  - 6.3. No communication with the Z-Wave module - a mechanism for counting communication failures and blocking device communication in the Z-Wave network
  - 6.4. Z-Wave network configuration tips
  - 6.5. Clearing information about nodes
- 7. Sending the configuration to the CLU
- 8. Initial values of features
- 9. Creating basic connections
- 10. Performing an update
  - 10.1. The process of updating the interface database
  - 10.2. The process of updating the firmware on the CLU
  - 10.3. The process of updating the firmware of the 2.0 series modules
  - 10.4. CLU / modules status in the firmware update window
  - 10.5 Forcing the module update
- 11. Diagnostic view
  - 11.1 Configuration of the diagnostic view
- 12. Other operations on the system

## **VII. Advanced configuration functions**

- 1. Containers
- 2. Scripts
  - 2.1. Script creation in the graphic mode
  - 2.2. Script creation in the editor
  - 2.3. Script parameters
  - 2.4. Scripts invocation
- 3. Date and time

## **VIII. Visual Builder - Smartphone control**

- 1. System control on the level of smartphone
- 2. Interface structure
- 3. Application for smartphone - GRENTON HOME MANAGER
- 4. New interface creation
  - 4.1. Graphic skin selection
  - 4.2. Interface pages creation
  - 4.3. Components
  - 4.4. Panels
  - 4.5. Containers
  - 4.6. Adding components and connecting to the system objects
  - 4.7. Sending interface to mobile device
- 5. Automatic interface creation - GUI generator
  - 5.1. Creating an interface with available resolution
  - 5.2. Creating an interface with its own resolution
  - 5.3. Changing the orientation of the interface with its own resolution
- 6. Video intercom configuration

- 6.1. Connection and configuration of a video intercom
- 6.2. Creation and configuration of the application interface
- 6.3. Making a call from the intercom
7. IP cameras image operation
8. Remote access of the mobile application to the system
  - 8.1. System configuration
  - 8.2. Port routing setting on the local network router
  - 8.3. Configuration of the Home Manager mobile application
  - 8.4. Starting remote access

## **IX. CLU Objects**

1. Timers
2. Calendar
3. Schedule
4. PID controller
5. Thermostat
6. Push
7. Presence Sensor
8. Sunrise and Sunset Calendar

## **X. Media measurement**

1. Virtual media measurement
  - 1.1. Launching media measurement on the Object Manager page
  - 1.2. Using media measurement on the Home Manager application side
2. Real media measurement
  - 2.1. Real media measurement settings in Object Manager

## **XI. CLU service functions**

1. Restoring factory settings CLU - *Hard Reset*
2. System diagnostics - *Save the diagnostic package*

## **XII. SMART PANEL**

1. Smart Panel equipment
2. Connection of the Smart Panel to the CLU
3. Information to help you create a configuration
4. Configuration of the Smart Panel module in the version v3
  - 4.1. Configuration parameters
  - 4.2 Creating button and display configurations
  - 4.3 Creating a gesture sensor configuration
  - 4.4 Configuration of the proximity sensor
  - 4.5 Creating a multi-panel configuration of the touch panel
5. Configuration of the Smart Panel v4
  - 5.1. Configuration parameters
  - 5.2. Creating a gesture sensor configuration
  - 5.3. Configuration of the proximity sensor
  - 5.4. Panel object - new functionality
  - 5.5. Panel object - page management mechanism
  - 5.6. Backward compatibility
  - 5.7. Creating a configuration using the Buttons page object
  - 5.8. Creating a configuration using the FreeDraw site object
  - 5.9. Creating a configuration using the Thermostats page object
  - 5.10. Connecting objects to larger buttons
6. Configuration of the Smart Panel v6
  - 6.1. Configuration parameters
  - 6.2. New functionality
  - 6.3. Changing the UI and the mechanism of operation of Thermostats pages

## **XIII. GATE ALARM Module**

1. General information
2. Module configuration

3. Integration with the Satel alarm control panel
  - 3.1. General information
  - 3.2. Configuration for the Satel system
  - 3.3. Virtual Objects
4. Integration with the Jablotron control panel
  - 4.1. General information
  - 4.2. Configuration for the Jablotron system
  - 4.3. Virtual objects
5. Virtual object - Timer
6. Restoring factory settings - *Hard Reset*
7. Configuration parameters

#### **XIV. GATE MODBUS module**

1. General information
2. Module configuration
3. Parameters of registers
4. Restoring factory settings - *Hard Reset*
5. Configuration parameters

#### **XV. GATE HTTP Module**

1. General information
2. Module configuration
  - 2.1. Virtual objects
    - 2.1.1. HTTP Request
    - 2.1.2. Downloading certain values from the received response (XML, JSON)
    - 2.2.1. HttpListener
    - 2.2.2. Preparation of the response sent to the server
    - 2.2.3. Reading key values from the querystringparams parameter
    - 2.3.1. Timer
3. The ability to connect to the Gate using TELNET
4. Comprehensive integration with external systems using the GATE Http device
  - 4.1. System
  - 4.2. Output control
  - 4.3. Status download
  - 4.4. Event order
  - 4.5. Event synchronization
  - 4.6. Feedback confirmation
  - 4.7. Timeout
  - 4.8. A lot of objects
  - 4.9. Status for the complex system
  - 4.10. Push Notifications
5. Restoring factory settings - *Hard Reset*
6. Configuration parameters

#### **XVI. DALI Controller Module**

1. General information
2. Module configuration
3. Objects

#### **XVII. Z-Wave modules**

1. Fibaro UBS
  - 1.1. General information
  - 1.2. Objects
2. NEO Coolcam Motion Sensor (PIR)
  - 2.1. General information
  - 2.2. Objects
3. NEO Coolcam Door / Window Sensor
  - 3.1. General information
  - 3.2. Objects
4. INFIBITY Motion Sensor (PIR) [NEO Coolcam]

- 4.1. General information
- 4.2. Objects
- 5. INFIBITY Door/Window Sensor [NEO Coolcam]
  - 5.1. General information
  - 5.2. Objects
- 6. INFIBITY Water Sensor [NEO Coolcam]
  - 6.1. General information
  - 6.2. Objects
- 7. Heiman Smart Smoke Sensor
  - 7.1. General information
  - 7.2. Objects
- 8. INFIBITY Siren Alarm [NEO Coolcam]
  - 8.1. General information
  - 8.2. Objects
- 9. Danfoss Living Connect
  - 9.1. General information
  - 9.2. Objects
- 10. POPP Z-Weather
  - 10.1. General information
  - 10.2. Objects
- 11. FAKRO AMZ Solar
  - 11.1. General information
  - 11.2. Objects
- 12. FAKRO ARF
  - 12.1. General information
  - 12.2. Objects
- 13. FAKRO FTP\_V
  - 13.1. General information
  - 13.2. Objects
- 14. Fibaro RGBW
  - 14.1. General information
  - 14.2. Objects
- 15. Remotec ZXT-120
  - 15.1. General information
  - 15.2. Description of device configuration
  - 15.3. Objects
- 16. Remotec ZXT-310
  - 16.1. General information
  - 16.2. Device configuration
  - 16.3. Objects
- 17. Aeotec Nano Switch
  - 17.1. General information
  - 17.2. Objects
- 18. Aeotec Dual Nano Switch
  - 18.1. General information
  - 18.2. Objects
- 19. Aeotec Nano Dimmer
  - 19.1. General information
  - 19.2. Objects
- 20. Aeotec Nano Shutter
  - 20.1. General information
  - 20.2. Objects
- 21. Aeotec Nano Shutter (V2)
  - 21.1. General information
  - 21.2. Objects

## **XVIII. MyGrenton mobile application**

- 1. MyGrenton installation

- A. Android
- B. iOS
- 2. Creating the interface
  - 2.1. Adding a page to the interface
  - 2.2. Deleting a page from the interface
  - 2.3. Copying the interface
- 3. Widgets
  - 3.1. Header
  - 3.2. Value
  - 3.3. On/Off
  - 3.4. On/Off Double
  - 3.5. Scene
  - 3.6. Open/Close (SCENE\_DOUBLE)
  - 3.7. Dimmer
  - 3.8. LED lighting
  - 3.9. Thermostat
  - 3.10. Roller shutter
  - 3.11. Roller shutter v2
  - 3.12. Camera
  - 3.13. Text
  - 3.14. Scheduler
  - 3.15. Personalization of the widget
  - 3.16. Widget removal
  - 3.17. Copying widgets
- 4. Personalization of the interface
  - 4.1. Change the name of the interface
  - 4.2. Change the interface icon
  - 4.3. Change the color of the interface
  - 4.4. Blocking access through the cloud
- 5. Sending the interface to the device
  - 5.1. Sending myGrenton interface to your phone using a QR code or manually
  - 5.2. Sharing myGrenton interface via the cloud
- 6. Application and interface settings
  - 6.1. Application settings
  - 6.2. Interface Settings

## **XIX. Grenton 2.0 Logic Distribution**

- 1. Configuration of the Distributed Logic mode
  - 1.1. Operation of Distributed Logic between DIN and output objects
  - 1.2. Operation of Distributed Logic between BUTTON and output objects
  - 1.3. Operation of Distributed Logic between PANEL\_PAGE with PANEL\_BUTTON and output objects
- 2. Default Mode
  - 2.1. Default Mode for input modules and output modules
  - 2.2. Default Mode for modules with their own inputs / outputs
- 3. Restoring communication between the CLU and the module

# Important information

---

**NOTE!** This documentation covers the functionalities and operating principles for Grenton 2.0 series modules. The functional description for the Object Manager as well as the Home Manager is preserved. The myGrenton application is completely compatible with Grenton 2.0 systems - in the case of 1.0 systems, access to specific functions may be limited or completely unavailable.

# I. System structure

---

GRENTON Smart Building System was designed to operate small, medium, and large objects. System created on its basis can be easily modified, expanded, and integrated with other systems.

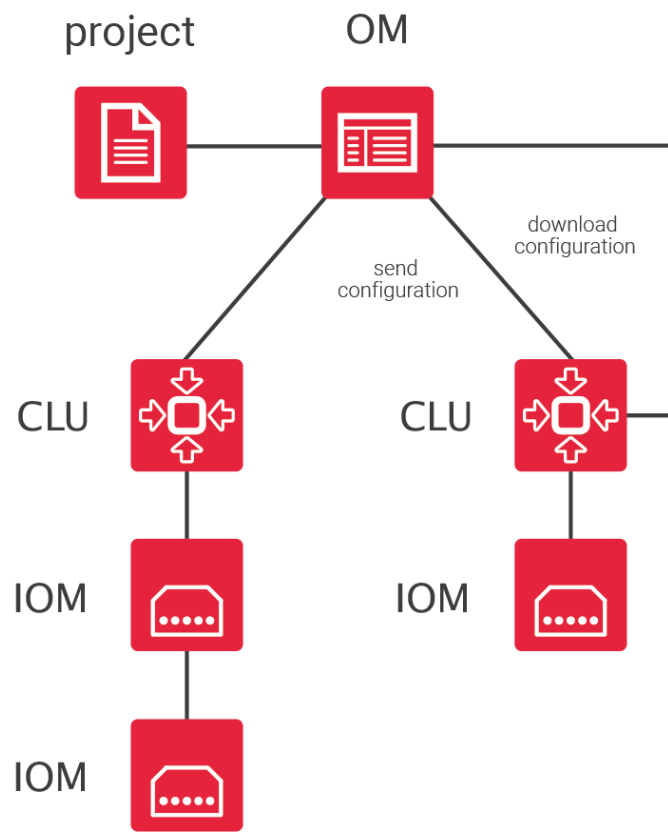
The system consists of: CLU modules, IOM modules, Object Manager, sensors, and applications for smartphone

- CLU (Common Logic Unit) modules. Their function is to process logic and store configurations. CLU is a basis of every system. CLU modules communicate with each other using system bus, working on the basis of standard Ethernet 100 Mbps. CLU module ensures also communication with IOM modules using field bus.
- IOM modules fulfil function of inputs/outputs. They are connected to CLU through TFBUS field bus or in a wireless way, using Z-wave standard. IOM modules may include various types of inputs/outputs, such as relays, switches, light sensors, temperature sensors etc., and their combinations.
- Object Manager - Software that enables configuration of system, logical functions, etc.
- Control applications - they allow activation of designed in OM graphic user interfaces, that enable system functions control using smartphones, tablets, PCs, TV sets, etc.

System configuration is stored as a project file and set using Object Manager (OM) software. Set configuration is then sent to the CLU modules which store it in their memory. IOM modules do not store configuration, they are controlled directly from CLU which they are connected to.

In the case of losing project OM file, downloading data from CLU is recommended. However, downloading data from CLU is associated with losses of: graphic view of scripts, containers, mobile interfaces and objects types (source/load).





# II. Foundation - Grenton Logical Interface

---

## 1. Introduction

---

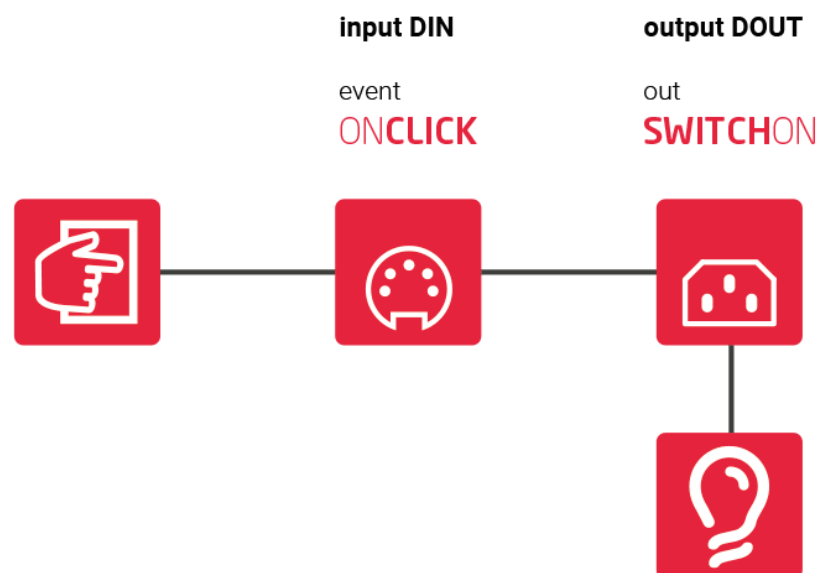
GRENTON system works on the basis of event driven model. Household members and their environment cause generation of events in the system, to which system reactions are connected, e. g. turning on the lamp in response to pressing a switch.

Objects are a basis of the logical interface. In GRENTON system, each object behaves and is treated as a physical object, e. g. a ball. Each object has its own features, we can perform certain actions on it, and it can cause events. In reference to the ball: it is red (so it has its own features), we can kick it (thus controlling it), and it can knock over a bottle while rolling (thus causing an event).

In the system each input and output has its own compilation of features, methods, and events, which is called its logical interface.

A solution unique for the GRENTON system is availability of each feature or method in any place of the system, on each CLU, regardless of where (on which CLU, input, or output) it is placed physically. Thus, it is possible to invoke methods from output connected to CLU A as a result of event that occurred within CLU B.

Moreover, every output has events specific for itself, which enables e.g. switching on one light as a consequence of switching on another. You can find full list of methods and features of each input / output in the catalogue card of the module.



## 2. Features

---

## 2.1. Built-in features

Built-in features is a group of parameters / information describing specific object (input, output, etc.). Some of these features can be set during system operation and are used to determine working method of an object (work mode of a button), while others can only be read, since e.g. they show physical parameters (temperature feature for thermometer).

## 2.2. User features

In CLU you can define features that will be used as variables in storing parameters during system operations, e.g. counters, markers (flags). User features can be used exactly as built-in features, except all user features can be saved and read.

## 3. Methods

---

Methods are commands which can be given to a specific object. Each object has its own characteristic methods. For relay output, it can be methods `SwitchOn` and `SwitchOff`. Additionally, the methods can include required or optional parameters, which specify details of method invocation e. g. switch-on time.

## 4. Events

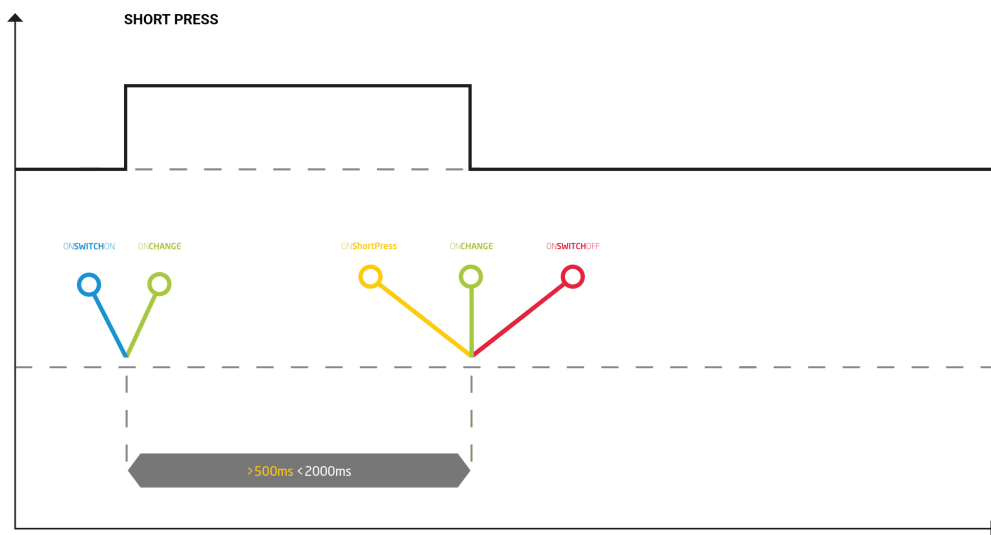
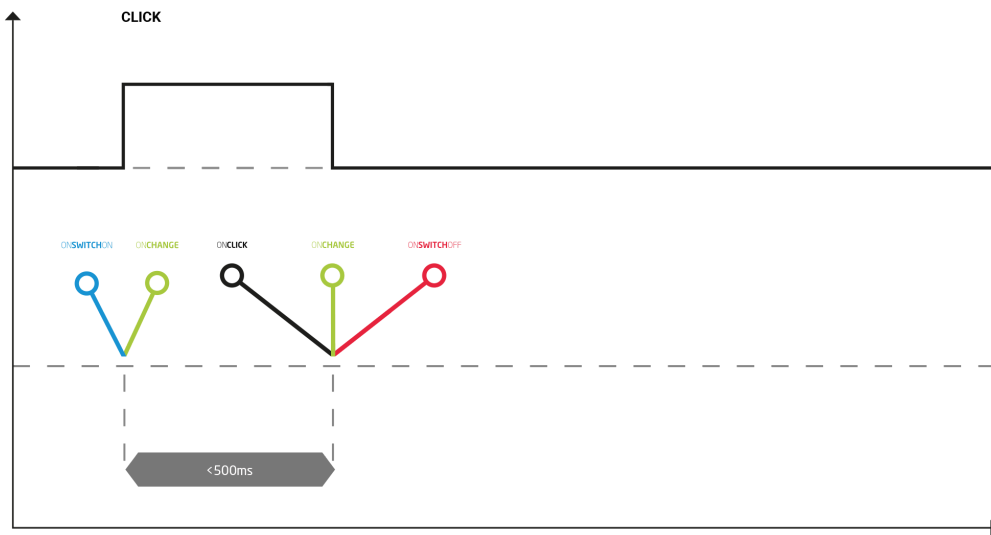
---

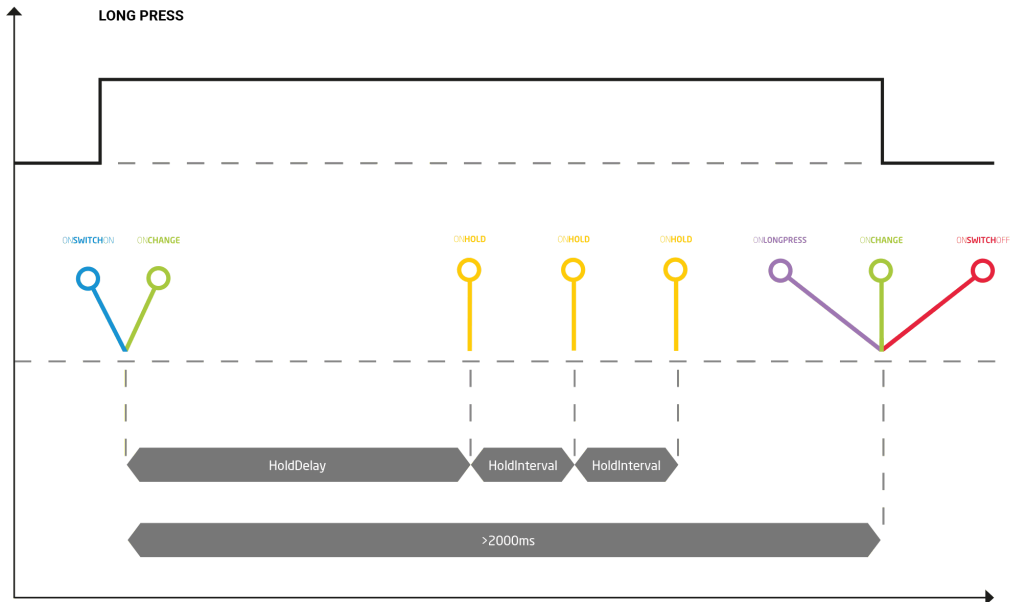
Events are elements of logical interface, invoked in reaction to the changes occurring in relation to an object (e.g. pressing a button, temperature change, etc.). We can connect one or more methods to each event, which will be invoked when an event occurs, e.g. when the button is pressed, the lights will be switched on. By connecting events of one object (mainly inputs, but sometimes also outputs) with methods of other object, we create logical configuration of the system.

Each type of object (type of input/output) has its own list of events, which are invoked in a precisely specified way, depending on the actions performed by the user. For instance, binary input has the following list of events:

- `OnChange`
- `OnSwitchOn`
- `OnSwitchOff`
- `OnShortPress`
- `OnLongPress`
- `OnClick`
- `OnHold`

which are invoked according to the following scheme:





## 5. Features and methods addresses

Each feature and method has an address in the system, thanks to which they can be invoked in scripts and during creating connections with events. Address consists of 3 parts, joined by “->” mark:

- CLU or container identification
- Object name (input, output, CLU)
- Name of feature/method and its parameters (if there are any)

For example: `CLU1->Lamp1->SwitchOn()` - method causing switching on output `Lamp1`

`Lights->Lamp1->value()` - feature showing whether the lamp is switched on or off, for lamp placed in the “Lights” container.

# III. Project preparation

## 1. Electrical system preparation

**NOTE!** Electrical systems in residential and public utility buildings may be established only in accordance to mandatory provisions and electrical standards and only by authorized, qualified specialists

### A. Electrical system topology

GRENTON System enables creation of both centralized and distributed systems. For newly designed buildings, we recommend connecting all its circuits to one electrical distribution board, which ensures more flexibility in systems design and more sustainable resources management. Every device that will be connected to the system should have its own, separated electrical circuit, ending in the electrical distribution board. Select wires diameter in accordance to the mandatory standards. If there is no chance of connecting electrical distribution board and the controlled device directly, there are three possible solutions:

1. Using CLU module together with IOM modules, CLU modules in the distribution board are connected to the device module using system bus - recommended solution when there are two or more buildings integrated into one system.
2. Using one or more IOM modules, the modules are connected using field bus - recommended solution when there are only a few device modules.
3. Using IOM radio modules based on Z-WAVE - recommended solution when there is no chance of using wiring installation (pre-existing buildings etc.).

### B. Bus

There are 2 buses in the system:

1. **System bus**, used for connections between CLU-CLU and CLU-SMARTPHONES modules etc.

System bus - Ethernet. Modules can be connected to each other using serial connection. The maximum length of wire between two CLU modules is 90 m.

UTP wire is recommended (minimum 5e category).

System bus length could be increased by using network devices such as switch, router etc.

2. **Field bus**, used for connections between CLU-IOM modules.

Field bus - IOM modules can be connected to the field bus using bus (TF-Bus) wire and also connected to the bus using the BUS MODULE. Modules must be connected to each other using serial connection. Maximum length of the bus from one end to another is 300 m.

**NOTE!** Separate power supply for the bus may be necessary.

Wire with constant surge impedance and minimum cross-section of 0.5 mm<sup>2</sup> is recommended, e.g. UTP wire (optionally, covered wire: FTP or E-BUS). With larger number of modules or more extended bus, potential drops should be considered when choosing diameter of the bus wire.

## C. Useful tips

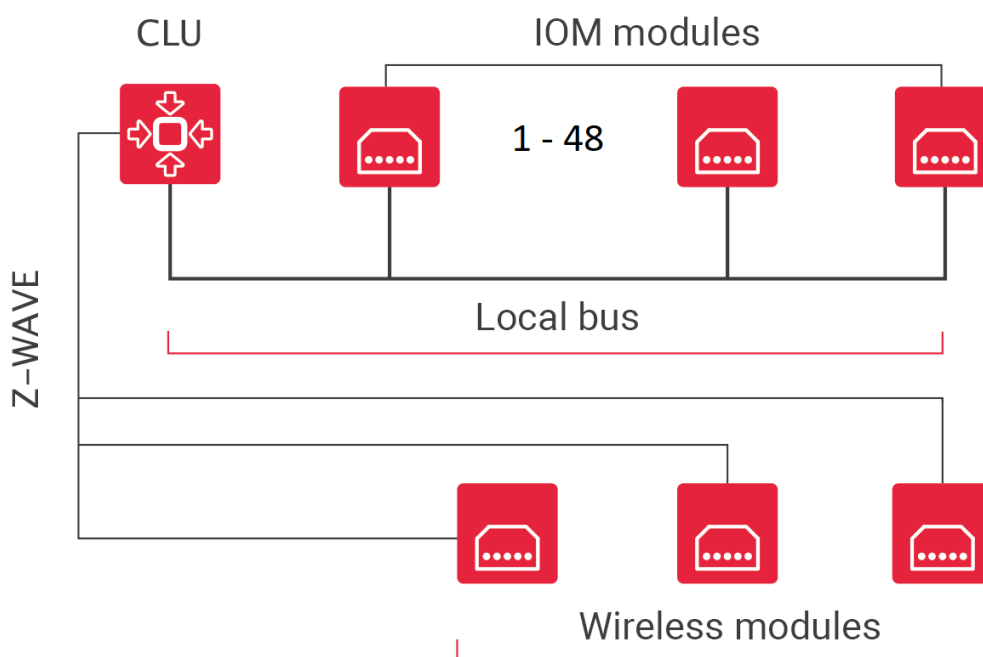
- Before implementing electrical system project, prepare smart house system project.
- If you don't know yet, which devices will be controlled by the system, make sure the wiring reaches all possible places.
- For light switches, any thin wire may be used, e.g. YTDY - it will allow saving on wire.
- Remember to prepare the system for temperature sensors and weather station.
- Place power outlet on the terrace and connect it to a separate power supply - you will be able to control the power in the outlet through the system.

## 2. System architecture selection

Various configurations may be used depending on type, size, and requirements of objects - the system is fully scalable. Depending on scale and needs, there are several available configurations:

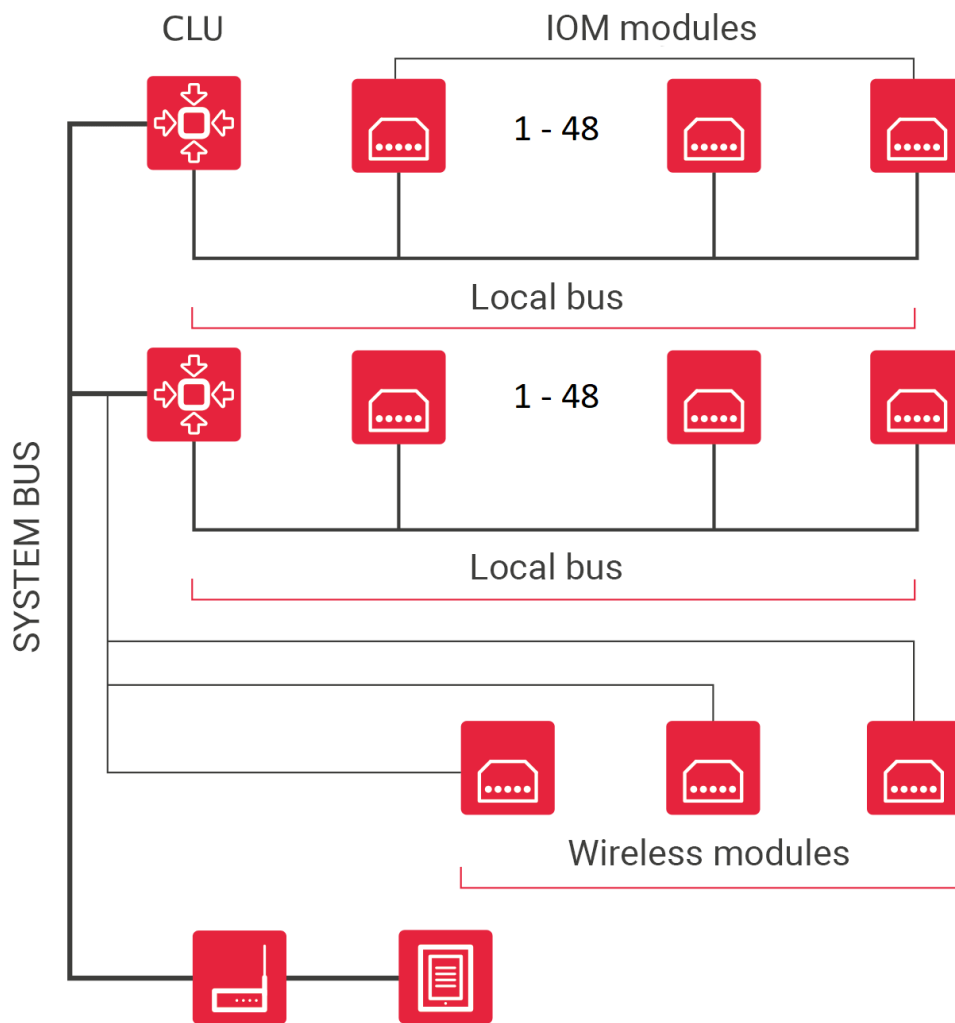
### A. Basic configuration - centralized system with one CLU

The diagram presents a system built on the basis of one CLU. In a system configured this way the maximum number of IOM modules equals 48 (or up to 400 objects). Remember to provide the bus with power adequate to its load.



### B. Advanced configuration - tablet-controlled distributed system with many CLU

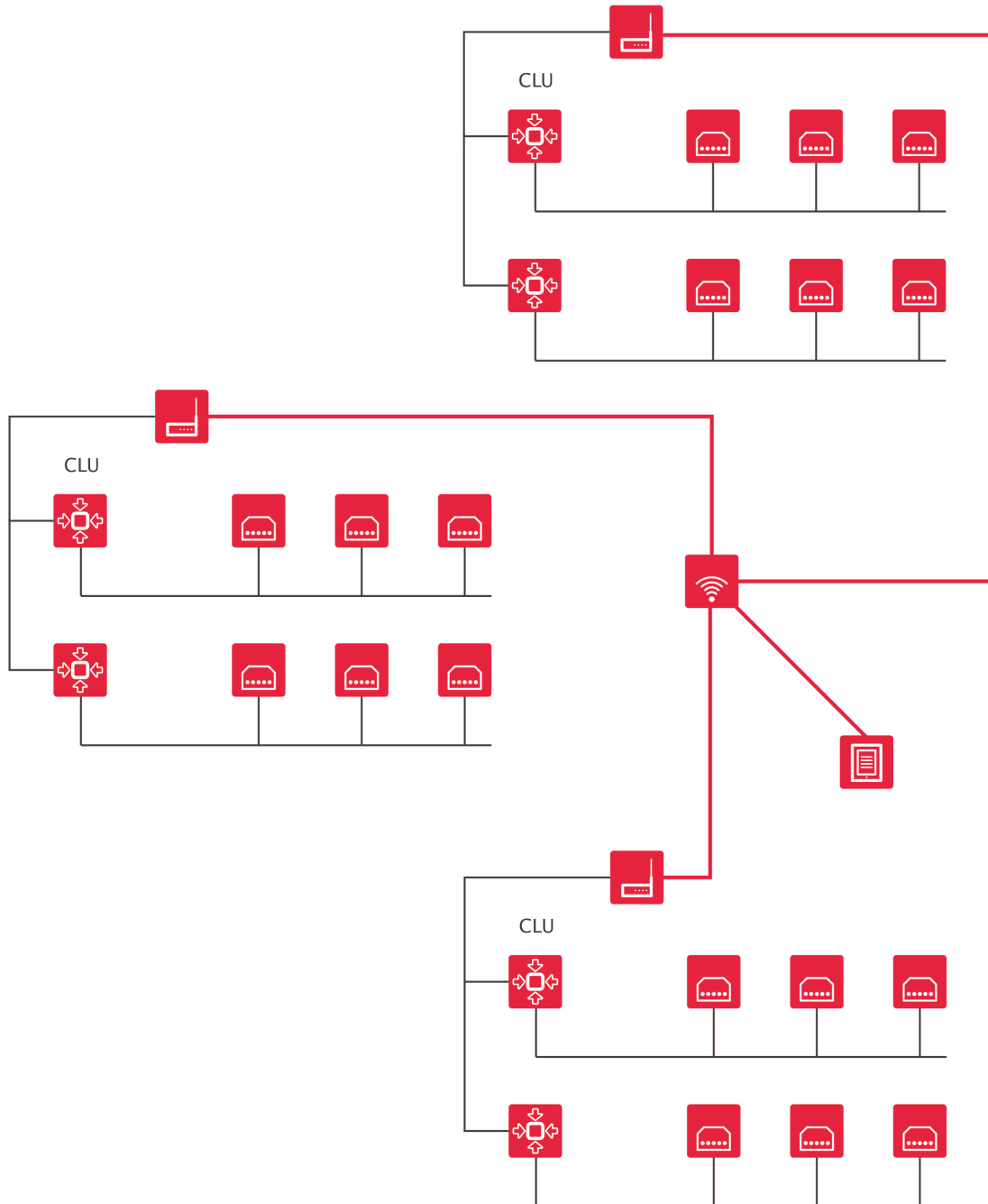
System capacity can be increased by adding next CLU modules. CLU units are connected to each other using system bus. The system may be additionally expanded with smartphones, tablets, etc.



### C. Integrating several buildings into one system

System expansion is practically unlimited. Several objects can be connected to one system. Thanks to that, you can have central control using only one system.

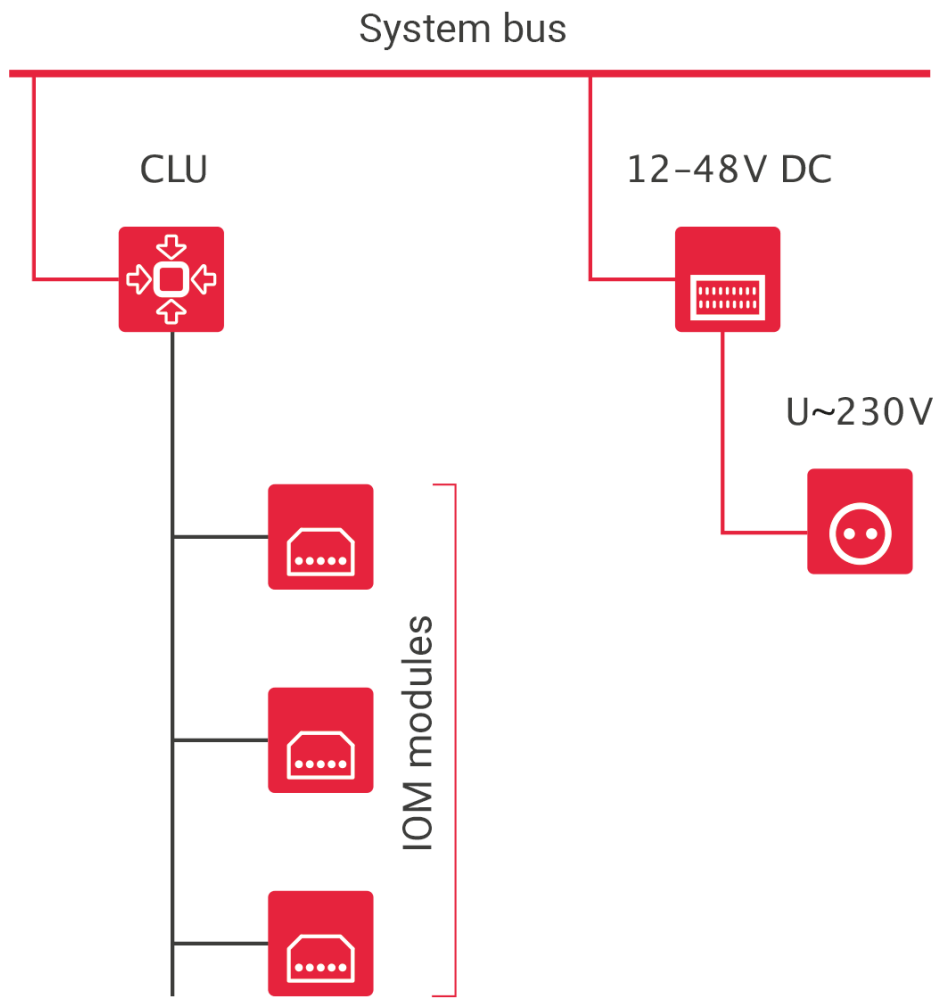




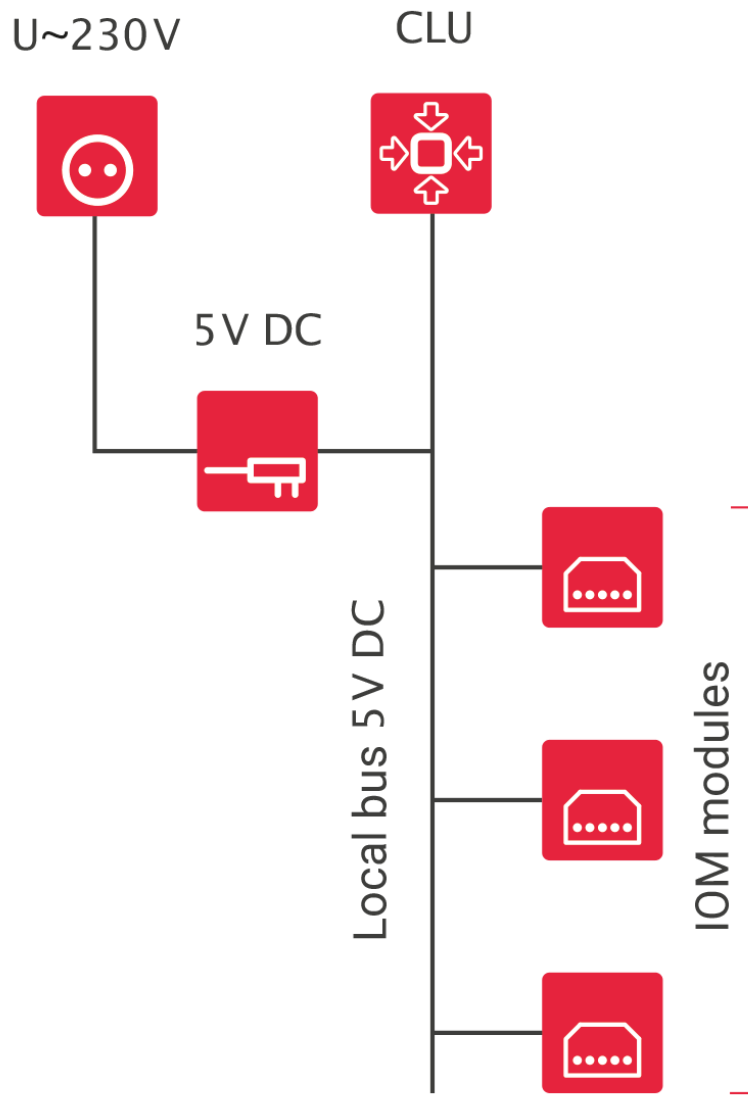
### 3. Modules power supply

The CLU and IOM modules can be powered in two ways:

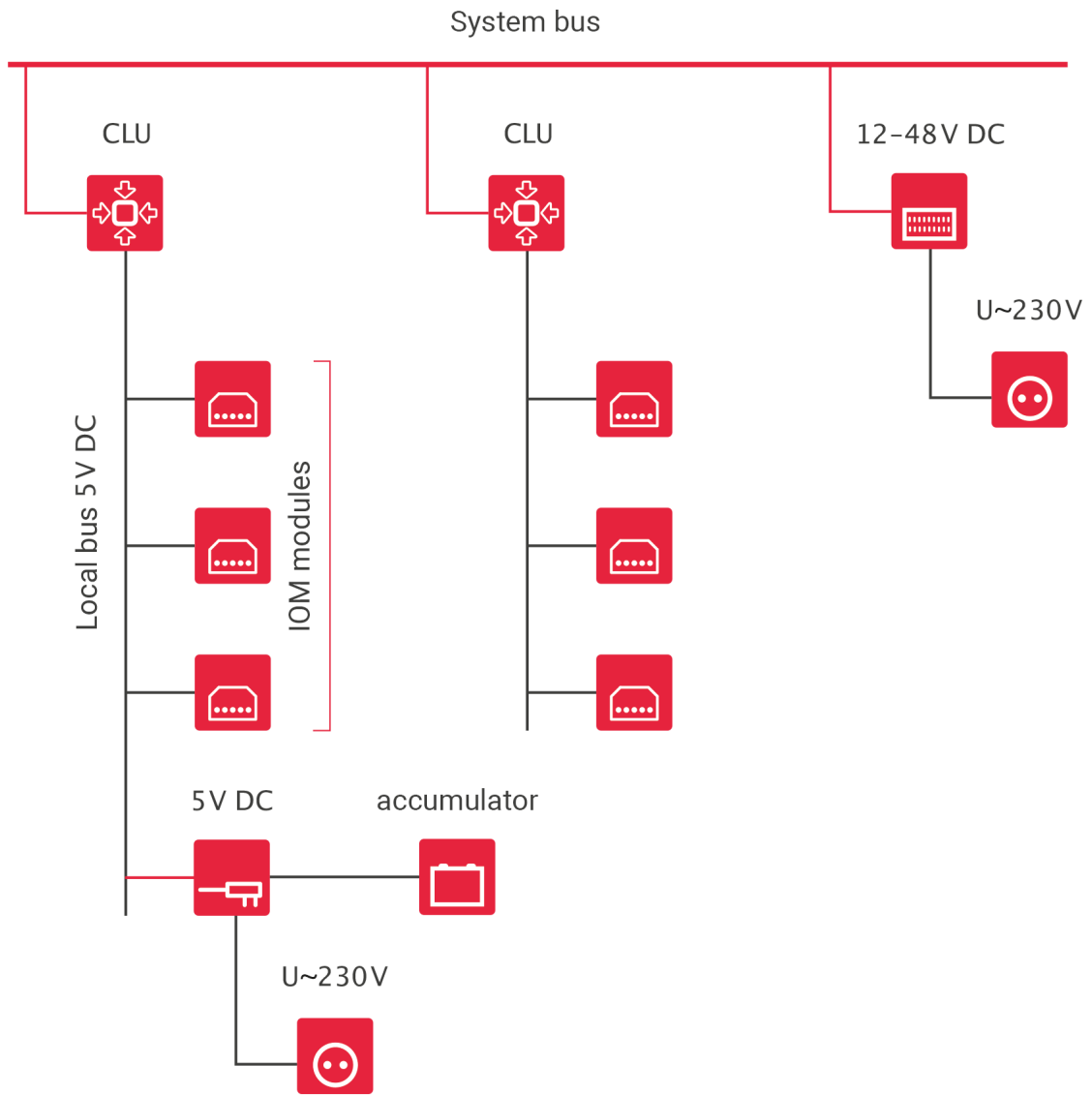
1. By connecting the power supply to the 24V DC system bus - in this case the CLU module will power the IOM modules connected to it via the local bus. A maximum current of 1000 mA can flow through the local bus (TFbus).



2. By connecting the 24 V DC power supply to the local bus. In this case, the CLU will be powered from the local bus.



For flush-mounted modules, it is possible to optionally use a 24 V DC flush-mounted power supply.



**NOTE!** CLU can be simultaneously connected to the power supply from the system bus and local bus!

# IV. Components installation

---

Majority of modules is provided in two versions: on the DIN rail to be assembled in the distribution centre, and flush-mounted. In addition, Z-Wave modules are available: Relay, Roller Shutter and Digital IN.

## 1. Modules installation in the switching action

---

Modules offered by GRENTON are provided in cases adjusted to assembly in the distribution centre on a DIN rail. To assembly a module, place it on the rail and block the latch on the underside of the module. Then, connect the modules to the system bus using special bus connectors, and attach connecting wires according to the assembly manual attached to the modules.

**NOTE!** Modules in the OM are identified using a serial number. After installing a module, write down its serial number and physically connected inputs / outputs, it will facilitate identification of specific objects.

## 2. Flush-mounted wire modules installation

---

Modules designed for flush-mounted installation are adjusted for installation in junction boxes of 70mm diameter, as well as majority of boxes of 60mm diameter. In the case of boxes of 60mm diameter, check beforehand if the module fits in the specific type of a junction box.

In the case of planned installation of larger number of modules, use deepened junction boxes.

## 3. Z-Wave flush-mounted modules installation

---

Wireless modules are adjusted to assembly in junction boxes of minimum 60mm diameter. It is For flush-mounted modules it is recommended to use cans with a side pocket.

# V. Object Manager

---

## 1. OM installation

---

Minimum system requirements for the computer and detailed Object Manager configuration software installation manual is attached to software installation files.

Current Object Manager version could be downloaded from: <https://www.grenton.com/wsparcie/materialy-do-pobrania.html>

**NOTE!** The folder in which the Object Manager will be installed can not contain special characters in the name ie. %, !, # etc.

### A. Windows

- Download the file .exe.
- Run the file.
- Select the Object Manager installation path.
- Start unpacking by pressing Extract.
- After unpacking, run the om.exe file located in the */object-manager* directory.

### B. macOS

**NOTE!** The application name contains the version number, which allows multiple OM versions to exist on one computer. Before removing previous versions, it is recommended to migrate / copy project files stored in the application file by default. To do this, in the Finder, select (Ctrl-Click) "Show Package Contents" and copy or move the folder containing the project files (om.app/Contents/MacOS/projects) to the new version of the OM application.

- Download the file.
- Run the file.
- Copy the Object Manager application to the Application folder as suggested.
- Start the Object Manager application in the standard way.

### C. Linux

- Download file object-manager.tar.gz.
- Extract the downloaded file in the selected location.
- After unpacking, run the ./om file located in the */object-manager* directory.

## 2. OM structure

---

Object Manager is operated through three available for the user menu panels:

- **MAIN MENU**

File Edit Tools Window Help

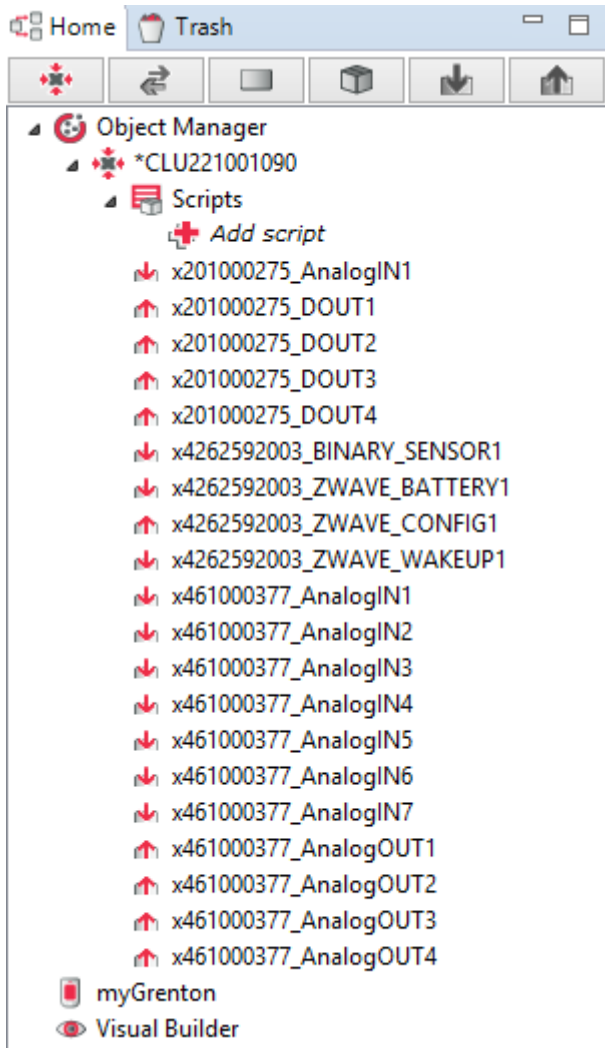
Contains basic commands for handling the project.

- **ACTIONS MENU**



Icons in the menu are used for programming and configuration of devices. Only icons that can be used at the moment are illuminated - it comes from the context which you are present in, e. g. if you selected CLU in the side tree, icons connected to CLU become active.

- **OBJECTS MENU**



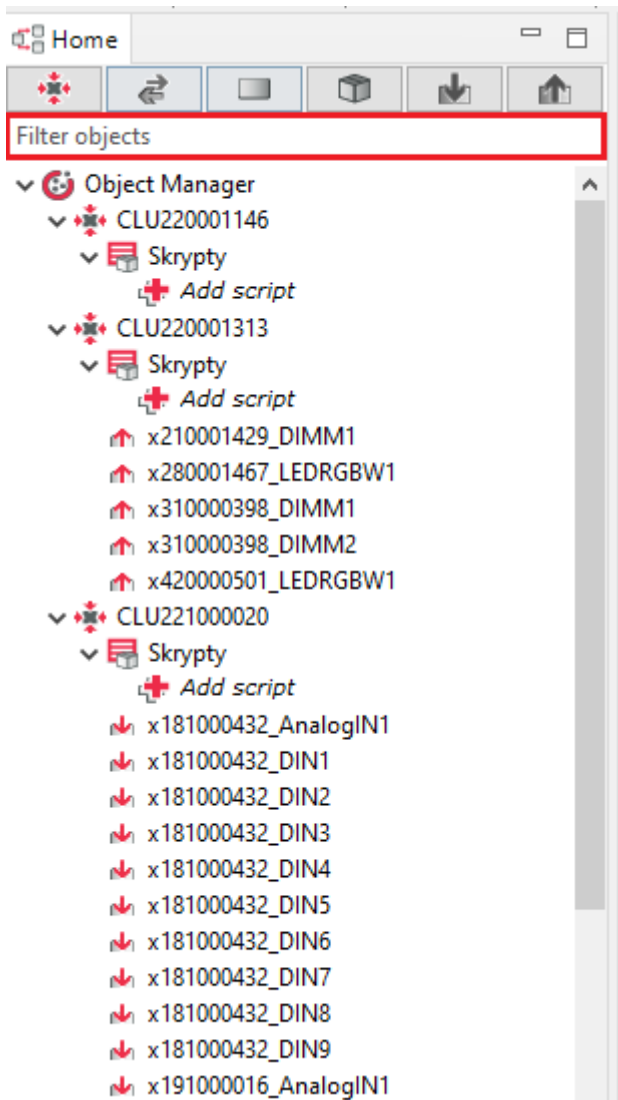
Consists of three parts: Object (CLU, inputs, outputs) list, the myGrenton tab (creating the myGrenton application interface) and Visual Builder (creating the Home Manager application interface).

All system configuration data is stored in the project file. In OM, any number of projects can be stored, each of them connected to different installation / building / apartment.

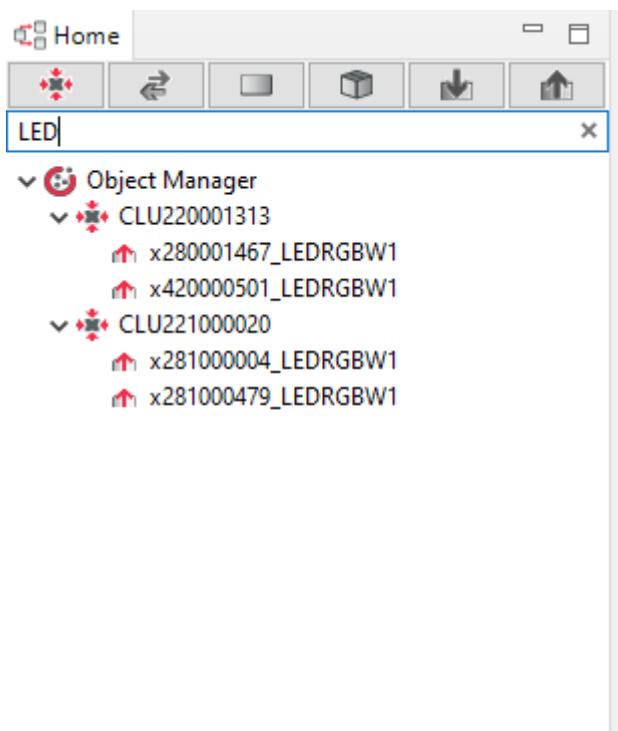
## 2.1. Object filtering

**NOTE!** The `Filter objects` functionality is available for Object Manager version 1.5.0 or higher.

For each grouping view (by CLU, by module, by type, by containers, only inputs, only outputs) it is possible to filter the displayed elements using the `Filter objects` option.



After entering the name to be searched, all elements (objects) containing the entered phrase in the given view are displayed.



The given phrase is included in the filtering after switching to another group view (tab).

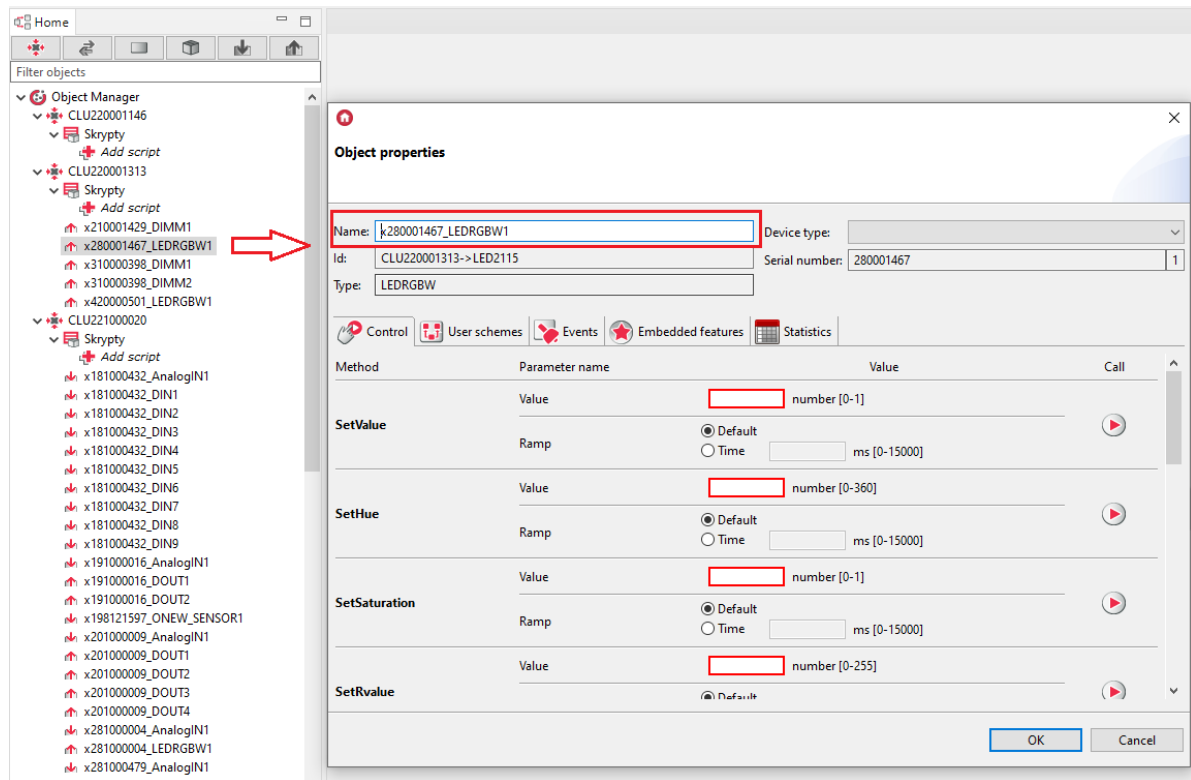


## 2.2. Renaming an object

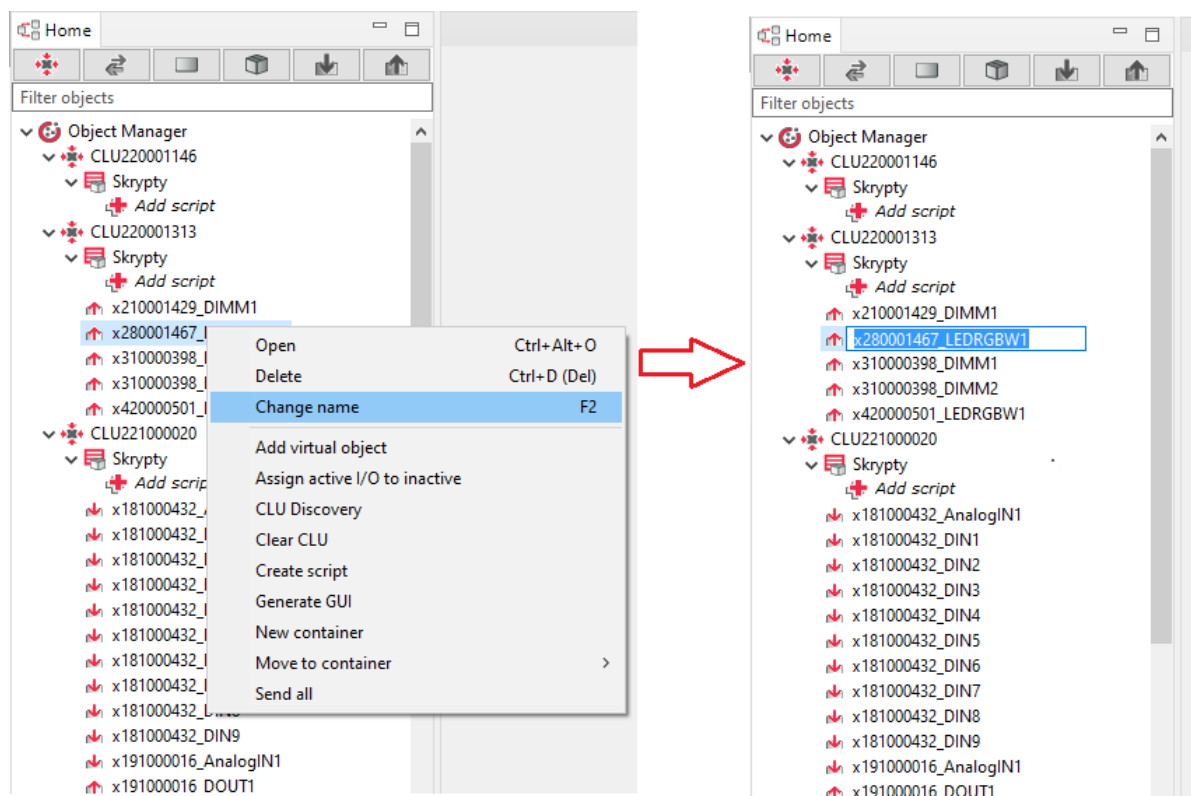
**NOTE!** The possibility of renaming an object using the context menu / **F2** is available for Object Manager version 1.5.0 or higher.

Renaming a given object can be done in the following way:

- In the Object properties window, enter a new name in the **Name** field.



- selecting the object and calling the rename option from the context menu or by using the **F2** keyboard shortcut.



**NOTE!** Renaming from the context menu / `F2` is not available for Home Manager interface elements.

## 3. Project files

---

### 3.1. Saved projects catalogue

After Object Manager installation, select a catalogue, in which the saved projects will be stored.

Default destination path for the catalogue: `C : \.... \OM\projects`

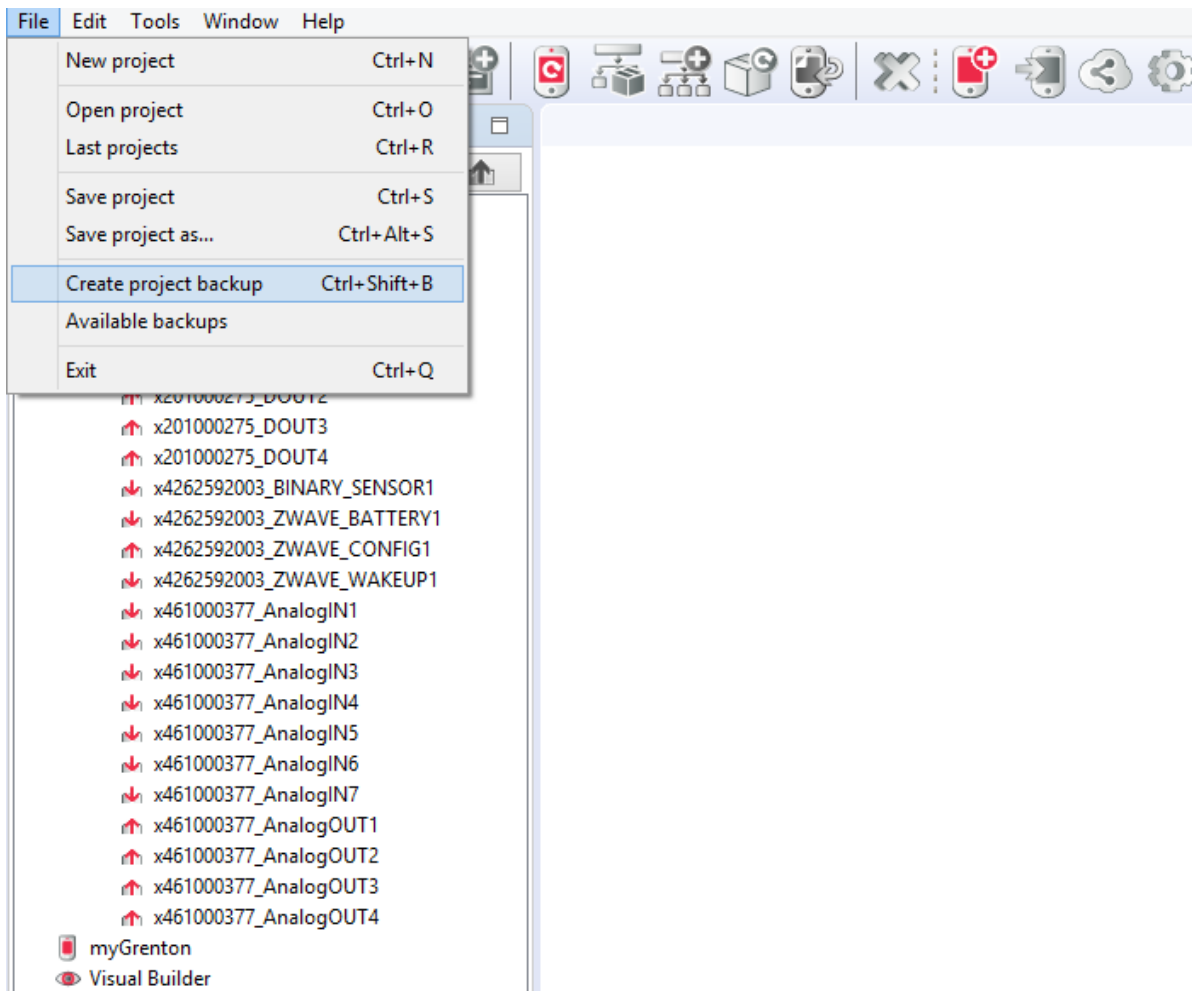
All files of created and saved projects are saved in the catalogue with `*.omp` extension. (e.g. `projekt.omp`).

### 3.2. Project backup

During work on a project there is an option to make a backup of the project which won't be modified despite making changes in the project. This way, it is possible to recover earlier version of the project if the user made unwanted configuration changes. Any number of backups can be made for each project.

**NOTE!** It is recommended to make backups as often as possible, especially before making significant changes of system configuration.

To make a project backup, click `File->Make project backup` in the main menu (backup can also be made by keyboard shortcut `CTRL+Shift+B`).



Saved backups are available in the list opened by clicking `Available backups`, or in the project opening window in a tab `Backups`.

**NOTE!** After selecting a backup from the list it will be loaded, and any current changes in the project that haven't been saved will be lost.

## 4. Basic elements

### 4.1. Objects configurator

Each input, output, sensor, or other physical device connected to the system is visualised as an object in the OM. Objects do not show physical modules, but specific inputs and outputs. Each object has its initial values, built-in features, and events, displayed in the object configurator. After clicking on an object, this form opens.

**Object properties**

Name:  Device type:

Id:  Serial number:

Type:

Control | User schemes | Events | Embedded features | Statistics

Method	Parameter name	Value	Call
SetValue	Value	<input type="text" value="Off"/>	<input type="button" value="▶"/>
Switch	Time	<input checked="" type="radio"/> Unlimited <input type="radio"/> Time <input type="text"/> ms	<input type="button" value="▶"/>
SwitchOn	Time	<input checked="" type="radio"/> Unlimited <input type="radio"/> Time <input type="text"/> ms	<input type="button" value="▶"/>
SwitchOff	Time	<input checked="" type="radio"/> Unlimited <input type="radio"/> Time <input type="text"/> ms	<input type="button" value="▶"/>
SetOverload	Overload	<input type="text" value=""/> W [0-3000]	<input type="button" value="▶"/>

The form above consists of the following sections:

### 1. Basic information

Name:  Device type:

Id:  Serial number:

Type:

This section is located in the upper part of the form and contains basic information on each object, e. g. name, Id, module type, serial number and input/output number within the module. In this section the user can also define type of source or receiver, physically connected to the object.


### 2. Control Tab

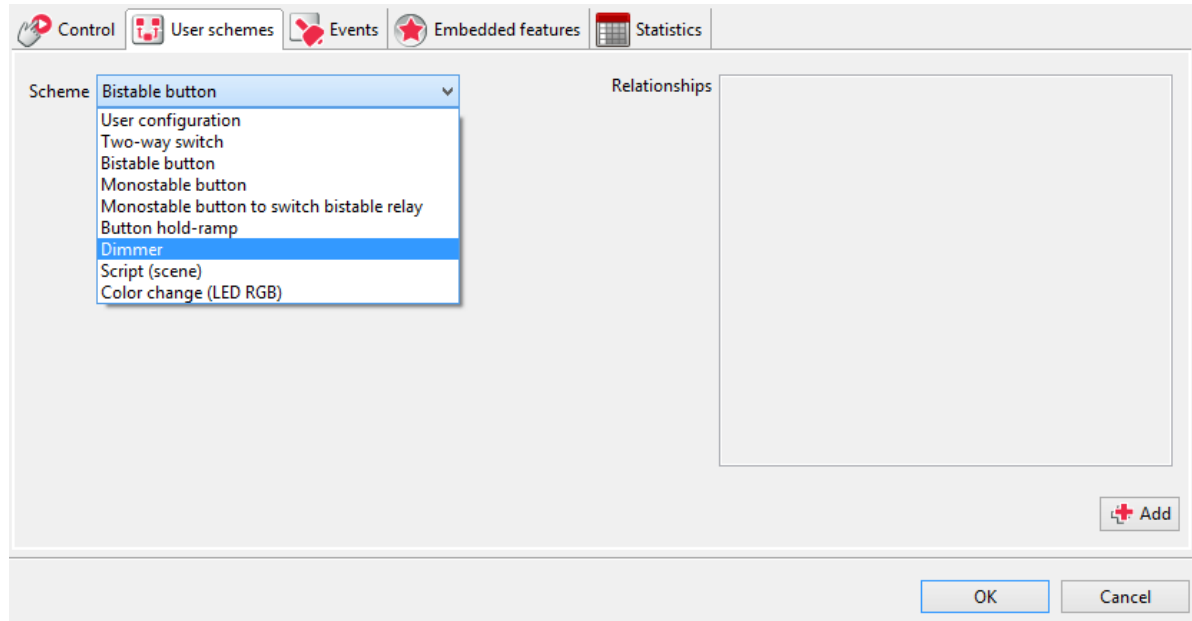
Control | User schemes | Events | Embedded features | Statistics

Method	Parameter name	Value	Call
SetValue	Value	<input type="text" value="Off"/>	<input type="button" value="▶"/>
Switch	Time	<input checked="" type="radio"/> Unlimited <input type="radio"/> Time <input type="text"/> ms	<input type="button" value="▶"/>
SwitchOn	Time	<input checked="" type="radio"/> Unlimited <input type="radio"/> Time <input type="text"/> ms	<input type="button" value="▶"/>
SwitchOff	Time	<input checked="" type="radio"/> Unlimited <input type="radio"/> Time <input type="text"/> ms	<input type="button" value="▶"/>
SetOverload	Overload	<input type="text" value=""/> W [0-3000]	<input type="button" value="▶"/>

Contains methods (with all parameters) relevant for the browsed object. Enables invocation of specific method from level of OM. For instance, for relay output you can invoke `SwitchOn` method with Time parameter (e.g. 30s), which will cause switching the output on for 30s. To invoke method at OM level, enter parameter values (if they are necessary) of the invoked method in the control tab and click "invoke" button.

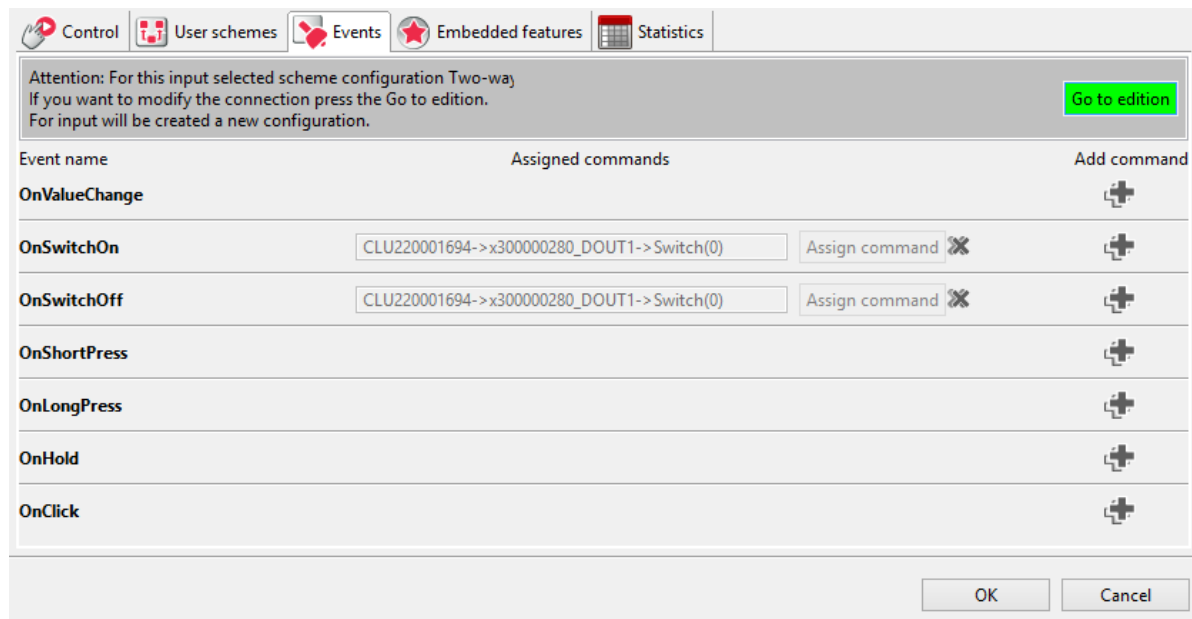
### 3. Configuration chart

Configurations charts define object behavior and allow simplified logic configuration. To add a configuration scheme, select the available scheme from the list, and then add a link with the module by clicking  - [look up VI.9.](#)



### 4. Events - tab description

The tab contains list of events applicable for the specific object type and methods connected to them, which are invoked after event occurrence (if the user defined them). If configuration chart was selected, the tab is in the read only mode and shows only connections created as a part of the selected chart.



You can go to event-method connections edition any time by clicking "Go do edition". In this case, "user chart" will be created, which will show up on the list in the configurations charts tab.

Control User schemes Events Embedded features Statistics

Event name	Assigned commands	Add command
OnValueChanged		
OnSwitchOn	CLU220001694->x300000280_DOUT1->Switch(0) <input type="button" value="Assign command"/>	
OnSwitchOff	CLU220001694->x300000280_DOUT1->Switch(0) <input type="button" value="Assign command"/>	
OnShortPress		
OnLongPress		
OnHold		
OnClick		

After adding command to selected event, objects list opens. Then, after selecting proper object, list of methods that can be invoked on it appears. Adding a selected method results in creation of new dependence between objects.

## 5. Built-in features

This part presents values which the selected objects currently possesses, and initial values which was saved in it (initial values set in the case of system restart, e.g. after power supply break). Entering value in the "Initial values" field will result in setting it during CLU start.

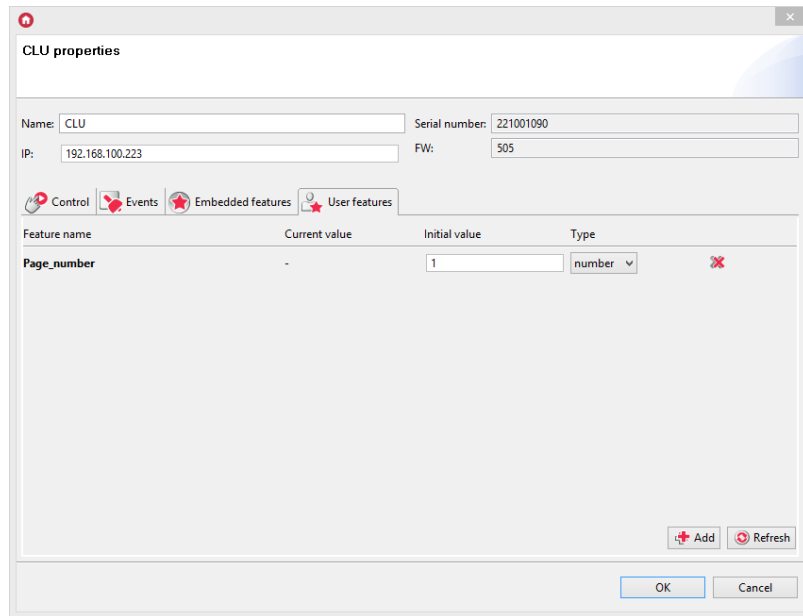
Control User schemes Events Embedded features Statistics

Feature name	Current value	Initial value	Unit	Range
Value	0	Off <input type="button" value="v"/>	bool	0,1
StatisticState	0	Off <input type="button" value="v"/>	number	0,1
VoltageType	2	Signal <input type="button" value="v"/>		0,1,2
VoltageValue	230	<input type="text" value="230"/>	V	[0-230]
Power	0		W	[0-3000]
Overload	3000	<input type="text" value="3000"/>	W	[0-3000]
DistributedLogicGroup	0	<input type="text" value="0"/>		[0-10000]

Auto refresh

## 6. User features (only CLU)

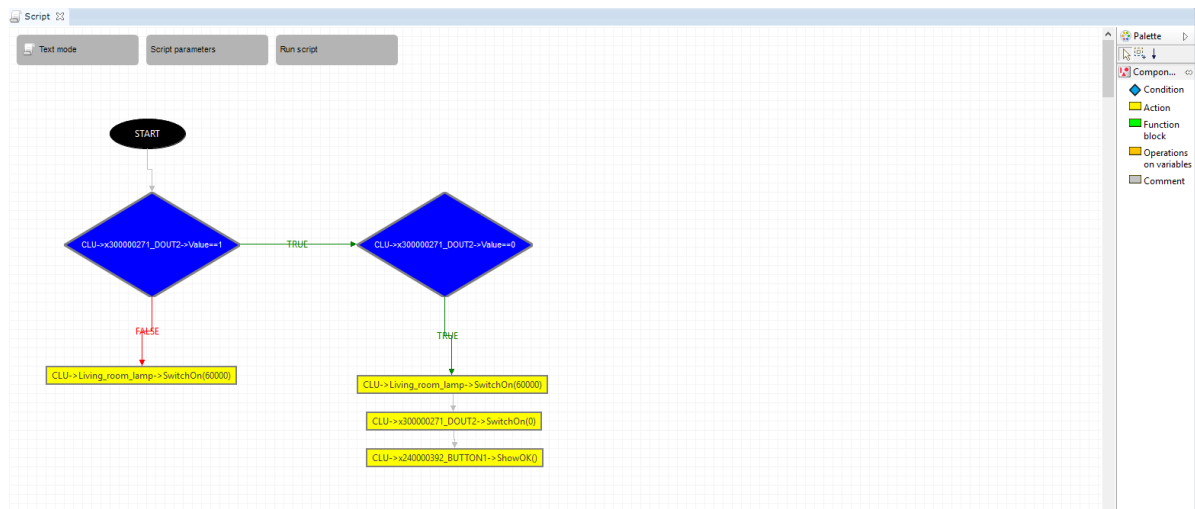
The tab allows user to define in the CLU his own list of features, which then can be used to store various type of data (counters, markers). Adding user feature happens after clicking the **Add** button and entering feature name. Then, feature's initial value and type (text, numeric, boolean) have to be defined.



## 4.2. Script builder

It's a tool for script creation, which can work in two modes:

1. **Graphic (simplified) mode**, in which the chart can be created in an easy way by dragging and connecting elements.



The graphic mode allows to create complicated scripts made of numerous conditions and methods. It is also possible to use variables and parameters. The only limitation is no possibility of creating loops, which require using text mode.

2. **Text (full) mode**, in which the user can create the logic using advanced LUA language. Thanks to that, creation of very complex charts using all elements of LUA language (including loops, tables, etc.) is possible

```

Script
Graphical mode Parameters Run script
1 if(not (CLU->x300000271_DOUT2->Value==1)) then
2 CLU->Living_room_lamp->SwitchOn(60000)
3 else
4 if(CLU->x300000271_DOUT2->Value==0) then
5 CLU->Living_room_lamp->SwitchOn(60000)
6 CLU->x300000271_DOUT2->SwitchOn(0)
7 CLU->x240000392_BUTTON1->ShowOK()
8 end
9 end
10

```

In comparison to the standard LUA, the language was expanded with possibility of direct linking to addresses of methods and features, which are treated same as other functions of LUA.

### 4.3. Connections diagram











A tool showing dependencies and connections between all objects in the system. Thanks to that tool, you can easily and quickly find a dependency that interests you, or check dependencies of a specific module without going through configurations.

Connections diagram may be run from the main menu: Tools -> Connections diagram, or using keyboard shortcut [ ALT+Q ].

Each object in the system is presented in the diagram by a circle with its address displayed next to it. The colour of a circle depends on the object type:

- CLU - red colour;
- Input / output - cherry colour;
- Events of inputs or outputs - light blue colour;
- Events generated by timers - dark blue colour;
- Built-in methods - dark green colour;
- Script methods - light green colour
- Built-in features - yellow colour;
- Defined features - orange colour;

Connections between objects are displayed as arrows which heads point to the invoked object.

TYPE OF CONNECTION	ENDING FROM THE SIDE OF THE ACTIVE OBJECT	ENDING FROM THE SIDE OF THE PASSIVE OBJECT
Unconditional method call		
calling methods covered by the condition		
reading of the values of features		
record of the values of features		
Bidirectional objects binding		

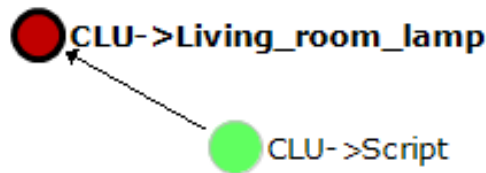


## Connections are displayed on three different levels:

1. CLU-CLU - displays connections between two CLU, if any of objects of one CLU (input, output) is connected to another CLU.
2. Connections between objects - displays connections between specific objects (inputs, outputs) without showing specific events, features, or methods.
3. Connections between events, methods, and features - displays the most detailed view, showing what specific events cause etc.

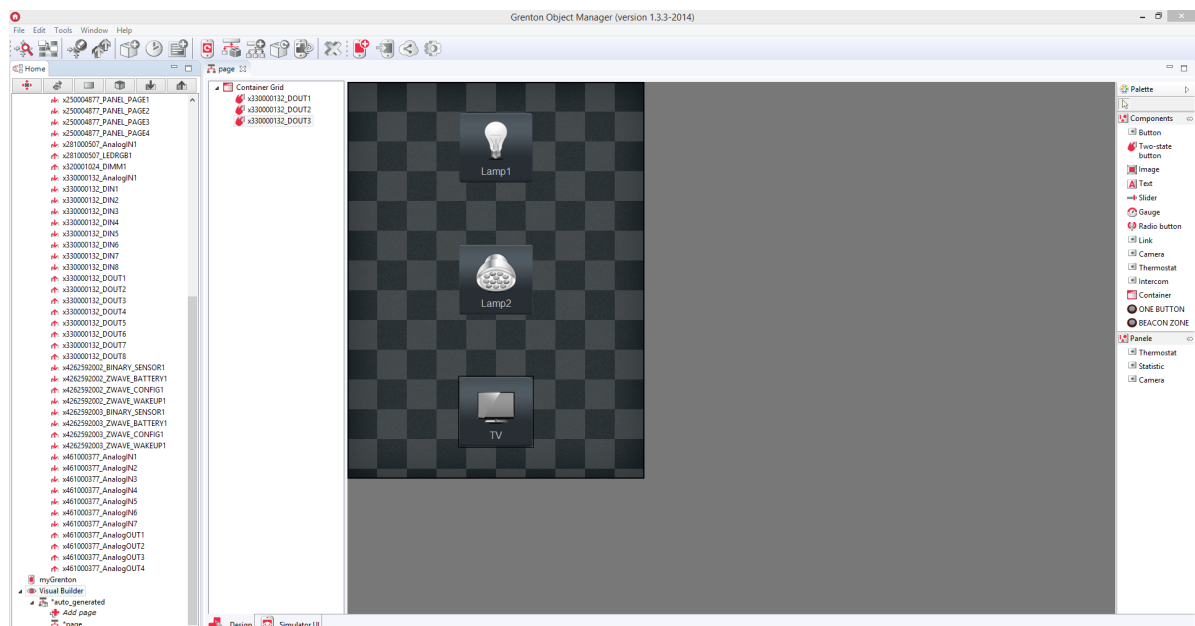
## Navigation also happens in two planes:

1. In the vertical plane - allows switching between objects on the same level by clicking any object except central in the chart.
2. In the diagonal plane - allows going up and down between levels by clicking on the central object and selecting an object from the appearing list (for going down) or by pressing "up" button in the upper part of the chart (for going up).



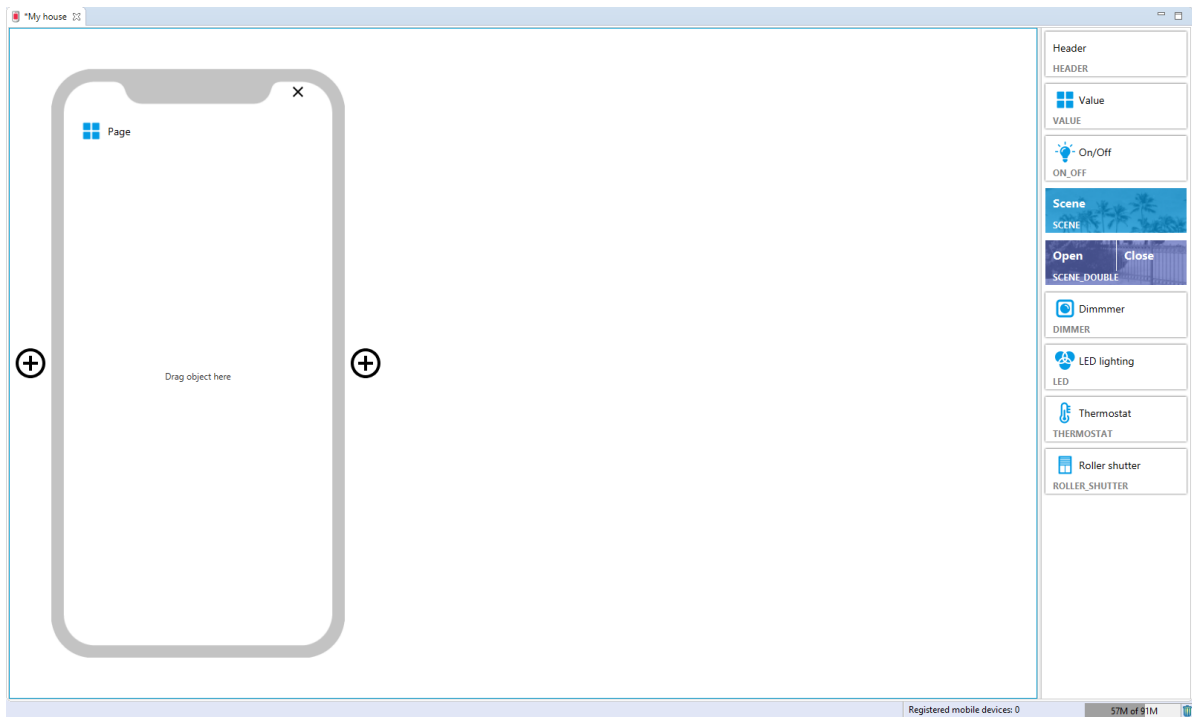
## 4.4. Visual Builder

Visual Builder is a tool used to create user interface for mobile devices. The interface can be created automatically on the basis of installation project, or can be designed and created by the user acc. to their personal preferences. The user has an option of using their own graphics. Interface is created through drag&drop of Visual Builder components. It enables creation of interface for all popular resolutions. The icon switching on the VB is placed at the end of expanded objects tree.



## 4.5. myGrenton

A tool for creating a user interface for mobile devices for the myGrenton application. Creating the interface is done by dragging & dropping elements from the object tree available in the project (modules connected to CLU) - the widget will have a predefined template.



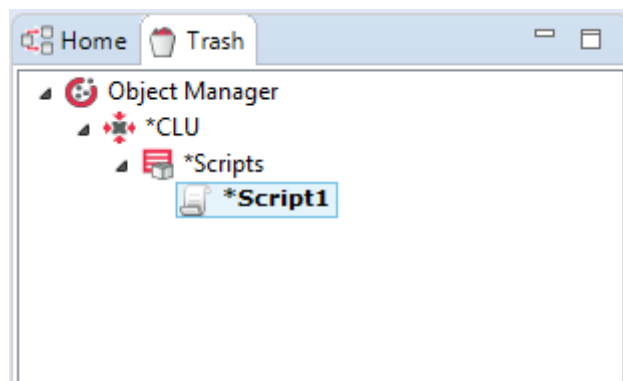
## 4.6. Bin

It is modelled after solution known from operating systems. Deleted object, script, or application in the project is not irretrievably removed, but moved to the bin, thus giving the user a possibility of retrieving deleted data in the case of change in the concept.

The bin has a form of a tab placed in the objects tree, and appears whenever an object is deleted. Objects from the bin can be restored at any moment by right-click and selecting "restore" from the context menu.

Objects can be irretrievably removed from the bin by selecting "delete" from the context menu. Restoring a module removed in this way to the project is only possible by executing *CLU Discovery*.

The bin is a great solution for storing objects which are not used at the moment but might be useful in the future.





# VI. Basic system configuration

---

## 1. Connecting OM to CLU

---

To configure devices in the system, the computer has to be connected to the CLU modules. During operations performance, all CLU modules must be connected to each other using Ethernet cable.

There are two connection methods:

1. Direct connection to the computer Connect network cable to the network card in the computer and to the network socket in the CLU module.
2. Connection through local network It is possible to connect with GRENTON system using local network. In order to do that, both CLU module and the computer which will be used for establishing connection must be in the same sub-network.

## 2. IP addressess

---

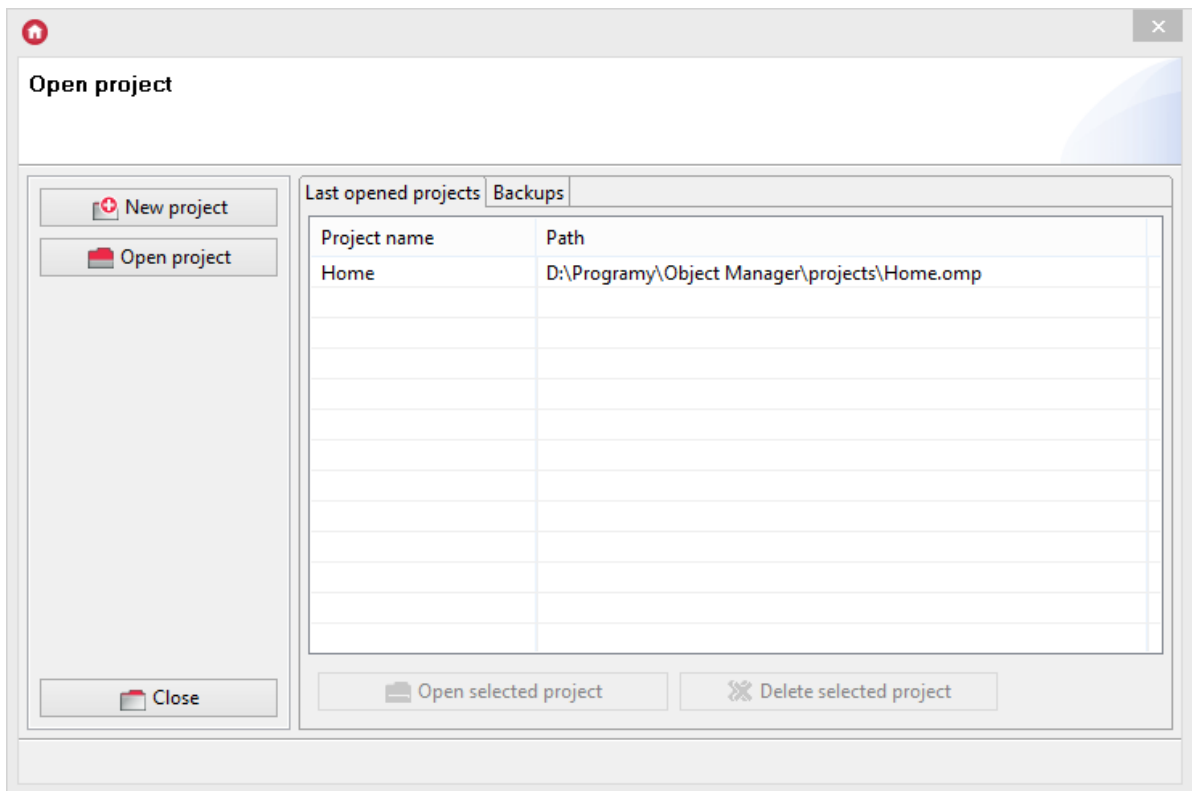
CLU modules, as all network devices, have their own IP address. Each of the modules installed in the system must have its own unique IP address, however, all CLU modules in the system must work in the same sub-network so they can communicate with each other. IP address of a specific CLU can be changed by the user at any time. The address can be changed through device configurator for the selected CLU and by entering the new address into the field containing the old address.

**NOTE!** After connecting CLU (or several CLU) to computer's network card, it will receive a new IP address consistent with pool of addresses in which computer's network card is.

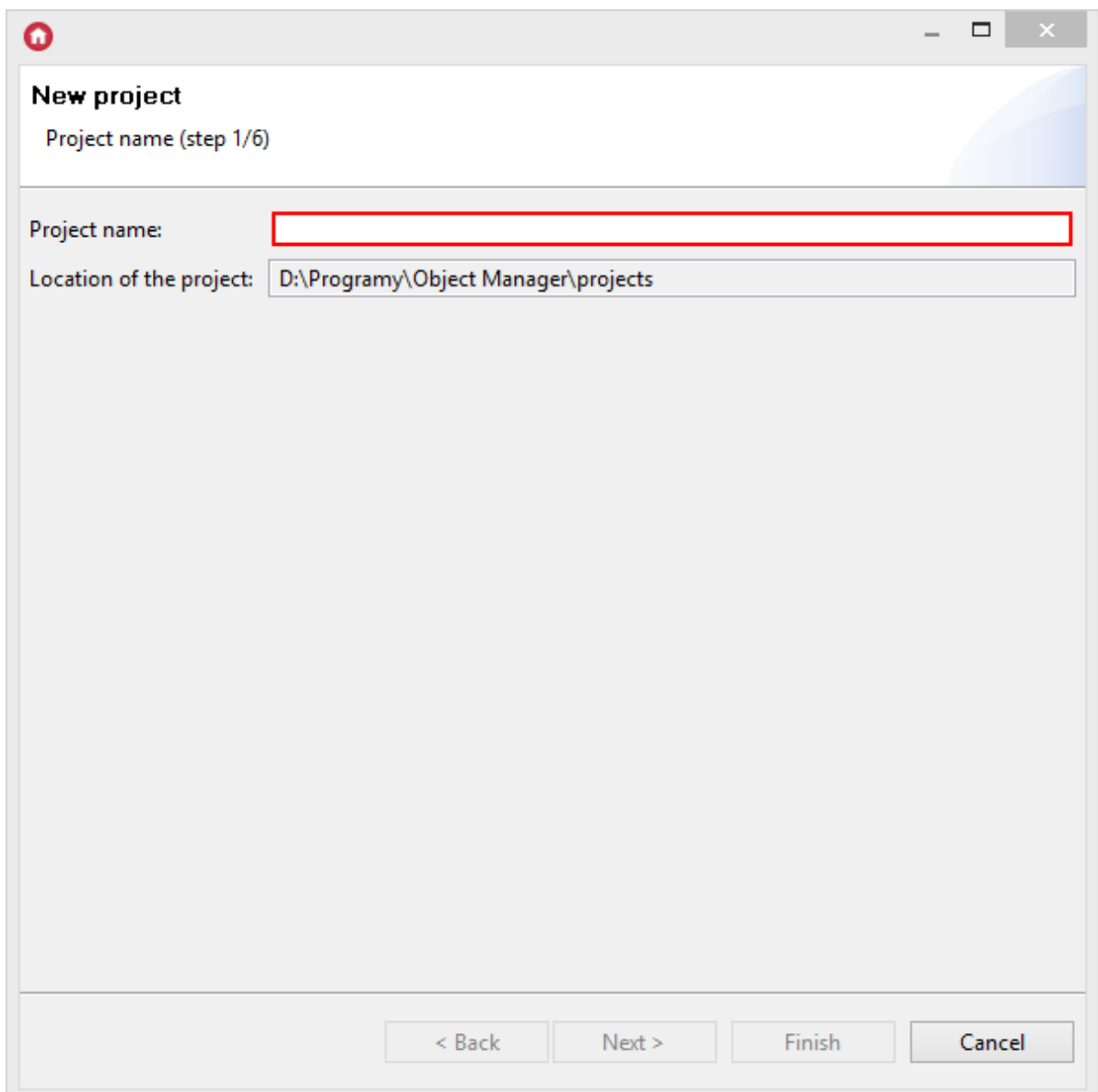
## 3. Opening new project

---

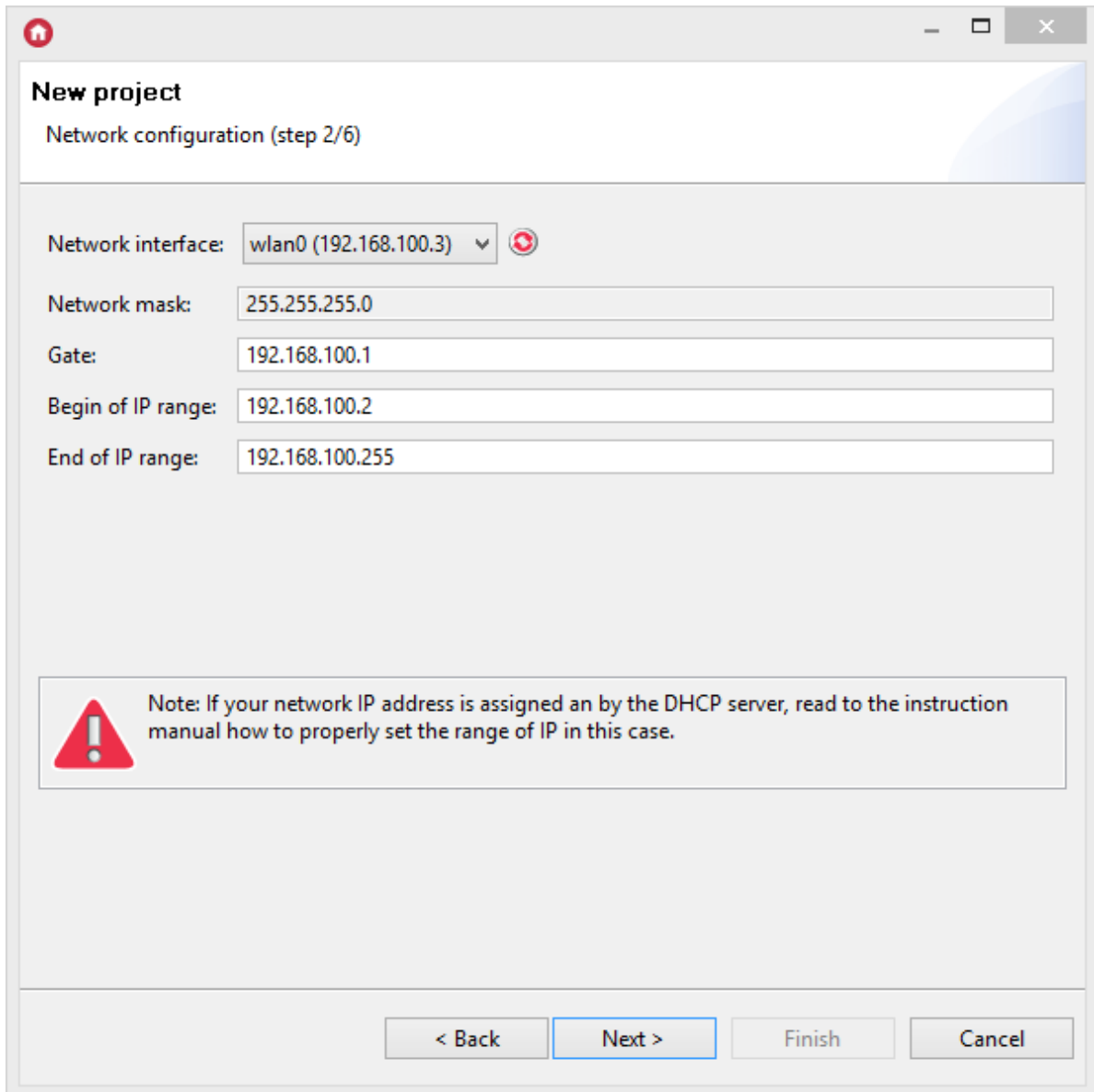
After opening Object Manager, a new window with two options appears: opening a saved project and creating a new project.



1. Select new project creation, then name the created project.



2. Object Manager software will display network configuration window. The settings for the available interface are loaded automatically. Other network interfaces can be selected from the list. It is possible to indicate the range from which the system will automatically assign available IP addresses to the modules found.




The screenshot shows a window titled "New project" with the subtitle "Network configuration (step 2/6)". The window contains several input fields for network settings:


- Network interface:** A dropdown menu showing "wlan0 (192.168.100.3)" with a refresh icon to its right.
- Network mask:** A text input field containing "255.255.255.0".
- Gate:** A text input field containing "192.168.100.1".
- Begin of IP range:** A text input field containing "192.168.100.2".
- End of IP range:** A text input field containing "192.168.100.255".

Below the input fields is a warning box with a red triangle icon containing an exclamation mark. The text inside the box reads: "Note: If your network IP address is assigned an by the DHCP server, read to the instruction manual how to properly set the range of IP in this case."

At the bottom of the window, there are four buttons: "< Back", "Next >" (highlighted with a blue border), "Finish", and "Cancel".

**NOTE!** For Object Manager version below 1.2.1, the settings for the available network interface are not loaded automatically.

 — □ ×

**New Project - Network configuration (step 2/7)** 

Enter network parameters:

Network mask:

Gate:


Enter the range of IP addresses which to be assigned for CLU modules:

Let's system to set IP address on discovered CLU

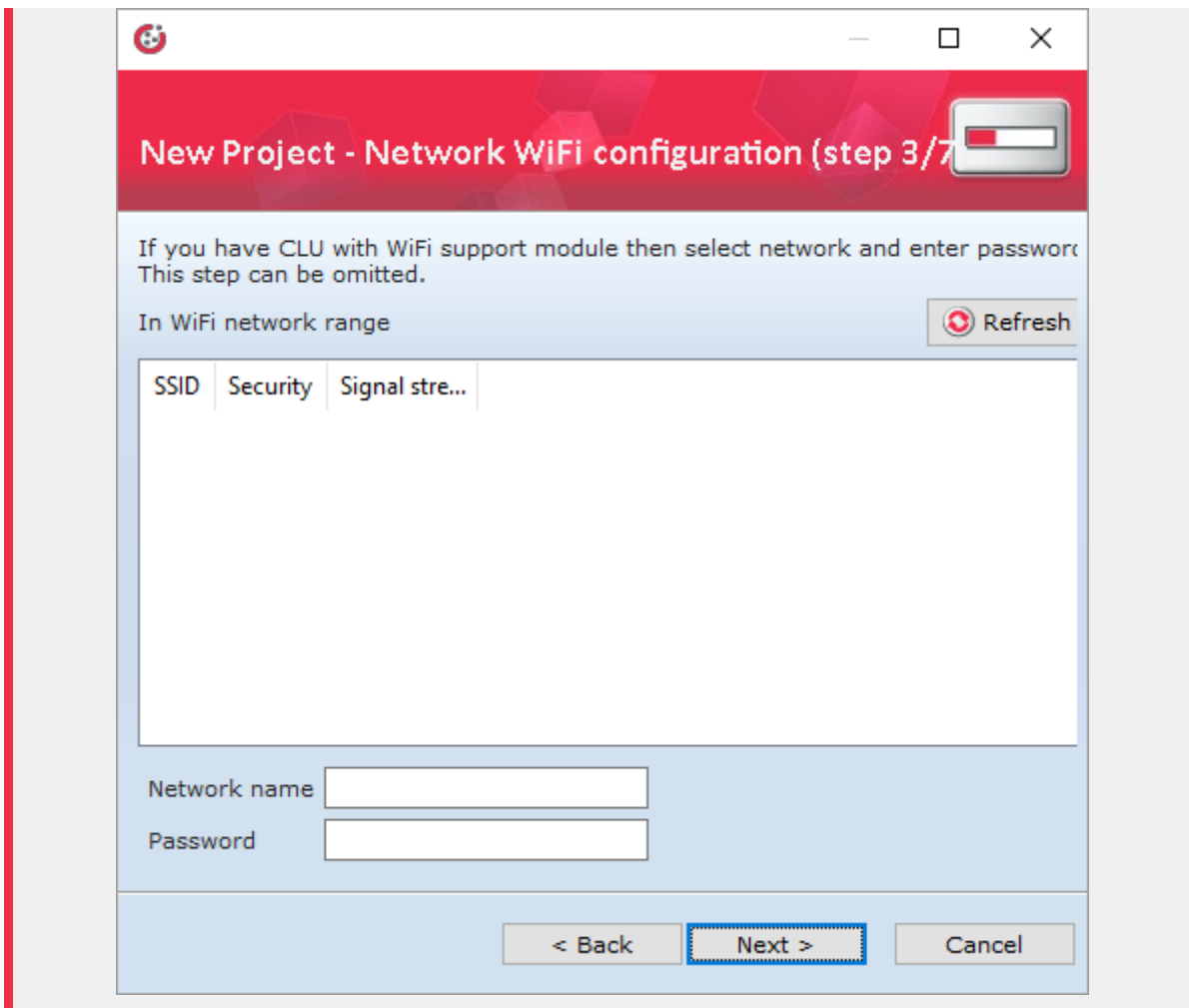
**Set IP address range**

Begin of IP range:

End of IP range:

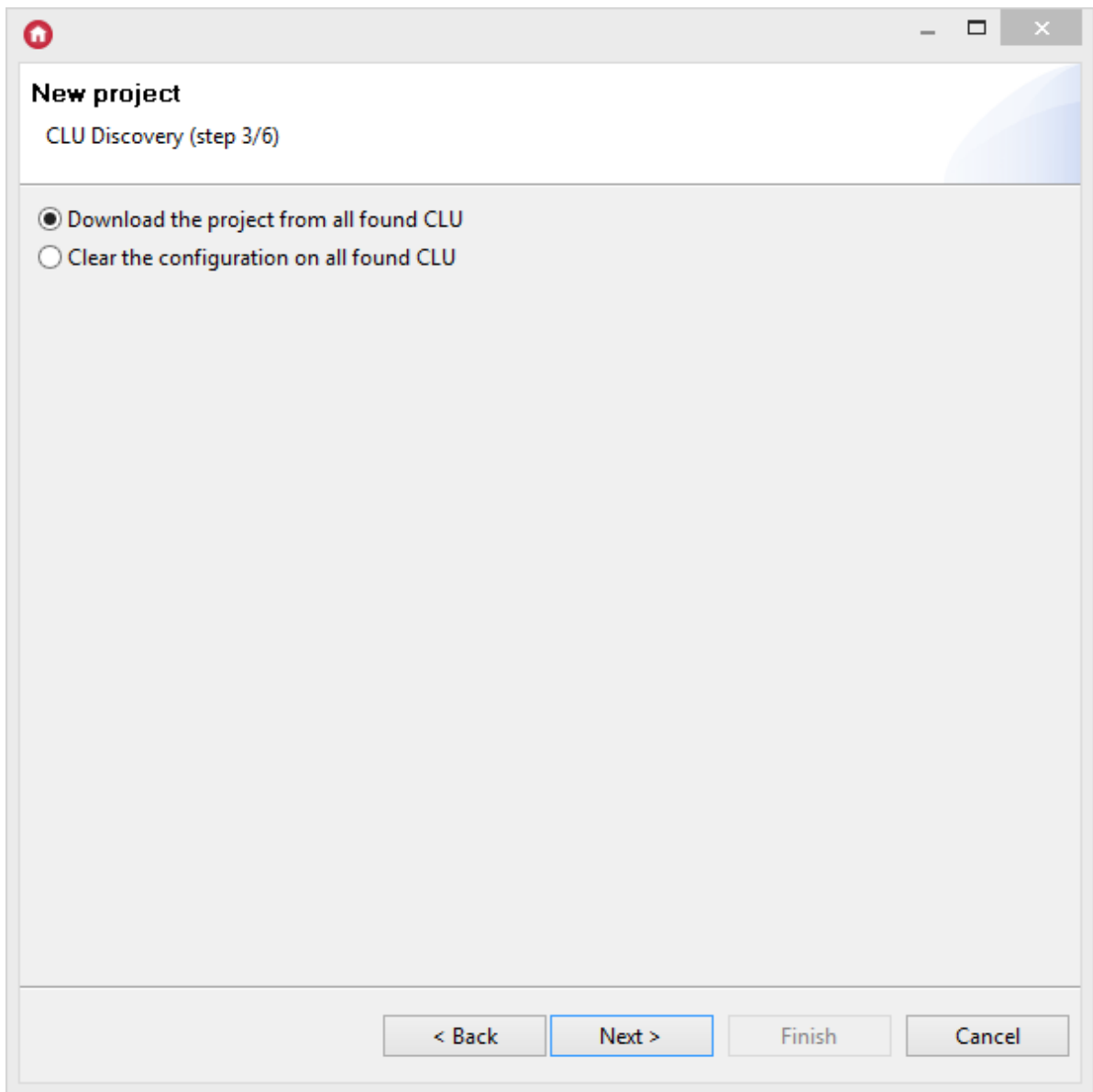
 Note: If your network IP address is assigned an by the DHCP server, read to the instruction manual how to properly set the range of IP in this case.

**NOTE!** In the older version of Object Manager, there is the "Network WiFi configuration" step that should be skipped.

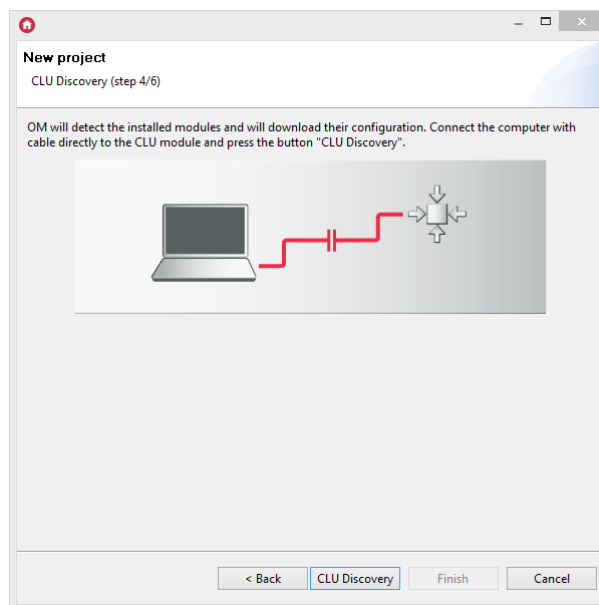


3. In the next step, you may choose between downloading existing system configuration to the newly created project, and complete configuration reset and starting a project from scratch. The first option is useful when necessity of recreating configuration after loss of project file occurs.

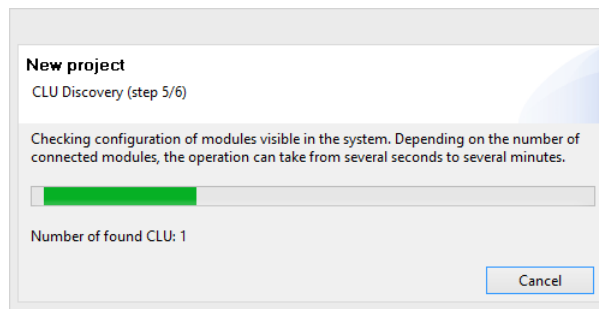




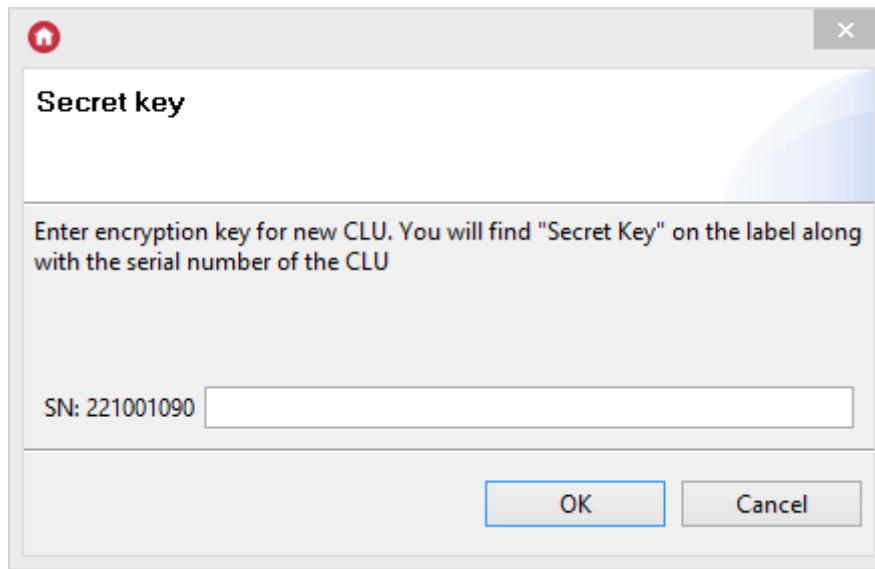
4. In the fourth step, available modules search procedure named CLU DISCOVERY should be launched.



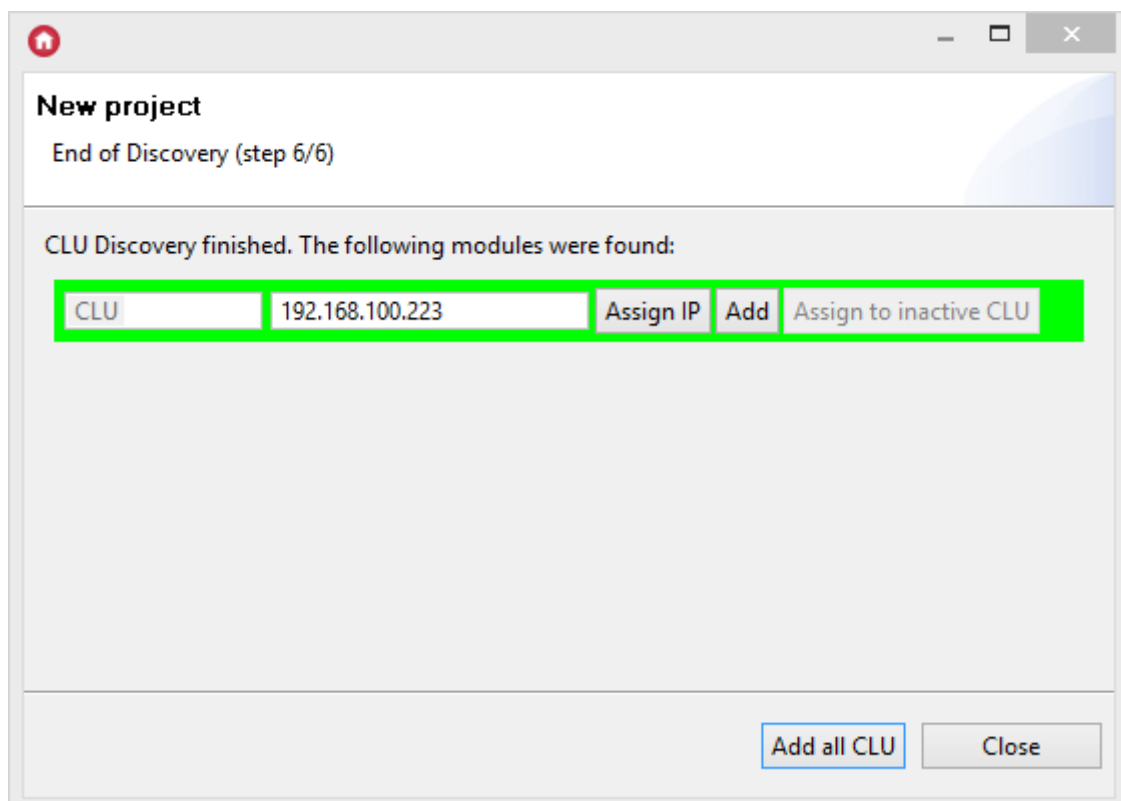
5. In the next step, OM starts searching available CLU modules.



To complete the creation of a new project - after searching for available CLU - in the window displayed, enter the *Secret Key* of the given CLU, which is located on the module's cover.



6. In the last step, OM will list the CLU modules found. In this window you can add all or only selected modules to the created project. You can also change the IP address that has been assigned automatically.



## 4. CLU Discovery function

CLU DISCOVERY function completely automatically finds CLU modules and connected to them IOM modules. It is launched obligatorily during opening a new project, but it can also be launched manually at any time from the actions menu.



Use CLU DISCOVERY function when:

- You connect new CLU or IOM module to the system
- You change CLU or IOM module for a different one
- You switch IOM module from one CLU to another
- There is a need to recover a completely deleted IOM object

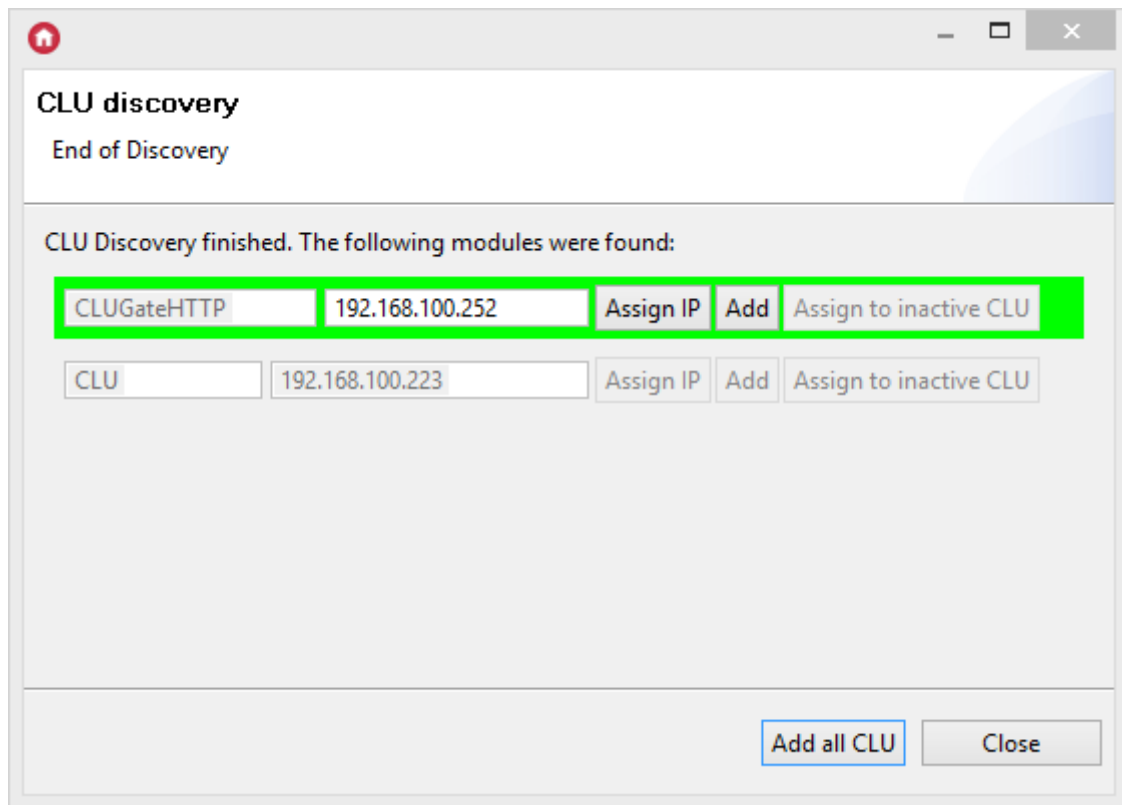
After properly conducted CLU DISCOVERY run, all changes will be found and added to the project.

Before running CLU DISCOVERY, make sure that:

- All modules are properly connected and powered
- CLU modules are connected to each other
- Computer on which OM is running is connected to the same network as CLU.

**NOTE!** If the network consists of router, it is recommended to connect the computer directly to the CLU with a network cable when running CLU DISCOVERY. In the majority of cases, CLU DISCOVERY will run successfully also while connected through the router, however, in the case of a specific router configuration, CLU DISCOVERY might not find CLU modules.

All found modules will be displayed as a List.



Colour of the position means:

- **Green** - newly found CLU, which can be added to the project
- **Red** - CLU, which for various reasons can't be added to the project (version not operated by OM etc.)
- **No colour** - CLU previously added to the project (only if CLU DISCOVERY was used on pre-existing project)








Modules may be added one by one by clicking "Add" button, or all at once by clicking "Add all CLU" button

After doing the above, the indicated CLU will be added to the project.

## 5. CLU status

### 5.1. Module diodes

Based on the LEDs of the CLU module - the user is informed about the current status of power supply, configuration and current device mode.

Status	Description
	No power supply
	Green LED flashes every 500ms - system OK
	Configuration error, system not configured or no communication with the IOM module
	The green LED flashes every 200ms - CLU in the mode of adding Z-Wave modules
	The red diode flashes every 200ms - CLU in the mode of removing Z-Wave modules
	The green LED is on for 1 second, then both LEDs blink three times (every 200ms) - confirmation of adding the Z-Wave module
	Both LEDs blink three times (every 200ms), then the red one goes out and the green one blinks every 500ms - confirmation of removing the Z-Wave module


## 5.2. CLU module icon in OM

Through the appearance of CLU module icon in the objects menu of the opened project, the user is informed about the current status of both configuration and connection between OM and CLU. For each CLU in the project, there are four work modes: normal, disconnected, configuration error, and emergency mode.

### Normal mode

CLU in the normal mode does not contain configuration errors, and the connection between OM and CLU is active. Name of the module is displayed in black, and the icon marking this status looks like this:



If the name of a specific CLU is preceded by  symbol, it means that there was a change in configuration which has not been sent to this CLU yet.

### Disconnected

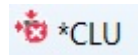
If there is no connection between CLU module and OM (no physical connection or error in LAN configuration), the name of CLU will be displayed in red, and the icon marking this status will look like this:



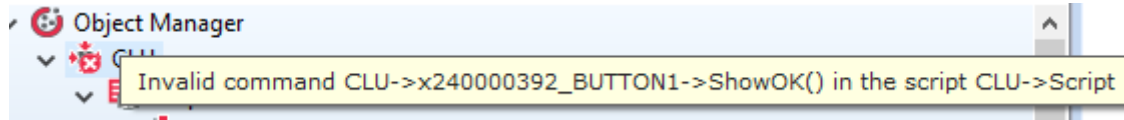
If the CLU is in disconnected mode, the user has an option of making and saving changes in the project, but the new configuration won't be sent to CLU - that is only possible in normal mode.

### Configuration error

If during work on the project there are changes made which contain configuration errors (e.g. creation of connection with non-existent object, or entering non-existent command), CLU in which the error was found will be switched to `Configuration error` work mode. Name of that CLU will remain black, but there will be error symbol displayed next to its icon:



After dragging cursor over the CLU, a field with list of errors will appear.



**NOTE!** OM does not allow sending configuration containing errors to CLU

### Emergency mode

If configuration containing syntax errors is sent to CLU (e.g. after sending script in the text editor), or if LUA interpreter crashed as a result of script's work, CLU will switch to EMERGENCY MODE. The name of the CLU will change its colour to orange, and the failure symbol will appear next to its icon:



If CLU switched to emergency mode, check accuracy of recently made changes and send configuration to CLU again.

**NOTE!** The CLUs taken out from the box (in the delivered condition) are in Emergency mode!

## 6. Connecting Z-Wave modules

Wireless IOM modules communicate with other system elements using Z-Wave protocol. They work and are recognisable (both from OM level and from control level) the same as other modules in the GRENTON system.

To enable using Z-Wave modules in the system, that system must contain at least one module CLU equipped with Z-Wave controller.

**NOTE!** Adding the Z-Wave module to the system should take place after placing it in the installation's destination - this is due to the requirements for creating the mesh network, the range of the device operation and disturbances of the Z-Wave network.

### 6.1. Adding Z-Wave modules

You have to add IOM Z-Wave modules to CLU for them to be present in the system. You can do it in two ways:

1. **By clicking LINK button on CLU module** In order to do that, press `Link` button placed on the CLU module with Z-WAVE controller.

After pressing the button, the CLU switches to the mode of adding modules - the ON diode blinks all the time at intervals of 200ms.

Then, press the button once on the added Z-Wave module (according to the instruction manual). The correct addition of the module will be signaled by lighting the ON diode for 1 second, and then by blinking the ON and ERR LEDs three times in intervals of 200ms. After completing the addition of the Z-Wave module, the ON LED will flash at 500ms. After completing the addition of Z-Wave modules, *CLU Discovery* should be performed - new Z-Wave modules will be added to the project.

2. **Using Object Manager software** This way of adding allows to define time for which CLU will await for wireless modules to “introduce” themselves, therefore it is very useful when you want to add modules located further away from the CLU and need more time to press the button on them.

To add wireless modules using OM, open object configurator of Z-Wave CLU module to which you will add wireless modules (double-click CLU icon on the objects list). Then, set time (as parameter) for `StartZwaveDiscovery` method in the `Control` tab and invoke this method.

Method	Parameter name	Value	Call
AddToLog	Log	<input type="text"/> string	
ClearLog			
SetDateTime	LocalTimestamp	09:04:24 06-07-2020	
StartZWaveDiscovery	Time	<input type="text"/> number	
StopZWaveDiscovery			

Set time will be the time for which CLU awaits for new Z-Wave modules to connect. When the time is up, the search is finished, even if no modules were found. Entering 0 will cause the search to end automatically after finding one new module.

After calling the `StartZWaveDiscovery` method, press the button located on the added Z-Wave module. The correct addition of the module will be signaled by lighting the ON diode for 1 second, and then by blinking the ON and ERR LEDs three times in intervals of 200ms. After correctly adding the Z-Wave modules, the ON LED will flash at 500ms. After completing the addition of Z-Wave modules, the *CLU Discovery* process should be performed - new Z-Wave modules will be added to the project.

**NOTE!** Calling the `StopZWaveDiscovery` method interrupts the search for Z-Wave modules.

**NOTE!** Do not add modules to the system that have already been connected to it. If you are not sure whether a module has been added before, you should first perform the removal procedure for this module.

The situation is similar when the Z-Wave module was connected and was not removed from another controller - the procedure of removing the module should be performed first.

## 6.2. Removal of Z-Wave modules

For the wireless module to stop appearing in the system configuration, it must be removed from it.

To do this, it is necessary to press the `Unlink` button on the CLU with the controller.

After pressing it, the CLU goes into the module removal mode - the ERR diode blinks all the time at 200ms intervals.

Then press the button on the wireless module to be removed. Correct removal of the module will be signaled by blinking ON and ERR LEDs three times in 200ms intervals. After completing the deletion of the Z-Wave module, the ERR LED will turn off and the ON will flash at 500ms. The last step will be *CLU Discovery* - the removed modules will be grayed out.

## 6.3. No communication with the Z-Wave module - a mechanism for counting communication failures and blocking device communication in the Z-Wave network

**NOTE!** The presented mechanism is available for CLU from version 04.07.41 (183201)

Failures in communication with a Z-Wave device may occur when:

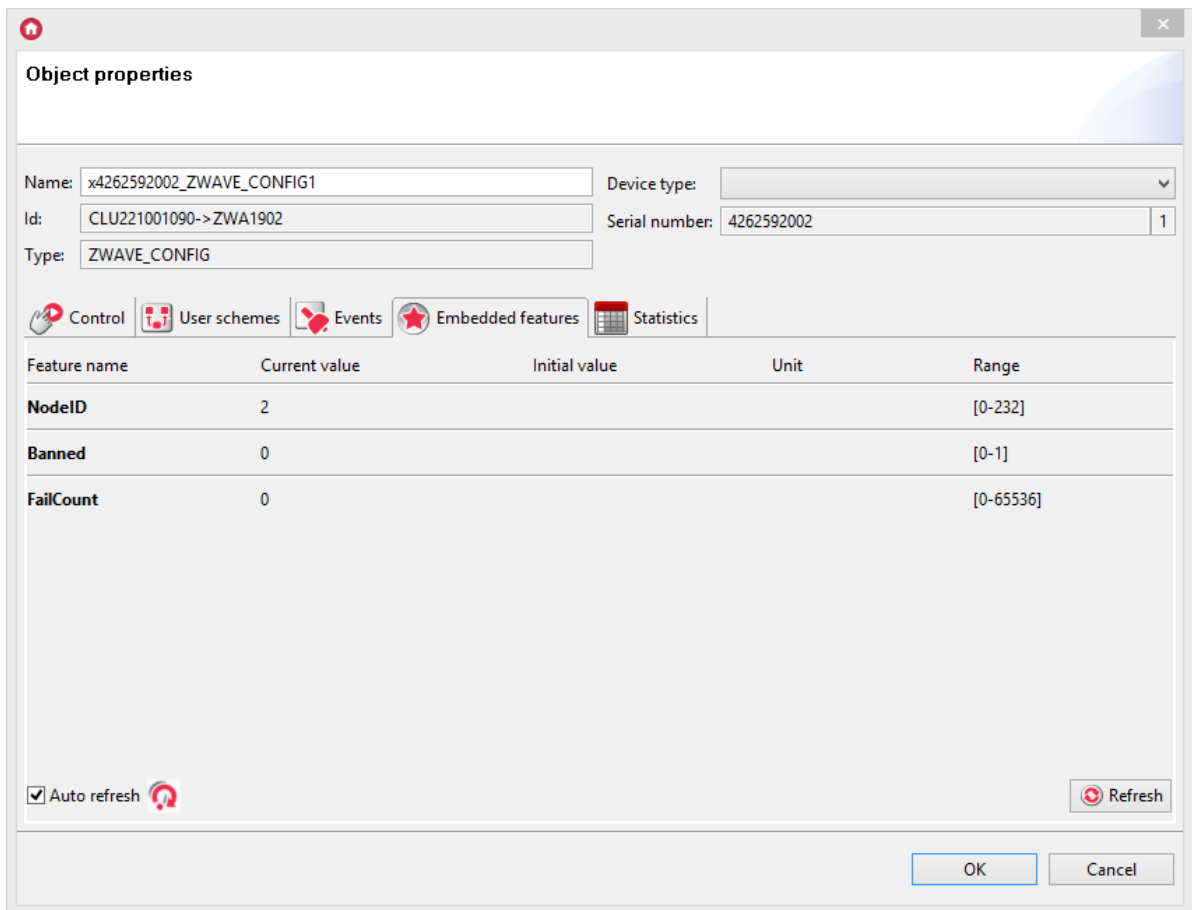
- the Z-Wave module is damaged,
- no power supply (230V) on the module / depletion of the battery supplying the module,
- the device works on the border of the range with the controller / it is not within the range of the controller,
- the controller (CLU) after sending the order will not receive confirmation from the device (ACK).

Information about the device status in the Z-Wave network can be read from the Object Manager using the ZWAVE object of the given Z-Wave module.

**NOTE!** ZWAVE\_CONFIG objects are not available for all Z-Wave modules - they have Grenton Z-Wave modules and selected modules that are supported by the Grenton system.

The following features are available for a given object:

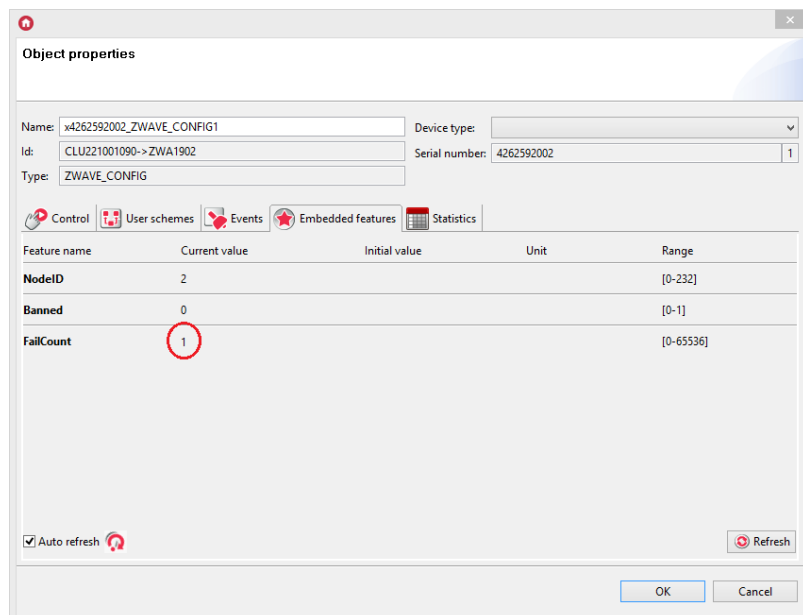




- `NodeID` - Number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
- `Banned` - Information on blocking Z-Wave communication with the module
- `FailCount` - The number of unsuccessful attempts to communicate with the Z-Wave module

#### Failure counting mechanism in communication:

- In the event of failure of communication with the module (no response, confirmation, etc.), the `FailCount` feature of the ZWAVE\_CONFIG object of the Z-Wave device is incremented.



- Another attempt to send an order to the retry device is every 15 seconds - 3 attempts are made to communicate with the device.

- In the case of 3 attempts to communicate with the module, the `Banned` feature is set to 1 and all communication with the module is blocked.

**Object properties**

Name:  Device type:

Id:  Serial number:

Type:

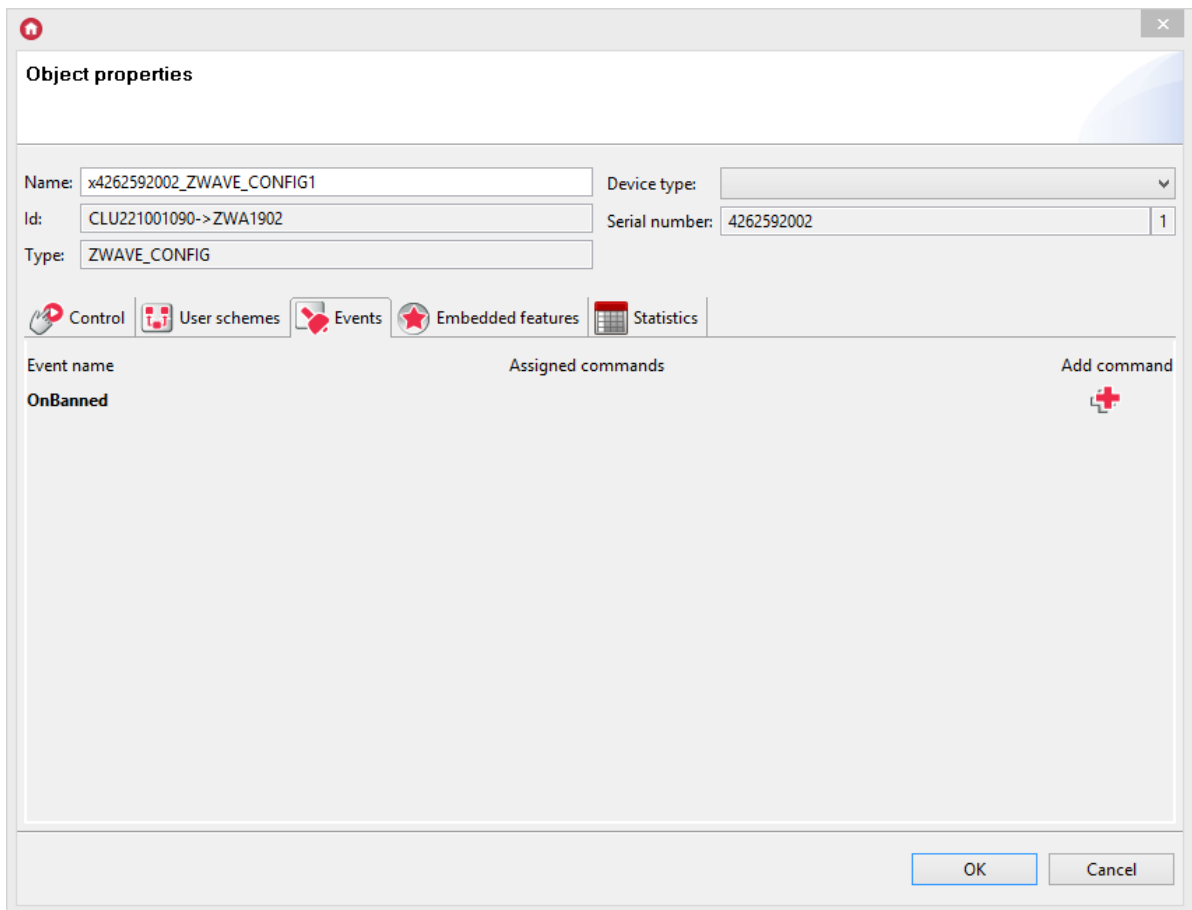
Control | User schemes | Events | **Embedded features** | Statistics

Feature name	Current value	Initial value	Unit	Range
<b>NodeID</b>	2			[0-232]
<b>Banned</b>	1			[0-1]
<b>FailCount</b>	4			[0-65536]

Auto refresh

### Locking mechanism for communication with the module

- When the `Banned` feature is set to 1, communication with the Z-Wave device is blocked - this means that all action calls on the device (ie change of output status, query for parameters) are not sent by the CLU to the blocked module.
- You can assign any action when you block communication with a given module using the `OnBanned` event.



- A short query (NOP) is sent to the banned module every 1.5 minutes:
  - if the module does not confirm receipt of the query, the Banned attribute continues to be 1, and the next query is repeated every 1.5 minutes,

**NOTE!** If more than one module is banned, then the NOP is sent every 1.5 minutes to **the next banned module**. Example:

3 modules (A, B, C) banned

CLU - NOP -> module A

1.5 minutes break

CLU - NOP -> module B

1.5 minutes break

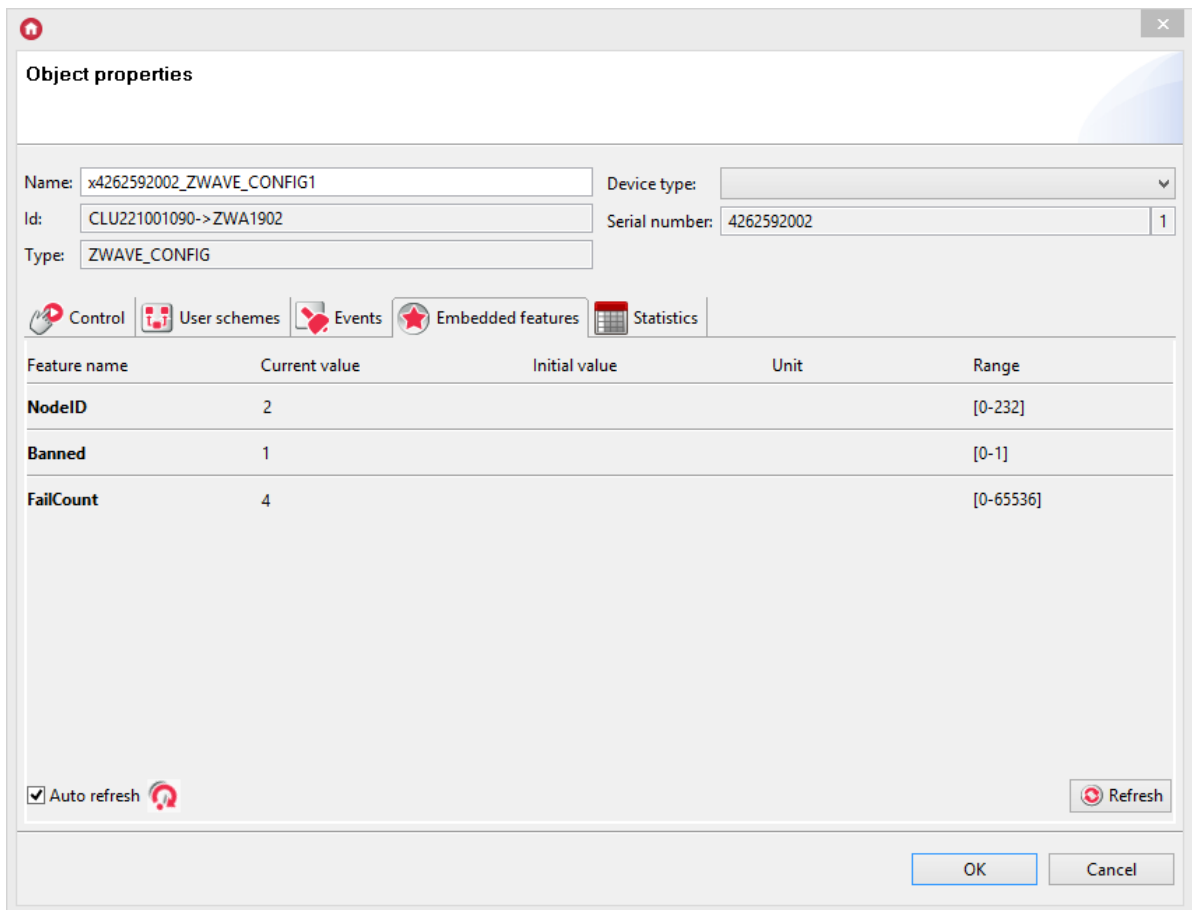
CLU - NOP -> module C

1.5 minutes break

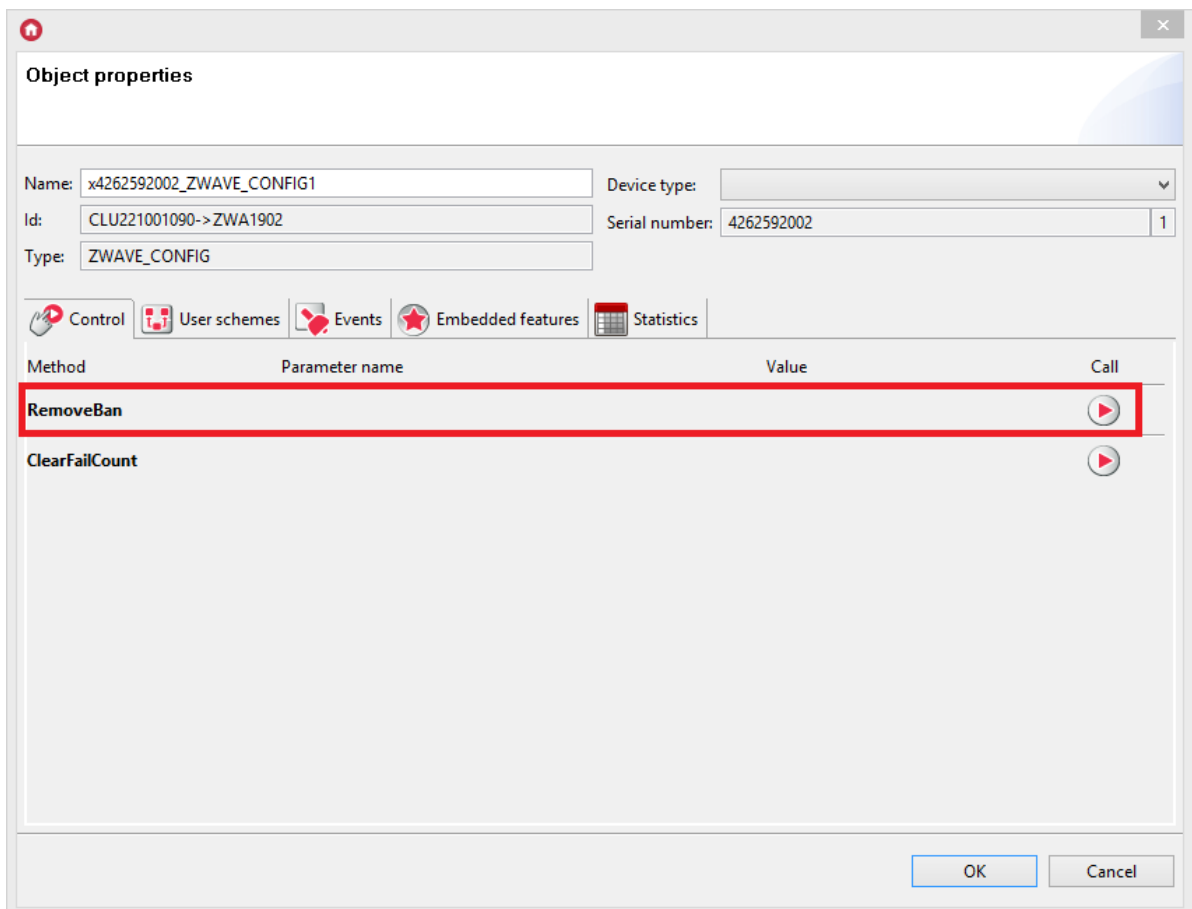
CLU - NOP -> module A

e.t.c.

- if the module confirms receipt of an inquiry (ACK), the Banned attribute changes to 0 - it means that it is possible to send commands again to a given device.



- It is possible to manually remove the lock - using the **RemoveBan** method.



- After calling this method, the `Banned` property changes value to 0 - it means that it is possible to send commands again to a given device.



The configuration is stored in the OM and until it is sent to the CLU, it is not taken into account in the operation of the system. To send the configuration to the CLU, press the 'Send configuration' button in the menu.

Object Manager detects on which CLU the change was made and sends the configuration.

**NOTE!** After sending the configuration, the CLUs will be restarted, so the lamps connected to the system may go out, and the system may not react for a few seconds to press the switches, etc.

## 8. Initial values of features

Each object in the system has its own list of features, some of which can be set. Features can be set during system startup (CLU restart), thanks to which it is possible to configure the behavior of objects once (eg setting the touch panel buttons as bistable, monostable). Initial values of features are set in the tab: *Embedded features* in the object's form (CLU, inputs, outputs):

Feature name	Current value	Initial value	Unit	Range
Mode	0	Monostable		0,1,2
HoldDelay	1000	1000	ms	[100-5000]
HoldInterval	100	100	ms	[100-2000]
Value	0		bool	0,1
Label	-		string	[0-15]
IconA	-		string	[0-9]
IconB	-		string	[0-9]

To set the selected feature, in the appropriate field, enter the desired value in the  column, and then send the configuration to the CLU.

## 9. Creating basic connections

Calling reactions in the system (eg switching on the lighting after pressing the key) is accomplished by creating links between objects. As a rule, these are connections between the entrance (eg switch) and the output (lamp). However, the system does not limit the creation of connections and allows them to create events between events of any other objects between events, which makes it possible, for

example, to switch on the LED lighting when the main lamp is turned off.

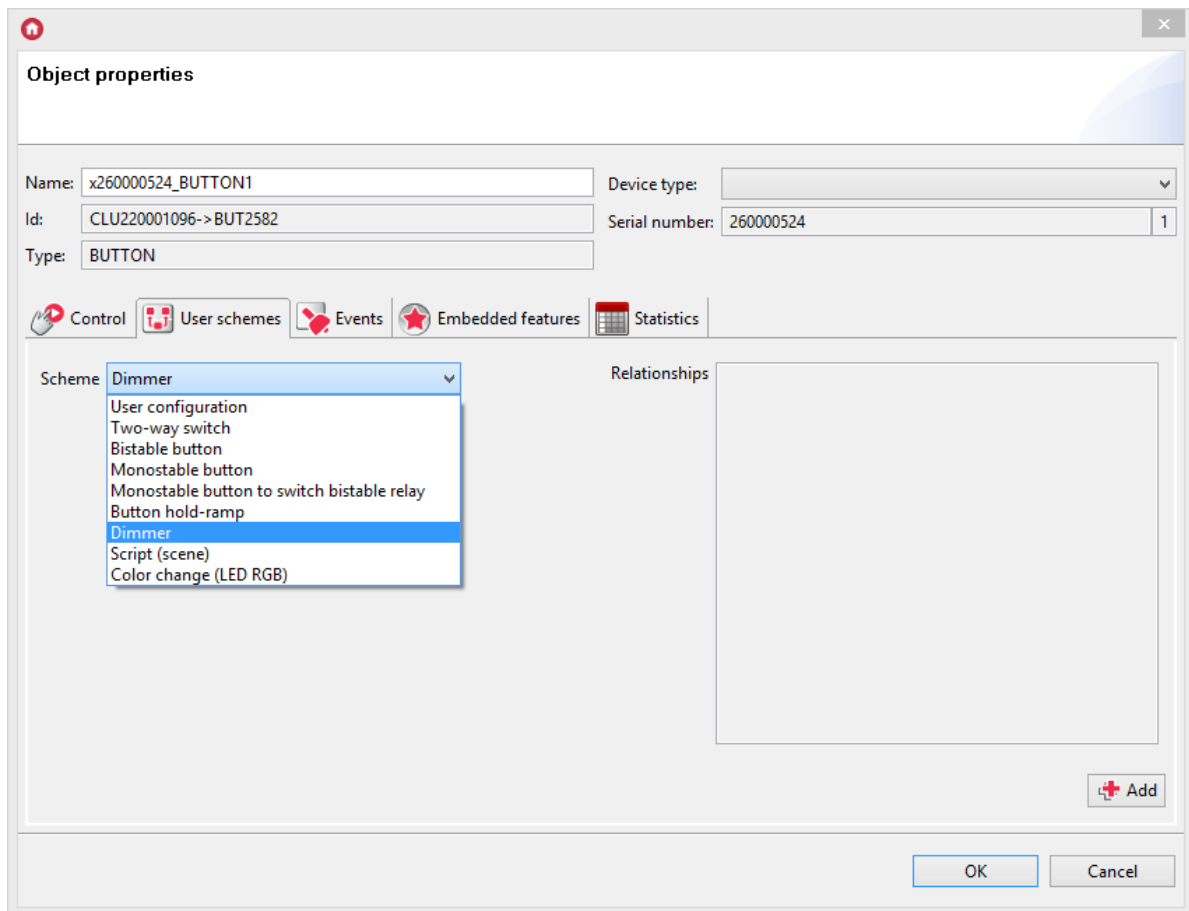
Associations can be created in two ways:

- By using configuration diagrams - it allows quick creation of typical switch-lamp connections;
- By manually creating event-method bindings - which will provide great flexibility in creating system logic.

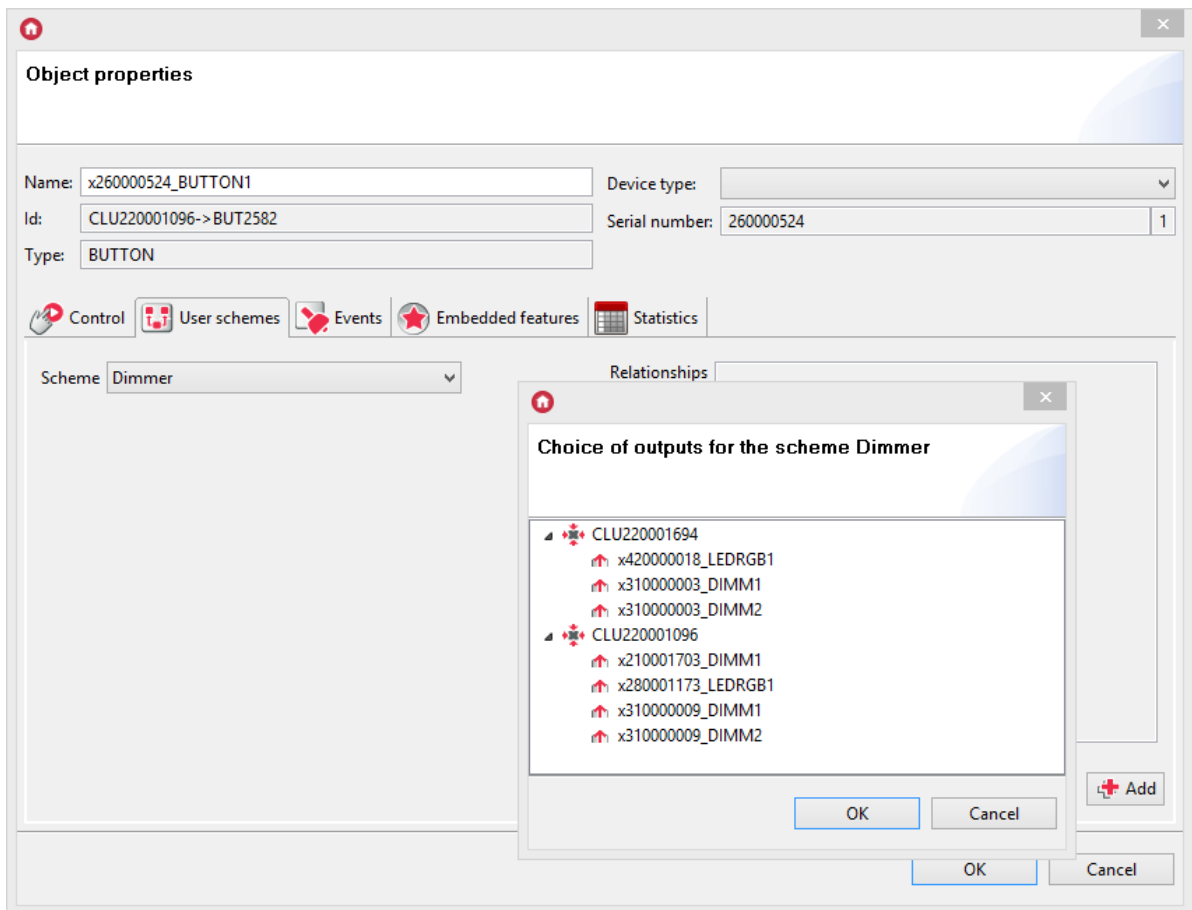
### Creating basic connections by using configuration diagrams

To create a binding using the configuration scheme, do the following:

- Click on the selected input;
- Go to the *User schemes* tab, select an interesting scheme from the list;



- By clicking  , select the outputs to be triggered;



Only outputs for which it is possible to assign a given logic will appear in the output selection.

To select more than one output, select objects by holding the `Ctrl` or `Shift` key on the keyboard. After confirming the selected outputs, Object Manager will automatically create event associations with object methods.

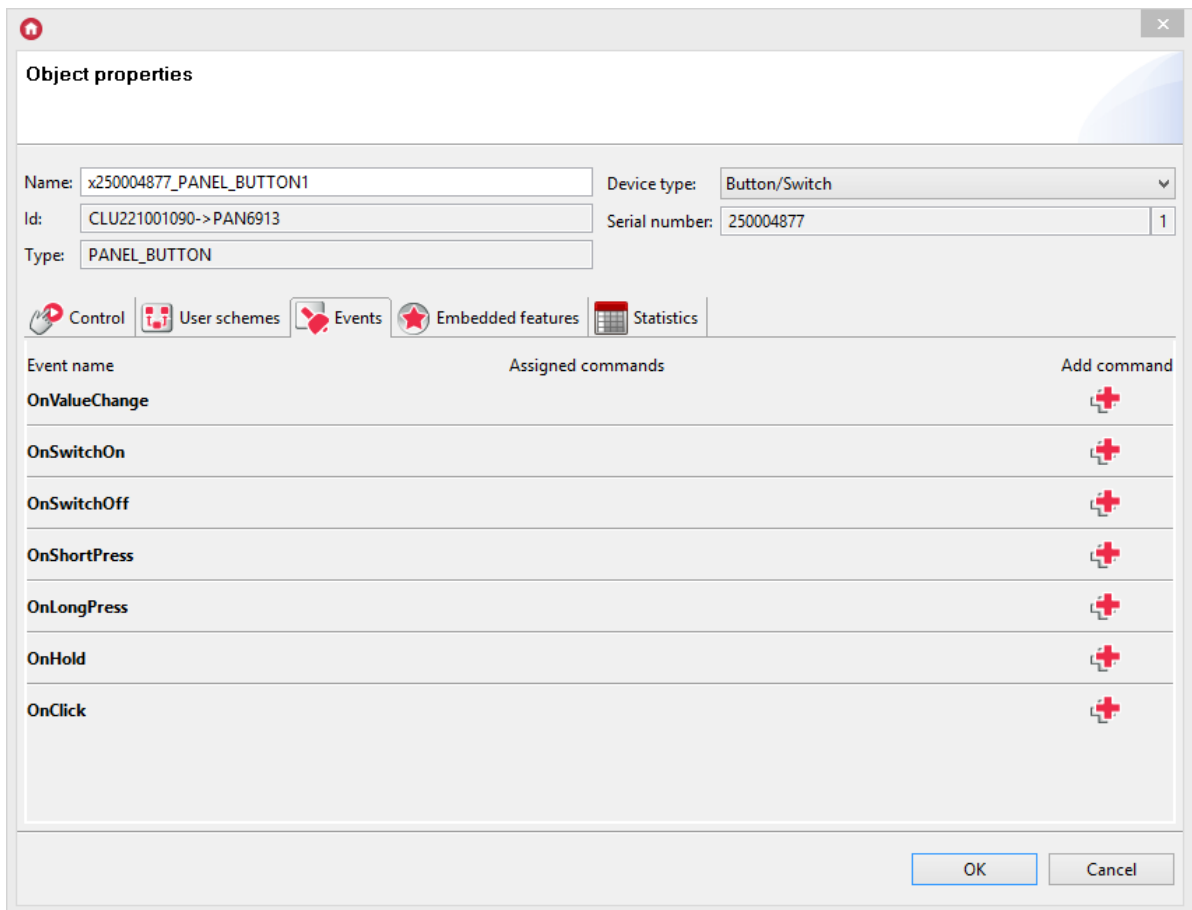
- Configure the remaining inputs and send the configuration to the CLU.

### Manually creating event-method bindings

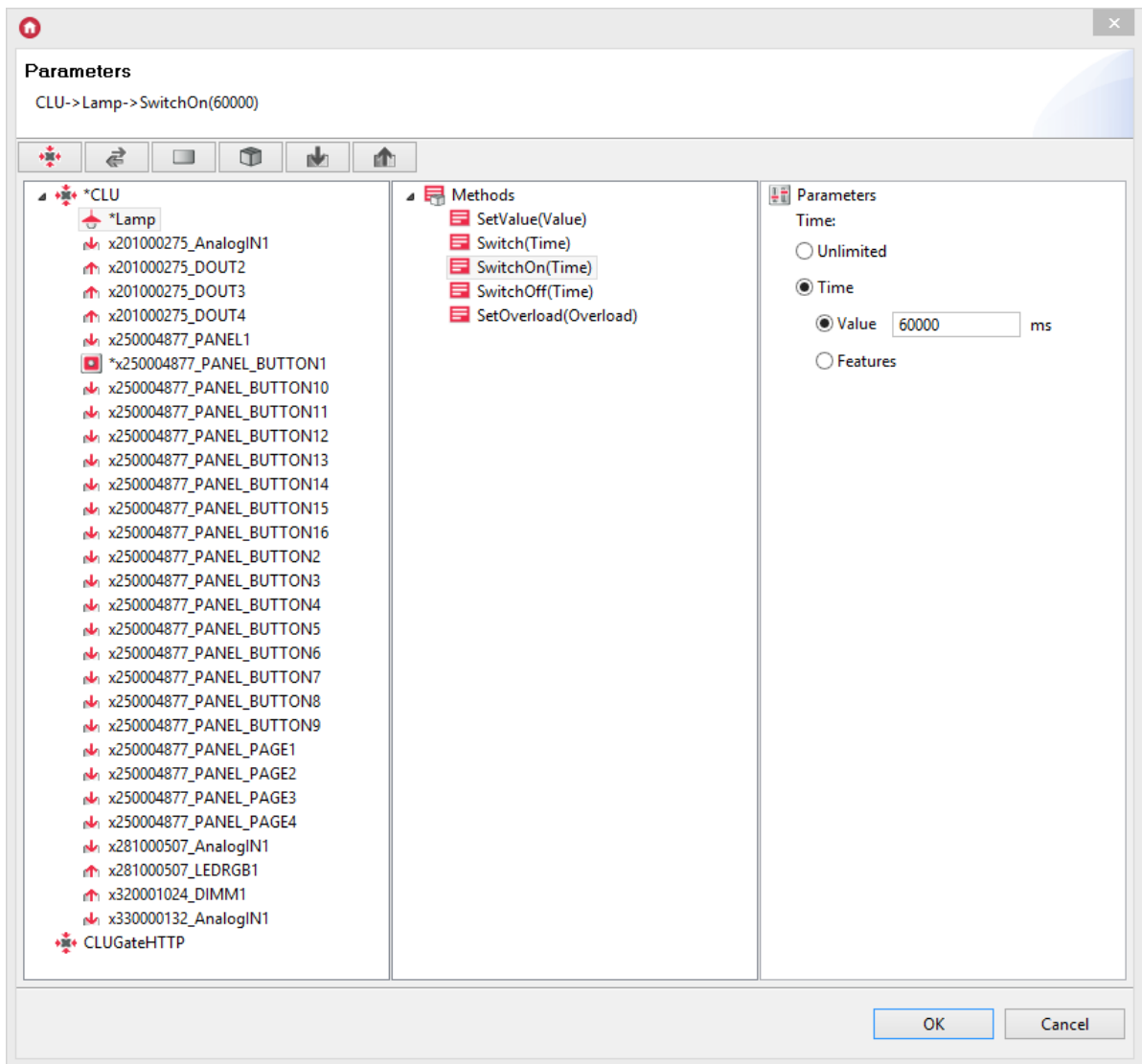
To manually create an event-method binding:

- From the list of objects in the system, select the object you are interested in, double-click it;
- Go to the `Events` tab:



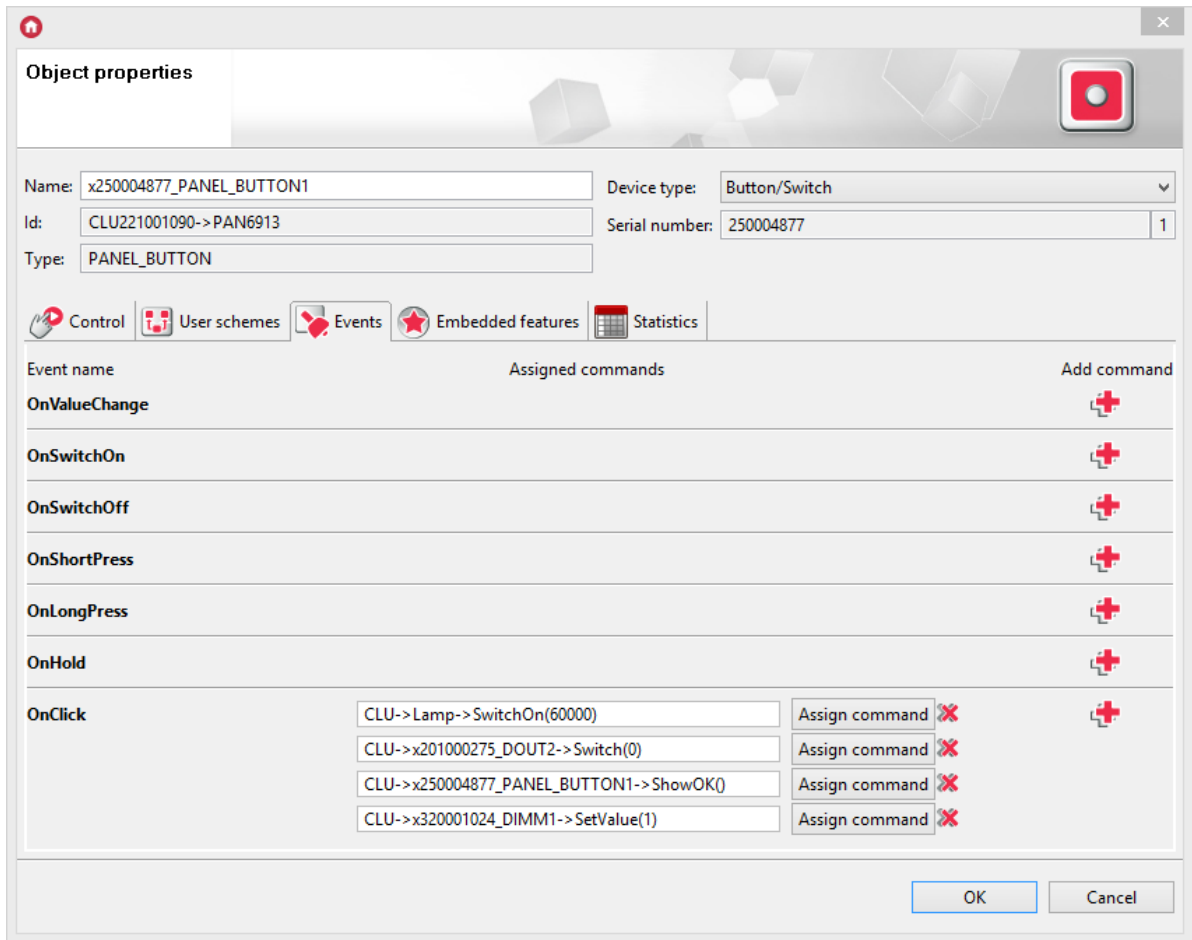


- Find the event to be linked from the list and click ;
- In the method selection format, select the object, method and parameters in sequence:

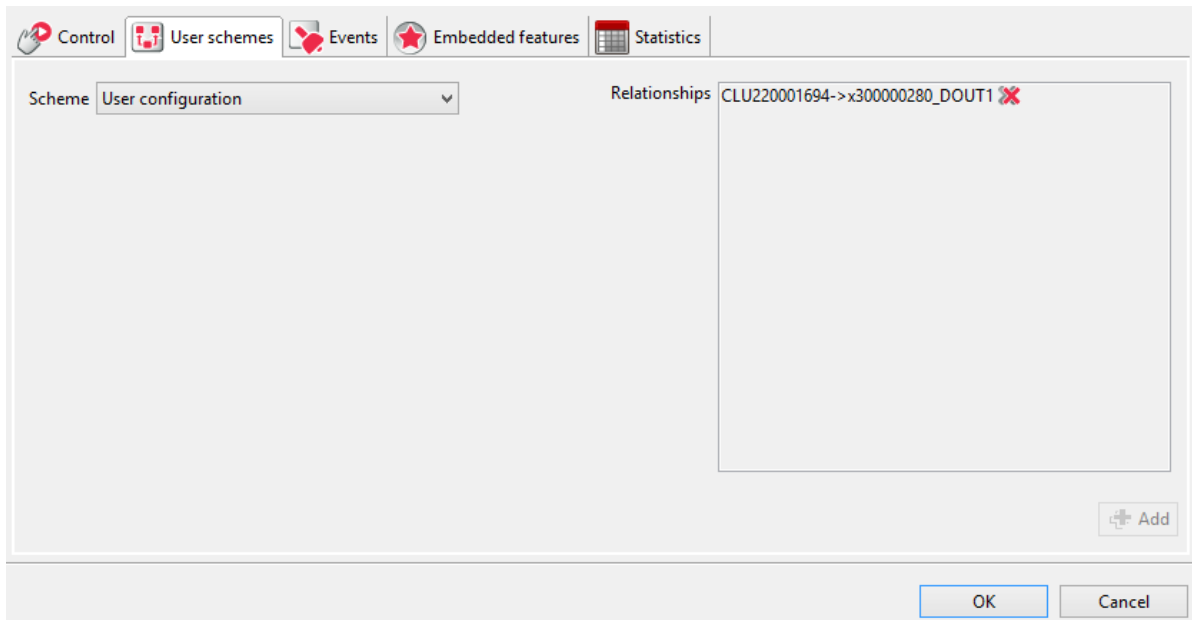


- Configure the remaining events and send the configuration to the CLU.

Up to 4 exit methods can be added to each event. If it is necessary to add more methods or conditions, it is suggested to create a script.



If the user created individual event-method connections using **Events** tab, they are visible on the list as **User configuration**.



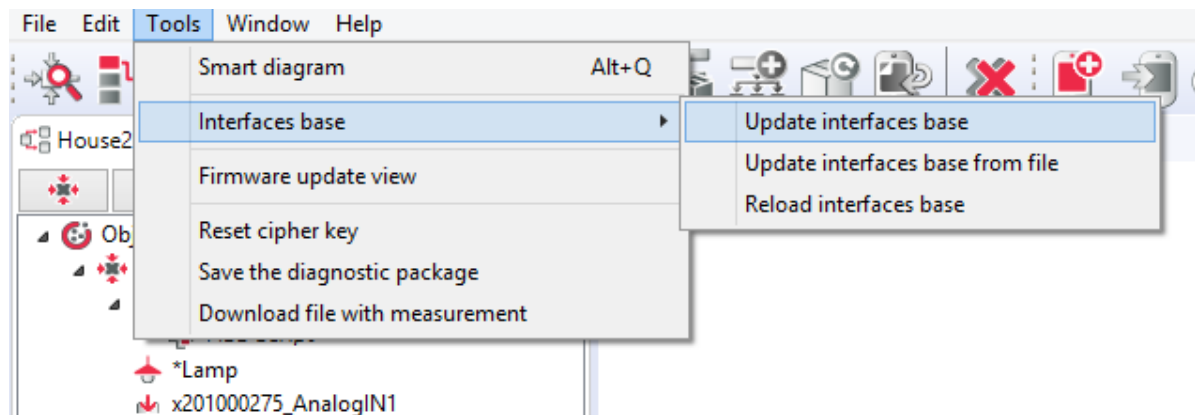
## 10. Performing an update

## 10.1. The process of updating the interface database

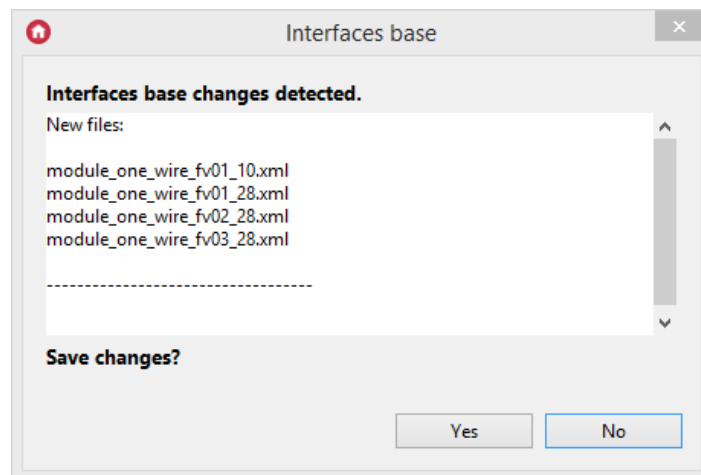
If the option *to automatically update the interfaces database* is marked when the Object Manager is started for the first time, there is no need to run it again. Otherwise, remember to update regularly. Updating the interfaces database should be done always before updating the software of a given Grenton module, and it is necessary to connect to the internet to perform it (the update takes place from the server).

In order to update the interface database in the Object Manager:

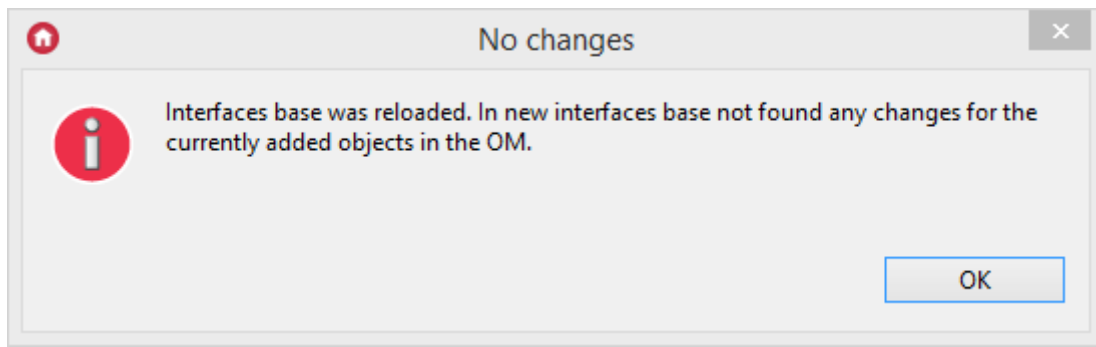
- Select **Tools** from the menu bar.
- Select the item *Interfaces base*.
- Select *Update interfaces database* from the list displayed:



- After a while a window will appear with detected changes in the interface database, which should be accepted by clicking the *OK* button:

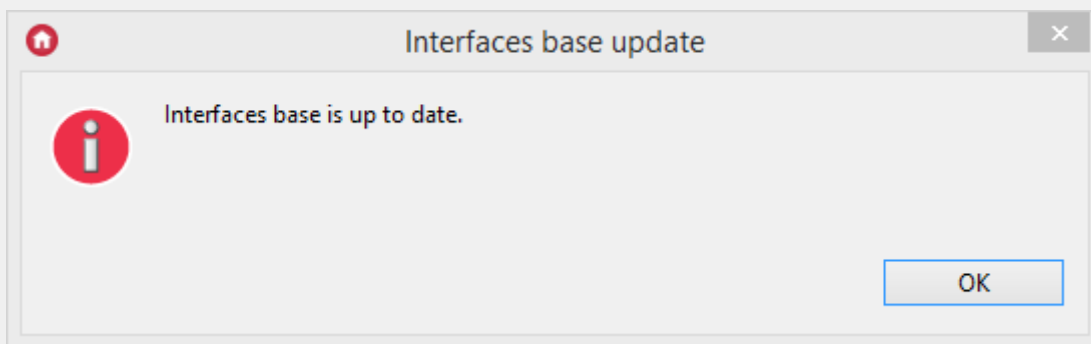


- Then a window will be displayed informing you that the interfaces base has been reloaded:



- The final stage is sending the configuration to the central logic unit, which follows automatically.

**NOTE!** If the configuration is up to date, then after choosing the option: *Update interface base*, the following message will be displayed:



## 10.2. The process of updating the firmware on the CLU

The firmware update on the CLU is carried out in order to: add support for new devices and increase the capabilities of the system. More details can be found in the Release Notes.

**NOTE!** Firmware update CLU 2.0 is only possible in Object Manager version 1.3.0.1927 or higher!

**NOTE!**

Device status display is available in Object Manager version 1.3.5.240201 or higher!

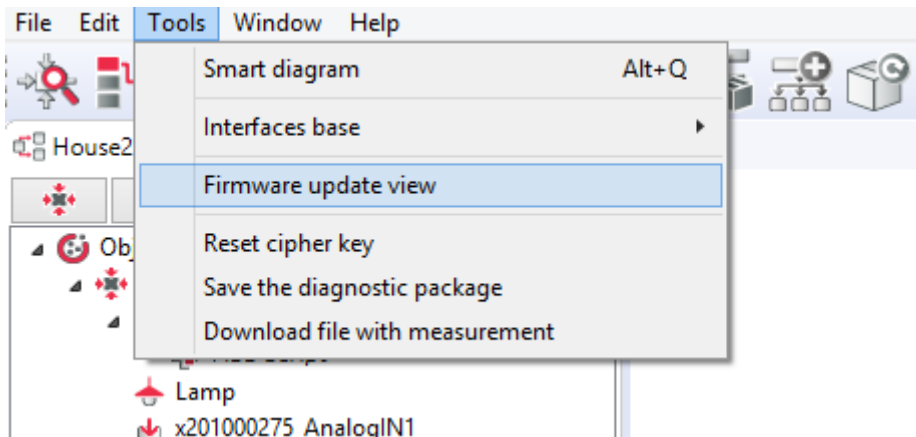
**NOTE!** The following requirements must be met for the update process to run correctly:

- The computer must be connected to the AC adapter, it cannot be on battery power.
- The network connection between the CLU<->router<->computer must be wired, the WIFI connection cannot be used.
- Do not perform any actions on the Grenton system while updating the firmware.

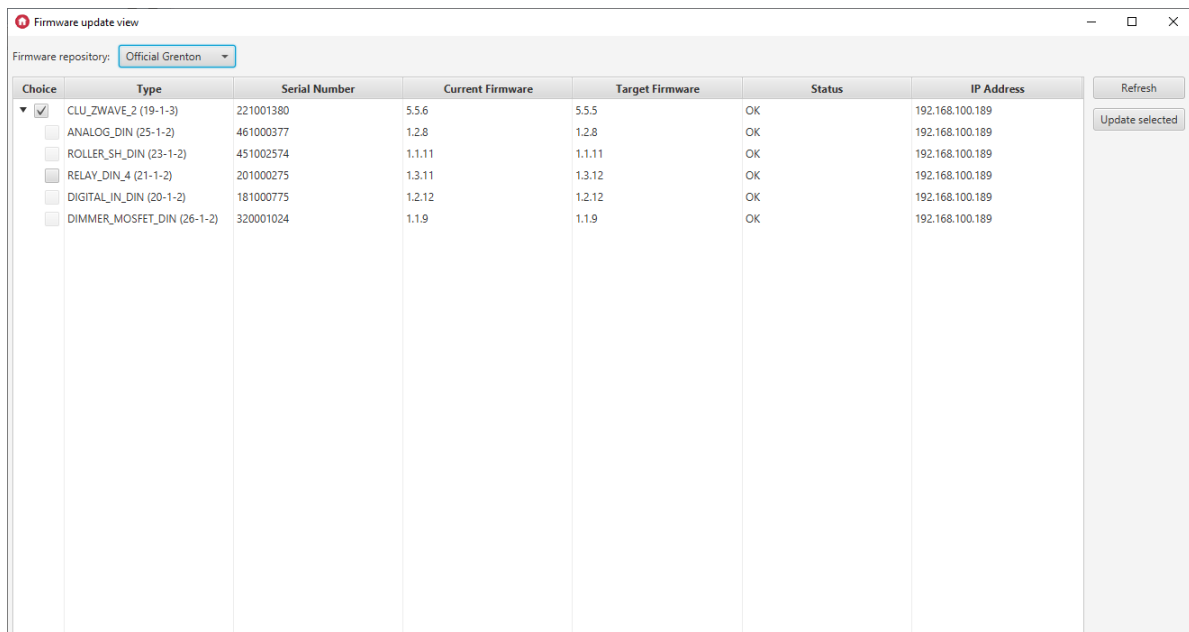
## A. Update from Grenton server

In order to to update the firmware on the CLU you should:

- Select `Tools` from the menu bar.
- Select item `Firmware update view`:

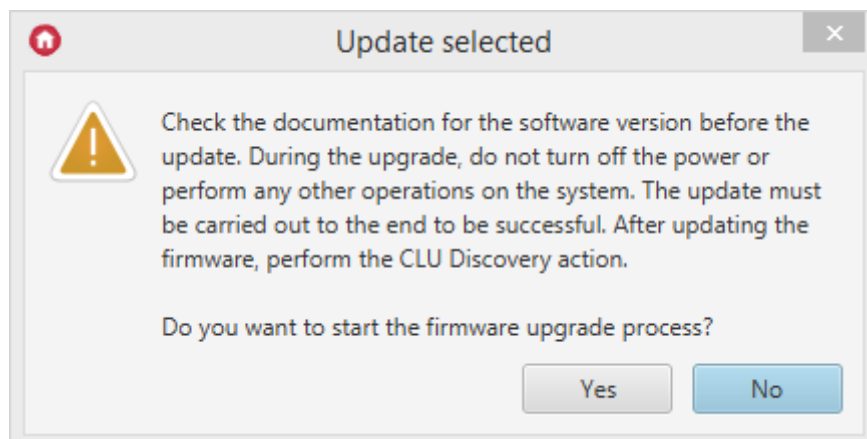


- Select the object type `CLU_ZWAVE_2`. Selecting the check box is only possible if the current firmware on the CLU is out of date:



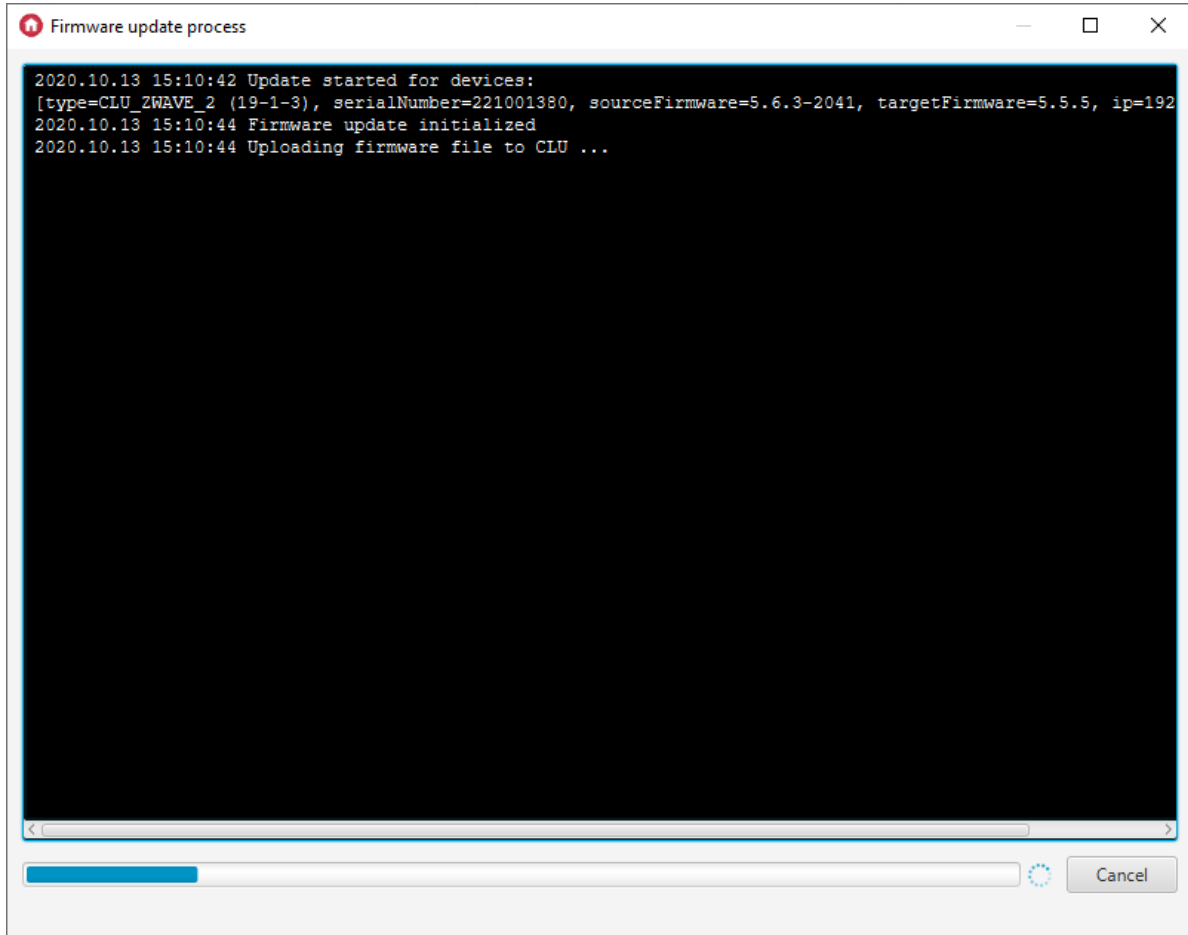
Choice	Type	Serial Number	Current Firmware	Target Firmware	Status	IP Address	
<input checked="" type="checkbox"/>	CLU_ZWAVE_2 (19-1-3)	221001380	5.5.6	5.5.5	OK	192.168.100.189	Refresh Update selected
<input type="checkbox"/>	ANALOG_DIN (25-1-2)	461000377	1.2.8	1.2.8	OK	192.168.100.189	
<input type="checkbox"/>	ROLLER_SH_DIN (23-1-2)	451002574	1.1.11	1.1.11	OK	192.168.100.189	
<input type="checkbox"/>	RELAY_DIN_4 (21-1-2)	201000275	1.3.11	1.3.12	OK	192.168.100.189	
<input type="checkbox"/>	DIGITAL_IN_DIN (20-1-2)	181000775	1.2.12	1.2.12	OK	192.168.100.189	
<input type="checkbox"/>	DIMMER_MOSFET_DIN (26-1-2)	320001024	1.1.9	1.1.9	OK	192.168.100.189	
<input type="checkbox"/>							

- Select the `Update selected` option. Read and accept to continue:



- Once accepted, the upgrade process will begin:

**NOTE!** During the upgrade process, do not turn off the power or perform other activities on the system.



- If the firmware update fails, there will be two attempts to update the module. If they also fail, the message [FAILED] will appear next to the module:

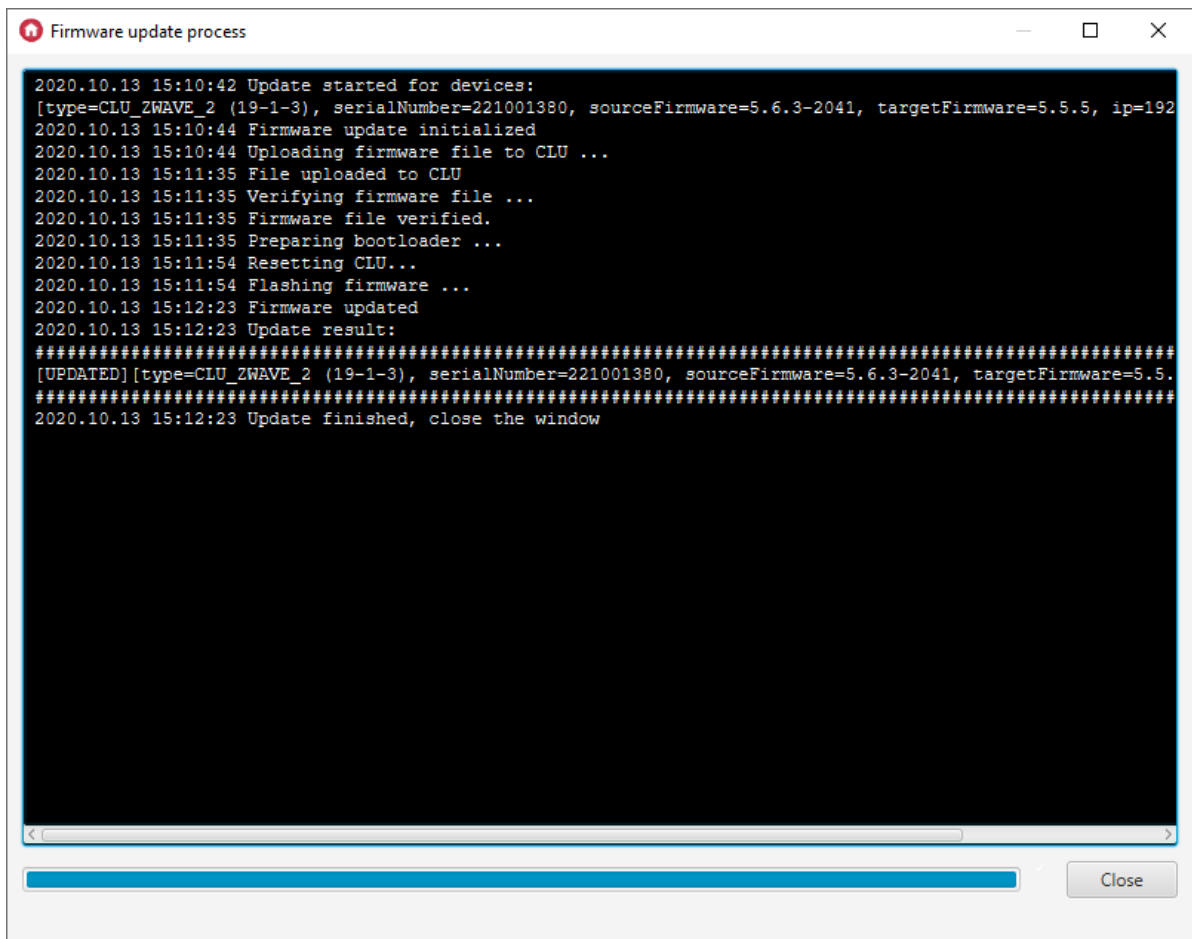
```
2020.10.14 08:46:00 Update started for devices:
[type=CLU_ZWAVE_2 (19-1-3), serialNumber=221001380, sourceFirmware=5.5.6, targetFirmware=5.5.5, ip=192.168.
2020.10.14 08:46:01 Firmware update initialized
2020.10.14 08:46:01 Uploading firmware file to CLU ...
2020.10.14 08:46:06 Firmware update failed - Uploading firmware file failed
2020.10.14 08:46:06 Retry update (2)
2020.10.14 08:48:07 Firmware update failed - Receive timed out
2020.10.14 08:48:07 Retry update (3)
2020.10.14 08:50:08 Firmware update failed - Receive timed out
2020.10.14 08:50:08 Update result:
#####
[FAILED][type=CLU_ZWAVE_2 (19-1-3), serialNumber=221001380, sourceFirmware=5.5.6, targetFirmware=5.5.5, ip=
#####
2020.10.14 08:50:08 Update finished, close the window
```

- If there are TF-Bus modules in the queue for updating and the CLU update fails, their update will be canceled:

```
2020.10.14 08:52:45 Update started for devices:
[type=CLU_ZWAVE_2 (19-1-3), serialNumber=221001380, sourceFirmware=5.5.6, targetFirmware=5.6.3-2041, ip=192
2020.10.14 08:54:45 Firmware update failed - Receive timed out
2020.10.14 08:54:45 Retry update (2)
2020.10.14 08:56:46 Firmware update failed - Receive timed out
2020.10.14 08:56:46 Retry update (3)
2020.10.14 08:58:47 Firmware update failed - Receive timed out
2020.10.14 08:58:47 Update started for devices:
[type=ANALOG_DIN (25-1-2), serialNumber=461000377, sourceFirmware=1.2.8, targetFirmware=1.2.7-1949, ip=192.
2020.10.14 08:58:47 Update started for devices:
[type=ROLLER_SH_DIN (23-1-2), serialNumber=451002574, sourceFirmware=1.1.11, targetFirmware=1.1.6-1922, ip=
2020.10.14 08:58:47 Update started for devices:
[type=DIGITAL_IN_DIN (20-1-2), serialNumber=181000775, sourceFirmware=1.2.12, targetFirmware=1.2.11-1949A,
2020.10.14 08:58:47 Update started for devices:
[type=DIMMER_MOSFET_DIN (26-1-2), serialNumber=320001024, sourceFirmware=1.1.9, targetFirmware=1.1.8-2008,
2020.10.14 08:58:47 Update result:
#####
[UPDATED][type=CLU_ZWAVE_2 (19-1-3), serialNumber=221001380, sourceFirmware=5.5.6, targetFirmware=5.6.3-2041
[CANCELLED][type=ANALOG_DIN (25-1-2), serialNumber=461000377, sourceFirmware=1.2.8, targetFirmware=1.2.7-19
[CANCELLED][type=ROLLER_SH_DIN (23-1-2), serialNumber=451002574, sourceFirmware=1.1.11, targetFirmware=1.1.
[CANCELLED][type=DIGITAL_IN_DIN (20-1-2), serialNumber=181000775, sourceFirmware=1.2.12, targetFirmware=1.2
[CANCELLED][type=DIMMER_MOSFET_DIN (26-1-2), serialNumber=320001024, sourceFirmware=1.1.9, targetFirmware=1
#####
2020.10.14 08:58:47 Update finished, close the window
```

- If the update is successful, [UPDATED] appears when the CLU is updated.





- To complete the update process, click  button.

If the update was successful, the firmware version should match the target version and the device status should be "OK". If the OM cannot establish a connection with the CLU then the status will be "DISCONNECTED".

**NOTE!** After the update is completed, perform CLU Discovery.

You can update more CLUs in one process. To do this, select all CLUs to be updated on the selection list.

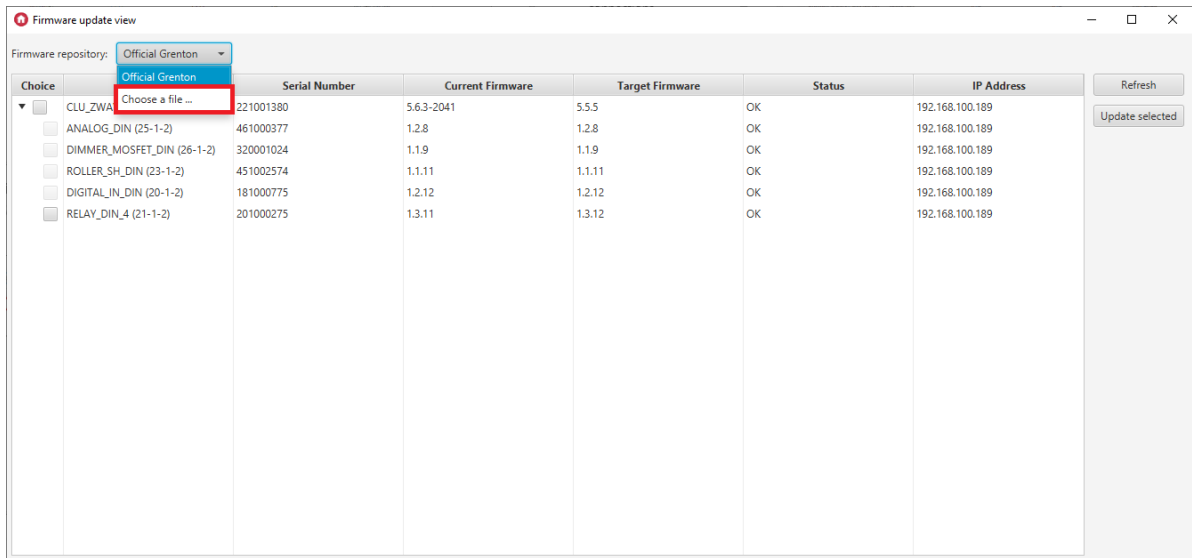
## B. Update with a .ZIP file

By default, information about the current firmware is downloaded from the Grenton server. However, you can update CLU from a local file. Updating from a file is done using .zip packages prepared by Grenton.

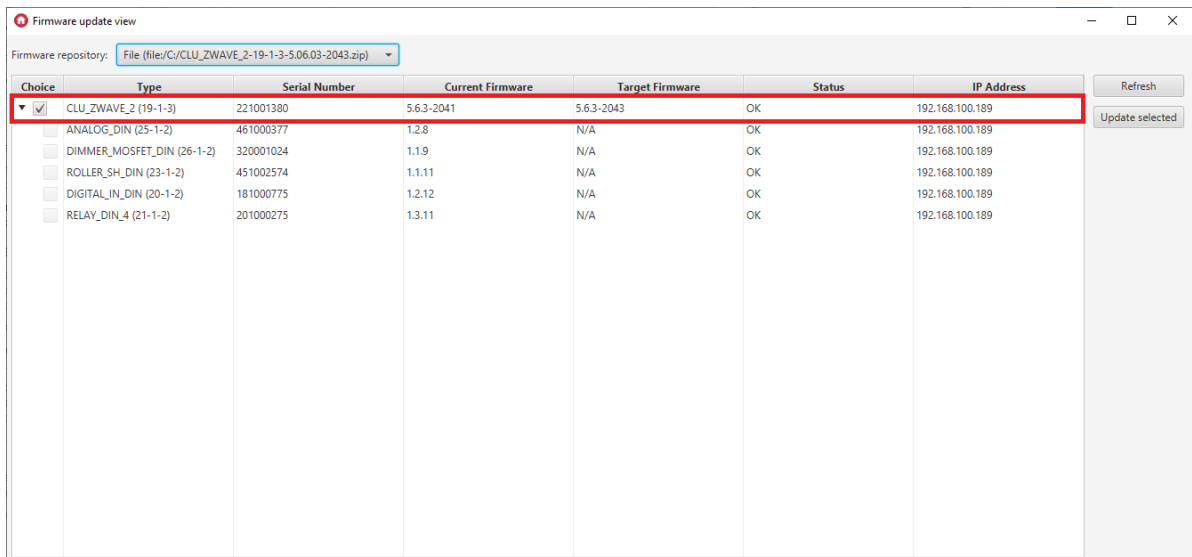
**NOTE!** Do not rename the .zip file provided by Grenton. The file must have an appropriate name to be properly loaded.

To update from a file you need to:

- Expand the *Firmware repository* and use the  option:



After loading the file, you will be able to select the module that can be updated. Under *Target Firmware*, the version number of the firmware to which the update will take place appears.



- After selecting the module, select the `Update selected` option and continue the installation, similarly to the standard update from the server [look up VI.10.2.A.](#)


### 10.3. The process of updating the firmware of the 2.0 series modules

**NOTE!** The device firmware update process is only possible for modules from the 2.0 series!

**NOTE!** The following requirements must be met for the TF-Bus device update process to run correctly

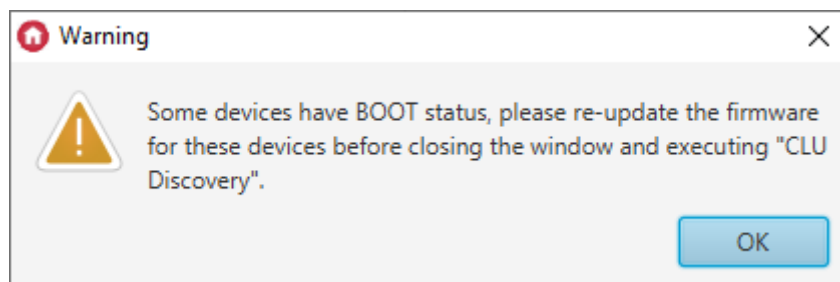
- The computer must be connected to the AC adapter, it cannot be on battery power.
- The network connection between the CLU<->router<->computer must be wired, the WIFI connection cannot be used.
- Do not perform any actions on the Grenton system while updating the firmware.
- You should start with the CLU firmware upgrade, then perform CLU Discovery and in the next step you can upgrade the modules, after which CLU Discovery should also be performed.

The 2.0 series modules update is similar to the CLU firmware update. Before starting the update, keep the following in mind:

- Firmware update of a given module is only possible if the firmware on the CLU is current. Otherwise, you must also select CLU, which will be updated first.
- The update is carried out for all modules of the same type. By selecting a given module, all modules of the same type on the list (if any) are selected.
- If an update is made for a given module to a version that changes the `firmwareApiVersion` of a module (for example, from version **1.x.x** to **2.x.x**) an icon  is displayed next to the module in the `Target firmware` column and after selecting the module a warning about the interface changes and creating new objects (`_UPGRADED`) for the device after CLU Discovery is displayed.



- At the start of the process, it is not possible to stop updating for a device that is currently being updated. The update will be aborted after the process is completed for a given group of devices (canceled for the next group of modules).
- In some cases, the update of a given device can be multi-step. In this case, after completing the upgrade process, check whether another new firmware version for the module is available.
- After updating the module, check if the firmware version is the same as the target version, and if the device status is "OK".
- If the module shows the "BOOT" status after the update, it means that the firmware update process has been interrupted and the device is still waiting for the new firmware. After closing the update process window, a warning will appear:



In such a case, the update should be repeated. The "DETACHED" status means that the CLU cannot establish connection with the module. In such a case, check the TF-Bus connections and perform a reset by disconnecting the power.

**NOTE!** After completing the update, perform CLU Discovery. It should not be performed if any module has the "BOOT" or "DETACHED" status!

## 10.4. CLU / modules status in the firmware update window

The modules status is displayed in the firmware update view table. If there is a status change while the update window is open, the list must be refreshed using the "Refresh" button.

### NOTE!

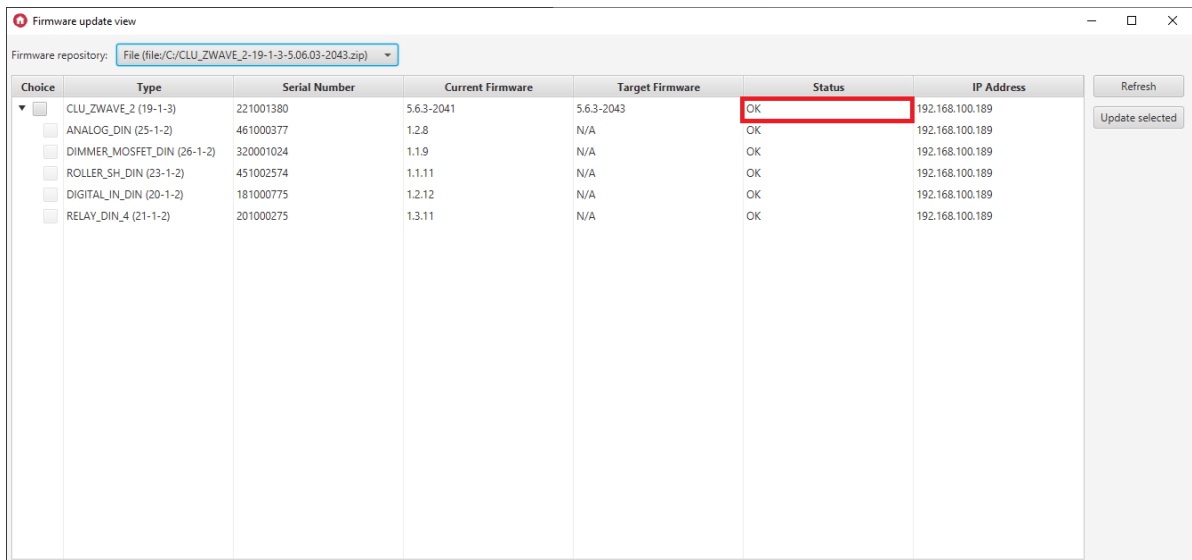
Device status display is available in Object Manager version 1.3.5.240201 or higher!

### NOTE!

The functionality of device statuses is available for CLUZ fw. version 5.06.03-2043 or higher!

## A. CLU status

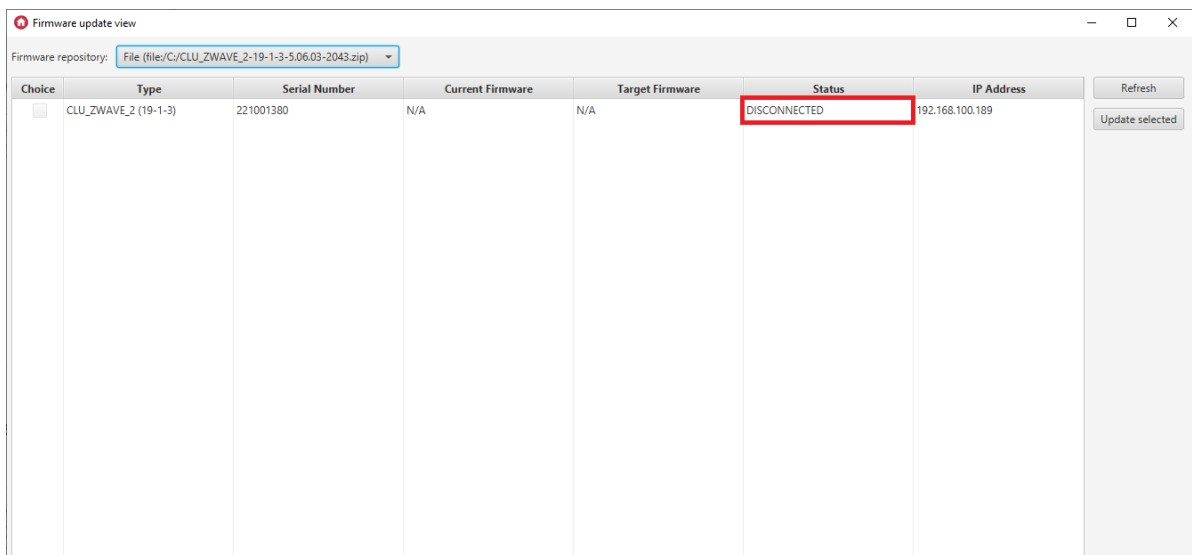
**Status: OK** - Correct connection with the CLU.



The screenshot shows the 'Firmware update view' window. At the top, there is a 'Firmware repository:' dropdown menu set to 'File (file/C:/CLU\_ZWAVE\_2-19-1-3-5.06.03-2043.zip)'. Below this is a table with the following columns: Choice, Type, Serial Number, Current Firmware, Target Firmware, Status, and IP Address. The table contains several rows of modules. The 'Status' column for the first row, 'CLU\_ZWAVE\_2 (19-1-3)', is highlighted with a red box and contains the value 'OK'. Other rows include 'ANALOG\_DIN (25-1-2)', 'DIMMER\_MOSFET\_DIN (26-1-2)', 'ROLLER\_SH\_DIN (23-1-2)', 'DIGITAL\_IN\_DIN (20-1-2)', and 'RELAY\_DIN\_4 (21-1-2)'. To the right of the table are two buttons: 'Refresh' and 'Update selected'.

Choice	Type	Serial Number	Current Firmware	Target Firmware	Status	IP Address
<input checked="" type="checkbox"/>	CLU_ZWAVE_2 (19-1-3)	221001380	5.6.3-2041	5.6.3-2043	OK	192.168.100.189
<input type="checkbox"/>	ANALOG_DIN (25-1-2)	461000377	1.2.8	N/A	OK	192.168.100.189
<input type="checkbox"/>	DIMMER_MOSFET_DIN (26-1-2)	320001024	1.1.9	N/A	OK	192.168.100.189
<input type="checkbox"/>	ROLLER_SH_DIN (23-1-2)	451002574	1.1.11	N/A	OK	192.168.100.189
<input type="checkbox"/>	DIGITAL_IN_DIN (20-1-2)	181000775	1.2.12	N/A	OK	192.168.100.189
<input type="checkbox"/>	RELAY_DIN_4 (21-1-2)	201000275	1.3.11	N/A	OK	192.168.100.189

**Status: DISCONNECTED** - The OM cannot connect to the CLU. This status is when the OM does not get a response from the CLU. In this case, check if the network cable is properly connected to the CLU / router / switch or perform a voltage reset of the CLU.

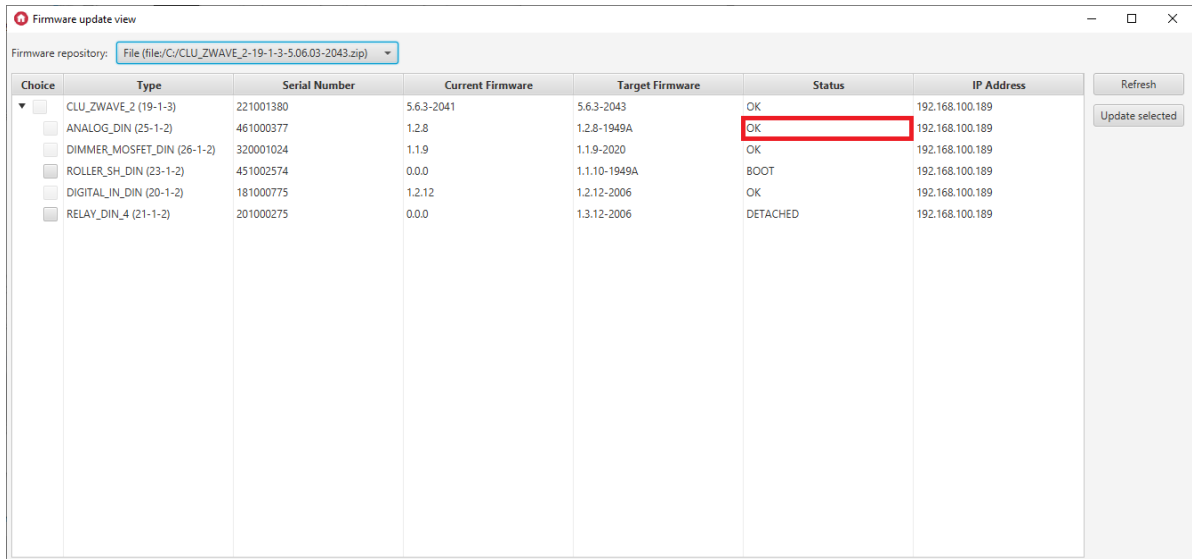


The screenshot shows the 'Firmware update view' window. At the top, there is a 'Firmware repository:' dropdown menu set to 'File (file/C:/CLU\_ZWAVE\_2-19-1-3-5.06.03-2043.zip)'. Below this is a table with the following columns: Choice, Type, Serial Number, Current Firmware, Target Firmware, Status, and IP Address. The table contains one row of modules. The 'Status' column for the first row, 'CLU\_ZWAVE\_2 (19-1-3)', is highlighted with a red box and contains the value 'DISCONNECTED'. To the right of the table are two buttons: 'Refresh' and 'Update selected'.

Choice	Type	Serial Number	Current Firmware	Target Firmware	Status	IP Address
<input type="checkbox"/>	CLU_ZWAVE_2 (19-1-3)	221001380	N/A	N/A	DISCONNECTED	192.168.100.189

## B. TF-Bus modules status

**Status: OK** - Correct connection of the module with the CLU.



Firmware update view

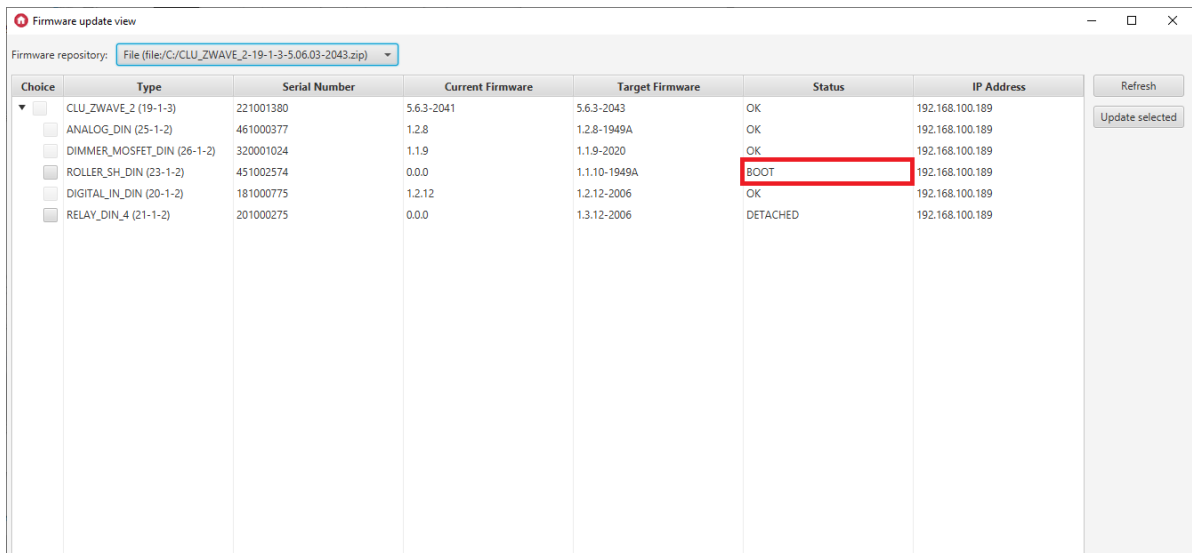
Firmware repository: File (file://C:/CLU\_ZWAVE\_2-19-1-3-5.06.03-2043.zip)

Choice	Type	Serial Number	Current Firmware	Target Firmware	Status	IP Address
<input checked="" type="checkbox"/>	CLU_ZWAVE_2 (19-1-3)	221001380	5.6.3-2041	5.6.3-2043	OK	192.168.100.189
<input type="checkbox"/>	ANALOG_DIN (25-1-2)	461000377	1.2.8	1.2.8-1949A	OK	192.168.100.189
<input type="checkbox"/>	DIMMER_MOSFET_DIN (26-1-2)	320001024	1.1.9	1.1.9-2020	OK	192.168.100.189
<input type="checkbox"/>	ROLLER_SH_DIN (23-1-2)	451002574	0.0.0	1.1.10-1949A	BOOT	192.168.100.189
<input type="checkbox"/>	DIGITAL_IN_DIN (20-1-2)	181000775	1.2.12	1.2.12-2006	OK	192.168.100.189
<input type="checkbox"/>	RELAY_DIN_4 (21-1-2)	201000275	0.0.0	1.3.12-2006	DETACHED	192.168.100.189

Buttons: Refresh, Update selected

**Status: BOOT** - The module is currently in the bootloader. This state appears when the module update is interrupted. In this case, the module must be updated again.

**NOTE!** Do not execute CLU Discovery if the module has the BOOT status! The module will not be discoverable by the Discovery process. If this happens, you will need to force an update for that module [look up VI.10.5.](#)



Firmware update view

Firmware repository: File (file://C:/CLU\_ZWAVE\_2-19-1-3-5.06.03-2043.zip)

Choice	Type	Serial Number	Current Firmware	Target Firmware	Status	IP Address
<input checked="" type="checkbox"/>	CLU_ZWAVE_2 (19-1-3)	221001380	5.6.3-2041	5.6.3-2043	OK	192.168.100.189
<input type="checkbox"/>	ANALOG_DIN (25-1-2)	461000377	1.2.8	1.2.8-1949A	OK	192.168.100.189
<input type="checkbox"/>	DIMMER_MOSFET_DIN (26-1-2)	320001024	1.1.9	1.1.9-2020	OK	192.168.100.189
<input type="checkbox"/>	ROLLER_SH_DIN (23-1-2)	451002574	0.0.0	1.1.10-1949A	BOOT	192.168.100.189
<input type="checkbox"/>	DIGITAL_IN_DIN (20-1-2)	181000775	1.2.12	1.2.12-2006	OK	192.168.100.189
<input type="checkbox"/>	RELAY_DIN_4 (21-1-2)	201000275	0.0.0	1.3.12-2006	DETACHED	192.168.100.189

Buttons: Refresh, Update selected

**Status: DETACHED** - CLU cannot establish connection with the module. In such a case, check the TF-Bus connections and perform a voltage reset of the CLU.

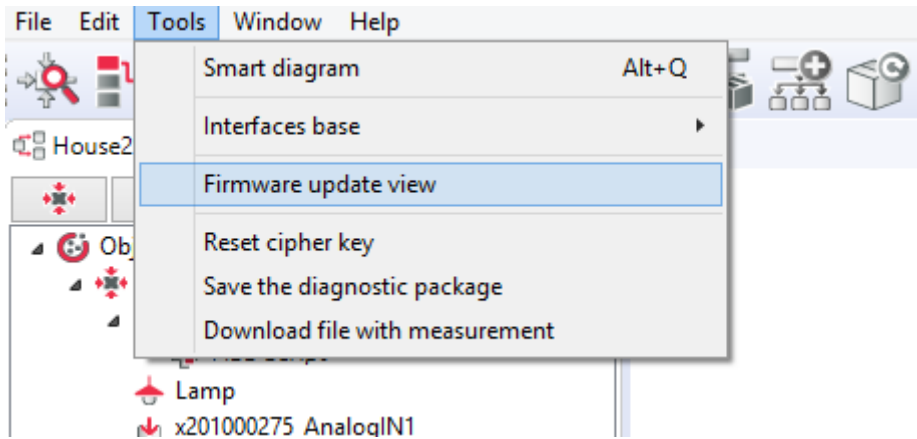
Choice	Type	Serial Number	Current Firmware	Target Firmware	Status	IP Address
<input type="checkbox"/>	CLU_ZWAVE_2 (19-1-3)	221001380	5.6.3-2041	5.6.3-2043	OK	192.168.100.189
<input type="checkbox"/>	ANALOG_DIN (25-1-2)	461000377	1.2.8	1.2.8-1949A	OK	192.168.100.189
<input type="checkbox"/>	DIMMER_MOSFET_DIN (26-1-2)	320001024	1.1.9	1.1.9-2020	OK	192.168.100.189
<input type="checkbox"/>	ROLLER_SH_DIN (23-1-2)	451002574	0.0.0	1.1.10-1949A	BOOT	192.168.100.189
<input type="checkbox"/>	DIGITAL_IN_DIN (20-1-2)	181000775	1.2.12	1.2.12-2006	OK	192.168.100.189
<input type="checkbox"/>	RELAY_DIN_4 (21-1-2)	201000275	0.0.0	1.3.12-2006	DETACHED	192.168.100.189

## 10.5 Forcing the module update

If a properly connected module is not detectable by the Discovery process, it is possible that the firmware for that module has not been properly loaded. In such a situation, it is necessary to force the module update.

In order to force a module update, you should:

- Select `Tools` from the menu bar.
- Select item *Firmware update view*.



- Right-click on the CLU module and select the option "Force update":

Firmware update view

Firmware repository: Official Grenton


Choice	Type	Serial Number	Current Firmware	Target Firmware	Status	IP Address
<input checked="" type="checkbox"/>	CLU_ZWAVE 2 (19-1-3) <b>Force update</b>	221001380	5.5.5	5.5.5	OK	192.168.100.189
<input type="checkbox"/>	RELAY_DIN_1 (19-1-3) Refresh	191000162	1.3.12	1.3.12	OK	192.168.100.189
<input type="checkbox"/>	ANALOG_DI	461000377	1.2.8	1.2.8	OK	192.168.100.189
<input type="checkbox"/>	DIMMER_MOSFET_DIN (26-1-2)	320001024	1.1.9	1.1.9	OK	192.168.100.189
<input type="checkbox"/>	ROLLER_SH_DIN (23-1-2)	451002574	1.1.10	1.1.11	OK	192.168.100.189
<input type="checkbox"/>	LED_RGBW_DIN (24-1-2)	281000507	1.4.7	1.4.6	OK	192.168.100.189
<input type="checkbox"/>	IO_MODULE_DIN_8 (30-1-2)	330000132	1.4.9	1.4.9	OK	192.168.100.189
<input type="checkbox"/>	RELAY_DIN_4 (21-1-2)	201000275	1.3.12	1.3.12	OK	192.168.100.189
<input type="checkbox"/>	DIGITAL_IN_DIN (20-1-2)	181000775	1.2.12	1.2.12	OK	192.168.100.189
<input type="checkbox"/>	CLU_ZWAVE (0-0-3)	220001096	N/A	N/A	DISCONNECTED	192.168.100.103
<input type="checkbox"/>	CLU_ZWAVE_2 (19-1-3)	221001090	5.5.5	5.5.5	OK	192.168.100.80

Buttons: Refresh, Update selected

- Select one module for which the forced update is to be performed and press the **Force** button:

**Force update**

Forcing the update should be performed when the module is connected to the TF-Bus, but it has not been updated correctly and is not found during "CLU Discovery". Forcing the update will also be performed for connected and correctly found devices.


 The action will be performed for modules connected to the following CLU:  
**Type:** CLU\_ZWAVE\_2 (19-1-3)  
**Serial Number:** 221001380

Type	Target Firmware
ANALOG_DIN (25-1-2)	1.2.8
ANALOG_DIN (25-2-2)	1.2.12
DIGITAL_IN_DIN (20-1-2)	1.2.12
DIMMER_MOSFET_DIN (26-1-2)	1.1.9
IO_MODULE_DIN_8 (30-1-2)	1.4.9
LED_RGBW_DIN (24-1-2)	1.4.6
RELAY_DIN_2 (22-1-2)	1.3.12
RELAY_DIN_4 (21-1-2)	1.3.12
RELAY_FM (31-1-2)	1.1.8
ROLLER_SH_DIN (23-1-2)	1.1.11
<b>ROLLER_SH_FM (32-1-2)</b>	<b>1.0.7</b>
TOUCH_PANEL_FM_4 (28-1-2)	1.1.5
TOUCH_PANEL_FM_8 (27-1-2)	1.1.3

Buttons: Force, Cancel

- After reading the message, select "Yes" to continue:

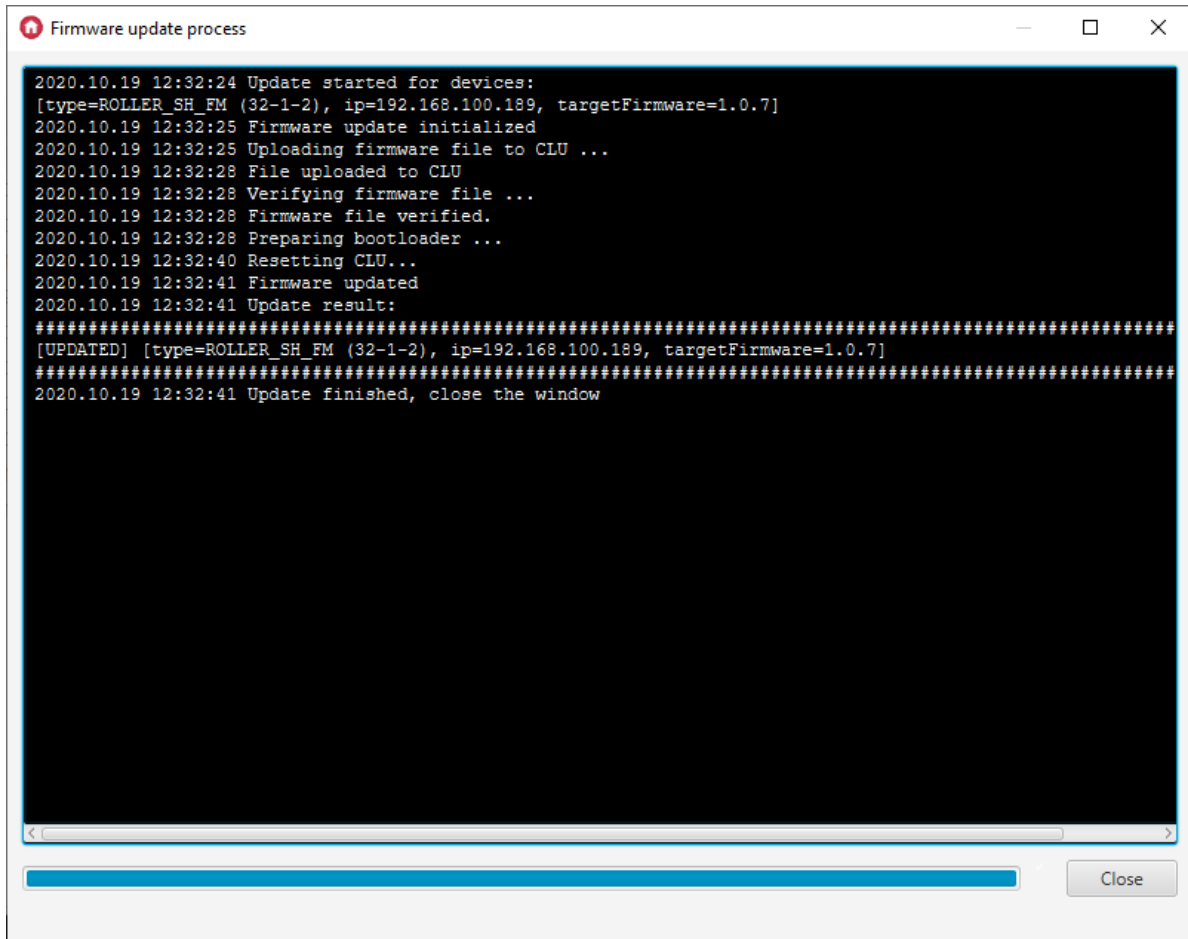
**Update selected**

 Check the documentation for the software version before the update. During the upgrade, do not turn off the power or perform any other operations on the system. The update must be carried out to the end to be successful. After updating the firmware, perform the CLU Discovery action.

Do you want to start the firmware upgrade process?

Buttons: Yes, No

- After a successful update, [UPDATED] will appear:



```
2020.10.19 12:32:24 Update started for devices:
[type=ROLLER_SH_FM (32-1-2), ip=192.168.100.189, targetFirmware=1.0.7]
2020.10.19 12:32:25 Firmware update initialized
2020.10.19 12:32:25 Uploading firmware file to CLU ...
2020.10.19 12:32:28 File uploaded to CLU
2020.10.19 12:32:28 Verifying firmware file ...
2020.10.19 12:32:28 Firmware file verified.
2020.10.19 12:32:28 Preparing bootloader ...
2020.10.19 12:32:40 Resetting CLU...
2020.10.19 12:32:41 Firmware updated
2020.10.19 12:32:41 Update result:
#####
[UPDATED] [type=ROLLER_SH_FM (32-1-2), ip=192.168.100.189, targetFirmware=1.0.7]
#####
2020.10.19 12:32:41 Update finished, close the window
```

- CLU Discovery must be performed when finished.

## 11. Diagnostic view

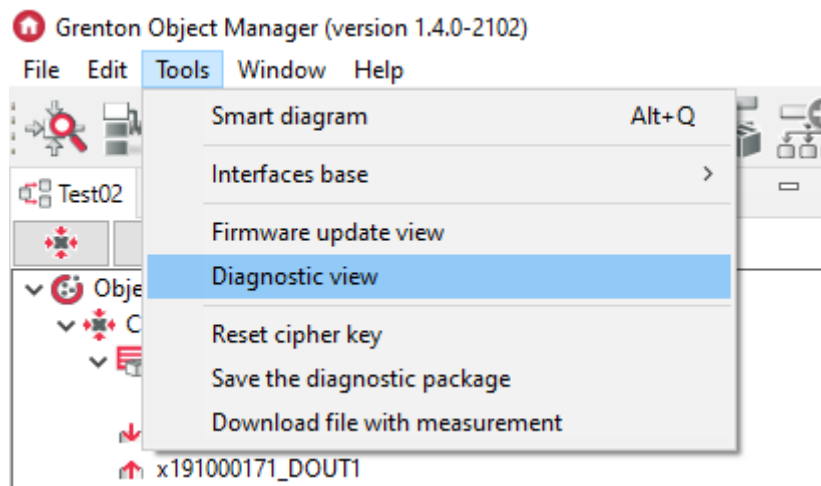
**NOTE!** Diagnostic view is available for Object Manager version v1.4.0 (or higher) and for CLUZ fw. v5.7.1 (or higher).

The diagnostic view presents information about all CLU in the project and the modules connected to them.

To open Diagnostic View in Object Manager:

- Select **Tools** from the menu bar,
- Select *Diagnostic view*.





A window showing a list of all CLUs in the project will appear:

Type	Serial Number	Status	IP Address	Cloud Connec...	Voltage	HwType	HwVersion	FwType	FwVersion	FwAPIVer.
CLU_ZWAVE_2	221001380	OK	192.168.0.217	Connected	24.16 [V]	19	1	3	5.7.1-2106	507
CLU_ZWAVE_2	221001090	DISCONNECTED	192.168.0.171	Disconnected	-	19	1	3	-	507
CLU_GATE_HTTP	521000148	DIAGNOSTICS OFF	192.168.0.252	Disconnected	-	18	2	3	1.1.0-2034C	1100

The view presents the following information for the CLU:

- `Type` - device type name,
- `Serial Number` - device serial number,
- `Status` - CLU status,
- `IP Address` - device IP address,
- `Cloud Connection` - cloud connection status,
- `Voltage` - CLU supply voltage value,
- `HwType` - hardware type,
- `HwVersion` - hardware version,
- `FwType` - firmware type,
- `FwVersion` - firmware version,
- `FwAPIVersion` - API firmware version.

CLU statuses:

- `OK` - CLU returns diagnostic data,
- `DISCONNECTED` - CLU not responding,
- `DIAGNOSTICS_OFF` - CLU responds but does not return diagnostic data.

After clicking on the CLU, a window appears with all TF-Bus modules connected to it:

The screenshot shows a 'Diagnostic view' window with two main tables. The top table lists CLU statuses, and the bottom table lists TF-Bus modules.

Type	Serial Number	Status	IP Address	Cloud Connec...	Voltage	HwType	HwVersion	FwType	FwVersion	FwAPIVersion
CLU_ZWAVE_2	221001380	OK	192.168.0.217	Connected	24.06 [V]	19	1	3	5.7.1-2106	507
CLU_ZWAVE_2	221001090	OK	192.168.0.171	Connected	23.83 [V]	19	1	3	5.7.1-2106	507
CLU_GATE_HTTP	521000148	DIAGNOSTICS OFF	192.168.0.252	Disconnected	-	18	2	3	1.1.0-2034C	1100

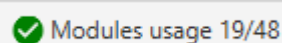
Type	Serial Number	Order	Connection	Voltage	Fails	Banned	HwType	HwVersion	FwType	FwVersion	FwAPIVersion
ROLLER_SH_...	550000453	Left 13	Connected	24.32 [V]	0	false	42	1	2	1.1.13	1
DOUModule	300000280	Left 12	Connected	0.0 [V]	0	false	1	1	2	0.0.0	12
ROLLER_SH_...	451000298	Left 11	Connected	24.01 [V]	4	false	23	1	2	2.1.1	2
LEDRGBmod...	420000010	Left 10	Connected	0.0 [V]	0	false	4	1	1	0.0.0	4
ANALOGMo...	670000031	Left 9	Connected	0.0 [V]	0	false	8	1	1	0.0.0	17
DIMMModule	310000003	Left 8	Connected	0.0 [V]	1	false	2	1	1	0.0.0	7
MODULE	290000521	Left 7	Connected	0.0 [V]	0	false	0	1	1	0.0.0	16
DIGITAL_IN_...	181000775	Left 6	Connected	24.12 [V]	3	false	20	1	2	1.2.12	1
RELAY_DIN_4	201000275	Left 5	Connected	23.95 [V]	3	false	21	1	2	1.3.12	1
ROLLER_SH_...	441000509	Left 4	Connected	23.78 [V]	1	false	32	1	2	1.0.7	1
LED_RGBW_...	281000507	Left 3	Connected	23.67 [V]	0	false	24	1	2	1.4.6	1
ROLLER_SH_...	451002574	Left 2	Connected	23.9 [V]	1	false	23	1	2	2.1.1	2
DIMMER_M...	220001201	Left 1	Connected	0.0 [V]	1	false	25	1	2	1.1.10	1

Modules usage 19/48

The view shows the following information for TF-Bus modules:

- `Type` - module type name,
- `Serial Number` - serial number of the module,
- `Order` - the sequence of connection to the TF-Bus (set manually),
- `Connection` - connection status with TF-Bus,
- `Voltage` - voltage value on the bus for the module,
- `Fails` - number of failed module responses,
- `Banned` - information if the module is banned,
- `HwType` - hardware type,
- `HwVersion` - hardware version,
- `FwType` - firmware type,
- `FwVersion` - firmware version,
- `FwAPIVersion` - API firmware version.

In the lower right corner there is information about the number of used modules:



**NOTE!** If the module does not have a voltage measurement, "0.0 [V]" will be displayed in the `Voltage` column.

If the CLU has Z-Wave modules connected, it will be possible to display them in the `Z-Wave` tab:

The screenshot shows a 'Diagnostic view' window with a table of Z-Wave modules. The table has columns for Type, Serial Number, Status, IP Address, Cloud Connection, Voltage, HwType, HwVersion, FwType, FwVersion, and FwAPIVersion. Below this table, there are tabs for 'TF-Bus' and 'Z-Wave'. The 'Z-Wave' tab is active, showing a table with columns for Type, Serial Number, NodeID, HomeID, ProductID, ManufacturerID, TypeID, Fails, Banned, Signal, Battery level, HwType, and FwAPIVersion. The data rows show details for four Z-Wave modules.

Type	Serial Number	Status	IP Address	Cloud Connec...	Voltage	HwType	HwVersion	FwType	FwVersion	FwAPIVersion
CLU_ZWAVE_2	221001380	OK	192.168.0.217	Connected	23.94 [V]	19	1	3	5.7.1-2106	507
CLU_ZWAVE_2	221001090	OK	192.168.0.171	Connected	23.67 [V]	19	1	3	5.7.1-2106	507
CLU_GATE_HTTP	521000148	DIAGNOSTICS OFF	192.168.0.252	Disconnected	-	18	2	3	1.1.0-2034C	1100

Type	Serial Num...	NodeID	HomeID	ProductID	Manufactu...	TypeID	Fails	Banned	Signal	Battery level	HwType	FwAPIVersi...
ZWAVE_GR...	390110687	2	fa1c420d	0000	3142	2700	4	false	91 [%]	N/A	3142270000...	255
ZWAVE_GR...	380100512	3	fa1c420d	0000	3142	2600	0	false	91 [%]	N/A	3142260000...	255
ZWAVE_GR...	400110056	4	fa1c420d	0000	3142	2800	0	false	88 [%]	N/A	3142280000...	255
ZWAVE_NE...	4196155007	7	fa1c420d	1082	0258	0003	0	false	83 [%]	86 [%]	0258000310...	255

The view presents the following information for Z-Wave modules:

- `Type` - module type name,
- `Serial Number` - serial number of the module,
- `NodeID` - Node ID of the module,
- `HomeID` - Home ID of the module,
- `ProductID` - Product ID of the module,
- `ManufacturerID` - Manufacturer ID of the module,
- `Failures` - number of failed module responses,
- `Banned` - information whether the module is banned,
- `Signal` - signal strength,
- `Battery level` - battery level,
- `HwType` - hardware type,
- `FwAPIVersion` - API firmware version.

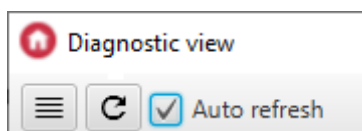
**NOTE!** The values for `Signal` and `Battery level` are updated after the module wakes up. If the module is banned (`Banned = true`) the current values will not be displayed.

**NOTE!** If the Z-Wave module is not a battery module, "N/A" is displayed in the `Battery level` column.

## 11.1 Configuration of the diagnostic view

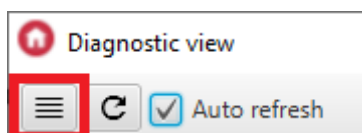
### A. Refreshing the view

The diagnostic view is refreshed when the "Refresh" button is pressed, or every 5 seconds if the `Auto refresh` option is selected.

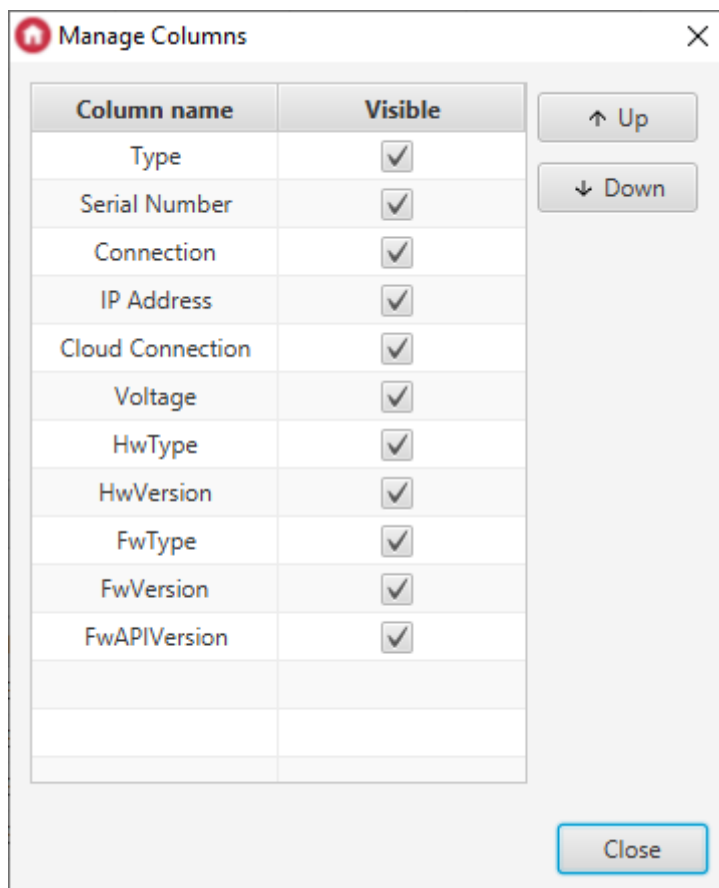


### B. Table configuration

Visibility and the order of displaying columns can be set after pressing the "Column settings" button, located in the upper left corner of the window.



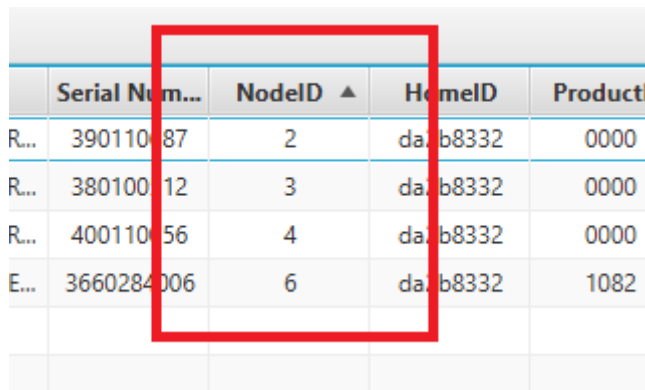
The configuration window will appear:



It is also possible to change the order of the columns by dragging their names in the main window.

### C. Sorting rows

By clicking on a column name, you can sort rows in ascending or descending order or return to the default display.

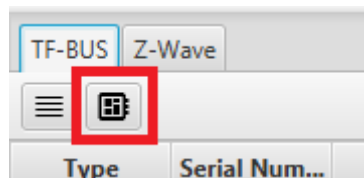


	Serial Num...	NodeID ▲	HomeID	ProductID
R...	390110087	2	da b8332	0000
R...	380100112	3	da b8332	0000
R...	400110056	4	da b8332	0000
E...	3660284006	6	da b8332	1082

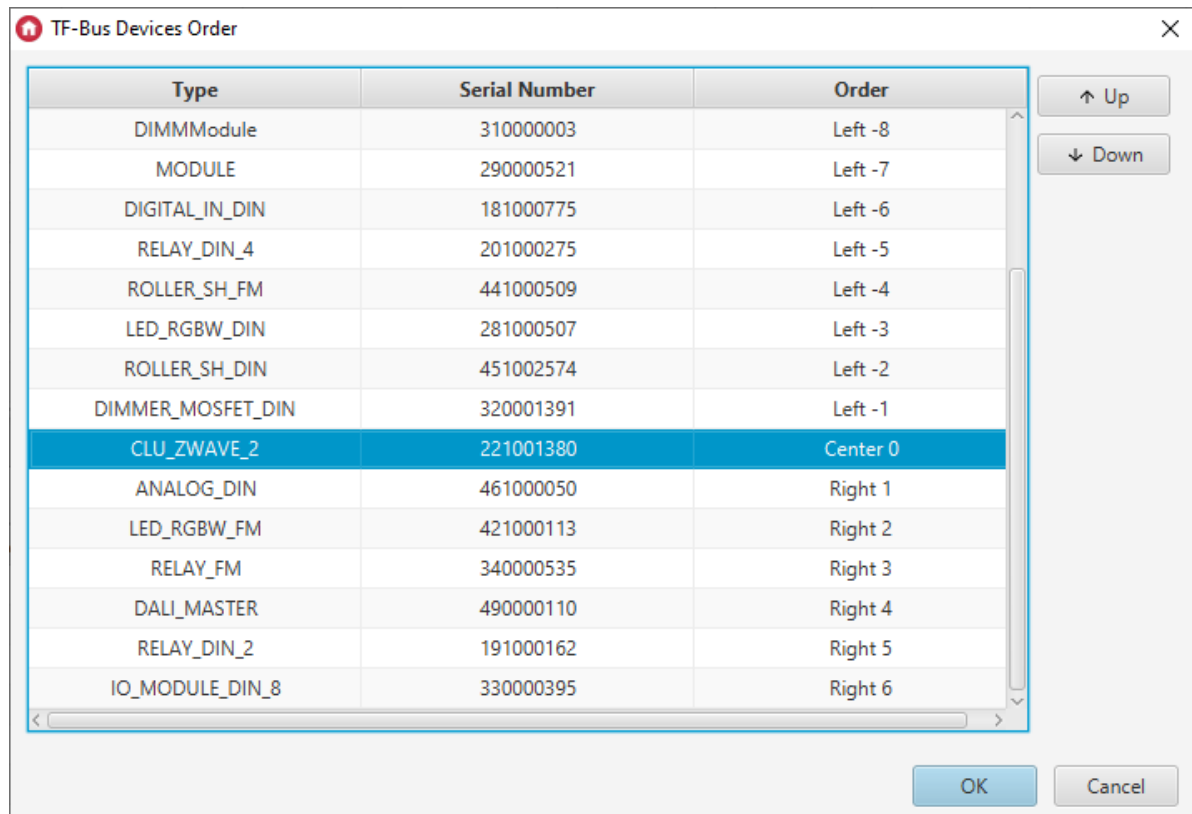
### D. Setting the sequence of connection to the TF-Bus

The order of the modules connected to the TF-Bus must be set manually, otherwise the modules in the `Order` column will be marked as "Unordered".

To set the order of modules, click the "Tfbus order" button:



Then a window will appear in which you can set the order of modules in relation to CLUZ (Center 0) using the `Up` and `Down` buttons:

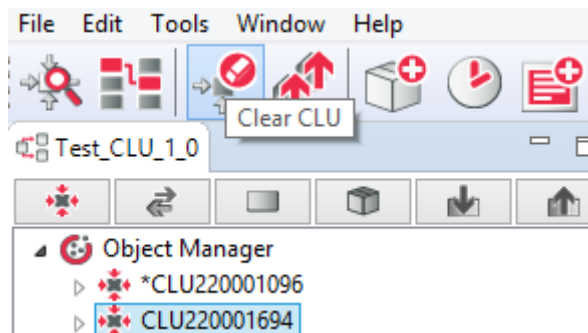


Type	Serial Number	Order
DIMMModule	310000003	Left -8
MODULE	290000521	Left -7
DIGITAL_IN_DIN	181000775	Left -6
RELAY_DIN_4	201000275	Left -5
ROLLER_SH_FM	441000509	Left -4
LED_RGBW_DIN	281000507	Left -3
ROLLER_SH_DIN	451002574	Left -2
DIMMER_MOSFET_DIN	320001391	Left -1
<b>CLU_ZWAVE_2</b>	<b>221001380</b>	<b>Center 0</b>
ANALOG_DIN	461000050	Right 1
LED_RGBW_FM	421000113	Right 2
RELAY_FM	340000535	Right 3
DALI_MASTER	490000110	Right 4
RELAY_DIN_2	191000162	Right 5
IO_MODULE_DIN_8	330000395	Right 6

## 12. Other operations on the system

### Cleaning configuration

The user always has the option of clearing the configuration of any CLU in the system. In order to clear the configuration on the selected CLU, first we need to select them, and then click on the cleaning icon.



Clearing the configuration deletes all changes and settings made and sets default values.

**NOTE!** After clearing the configuration on the given CLU, the links between the objects of the other CLU and the cleaned CLU objects will be lost!

### Downloading configuration from an existing object

Object Manager allows you to download the configuration located in an existing and operating system. The configuration can be downloaded only when creating a new clean project - it is not possible to download the configuration for a project that already has some data.

### Adding a new CLU or IOM module

After installing the new module, add it to the system. The module must be plugged into the system bus (before disconnecting the new module, the bus power supply must be disconnected). In the case of Z-Wave modules, add them to the controller - [look up VI.6.1.](#) After correct installation of the module, you should run CLU DISCOVERY, it will automatically search and add a new module. If there are unused I / O in the system, the system will launch a list that allows assigning inactive I / O to the I / O from the new module. After completing the above procedure, the module will appear in the list of objects.

### Replacing the IOM module (inputs / outputs)

If a given module is exchanged for a different one but with the same parameters (same type and same number of inputs / outputs), the module must also be replaced in the project in the Object Manager program. After correctly installing and connecting the module, the CLU DISCOVERY function must be started. The system will automatically search for and recognize a new module, and automatically assign an input / output from the "old" module to it. After searching, a list will be displayed with I / O assignments between the mentioned modules and an option to confirm and accept the change. If you accept the changes, nothing will change in the list of objects, and all assignments will be made automatically. Lack of acceptance will cause new items to appear on the list of objects, while at the same time inactive inputs / outputs will be displayed (marked in gray).

**Exchange of the module from one CLU to another in the same system.**

In situations where it is necessary to switch the IOM module from one CLU to another, physically overpass the module (switch cables), and then perform the CLU DISCOVERY function, which will update the list of modules in all CLUs

# VII. Advanced configuration functions

---

## 1. Containers

---

In order to manage available inputs / outputs more easily, OM has a function of containers, which allow to group inputs/outputs according to needs of the user. Containers can be used for example to sort inputs / outputs according to their function (lighting, heating, etc.) or their placement in the building (living room, kitchen, etc.).

To add new container, click container icon in the menu, then name it. New container icon will appear in the tree on the level of the main container. No Polish letters can be used in the container name.

The inputs / outputs are assigned to each other by: dragging from the CLU or after clicking on it with right mouse button and choosing option *Move to container*.

## 2. Scripts

---

Scripts enable creation of very complex logic using conditional functions, loops, and variables, which also allows to create complex scenes that modify their actions depending on external conditions.

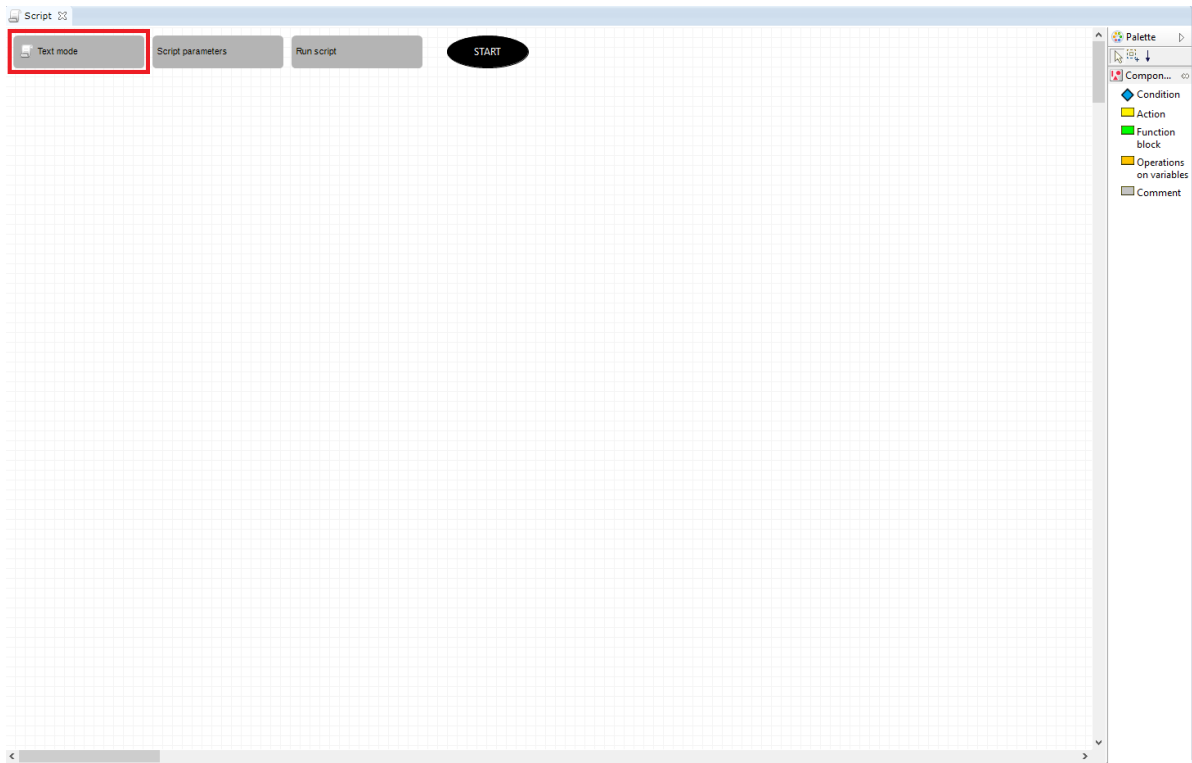
Created scripts are displayed in the system as CLU methods and can be invoked by being added to events of any object. They can also be invoked from the level of other scripts.

To create a scripts, click CLU on which the script will be stored, then select option `Create script` in the actions menu, as shows picture below:



After naming the script (no Polish letters allowed), script builder that enables script creation will open in a tab. Script builder can work in two modes: graphic and text. After new script creation, script builder automatically launches in the graphic mode. You can switch to text mode by clicking `Text mode`, as shows picture below.

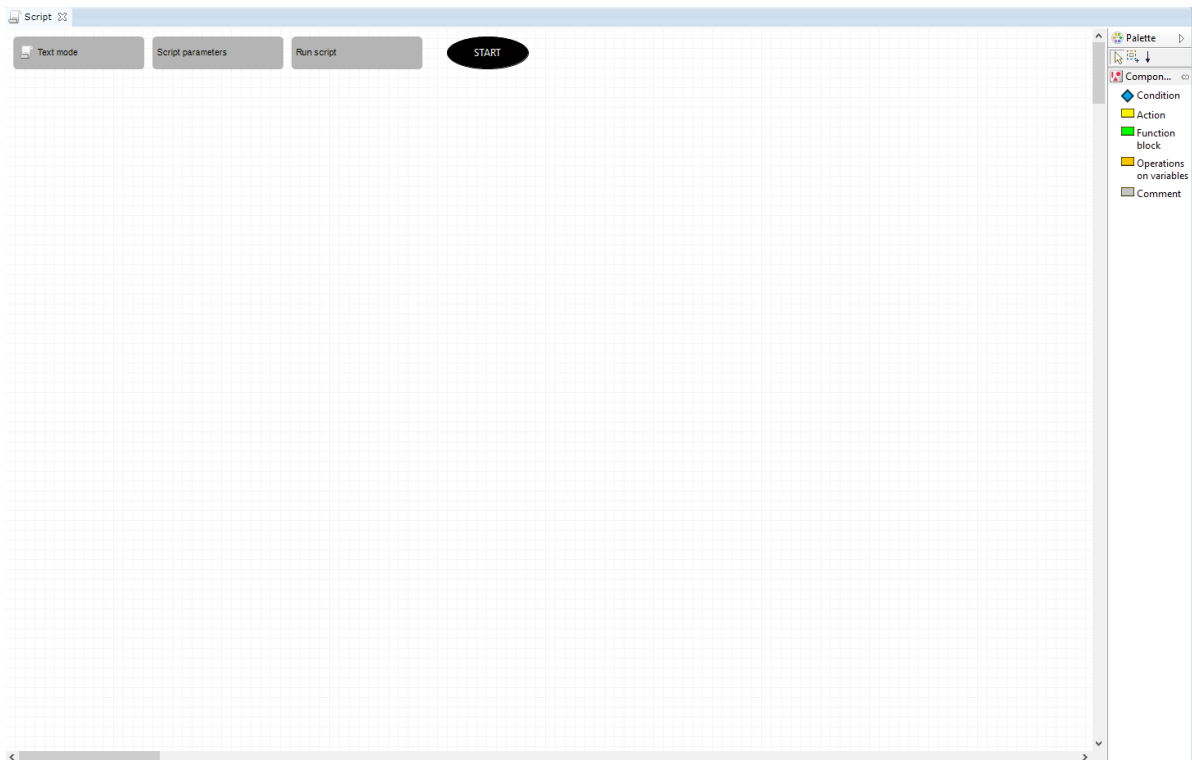




**NOTE!** Switching from graphic mode to text mode is irreversible. If a script was created in graphic mode, it will be converted to the text form. However, after edition in text mode, going back to graphic edition won't be possible.

## 2.1. Script creation in the graphic mode

After opening, a clear worksheet appears.



There is components list on the right of the worksheets. Drag commands from the list to the worksheet to add them. After dropping a command on the worksheet, a dialogue box open which allows to determine command parameters and conditional instructions. After adding a new component to the worksheet, a connection between last added component (or `start` if it's the first component) and currently added component is created automatically. Commands are fulfilled in order of connections, beginning with start. If you want to change the order of fulfilling commands, delete existing connection and add a new one (according to desired order) using `Link` tool.



**NOTE!** Leaving a component, which is not connected to other components, in the worksheet, will be seen as error and displayed as configuration error of CLU o which the script was created.

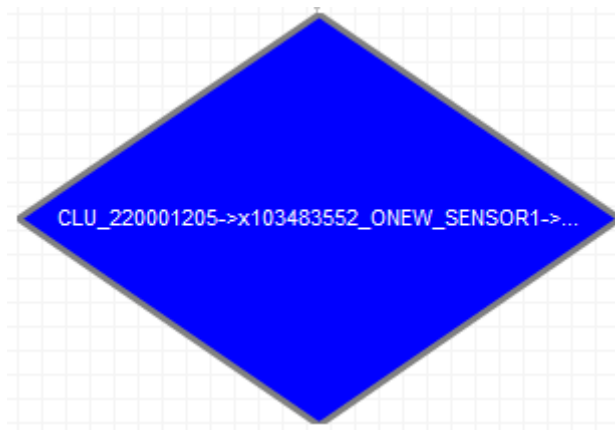
Script Builder uses the following components:

## A. Action

CLU\_220001205->x190000558\_DOUT1->SwitchOn(0)

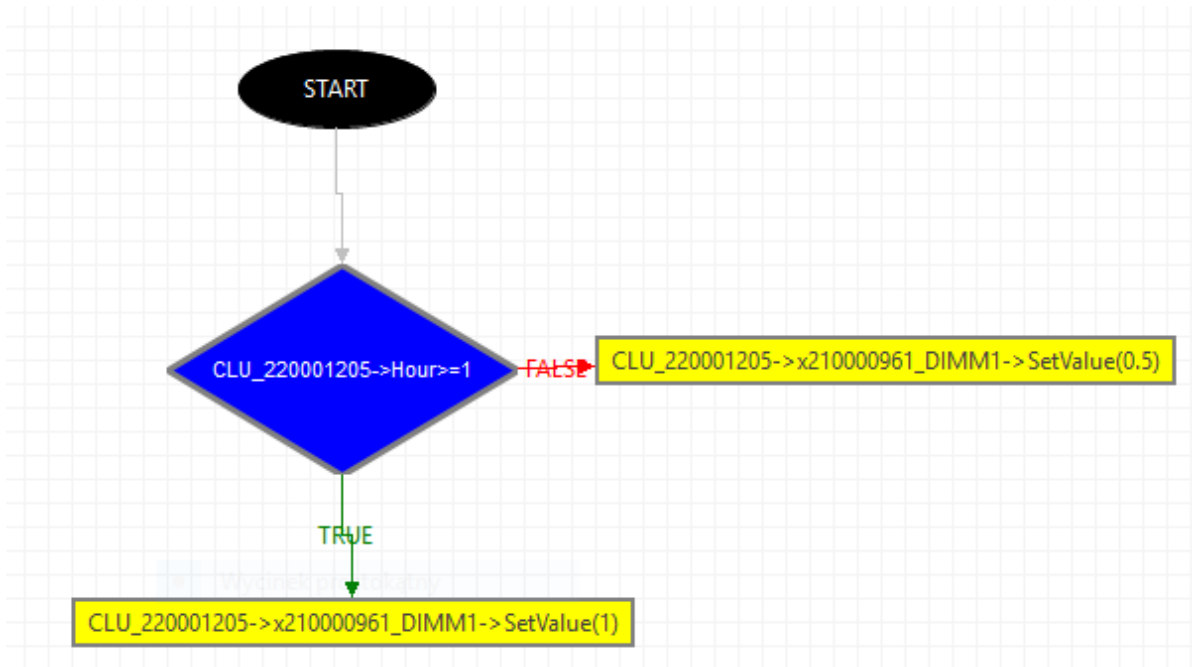
The block in which the order to be executed is entered. The command might be not only method invocation, but also value change or script invocation. After dragging action icon into the worksheet, a window with objects list and their methods opens. Scripts are available on the list as CLU methods after clicking CLU on which they are located.

## B. Condition

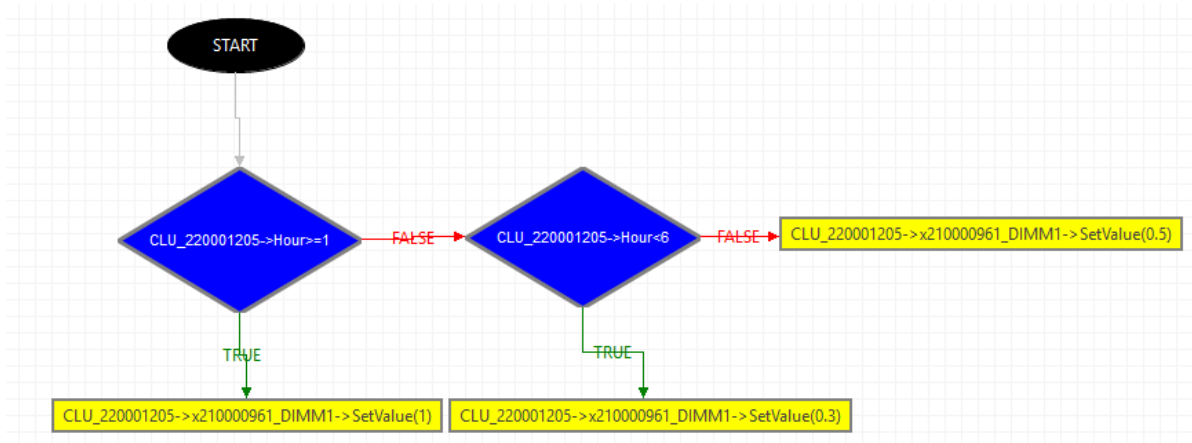


Logic block realising `IF then ELSE` function. Using this block makes it possible to make the action dependent on the conditions, eg if it is dark, turn on the light, if not, turn it off. After dragging block to the worksheet, enter condition which needs to be fulfilled in its parameters. After adding "condition" component, add at least one "action" or "Operation on variables" component and connect it with "condition" component with an arrow which head points at the action. After adding an arrow, OM will ask whether the action should be performed when condition is met (`true`) or when it is not (`false`). Two actions can be connected to one condition - when performed when it is fulfilled, the other when it is not. To change the `true` / `false` assignment, double click on one of the arrows coming out of the condition.

Picture below shows easy conditional instruction which changes light intensity depending on the hour.



Conditions can be connected via cascade connection, thanks to which operator `and` can be implemented (action is performed when two or more conditions are fulfilled). The following diagram shows an example of using cascade connection:



Conditions can compare any object feature or script parameter with a number, a text, other feature, or other script parameter.

### C. Function block

```
delay(500)
```

Contains instructions invoked within the script which can be used for creating more advanced scenes ("delay" function) and debugging ("print" function). After dragging the icon of the block into the worksheet, a window with list of function blocks opens. The list contains:

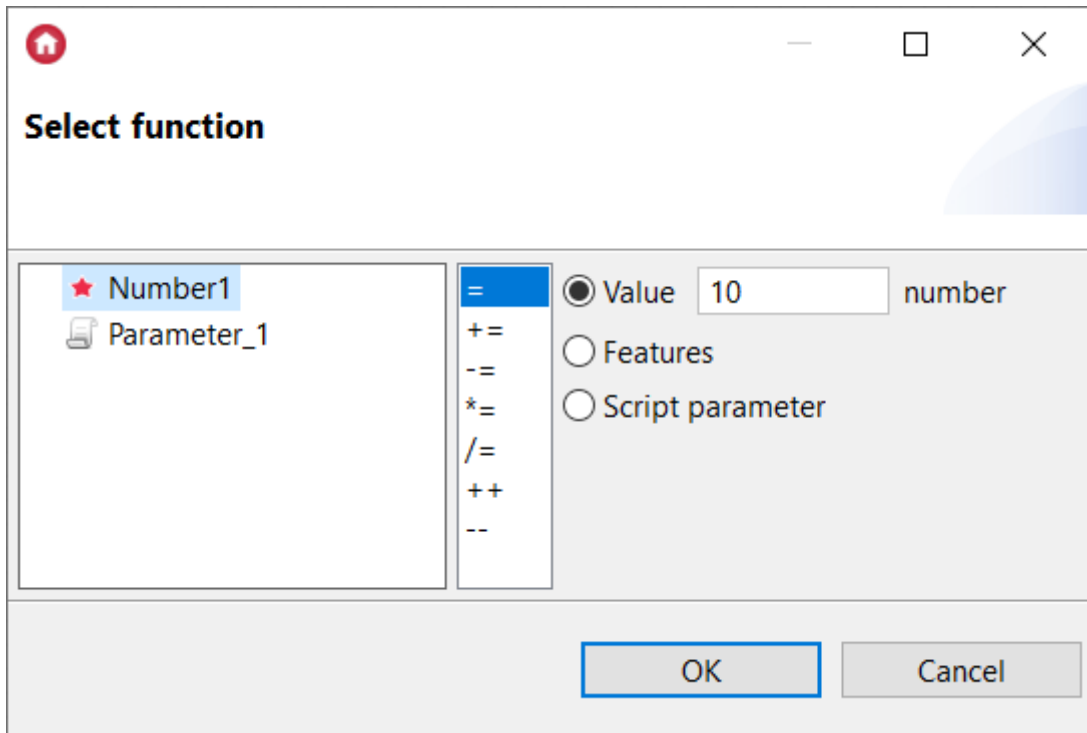
- **DELAY** Allows to set time delay between consecutive commands during script realisation
- **PRINT** The command that causes the display of the declared text in debugging.

## D. Operation on variables

CLU\_220001205->Page++

The block enables creation of complex logical functions using variables. The variables must be declared first so they can be used in the script. Variables can be declared in scrip parameters and CLU user features. A variable declared as script parameter can be used within the script to make calculations during running of the script. Data stored within that variable is not available outside the script. To store or use data from variables outside the script, use CLU user features.

When the user feature or script parameter is set to type `number`, several math operations are available.



Description mathematical operations based on the example above (with the given `number` value = 10):

Actions	Description
=	Setting the number 10 as the value of the user feature / script parameter.
+=	Adding the number 10 to the value of the user feature / script parameter.
-=	Subtracting the number 10 from value of the user feature / script parameter.
*=	Multiplying the number 10 by the value of the user feature / script parameter.
/=	Dividing the value of the user feature / script parameter by the number 10.
++	Increasing the value of the user feature / script parameter by +1.
--	Decreasing the value of the user feature / script parameter by -1.

## 2.2. Script creation in the editor

Another method of script creation is using text editor, which gives practically endless possibilities of script creation using LUA instructions expanded with possibility of using addresses of objects of logical interface.

Logical interface addresses are treated as functions and can be invoked and used as parameters in conditional instructions, loops, etc.

The script below shows way of using logical interface addresses in scripts:

```
Graphical mode | Parameters | Run script
1 if (CLU_220001205->Hour>=1) then
2 CLU_220001205->x210000961_DIMM1->SetValue(1)
3 else
4 if (not (CLU_220001205->Hour<6)) then
5 CLU_220001205->x210000961_DIMM1->SetValue(0.5)
6 else
7 CLU_220001205->x210000961_DIMM1->SetValue(0.3)
8 CLU_220001205->Page=CLU_220001205->Page+1
9 end
10 end
11
```

## 2.3. Script parameters

Scripts can have initial parameters, which are sent during their invocation (e.g. in the event) and then can be used inside the script, e.g. in conditional instructions. Script parameters are created by clicking on `Script parameters`, then a window will open in which you should select `Add parameter` and define the name, value to run, default value, type and restrictions. To delete a variable, click the button `-` on the right.

**Scripts Parameters**

Object name:

Name	Value to run	Default value	Type	Restrictions	
<input type="text" value="number_1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="number"/>	<input type="text" value="1-10"/>	<input type="button" value="-"/>
<input type="text" value="string_1"/>	<input type="text"/>	<input type="text" value="null"/>	<input type="text" value="string"/>	<input type="text"/>	<input type="button" value="-"/>
<input type="text" value="logic_1"/>	<input type="text"/>	<input type="text" value="true"/>	<input type="text" value="boolean"/>	<input type="text"/>	<input type="button" value="-"/>

**NOTE!** Variable names cannot contain spaces and start with a digit or a character.

**Value to run** - this is the value that is set when running the script using the "Run script" button in the Object Manager.

**Default value** - this is the parameter value that will be used if the parameter is not specified when calling the script.

**Type** - allows to determine type of data that will be stored in the parameter:

- **string** - for text data;
- **number** - for numerical data;
- **boolean** - for boolean variables  true /  false.

**Restrictions** - for numerical parameters, restrictions of maximum and minimum parameter values can be set. In the case of invocation of script outside this range, the script will be invoked with default parameter value. Restrictions must be specified in the format x-y, where x and y are the minimum and maximum values of the restriction. After entering the restriction, complete the *Value to run* field.

**NOTE!** Script parameter contains values which can be used only within it (local values). These values are not available in other scripts. If it's necessary to save values to use in other areas, use user features available in CLU, or send the value to another script using its parameter.

**NOTE!** Local variables in one script may only be used by the CLU on which the script was created. In order for the variable to be used by other CLUs, the **user feature** must be defined on the CLU, e.g.:

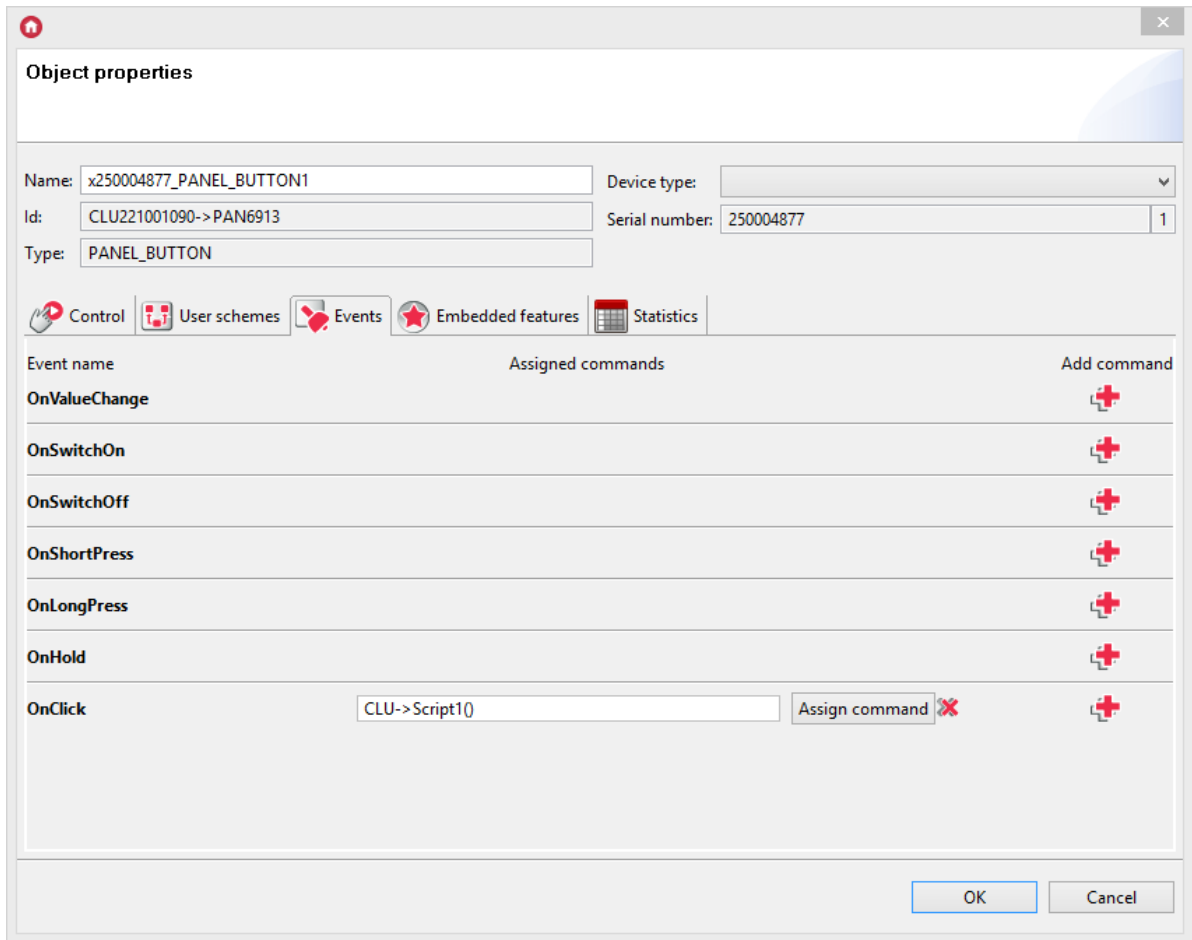
(CLU\_A - CLU on which the script was created)

```
CLU_A->Lampa1->SetValue(local_variable)
CLU_B->Lampa2->SetValue(user_variable)
```

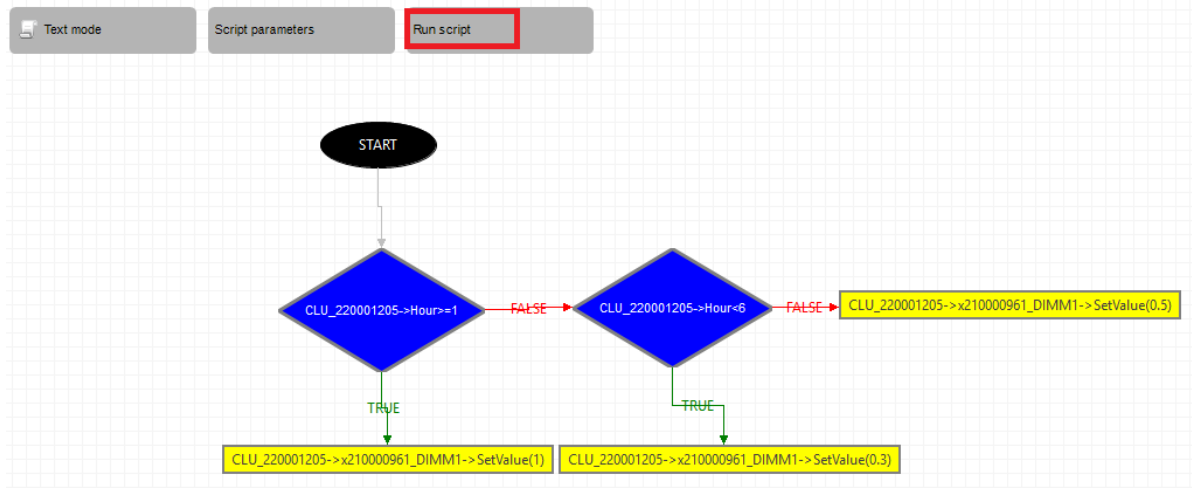
## 2.4. Scripts invocation

Scripts are displayed and treated as CLU methods. They can be invoked from events of any object, and from action block in another script identically as other methods.

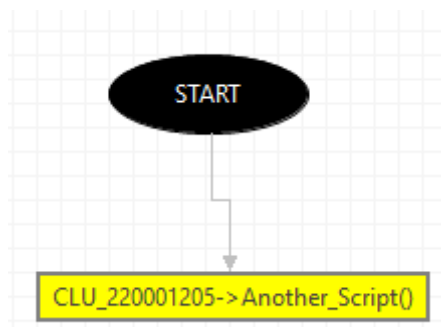
- **Invocation by an event** Picture below shows adding script to a switch. The script will be started after pressing the switch.



- **Invocation from script level** The following figure shows how to call from the script level using the `Run script` button.



- **Invocation from another script** Picture below shows fragment of diagram in which another script was invoked using action block.



- **Calling a script with a parameter**

To specify input parameters when calling the script, enter them in parentheses in the correct order:

```
CLU-> script (12, "text", true)
```

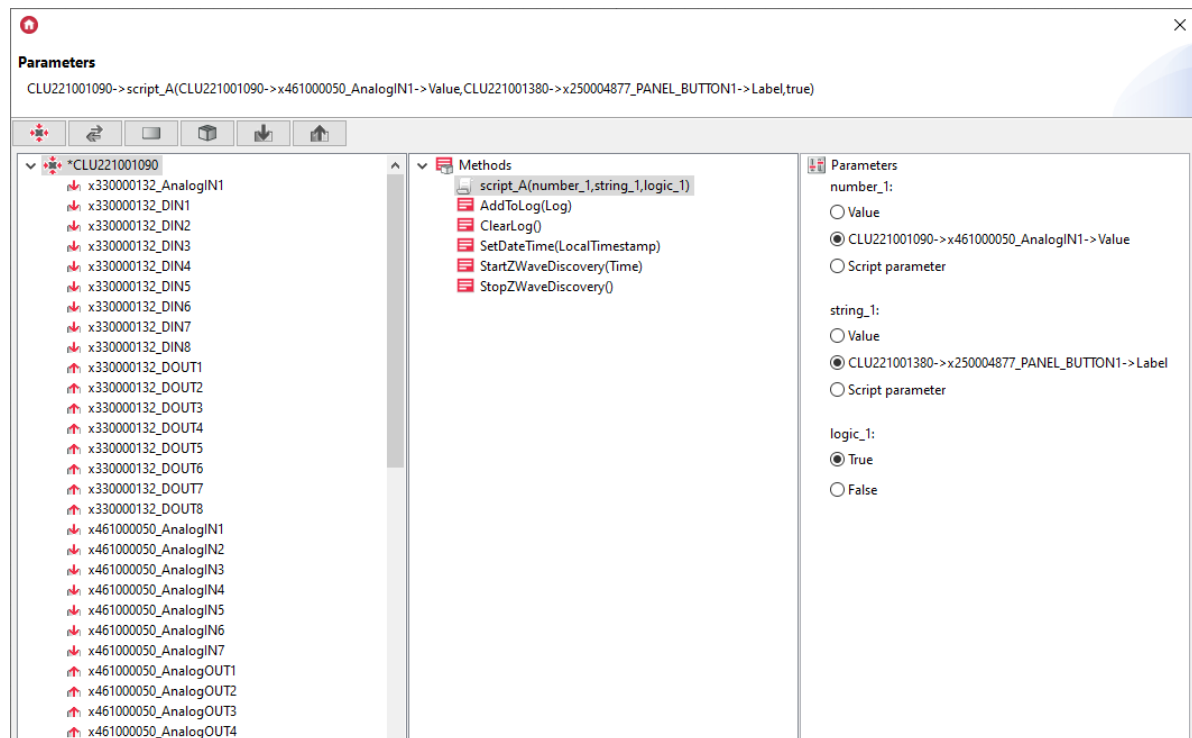
where: parameters were specified for a variable of type number, string and boolean.

To assign specific features to local variables, enter the full paths of given features:

```
CLU-> script_A (CLU-> AnalogIN1-> Value, CLU-> BUTTON1-> Label, CLU-> CloudConnection)
```

where: parameters were specified for a variable of type number, string and boolean.

The input parameters can be easily defined using the Parameters window:



### 3. Date and time

CLU is equipped with real time clock (RTC) powered by built-in battery. CLU provides several features which can be used in the script. The full list of time-related features reads as follows:

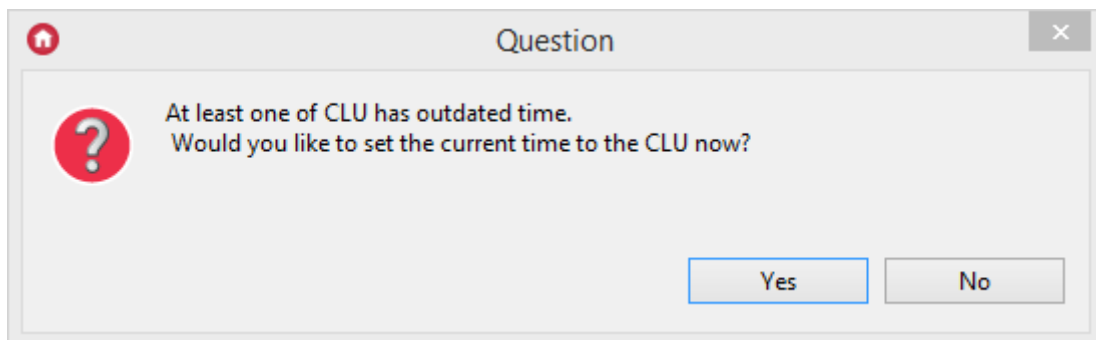


Name	Description
Uptime	Work time of the device since the last reset (in seconds)
Date	Shows current date
Time	Shows current time (hh:mm:ss)
Day	Shows number of current day of the month
Month	Shows number of the current month
Year	Shows number of the current year
DayOfWeek	Shows number of current day of the week (0=Sunday)
Hour	Shows current hour(without minutes and seconds)
Minute	Shows current number of minutes since the last full hour
LocalTime	Shows current local time maker
TimeZone	Shows the current time zone set
NTPServer	Shows the address of the UTC time server

`LocalTime` feature is worth noticing - it shows number of seconds since 1970 (local time) as one figure. It can be useful for checking how much time has passed from the previous script running or event.

The current time (in UTC form) is automatically downloaded to the CLU from the NTP server and corrected by the `TimeZone` set. You can also set it manually using the `SetDateTime` method.

If, while opening the project, the Object Manager detects that the time on the CLU is out of date, it will ask the user to set the current time to the CLU.



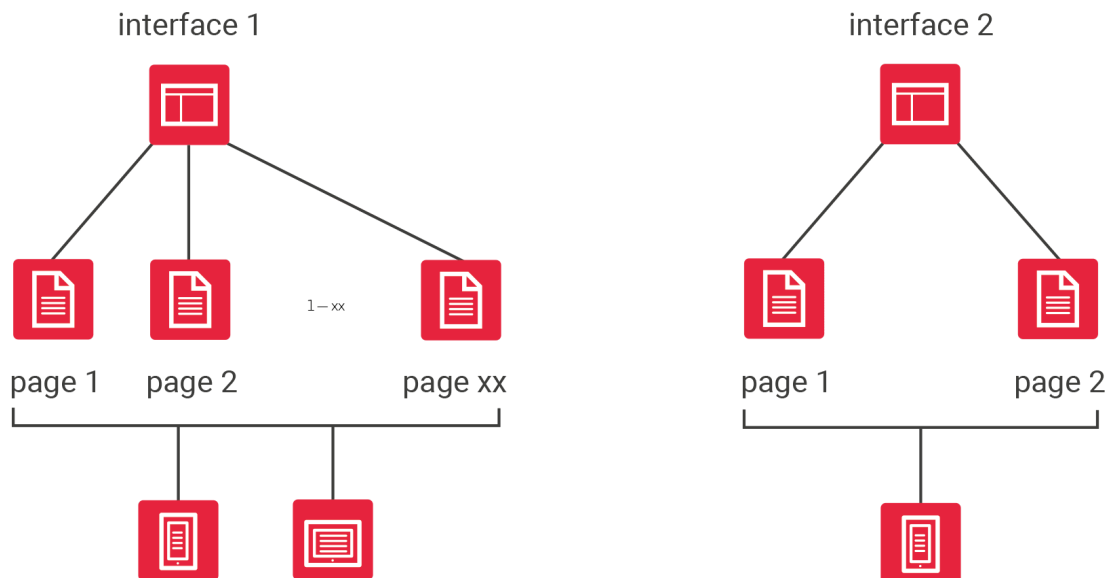
# VIII. Visual Builder - Smartphone control

**NOTE!** Support for Visual Builder functionality has been ended. The Home Manager interface creator will be removed in the next versions of Object Manager.

## 1. System control on the level of smartphone

The system enables control using any devices working on the basis of both android and iOS operational systems (tablets, mobile phones, media players). For each system, one or numerous interfaces can be prepared, each of which can contain many subpages. It enables creation of various interfaces for various users according to their needs and preferences, as well as logical sorting of control function within each interface (e.g. each room at separate subpage or dividing by function such as heating, lighting etc.).

Interfaces are created using Visual Builder tool (which is part of Object Manager), then sent to the application installed on the android or iOS device.

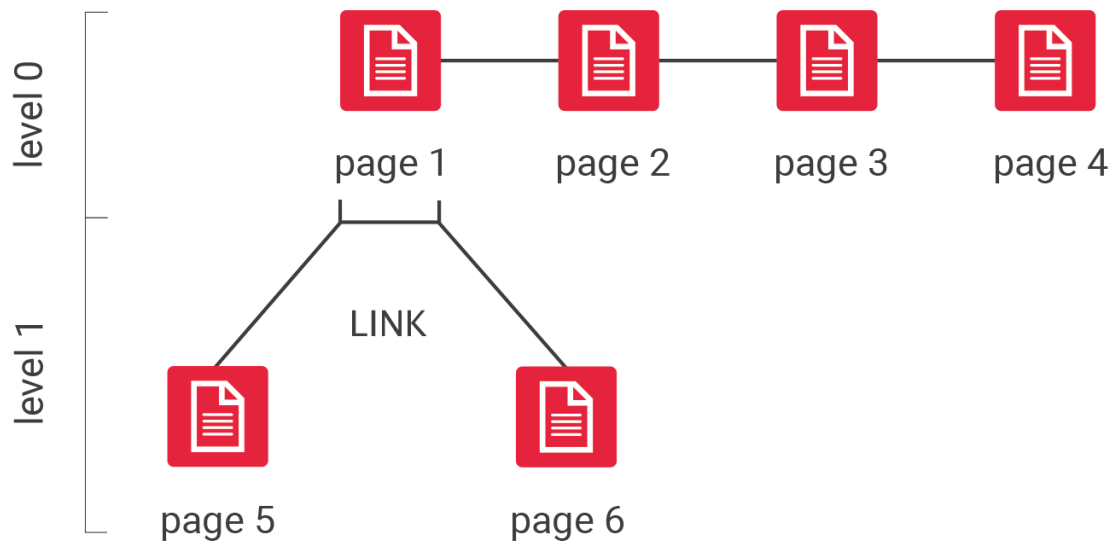


## 2. Interface structure

**NOTE!** Support for Visual Builder functionality has been ended. The Home Manager interface creator will be removed in the next versions of Object Manager.

Each interface consists of one or many subpages on which control elements (buttons, scroll bars) are placed. The designer can fully control page layout, arrangement of graphical elements, and interface appearance which is set by graphic skin selection.

Pages in the interface can be on two levels: level zero and level one. Pages located on level zero are available as basic interface pages used for navigation by scrolling left / right through them. The user can get to pages of level one by *Link* component.



### 3. Application for smartphone - GRENTON HOME MANAGER

GRENTON HOME MANAGER application allows to launch user interfaces designed in Visual Builder on android and iOS devices. A packet prepared in Visual Builder containing interface description, all files related to it, and configuration data is sent to the application.

Depending on the created interface, the GRENTON HOME MANAGER application allows to check current system status and control of all functions available in the system.

To control GRENTON system using smartphone, install the application on it, then send interface created using Visual Builder. The application can be downloaded for free from GOOGLE PLAY store for android devices and APP STORE for iOS devices. For application to work properly, it must be installed on a device connected to the same LAN network as GRENTON system, or there must be remote connection created in the WAN network.

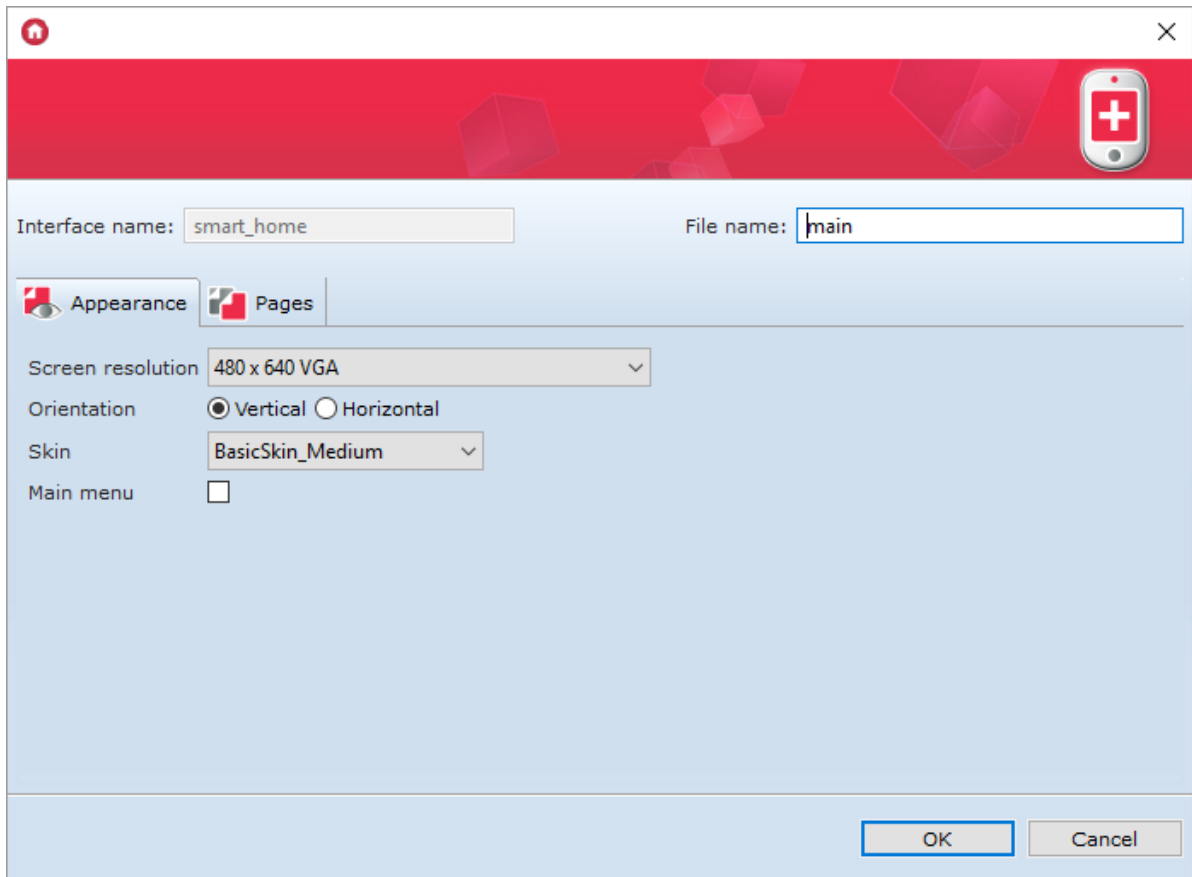
### 4. New interface creation

**NOTE!** Support for Visual Builder functionality has been ended. The Home Manager interface creator will be removed in the next versions of Object Manager.

To create new interface, select `Add interface` in actions menu.



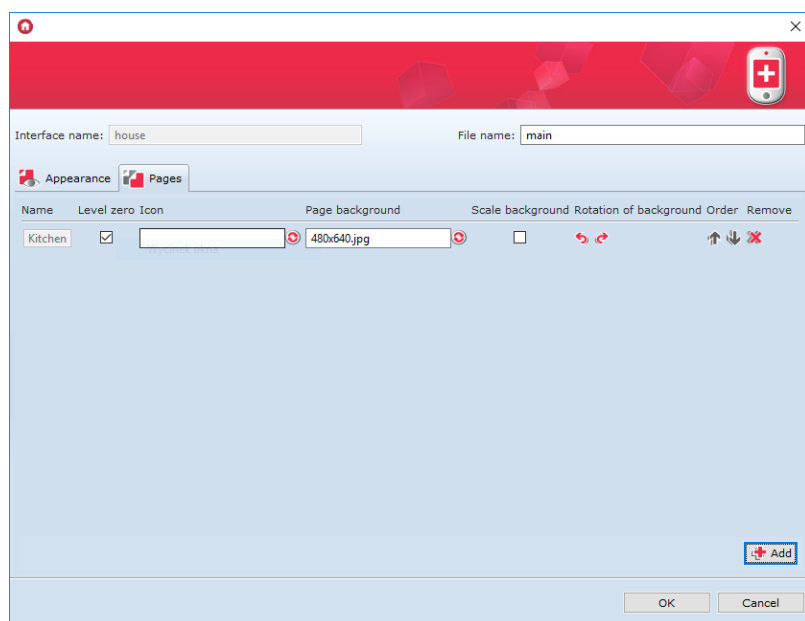
After entering name, new interface window will open. It contains two tabs: appearance and pages (interface window is also available through double-clicking icon of created interface in the objects menu). `Appearance` tab:



Contains information on the way of displaying interface, such as: resolution, orientation, available skins list, and box that creates main menu upon selection.

Upper right corner contains fields `File Name`. This name, after sending interface to mobile device is displayed on its interfaces list. In the case of sending more than one interface to one device, remember to give different name to each interface.

Tab `Pages` contains list of all created pages.

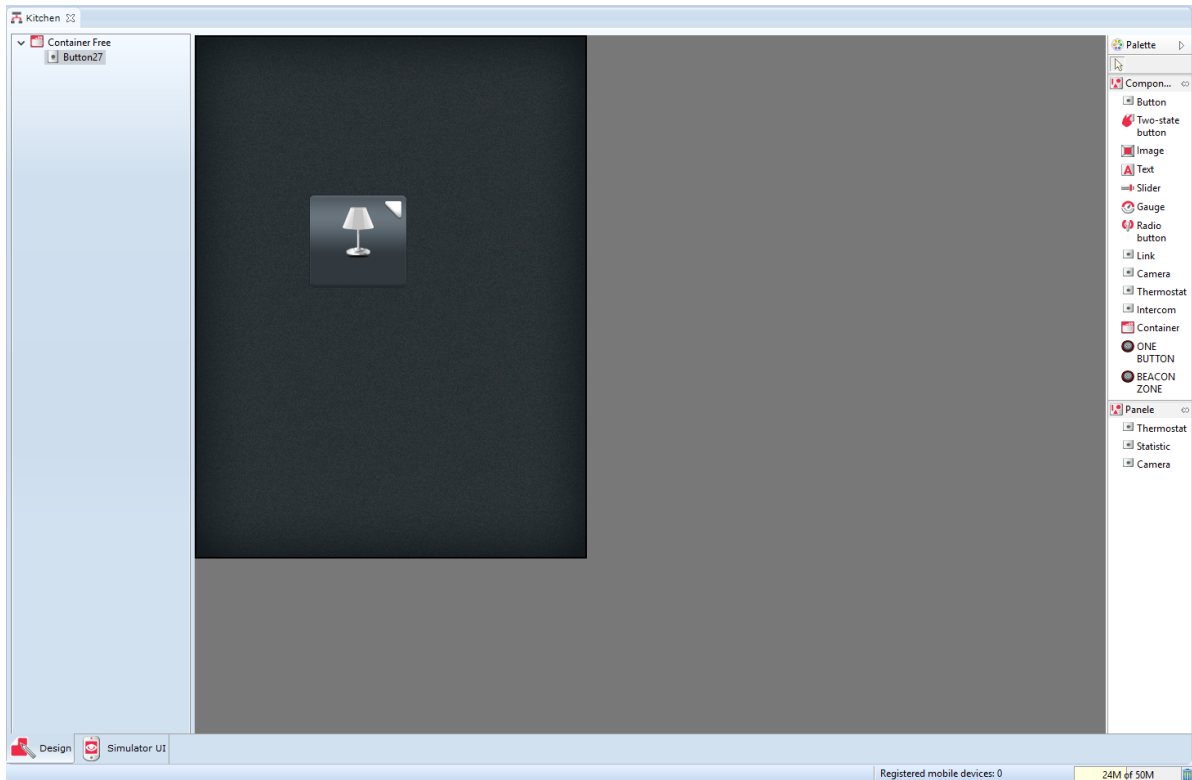


In the tab, you can change order of displaying pages and delete created pages. After selecting `level zero` option, the page will be visible in the main menu. You can also change page icon displayed in the menu on the bottom of the page and page background in this tab.

If the chosen background has orientation different than the one used in the interface, you can rotate it using `Rotation of backgrounds` buttons.



The DESIGN tab contains: objects list, main container, components and panels list.

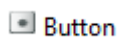


Objects list displays all objects used in the current worksheet.

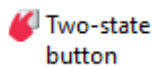
## 4.3. Components

**Components** - list of objects which can be used during interface creation. Components list includes:

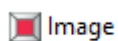
- **Button** - works as a monostable button



- **Button** - works as a bistable button



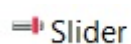
- **Picture** - enables adding picture from an external file



- **Text** - enables adding text box



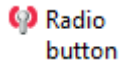
- **Slider** - enables fluid regulation



- **Measure** - displays object value in an analogue way



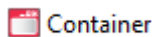
- **Radio** - displays object state in digital (on/off) way



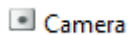
- **Link** - enables creating links to other pages within the same interface



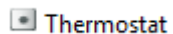
- **Container** - arranges components in the workspace in specific way



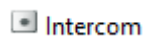
- **Camera** - allows to display image from an IP camera in the Home Manager application



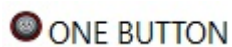
- **Thermostat** - allows displaying the virtual object Thermostat in the Home Manager application



- **Intercom** - allows you to configure the intercom (configure the connection to the SIP server, assign methods to specific events and display the image from the IP camera during the call)



- **ONE BUTTON** - allows you to assign the BEACON method to the event in ONE BUTTON mode.



- **BEACON ZONE** - allows you to configure BEACON in BEACON ZONE mode and assign specific methods to events (after adding BEACON ZONE to the page visible at the bottom).




Selected objects are put in the container by dragging them from the components list and their arrangement depends on type of used main container.

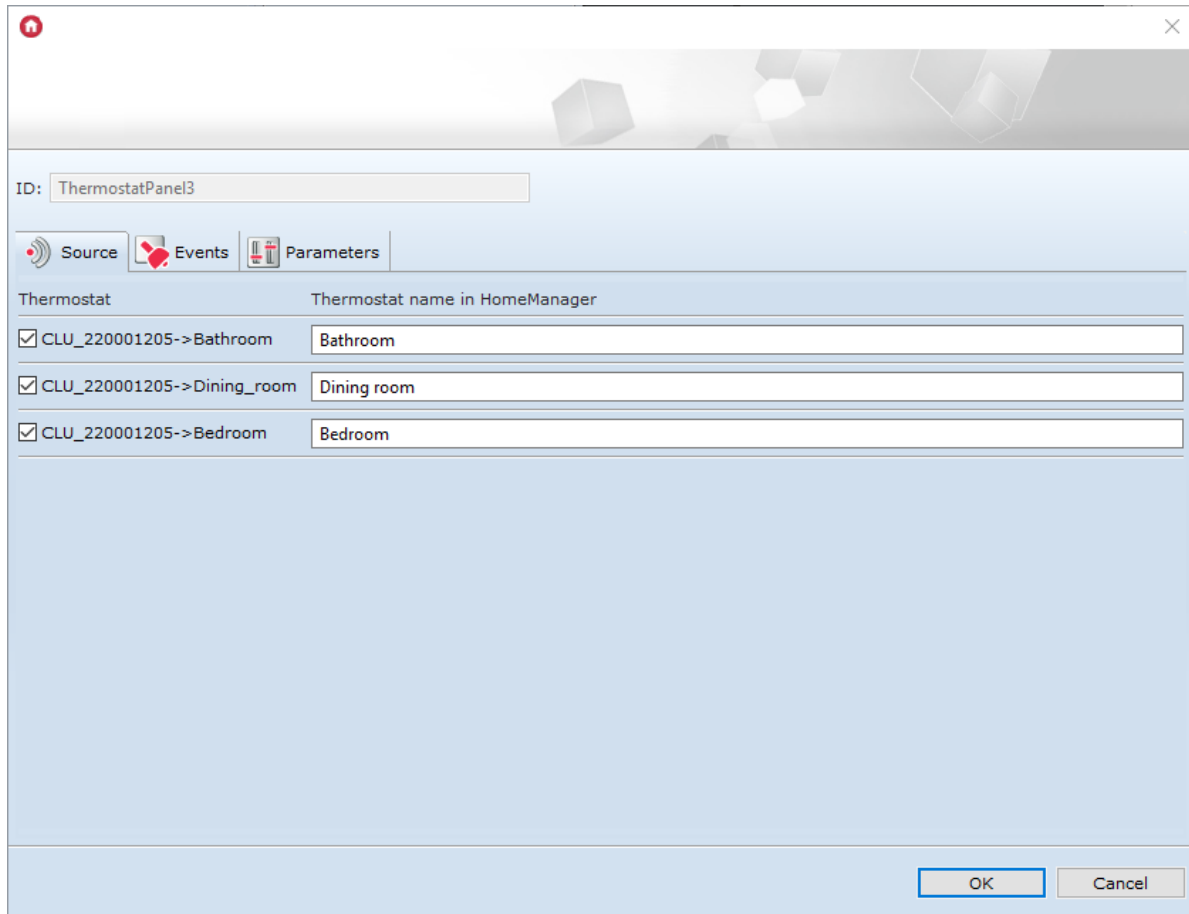
## 4.4. Panels


**Panels** - list of objects that can be used when creating the interface for a mobile device. Panels, unlike components, occupy the entire page of the mobile interface. The list of panels includes:

- **Thermostat** - creates a panel for the thermostat on the entire interface page in the HM.

 **Thermostat**

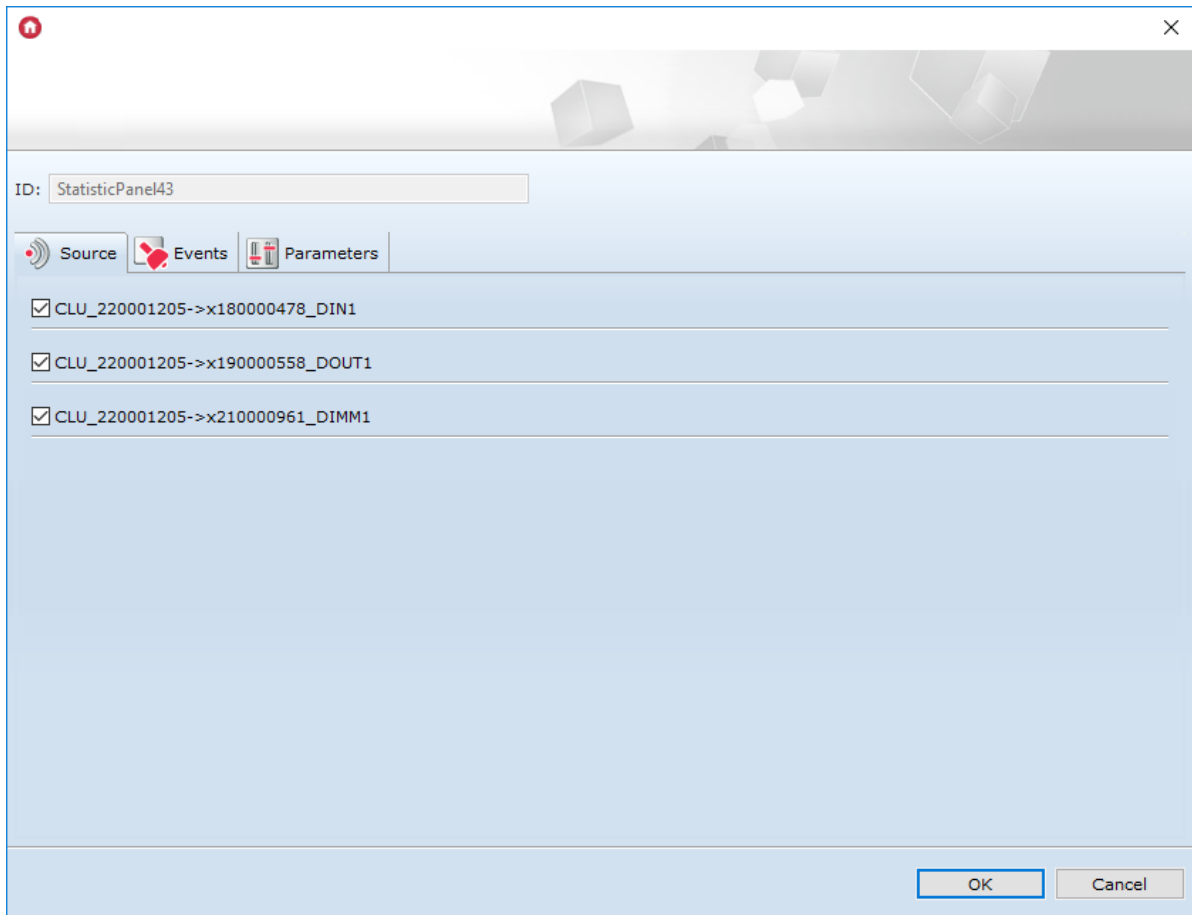
The previously created virtual object `Thermostat` is set as the thermostat panel source.



- **Statistics** - creates a panel for media measurement on the interface page in HM.  **Statistic**

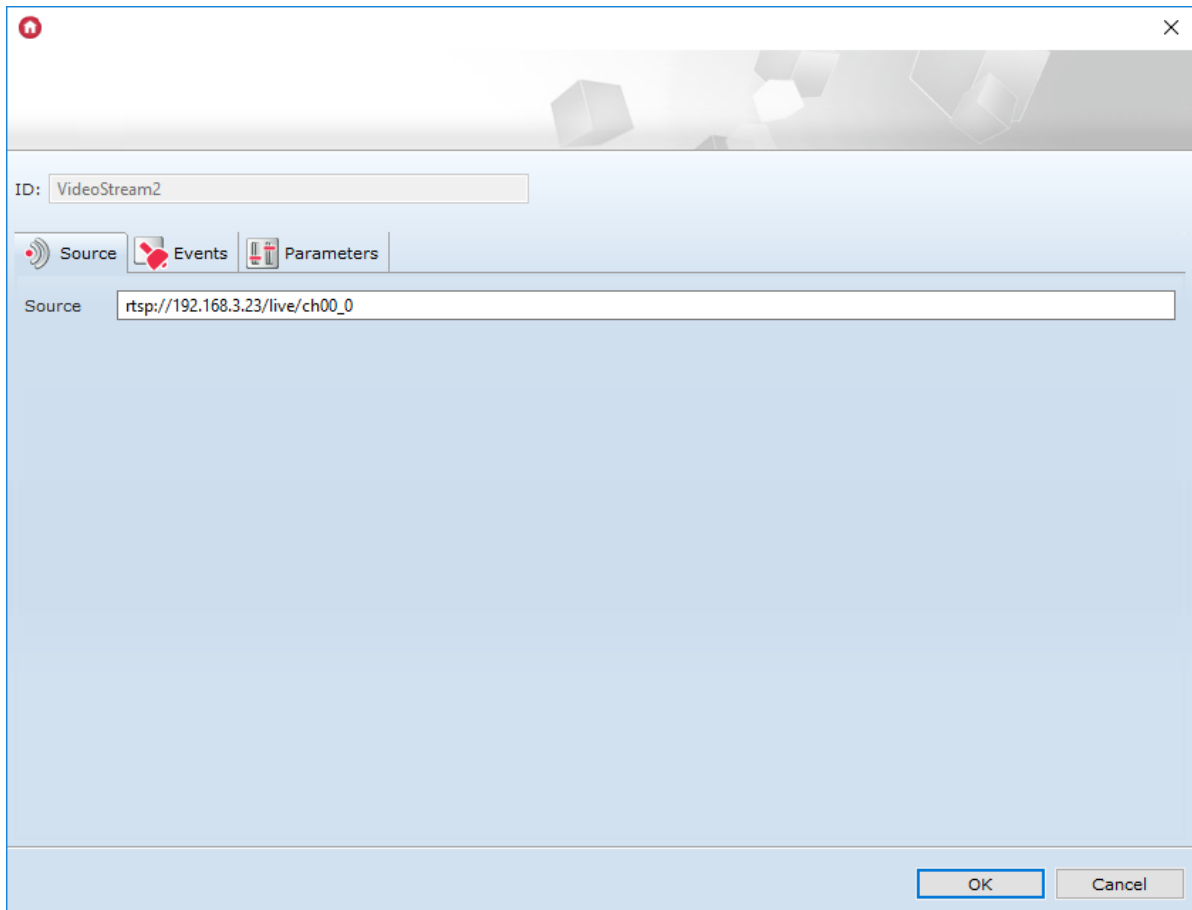
After dragging the panel to the interface page, select the objects for which the media measurement will be presented in the HM. The window will display only objects for which *Media Measurement* was previously attached.





- **Camera** - creates a panel for displaying the image from the IP camera on the defined space of the interface page in the HM.  **Camera**

The RTSP stream of the IP camera should be given as the source of the camera panel.

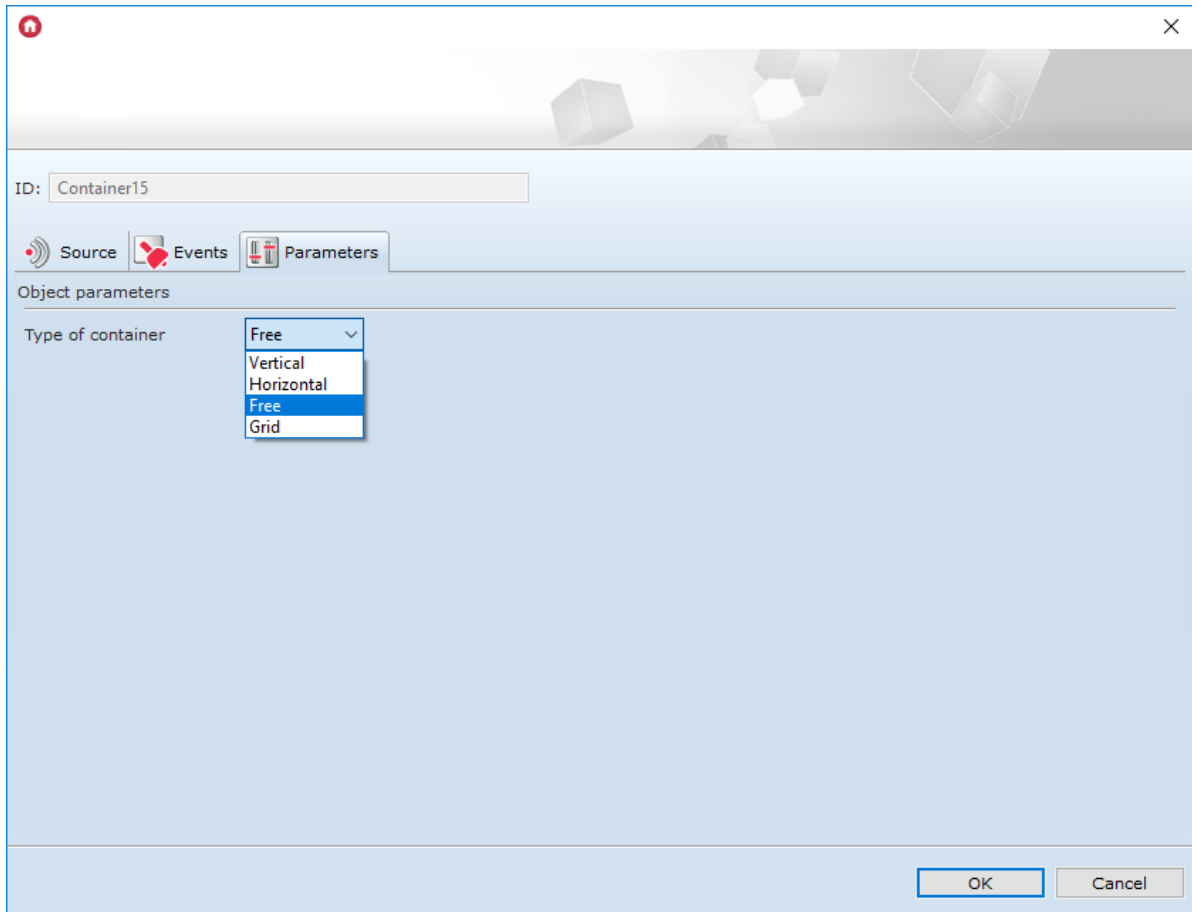


## 4.5. Containers

A container is objects compartment determining their arrangement in the workspace.

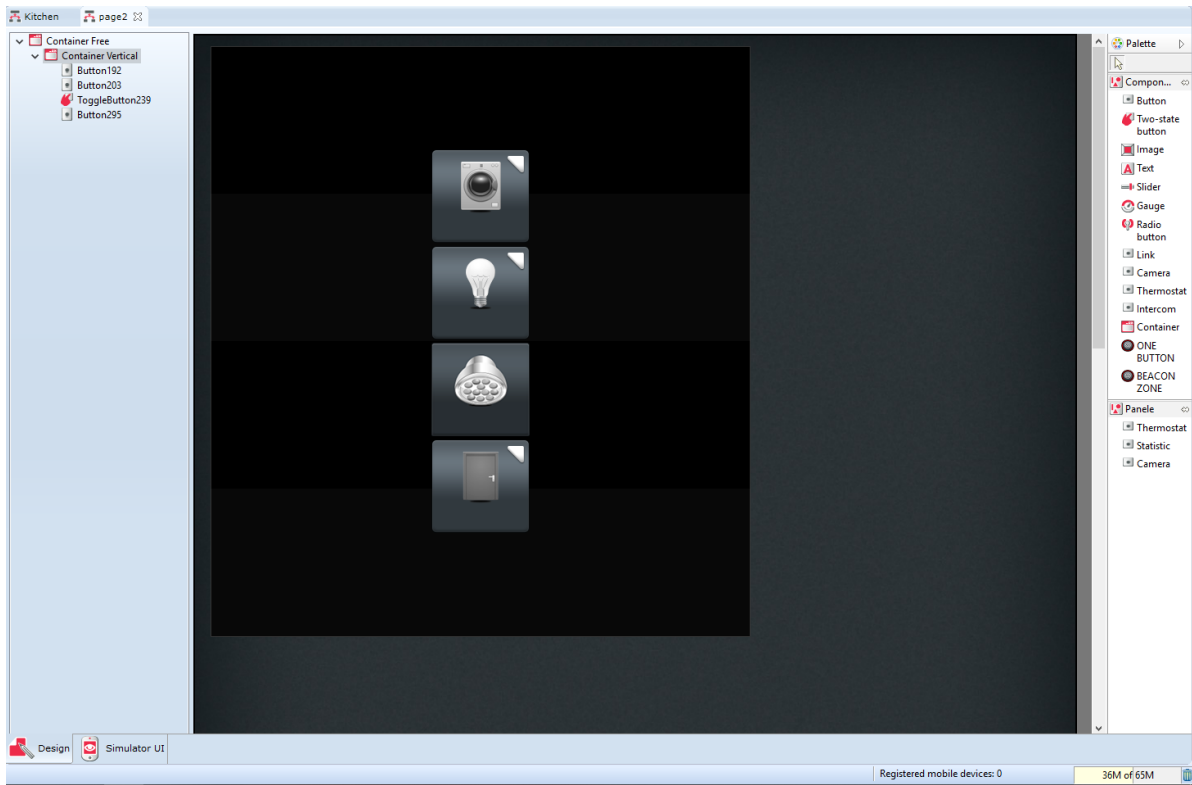
Objects within the workspace are arranged accordingly to the type of the selected container.

Container type can be changed in object parameters of the container. Parameters window opens after double clicking container object on the first place in the objects list.

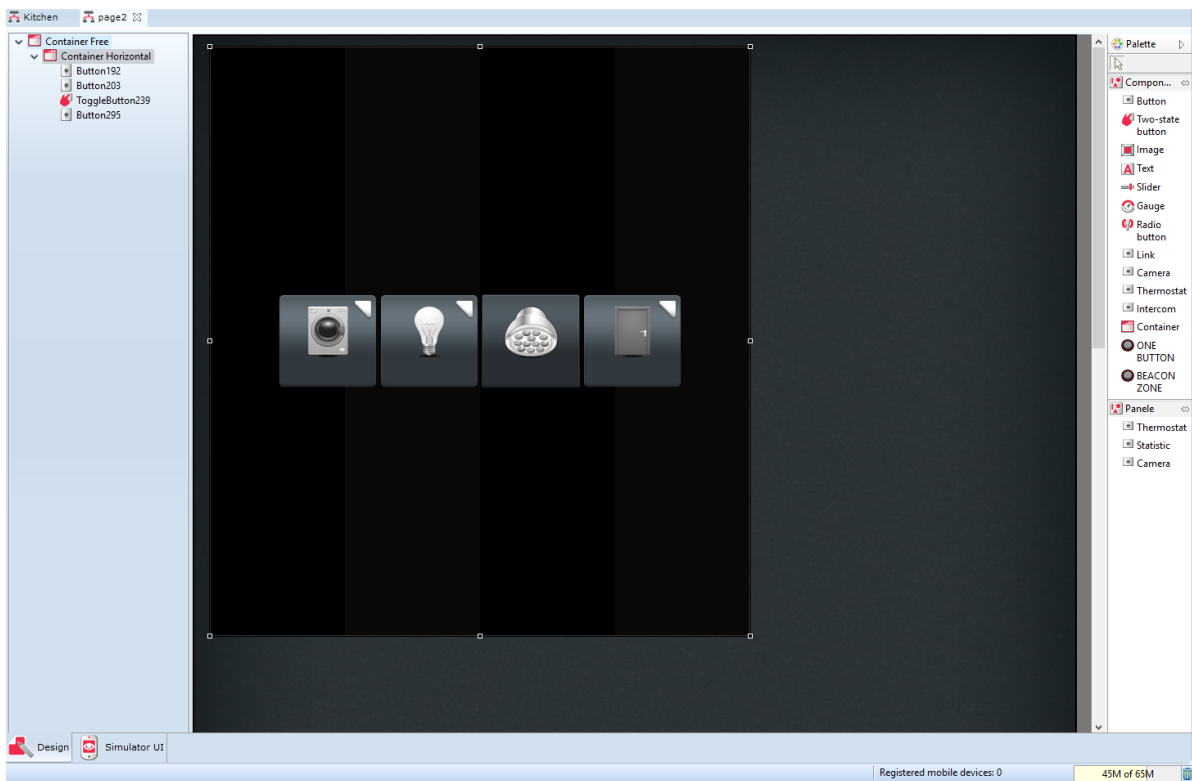


**There four types of containers:**

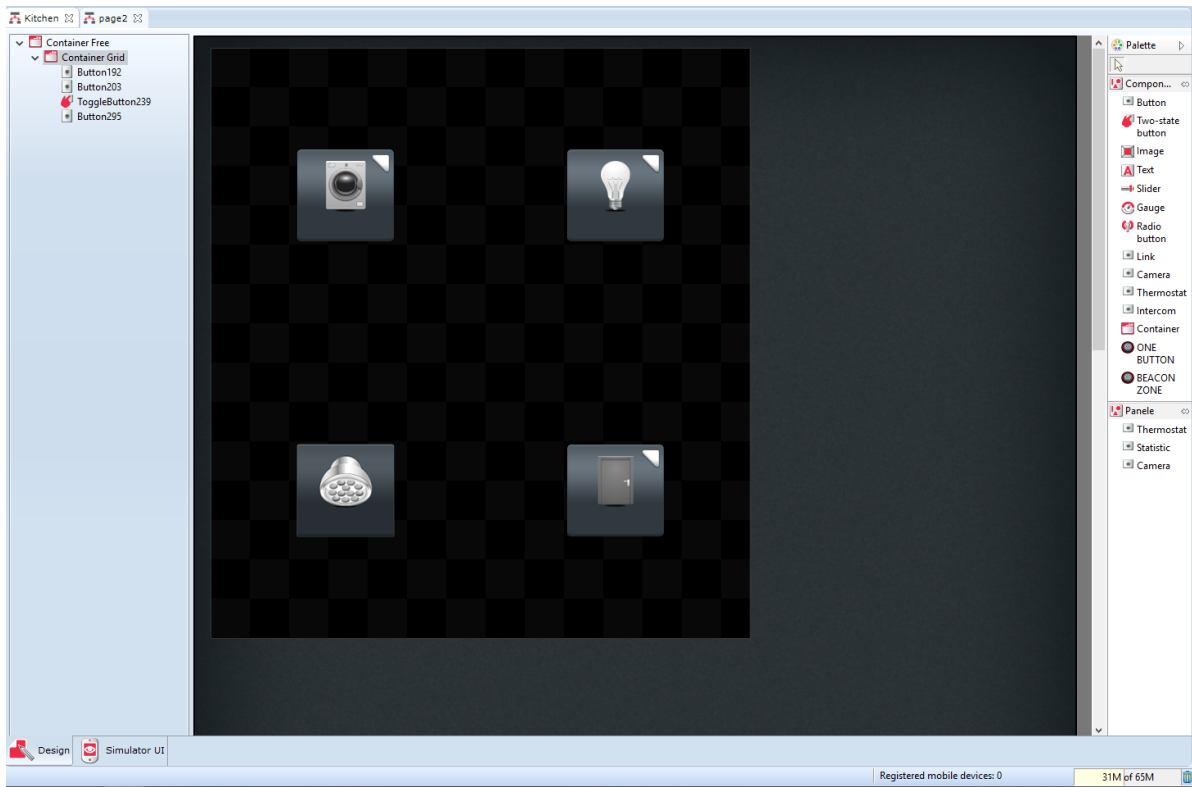
1. **Vertical** - the elements are arranged vertically in equal, automatically created sections.



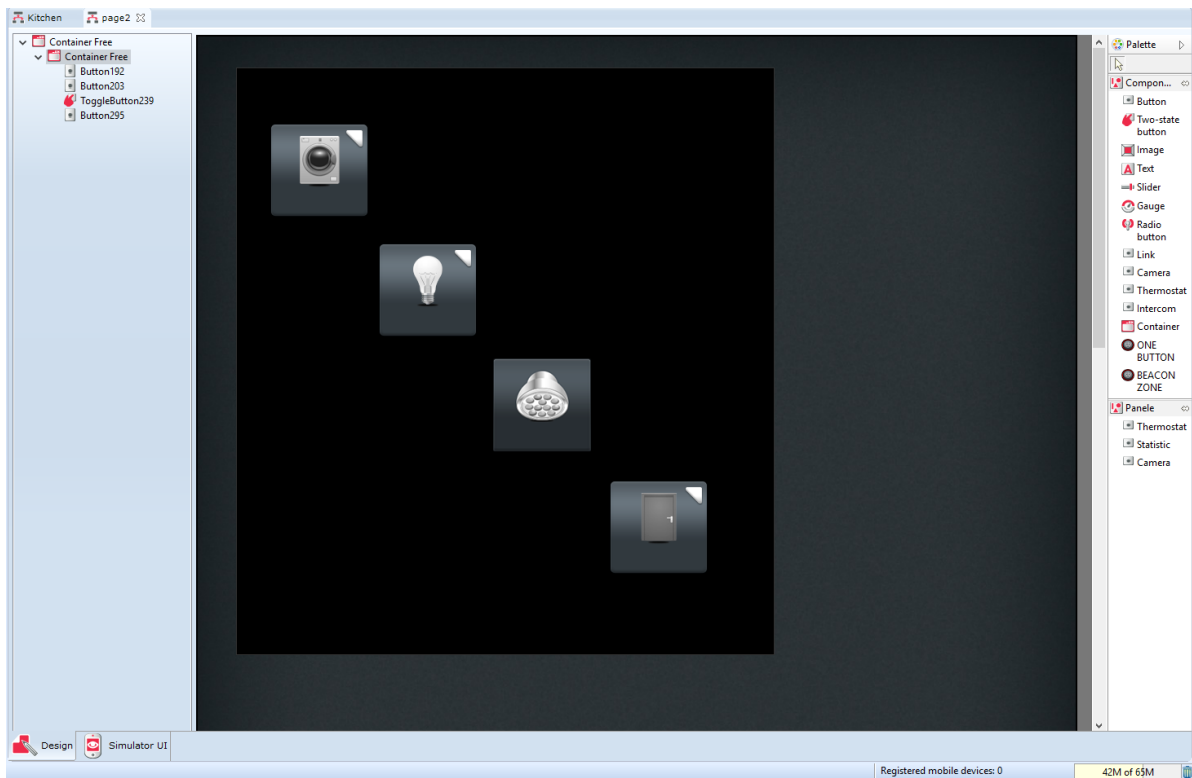
## 2. **Horizontal** - the elements are arranged in horizontal sections



## 3. **Net** - the elements are arranged in a symmetrical net.



4. **Random** - enables any arrangement of the elements within the whole container area

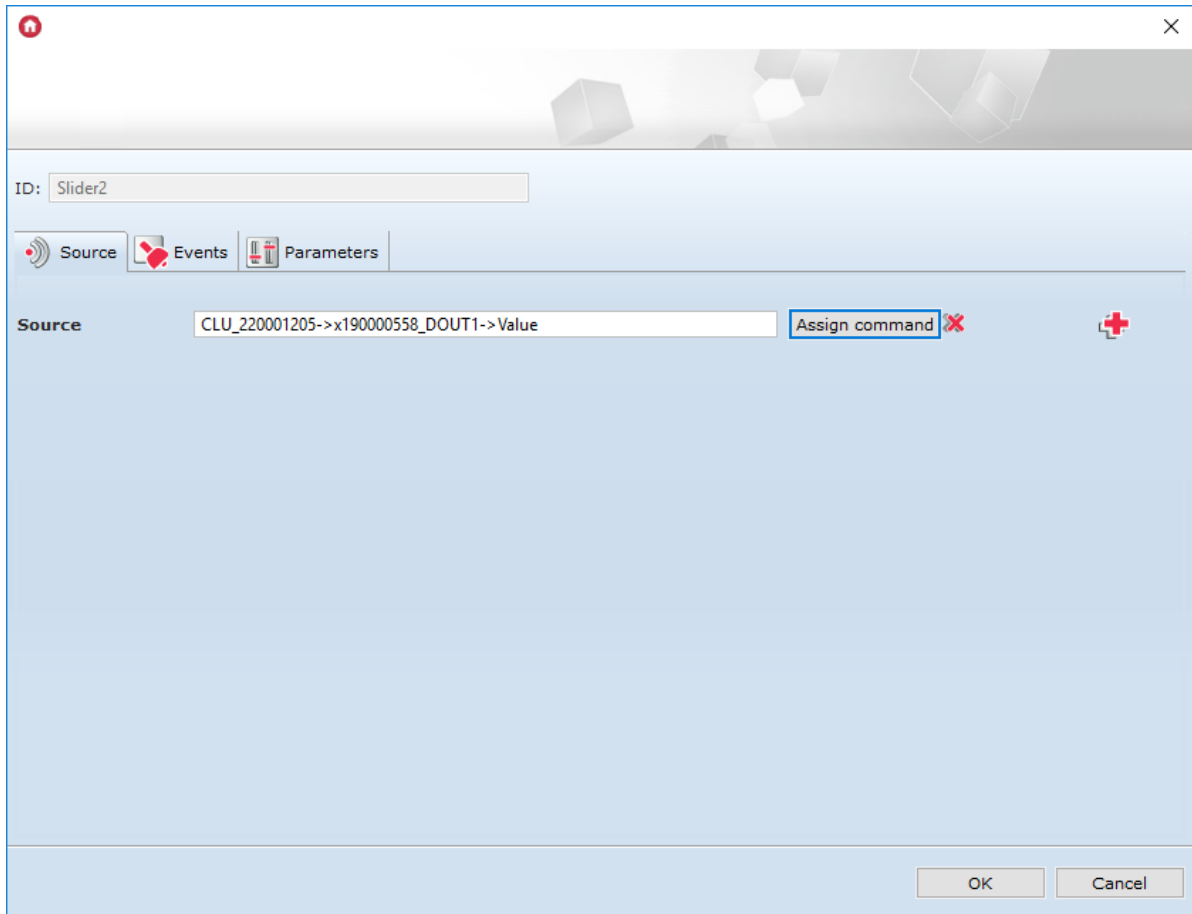


## 4.6. Adding components and connecting to the system objects

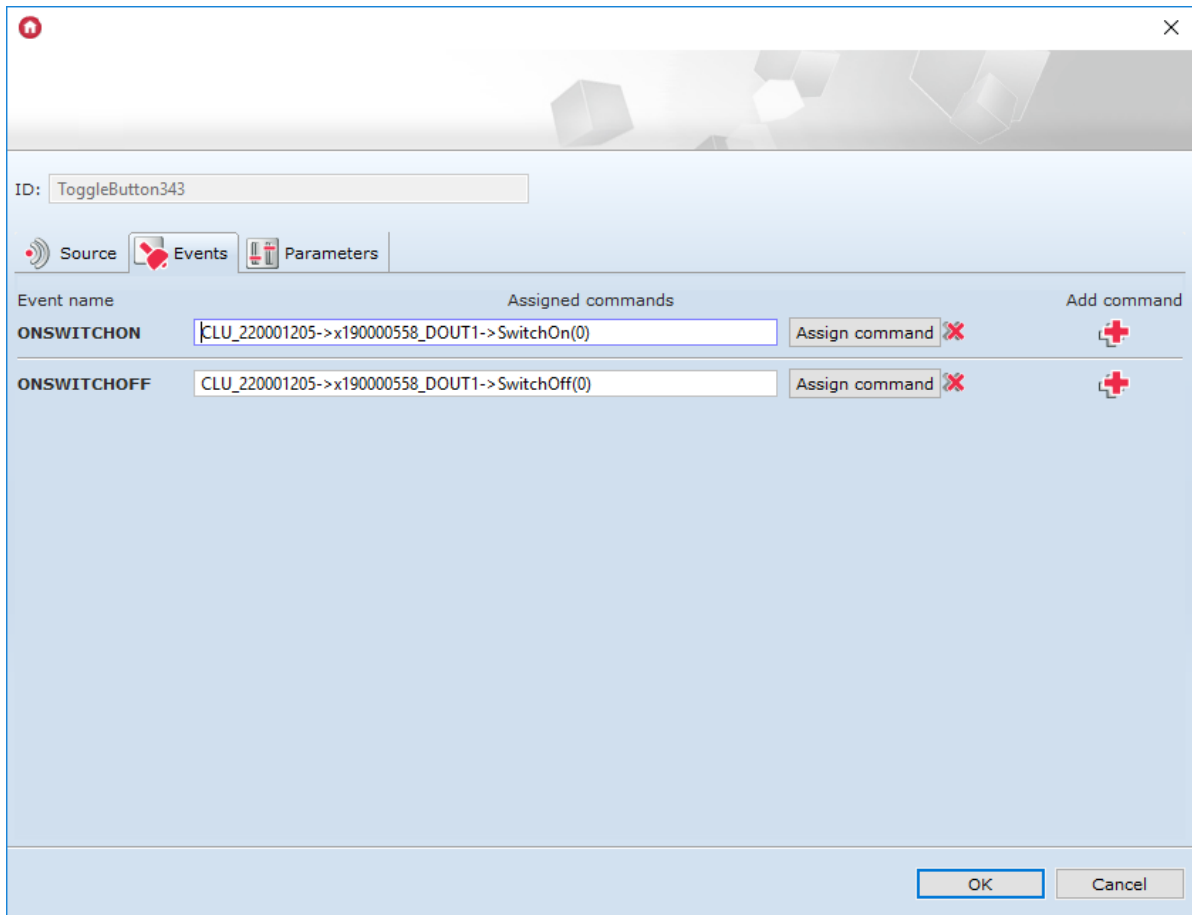
After selecting component from the list on the right and putting it in the main container, windows of its properties opens automatically. There are three tabs in the window ( `Source`, `Events`, and `Parameters` ), that need to be set as follows:

1. In the `Source` tab, select an object which value should be mirrored, and time of refreshing the value, e.g. if you put a slider controlling the dimmer in the interface, then the controlled dimmer

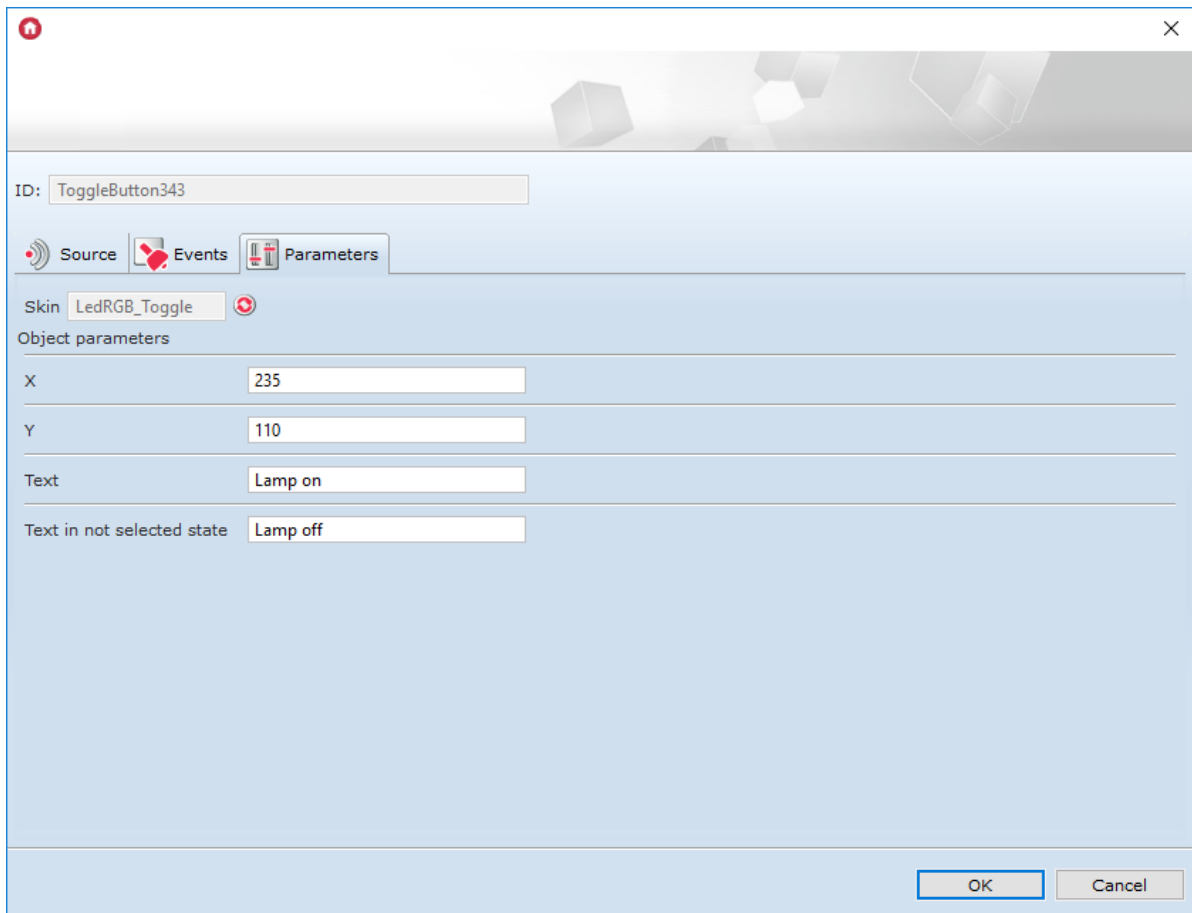
must be set as a source so the current value of light can be displayed on the smartphone.



2. **Events** tab is used for control objects, e.g. a button or a slider. In the tab, there are events applicable for certain type of objects, which needs to be connected to methods of the controlled objects.



3. In the `Parameters` tab there is data on displaying specific object in the interface. The user can change font and object size, and add edition skin.



**NOTE!** If the **\$value\$** command is entered in the **Text** field, it will display the current value of the **Value** feature of the object selected in the **Source** tab.

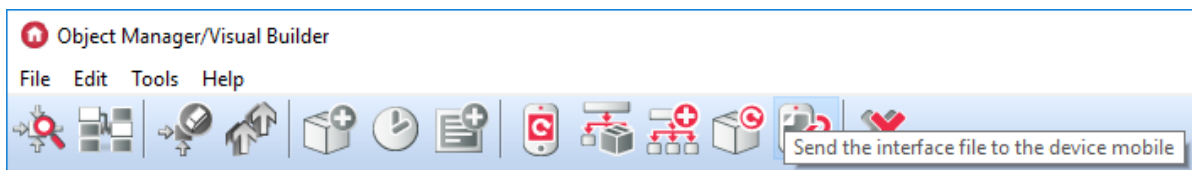
After and during interface creation the user has an option of checking its functionality and appearance. To do that, launch UI Simulator (second tab on the bottom of the page).

## 4.7. Sending interface to mobile device

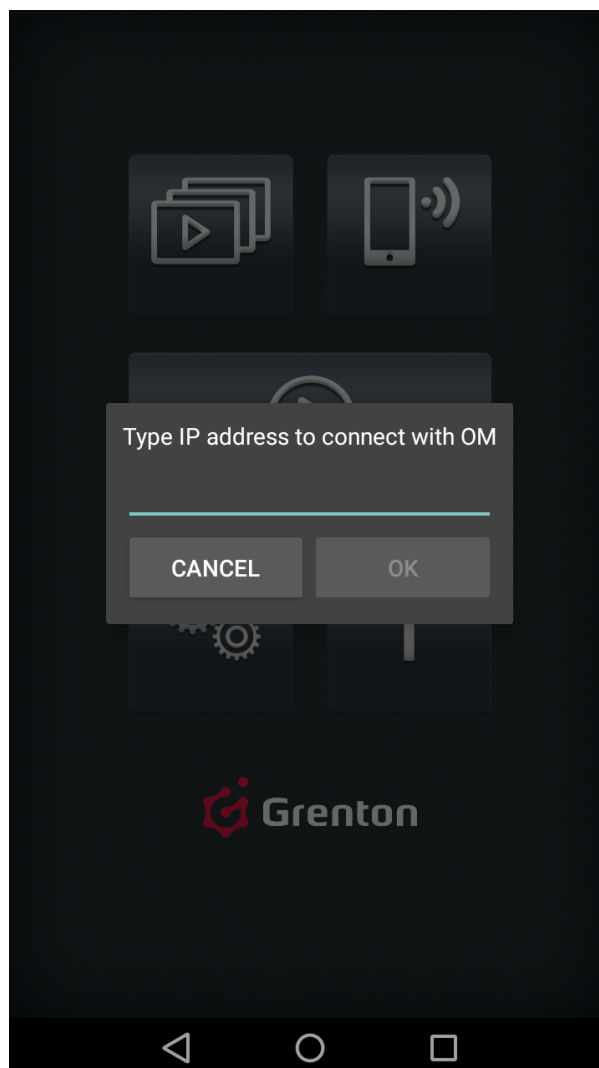
To enable system control by a mobile device, the created interface must be sent to GRENTON HOME MANAGER application installed on the device.

To do that:

- Select the interface you wish to send from the list of created interfaces in VISUAL BUILDER - the icon for sending the interface tool is in the main menu:



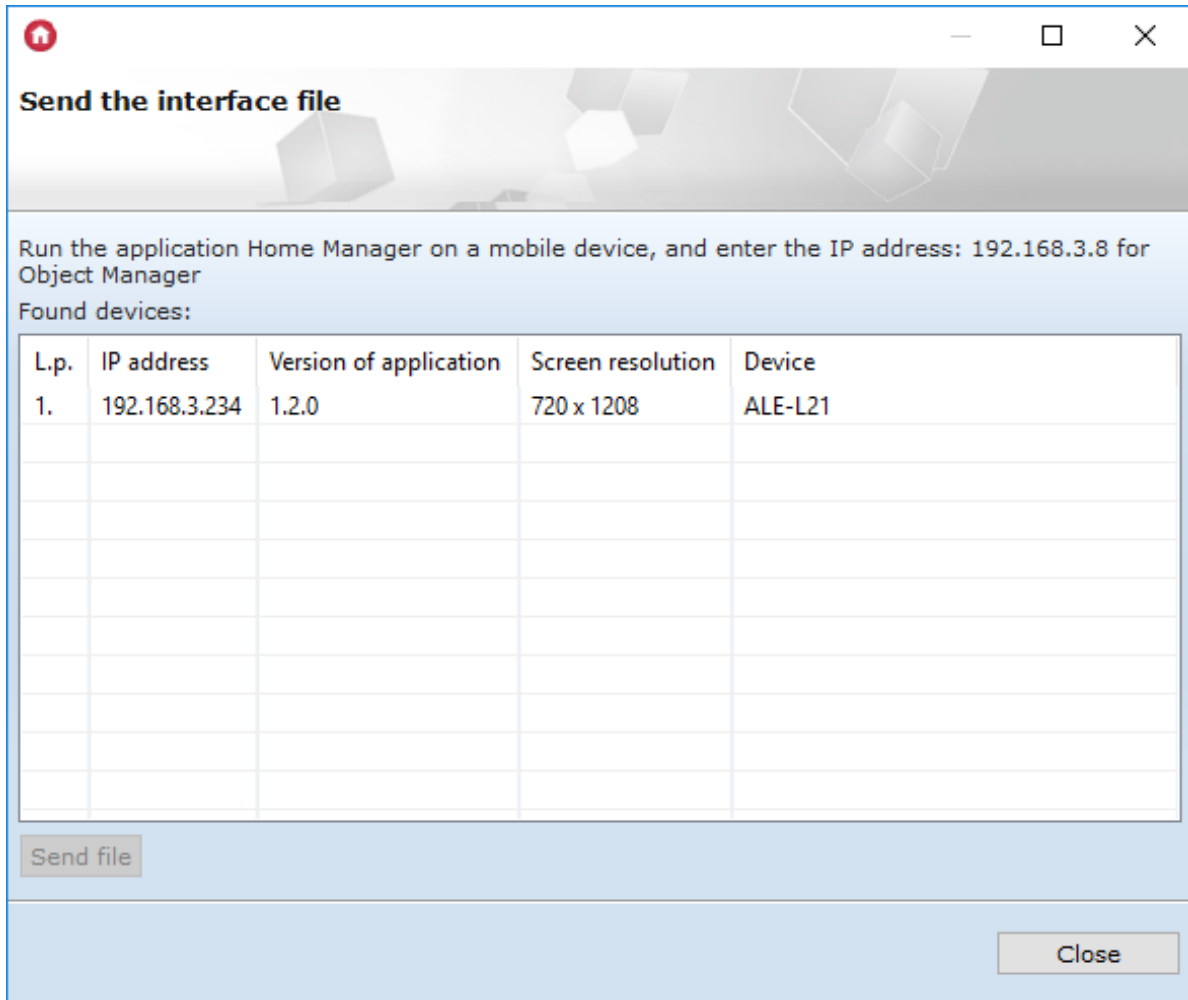
- On the mobile device, connect to the network in which the CLU is located (after displaying the send window in the Object Manager);
- In the open Home Manager application, select **Connect to OM from the main menu**;
- Enter the IP address of the Object Manager and choose *OK*:



- The mobile device will appear in the send window that was displayed in the Object Manager;

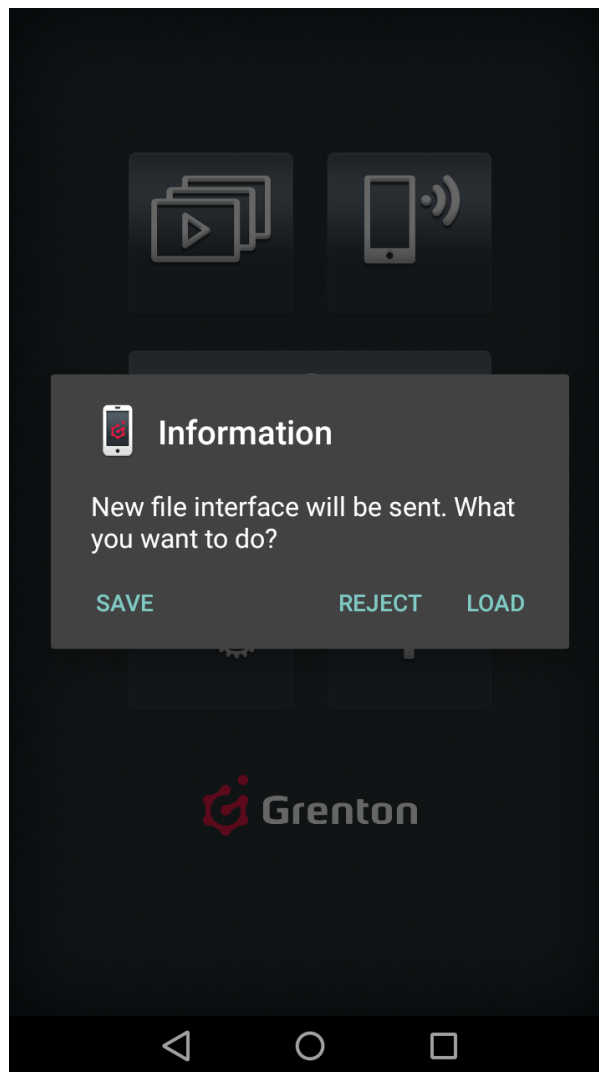
**NOTE!** The list displays the devices which have GRENTON HOME MANAGER application running, and have `connect to OM` option turned on in the application settings.

- Double click on its name or select and select *Send file*:



- In the mobile application, the window for accepting the interface will appear. Select *Save*:





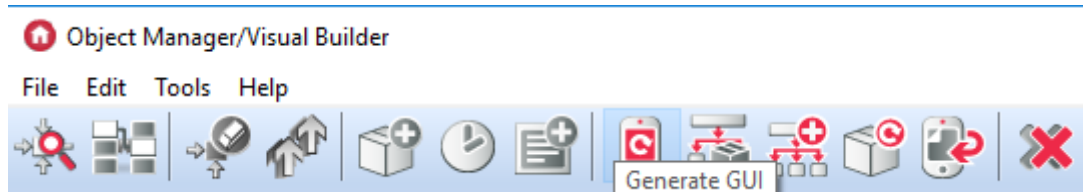
- The transfer status bar will appear on the screen. When finished, the information on the correct completion of the transfer will be displayed on the upper bar of the program.
- After sending the file with the interface to the mobile device, for the remote control to be possible, the uploaded interface must be loaded.

## 5. Automatic interface creation - GUI generator

**NOTE!** Support for Visual Builder functionality has been ended. The Home Manager interface creator will be removed in the next versions of Object Manager.

This function allows to quickly create interface through selection of objects which you want to control from all objects available in the system.

Start automatic user interface creation by launching GUI Generator. Generator icon is located in the objects menu:

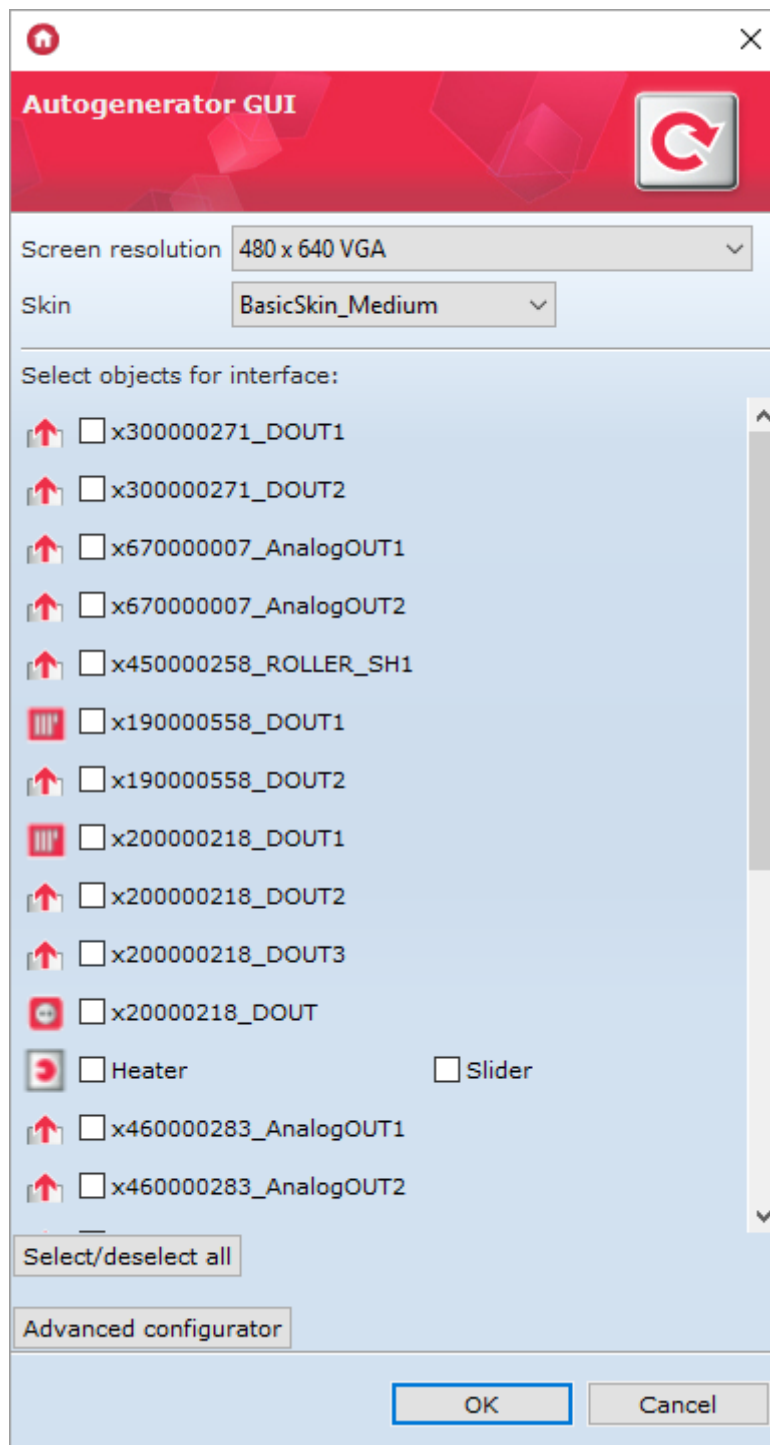


## 5.1. Creating an interface with available resolution

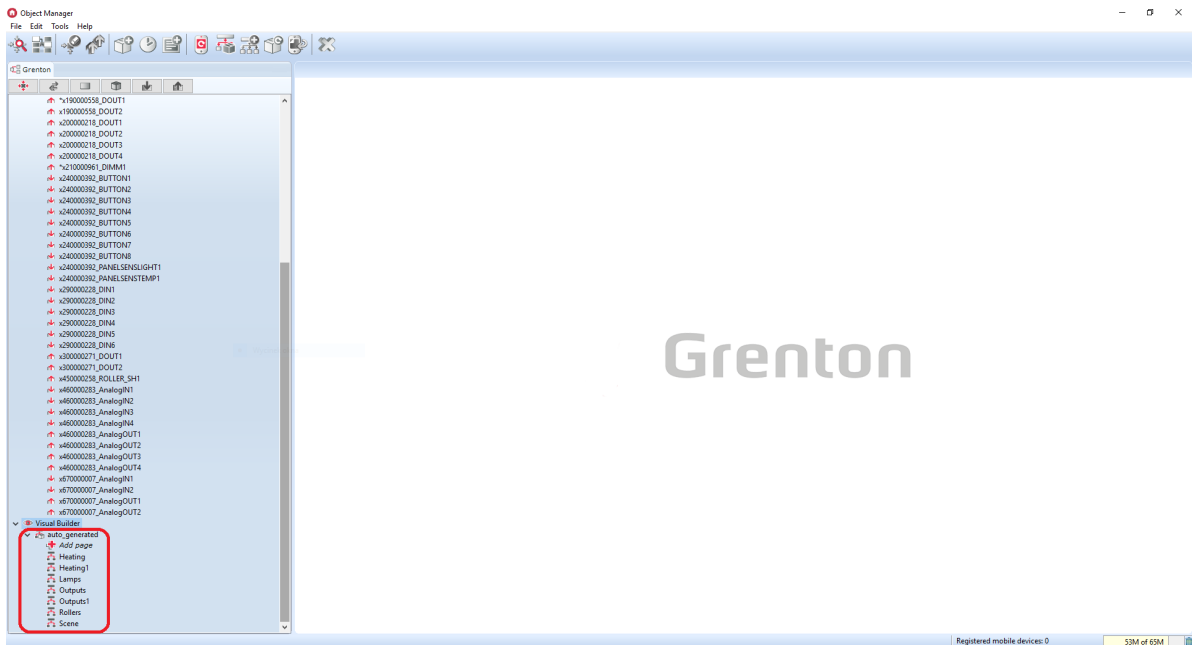
### A. Simple configurator

After clicking on the icon, the `GUI Autogenerator` window opens. It is a simple configurator in which you should choose:

- resolution of the mobile device
- a skin that determines the appearance of icons in the interface
- objects (from the list of objects) to be included in the created interface

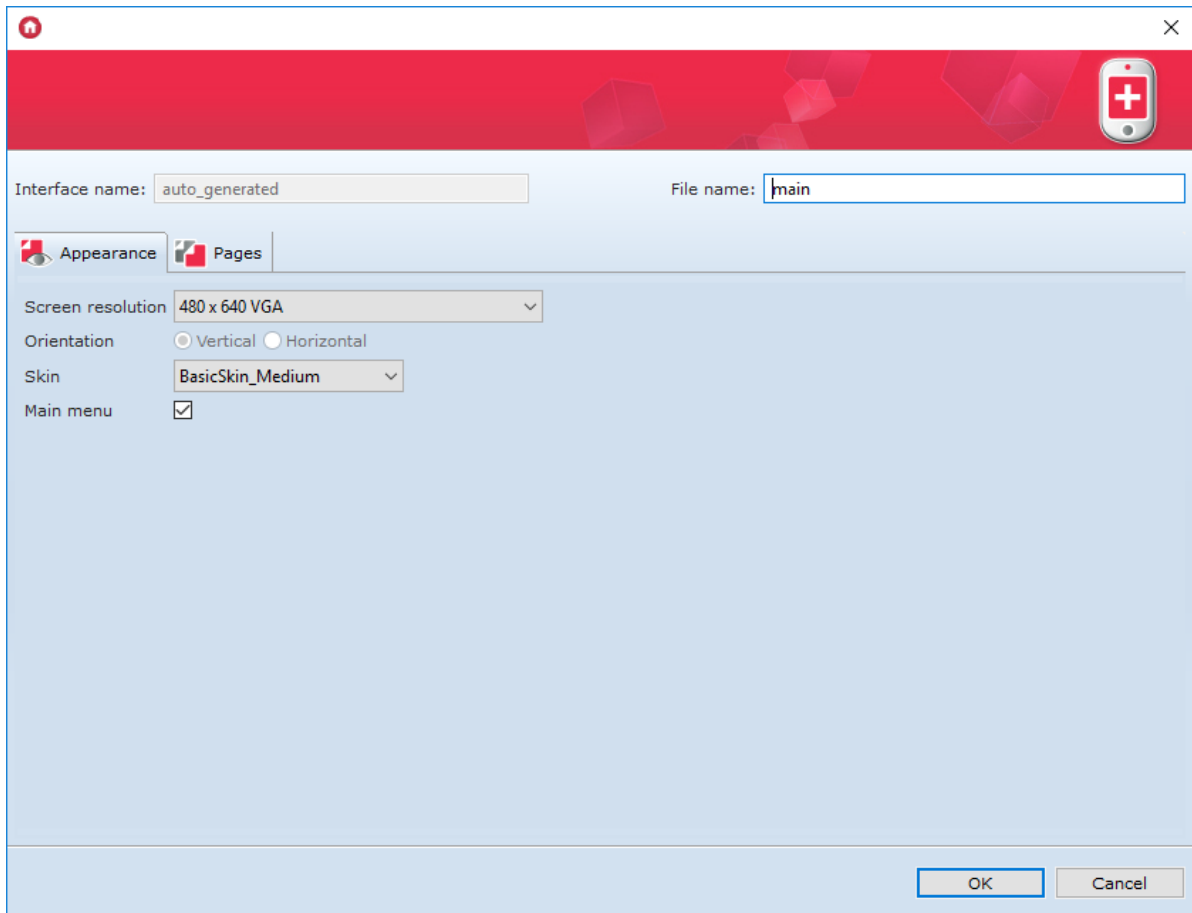


After selecting the objects of interest, click `OK`. As a result, the newly created pages appear in the list of objects (under the icon of the created interface) according to the following figure:



At any time, you can change the interface settings - just double click on its name, and a window will open with two tabs: `Appearance` and `Pages`.

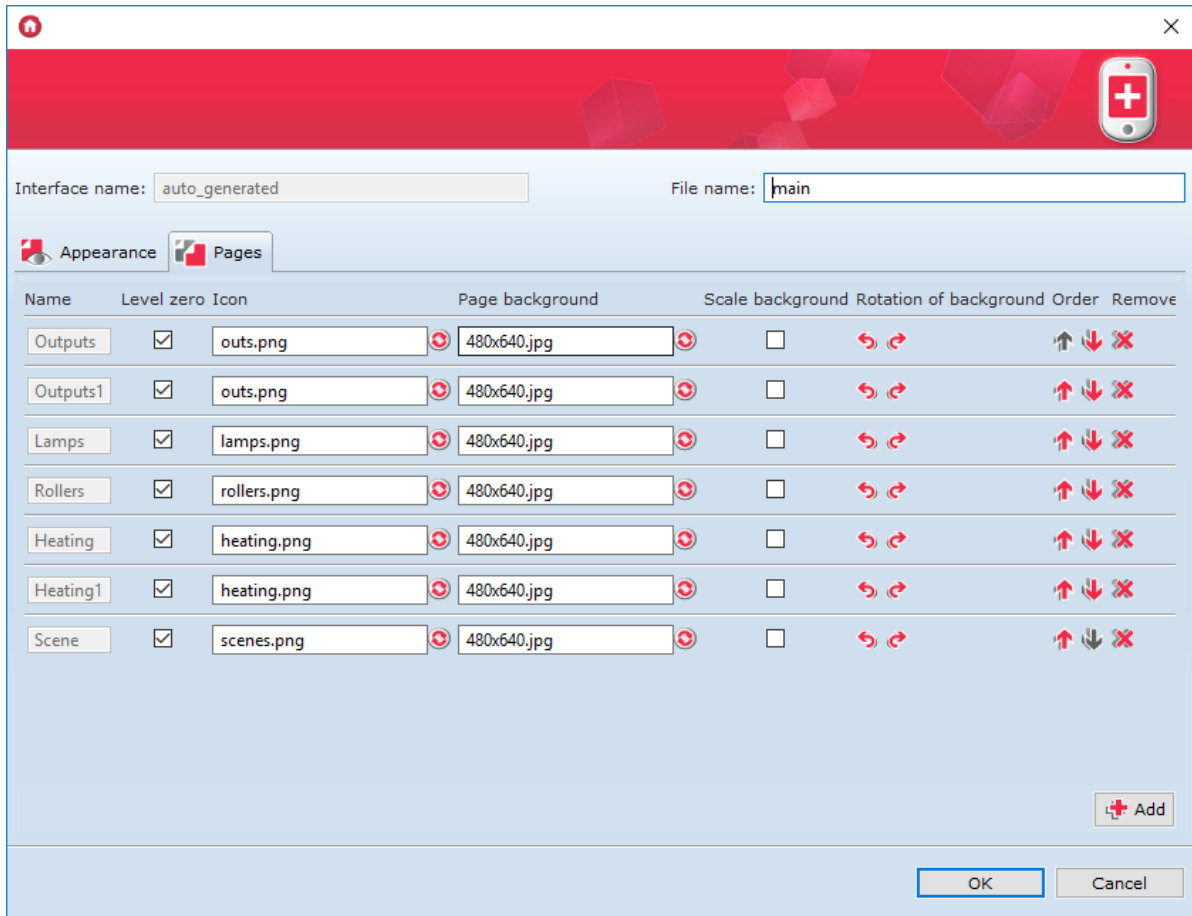
In the `Appearance` tab, the user can select skins visible in the interface. In this view there is also a field *Main menu*. After selecting it, a menu will be created containing all available and selected pages.



The `Pages` tab contains a list of pages created and allows you to change their parameters, such as:

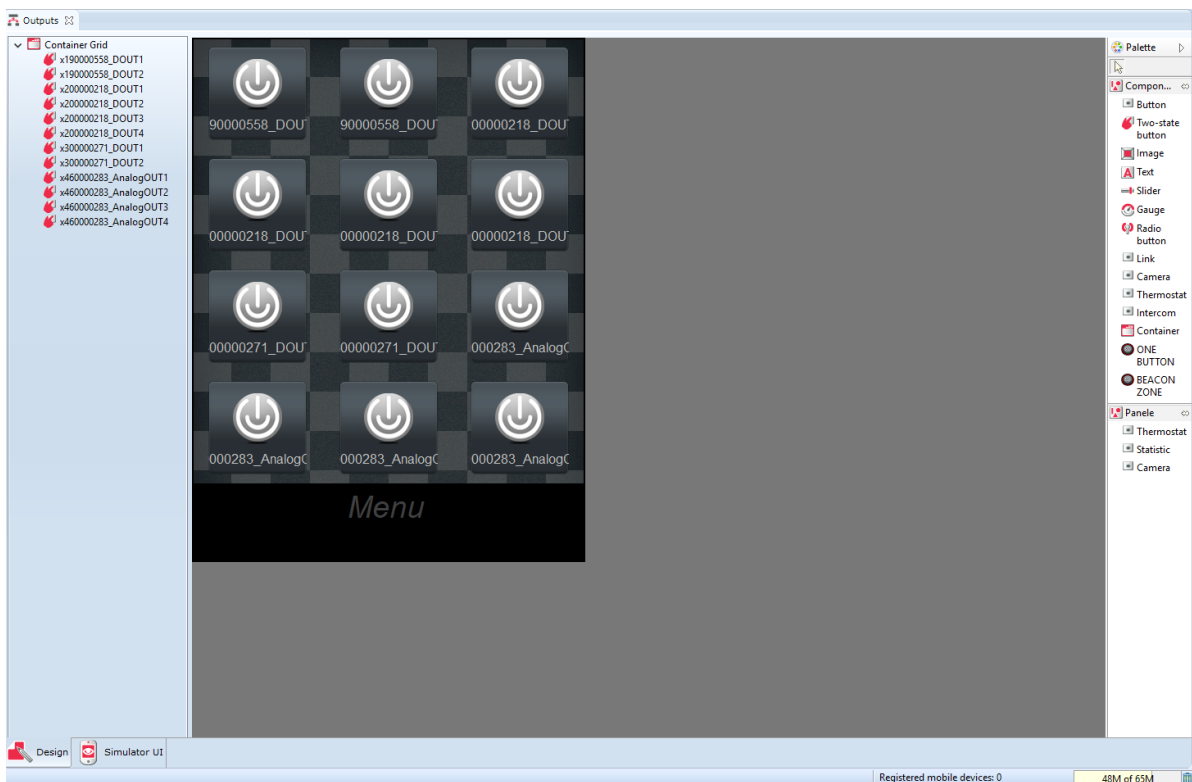
- **Level zero** - whether or not the page should be displayed in the menu;
- **Icon** - icon displayed in the menu (by default, it is icon from the selected skin);
- **Background** - background of the displayed page. By default, background from the selected skin is displayed, but the user can define their own background;
- **Scale background** - match the selected resolution to the resolution of the mobile device;

- **Background rotation** - change of the background orientation;
- **Order** - set the order in which the pages are displayed in the menu;
- **Delete** - complete deletion of the page from the interface.



The user also has the option of making changes to the generated pages. Double-clicking on the page icon will open the edit sheet, containing the two tabs `Design` and `Simulator`.

`Design` tab - displays the workspace contained in the container and allows you to edit the created page.



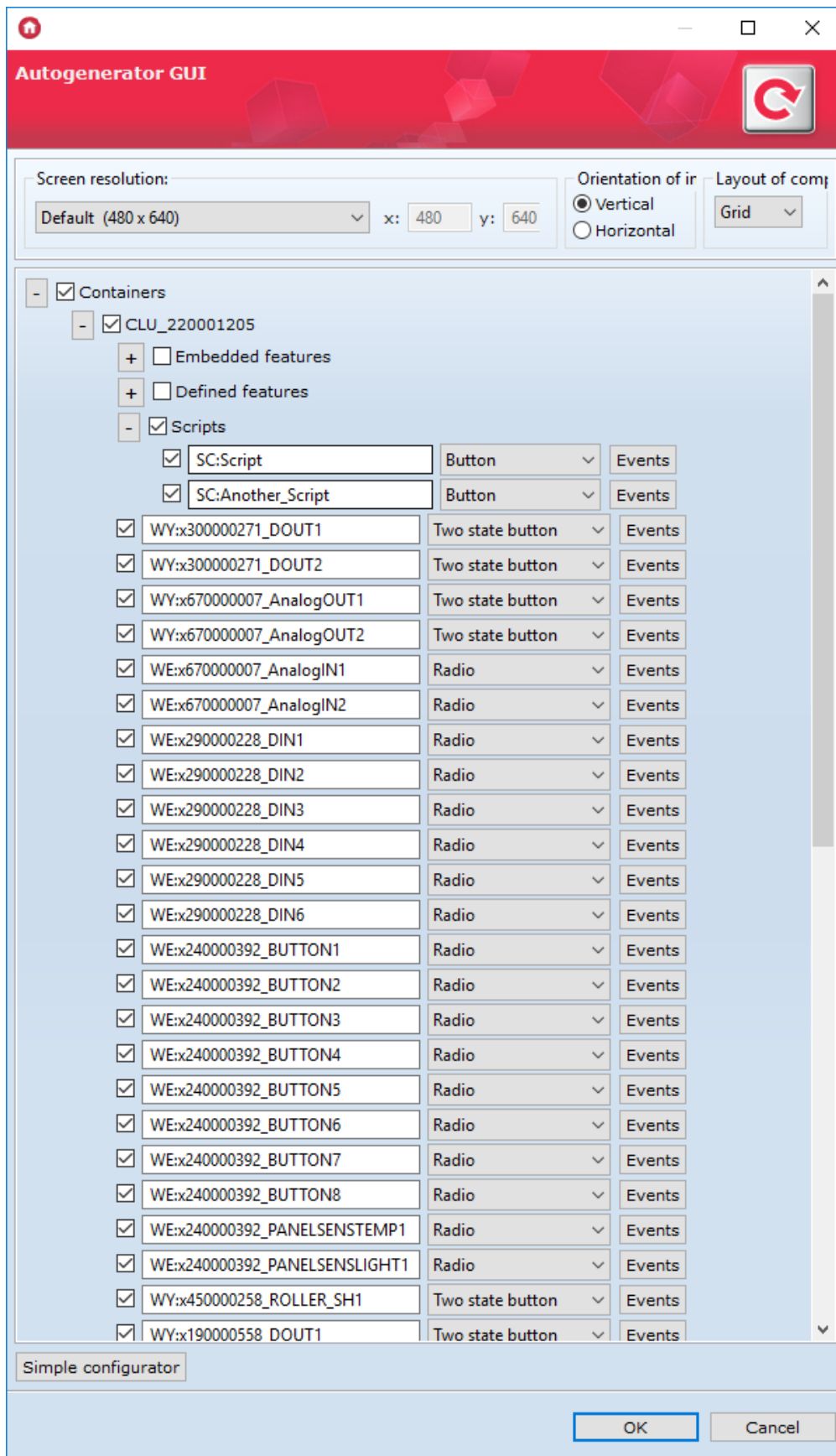
The `Simulator` tab - gives the user the ability to check the appearance and operation of the created interface from the computer screen (before it is sent to the mobile device).



## B. Advanced configurator

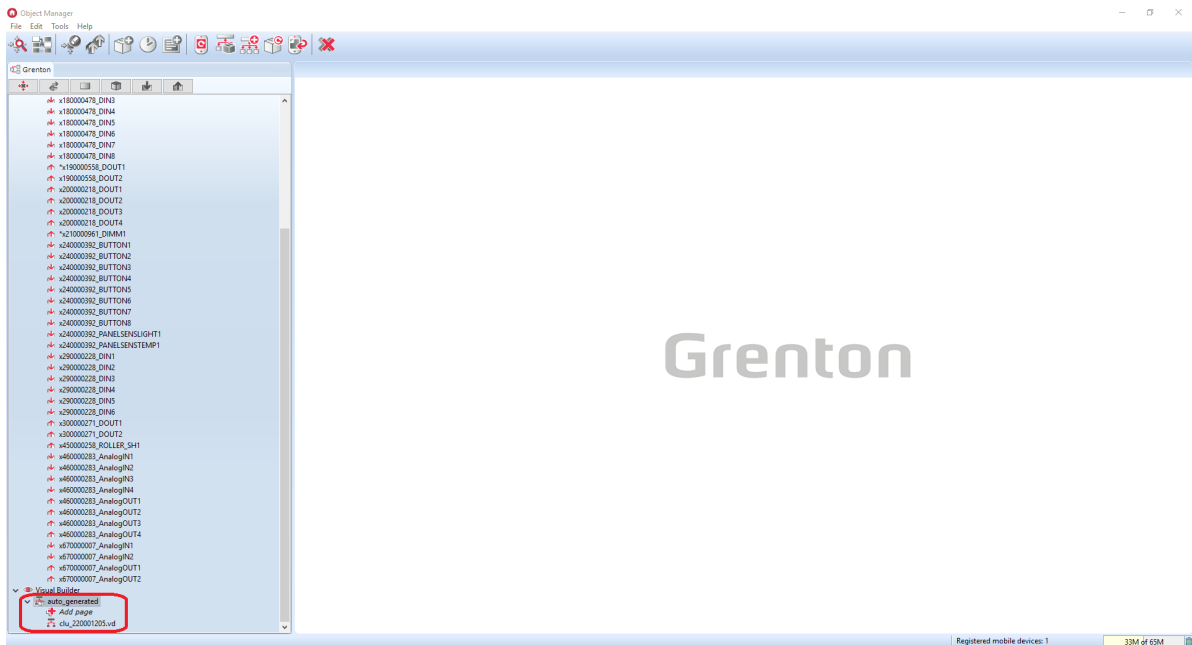
After clicking the `Generate GUI` icon in the `Autogenerator` window, you can select the `Advanced configurator` option. Selecting this option will open a new window in which you should select:

- the resolution with which the mobile device is working;
- interface orientation (vertical or horizontal);
- arrangement of components (grid or list);
- objects and features (from the list of objects) to be included in the created interface;
- displayed icon and events for each object.



Then, after setting all parameters and pressing , the window of the created interface opens. The window, in addition to the name field of the created interface, contains two tabs:  and . Their functionalities are exactly the same as in the case of the simple configurator.

After setting all parameters in the created interface window and clicking , the newly created pages appear in the list of objects (under the icon of the created interface) as shown in the following figure:

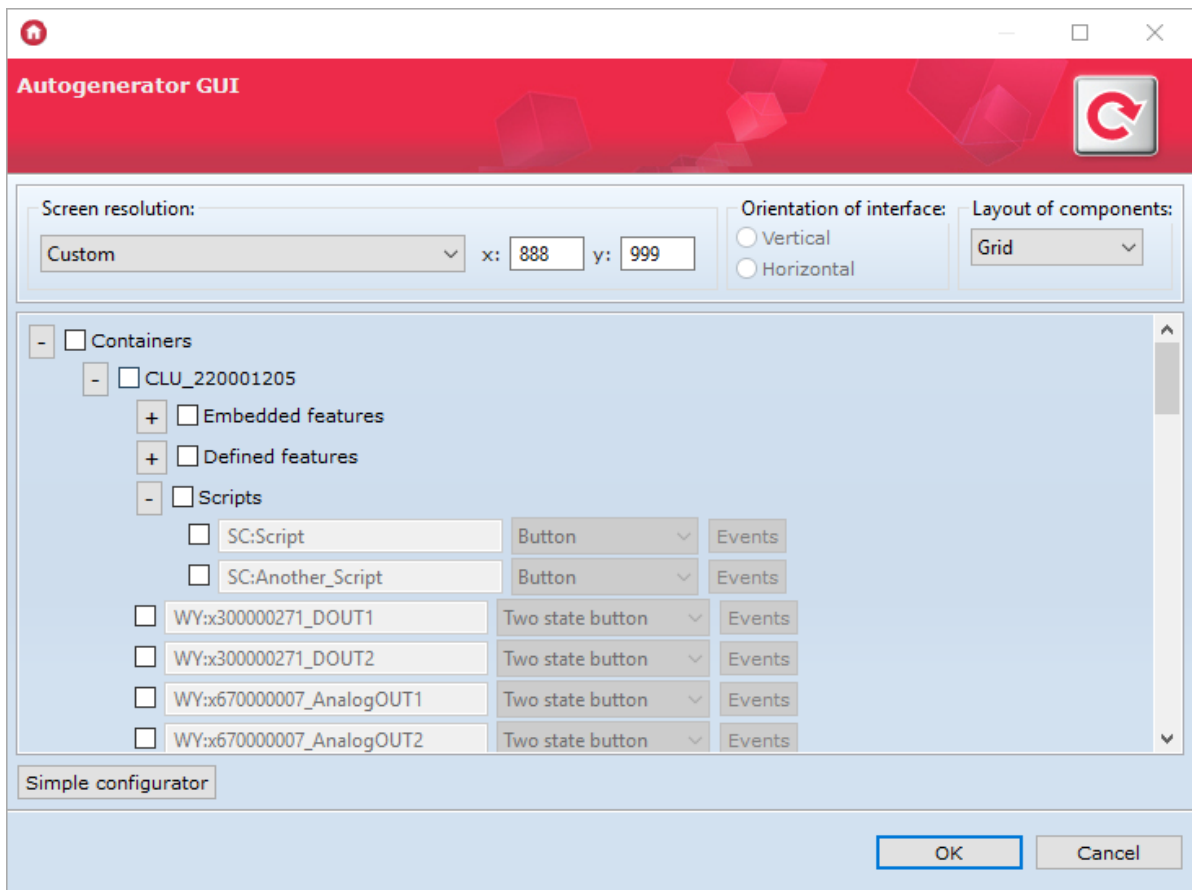


As with the simple configurator - the user has the option of making changes to the generated pages. Double-clicking on the page icon will open the edit sheet, containing the two tabs `Design` and `Simulator`.

## 5.2. Creating an interface with its own resolution

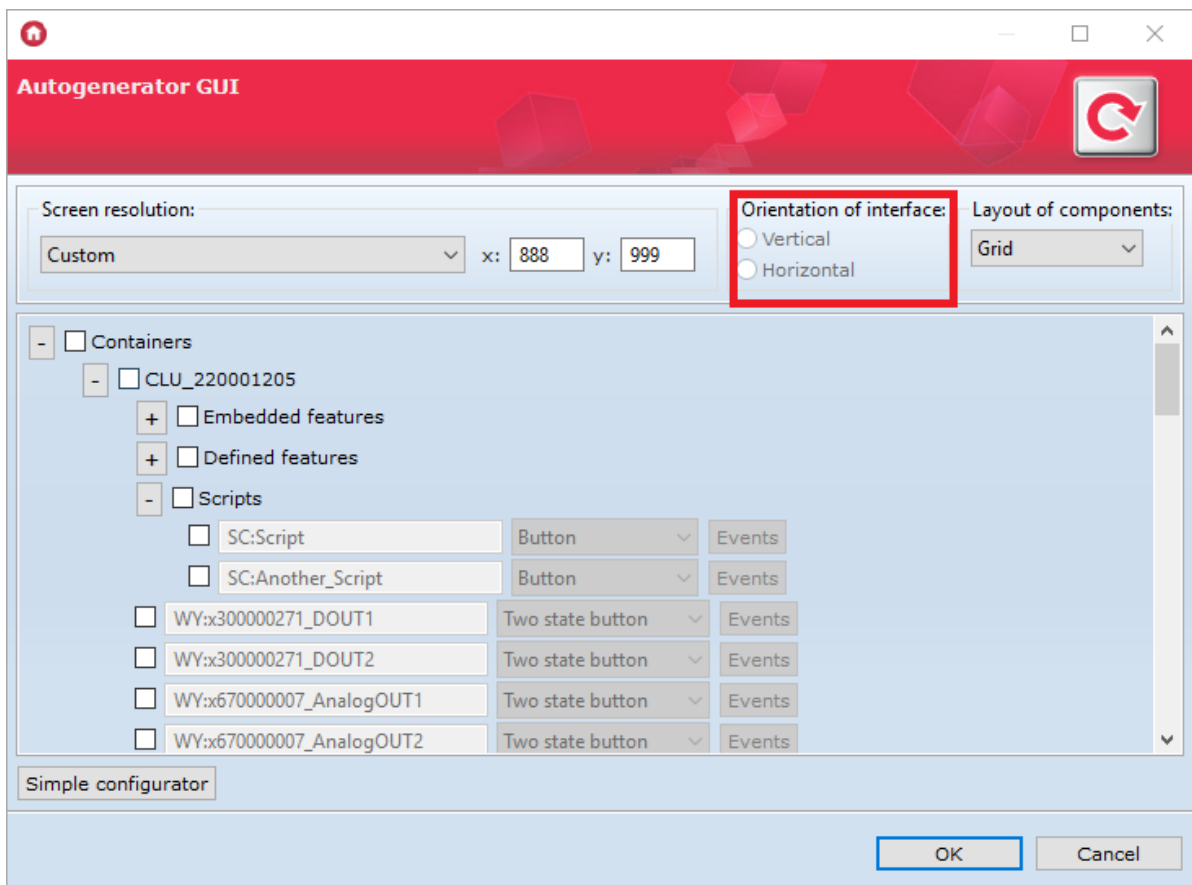
In the case of the advanced configurator, it is possible to create an interface with its own, selected resolution. To do this:

- Click on the `Generate GUI` icon in the top object window;
- Select the advanced configurator;
- In the window for selecting the resolution, select the option *Customize* and enter the dimensions of the interface;
- Select the remaining interface parameters;
- Accept the settings you have made.



### 5.3. Changing the orientation of the interface with its own resolution

Using the advanced configurator, the change of interface orientation does not take place in the `Autogenerator GUI` window.





- If you want to change the orientation of the interface with your own resolution, after creating it you have to:
  - Click twice on its name;
  - Go to the tab `Pages` ;
  - Delete all visible pages;
  - Go to the `Appearance` tab;
  - Select orientation - horizontal or vertical;
  - Go back to the tab `Pages` ;
  - Add pages to the interface;
  - Accept changes by clicking OK;
  - Send the interface to the mobile device.

## 6. Video intercom configuration

**NOTE!** Support for Visual Builder functionality has been ended. The Home Manager interface creator will be removed in the next versions of Object Manager.

### 6.1. Connection and configuration of a video intercom

The configuration of a video intercom with the Grenton system is possible for devices connected to a common network (*LAN*) or those using remote access to a given network, enabling the use of *rtsp* stream of IP camera built into the device. Two or more accounts on the *SIP* server are needed for correct video intercom configuration.

An exemplary configuration was made on the intercom *Akuvox R26*.

**NOTE!** The Video intercom panel is available for Object Manager version 1.2.0.180202 and higher.

#### A. Connection of a video intercom

You should:

- Connect the video intercom to the power supply;
- Connect the video intercom with the RJ45 network cable to the router.

#### B. Camera configuration

The video door entry panel in the Grenton Home Manager application uses the visualization of the camera embedded in the device - if you want to have access to the camera image, you should issue the appropriate port in the network settings. In order to configure the port, log in to the router settings using its IP address in the web browser, make appropriate changes, and then save the settings:

- Enter redirect settings <sup>1</sup> ;
- Find the port settings;
- Set the triggering and forwarding port to **554** <sup>2</sup> and the triggering and relaying protocol on **TCP**;
- Save settings;

**NOTE!** Please note that in order to allow remote connection of the application, it is necessary to set the port **1234** in the **UDP** protocol.

- Finally, go to the list of currently connected devices to the network and save the IP address of the video doorphone - it will be needed when configuring the *SIP* server.

### C. SIP configuration:

- To create a video intercom configuration you need at least two *SIP* accounts;
- Using the browser, log in to the video intercom <sup>3</sup>;
- It is necessary to find the *SIP* account settings <sup>4</sup>;
- Then select one of the available accounts (e.g. **Account\_1**) and set its status to activated (`enabled`);
- In the next step, set the *SIP* account number / name and password;
- After that, it is necessary to enter the *SIP* server settings (`Server IP`, `Port`, `Registration Period`) - these settings should appear when creating accounts;
- Then find the settings of the codecs used in the operation and activate the *PCMU* type codecs;
- At the end it is necessary to find the settings of the Intercom, where you must configure the number / name of the customer to which the video door phone should be called (second account *SIP*) and set (if possible) the device's behavior when the connection is missed.

**NOTE!** If in the setting of Intercom, it is necessary to select one account from several configurable, select the previously selected one - in the example **Account\_1**!

## 6.2. Creation and configuration of the application interface


### A. Adding a door intercom to the application interface in the Object Manager

In order to add a video intercom to the interface:

- From the main menu, click *Add interface*:



- Configure interface settings - choose: resolution, name, skin, add at least one page;
- To the created page - from the component palette - add the button *Intercom*:

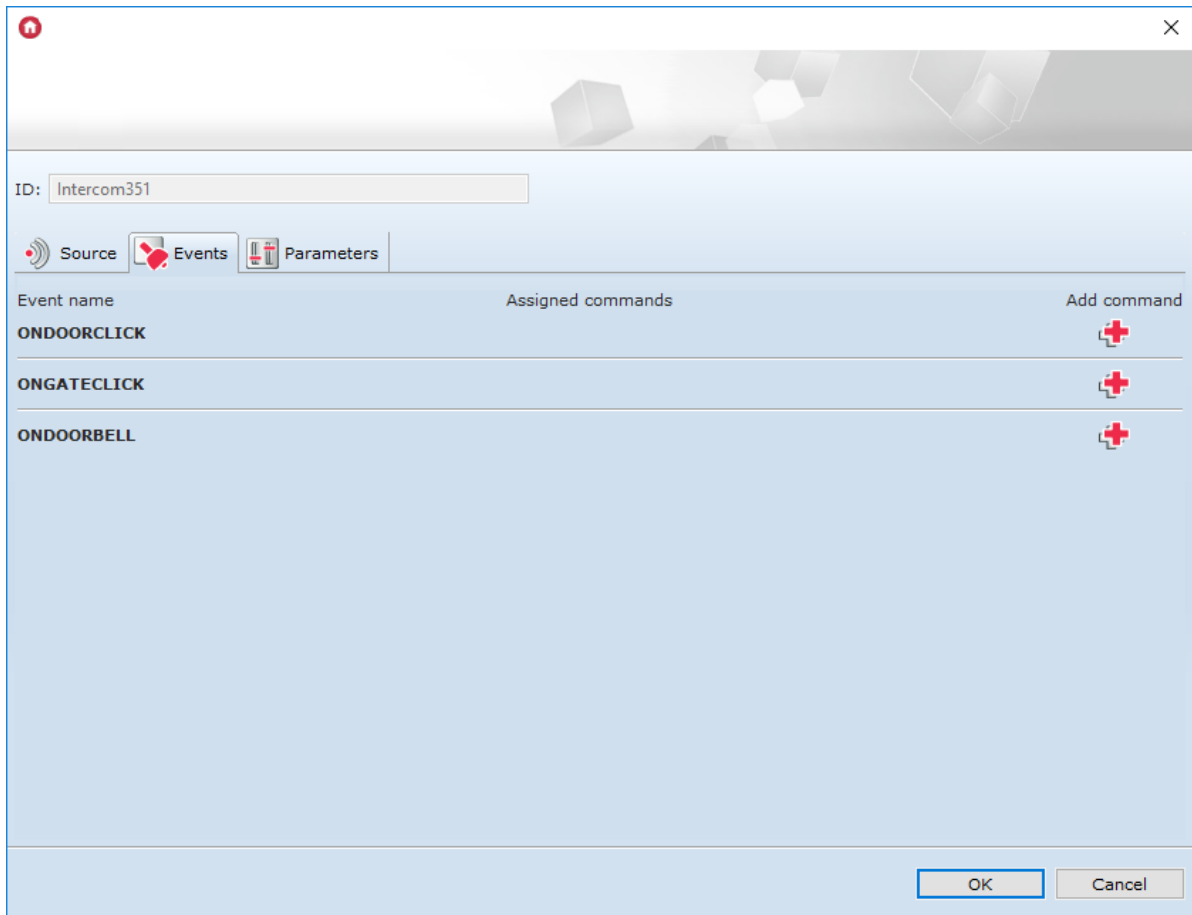
 **Intercom**

- In the window that will open after adding the button, set the parameters of the video intercom:
  - **Source** - stream *rtsp* found in the settings of the video intercom or its documentation;
  - **IP address** - IP address of the video door phone (previously saved when making its configuration);
  - **Account** - number / account name *SIP* entered first in the video intercom settings - account from which calls will be made (selected in item 3 of the chapter "Connection and

configuration of a video intercom”):

The screenshot shows a configuration window for a video intercom. The window has a title bar with a red home icon and a close button. The main area is light blue. At the top, there's a header with a decorative background of cubes. Below the header, there's a text field labeled 'ID:' containing 'Intercom351'. Underneath, there are three tabs: 'Source' (selected), 'Events', and 'Parameters'. The 'Source' tab contains three input fields: 'Stream source', 'IP address', and 'Account'. At the bottom right, there are 'OK' and 'Cancel' buttons.

- Go to the tab *Events*:
  - To the `OnDoorClick` event assign a method to be called after pressing the wicket opening button in the doorphone panel in the Home Manager application;
  - Associate the `OnGateClick` event with the method to be called after pressing the button for opening the entrance gate in the intercom panel in the Home Manager application;
  - To assign the `OnDoorBell` event to the method or script to be executed when the call is made - when the bell on the intercom is pressed:



- Click OK;
- Send the interface to the mobile device - [look up VIII.4.7.](#)

## B. Home Manager application configuration

In order to carry out the configuration:

- Open the Home Manager application;
- Select *Settings* from the main menu (gearstick pictogram);
- From section *Intercom* select *SIP configuration* <sup>5</sup> ;
- In the settings specify:
  - **Server address** - server IP address *SIP* on which the accounts were created;
  - **User name** - number / account name *SIP* to which calls will be made - specified in the entry phone settings, as the destination account for receiving calls (selected in item 3 of the chapter "Connecting and configuring a video intercom");
  - **Password** - password for the above *SIP* account, to which connections from the interphone will be made:

Server address

sip.freeconet.pl

Username

grenton

Password

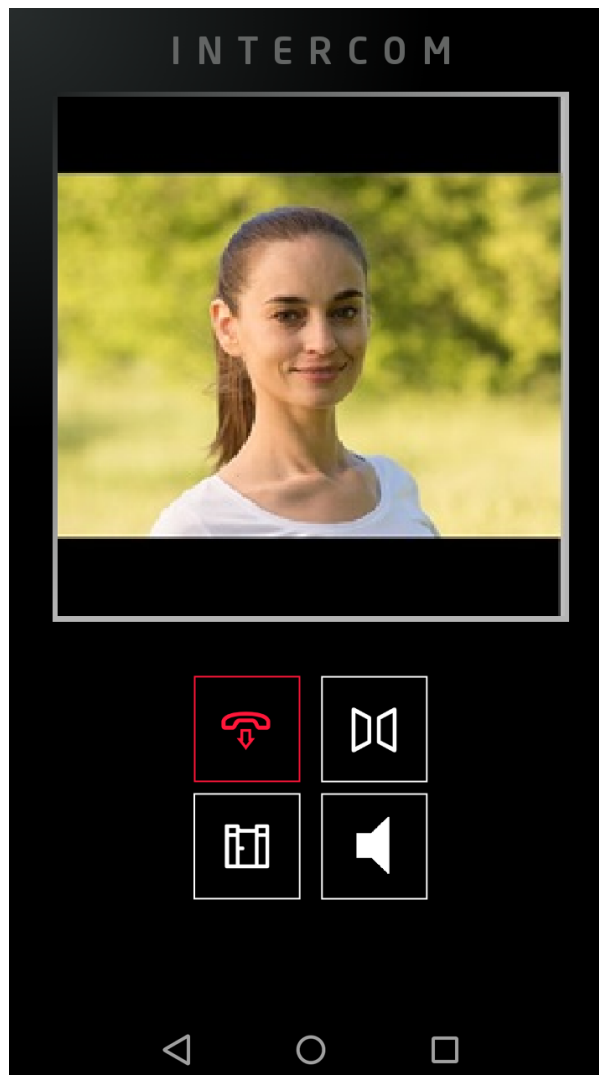
.....

SAVE CANCEL

- Confirm your entries by pressing *Save*;
- Correctly carried out configuration will cause the screen of the mobile device - in its notification bar - to display information about the connection to the *SIP* server;
- Exit the application settings.

### 6.3. Making a call from the intercom

1. On the intercom, press the call button.
2. Regardless of whether the Home Manager application on the mobile device is open, a connection will be established - the door intercom panel will appear on the screen.
3. The button on the top left is used to receive a call - until it is used - the caller will hear nothing and the interphone will still ring.
4. The `OnDoorClick` and `OnGateClick` events can be triggered from the door video panel position, which will work depending on the setting made in the Object Manager.
5. In the doorphone panel there is also a button for switching on / off the hands-free mode.



## 7. IP cameras image operation

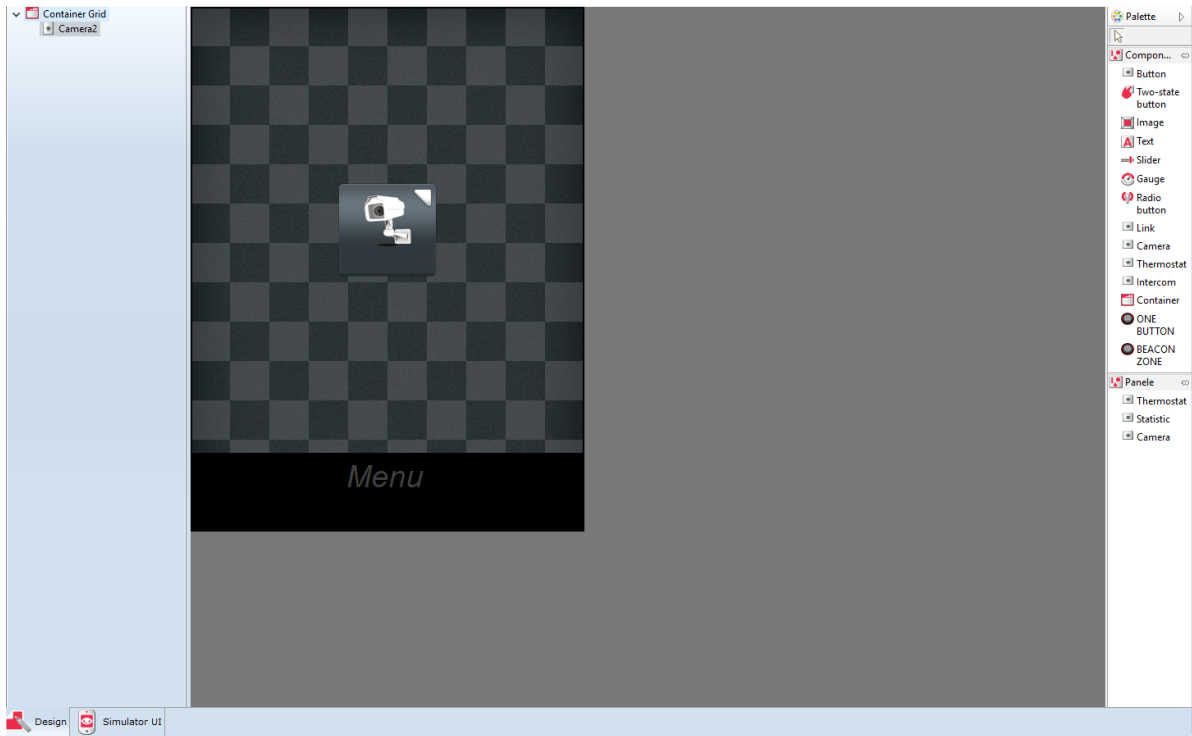
**NOTE!** Support for Visual Builder functionality has been ended. The Home Manager interface creator will be removed in the next versions of Object Manager.

Home Manager application allows to access image of IP camera via any interface. There is no limit of number of operated cameras, however, image of each of them will be displayed separately.

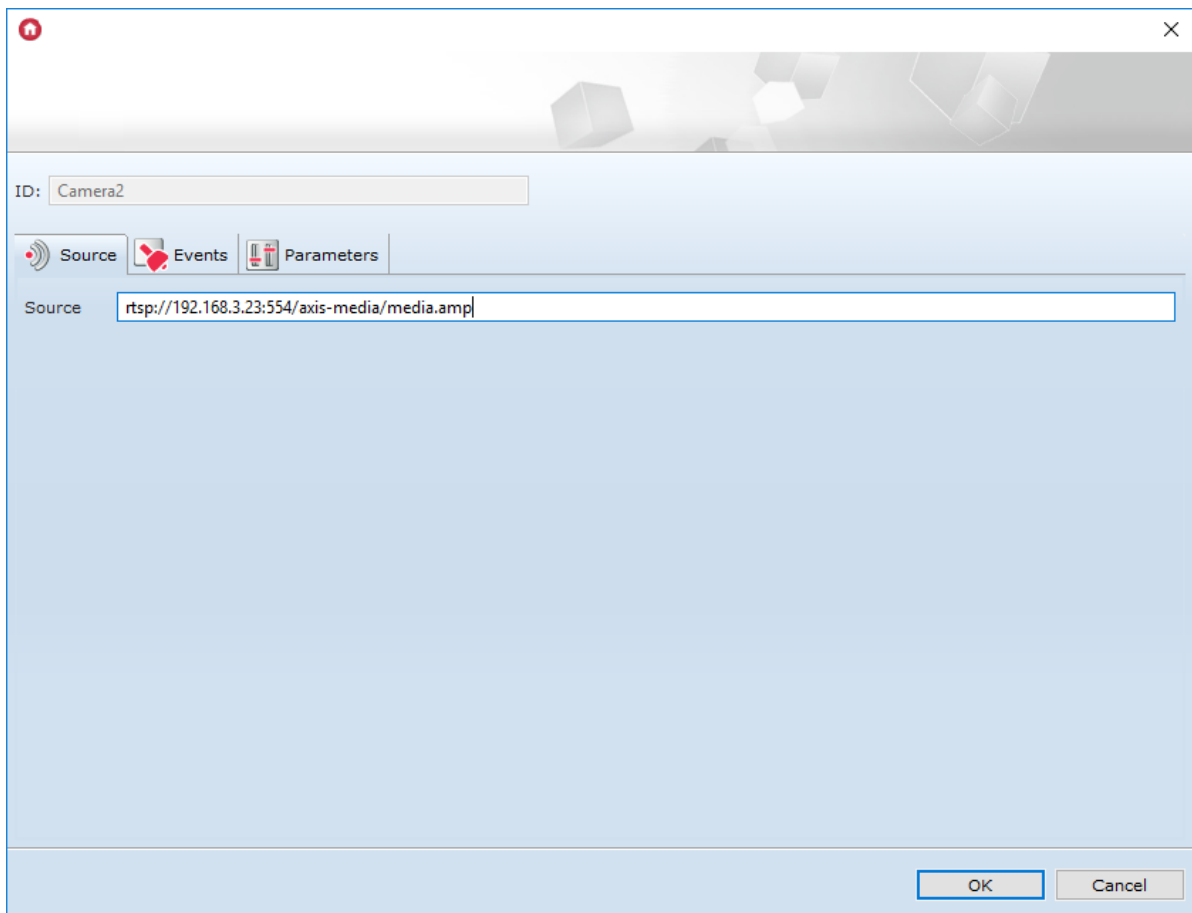
**NOTE!** Home Manager application displays correctly images of cameras supporting RTSP protocol and h264 codec in MPEG standard.

### A. Adding camera component

To add camera image to the interface, drag the "camera" object available on the objects list to the workspace:



Then, enter address of the camera which image will be displayed as a source of the object. Added camera needs to be pre-configured to enable opening images from it using RTSP protocol.

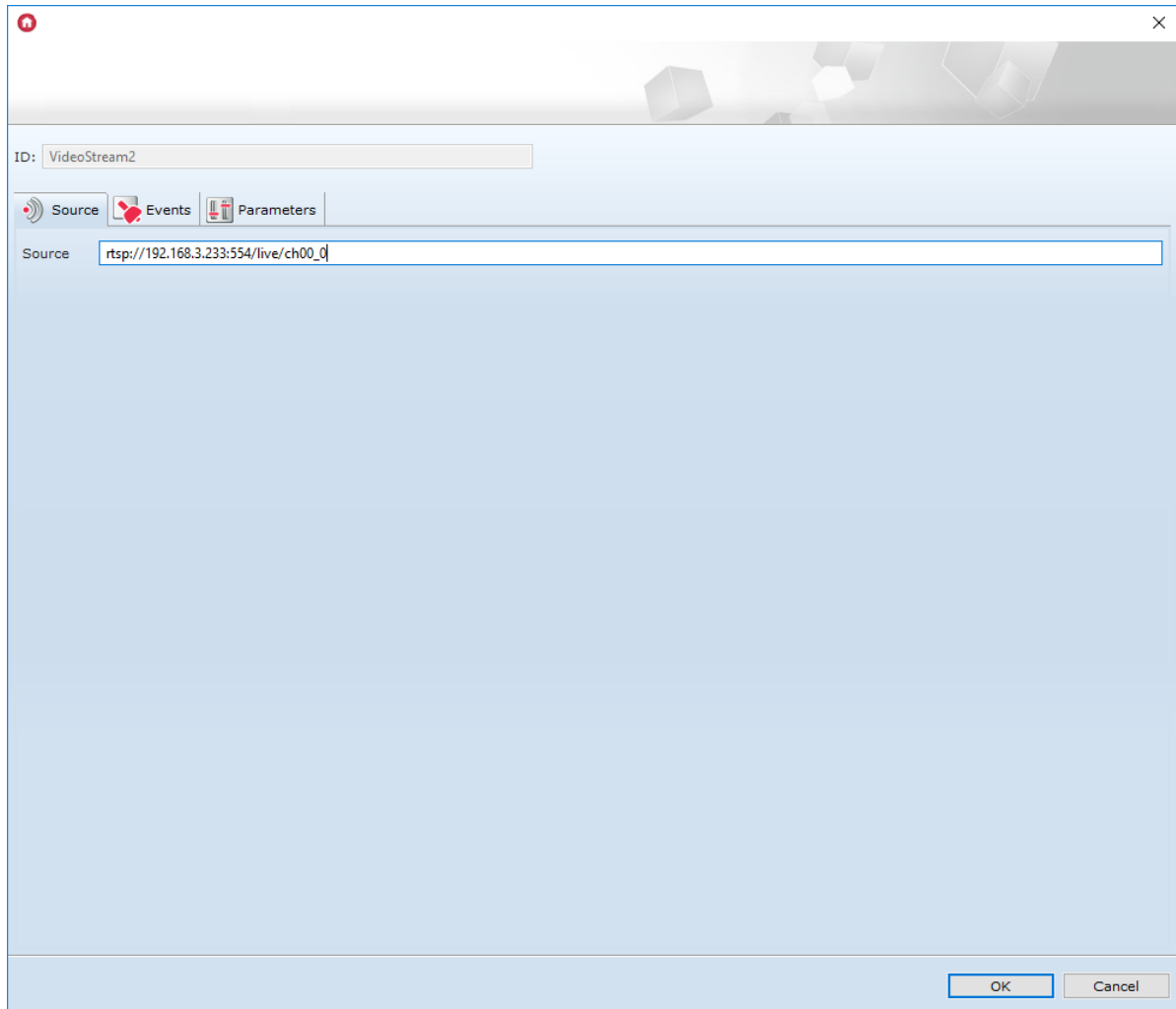


After sending created interface, image from the camera will be displayed on the screen of the mobile device after clicking added object.

## B. Adding camera panel

It is possible to add an image from the camera to the interface using the *Camera* panel. To do this, drag it to the blank page of the interface.

Then - as the source for the added object, it is necessary to enter the address of the camera whose image is to be displayed. The added camera must be set up in advance in such a way that it can be previewed via the RTSP protocol.



After sending the created interface, the image from the camera will be displayed on the screen of the mobile device after pressing the page with the added panel *Camera*.

## 8. Remote access of the mobile application to the system

The Grenton system gives you the freedom to control your home from anywhere in the world. Sitting at work, or being on a business trip, we can easily control the state of our investment and manage its functions in a very simple way.

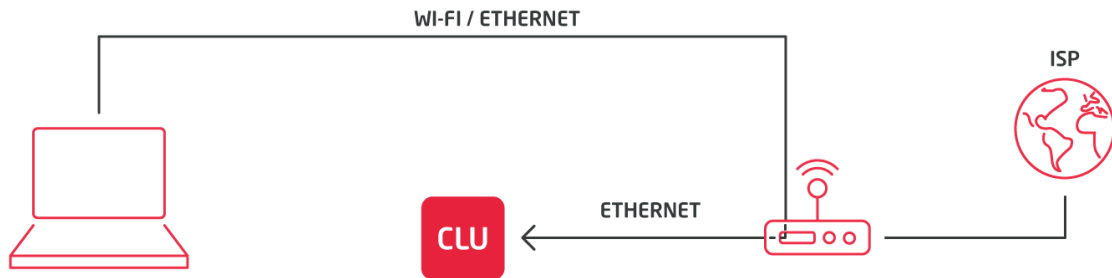
To be able to remotely access the Grenton system, it should meet the following requirements:

- the Grenton system must be fully configured;
- created mobile application interfaces must be sent to mobile devices from which remote access is to be performed;
- the internet service provider must provide access to a static external IP address;
- the router / access point must be able to route ports.



## 8.1. System configuration

The manual has been prepared for a system in which the central unit is connected to a router visible through an external static IP address.



Before configuring remote access, you must:

- make sure that the central unit has been connected to the router of the local network and that the address of the central unit has been sent from the router's address pool;
- check the address of the central unit assigned by the local area router (for this purpose, double-click on the central unit's icon);
- in the newly opened window, read the information from the box marked below:

The screenshot shows a configuration window for a central unit (CLU). The window title is 'CLU\_220001205'. The 'Name' field contains 'CLU\_220001205' and the 'ID' field contains '220001205'. The 'IP' field contains '192.168.1.2' and the 'FW' field contains '407'. The 'IP' field is highlighted with a red box. Below the fields are tabs for 'Control', 'Events', 'Embedded features', and 'User features'. A table lists various methods and their parameters:

Method	Parameter name	Value	Call
AddToLog	Log	<input type="text"/> string	<input type="button" value="▶"/>
ClearLog			<input type="button" value="▶"/>
SetDateTime	UnixTimestamp	13:11:22 15-02-2019	<input type="button" value="▶"/>
StartZWaveDiscovery	Time	<input type="text"/> number	<input type="button" value="▶"/>
StopZWaveDiscovery			<input type="button" value="▶"/>

At the bottom right, there are 'OK' and 'Cancel' buttons.

For the analyzed case, the central unit's address is: *192.168.1.2*. This address will be used to perform port routing.

## 8.2. Port routing setting on the local network router

**NOTE!** The port routing settings for each router may vary! The general procedure is presented below.

In order to set up port forwarding it is necessary to:

- access to the settings of the local network router - to do this, it is necessary to connect to the local network in which the central unit is located;
- opening an internet browser and entering the IP address of the local network router in the address field (in order to enter its settings) - the default address is usually found at its bottom;
- logging in using login data - the default login and password are most often in the form of a sticker on the bottom of the local network router (the default router data can also be found in dedicated internet tools);

**NOTE!** If the entered IP address or login details are incorrect, it means that they have been changed by the network administrator. To access the router's settings, please contact him.

- find the position regarding port forwarding in the router's settings (*Port Forwarding* or similar);
- execution of external port forwarding 1234 to internal port 1234 of the local address of the central unit using the UDP protocol - an example configuration is provided below:

The screenshot shows the Tomato router's web interface. The left sidebar contains a navigation menu with categories like Status, Bandwidth, IP Traffic, Tools, Basic, Advanced, Port Forwarding, Access Restriction, QoS, Bandwidth Limiter, USB and NAS, VPN Tunneling, and Administration. The 'Port Forwarding' section is selected, showing a table with the following configuration:

On	Proto	Src Address	Ext Ports	Int Port	Int Address	Description
On	UDP		1234	1234	192.168.1.2	CLU1
<input type="checkbox"/>	TCP					

Below the table, there are instructions for the fields:

- **Src Address** (optional) - Forward only if from this address. Ex: "1.2.3.4", "1.2.3.4 - 2.3.4.5", "1.2.3.0/24", "me.example.com".
- **Ext Ports** - The ports to be forwarded, as seen from the WAN. Ex: "2345", "200,300", "200-300,400".
- **Int Port** (optional) - The destination port inside the LAN. If blank, the destination port is the same as *Ext Ports*. Only one port per entry is supported when forwarding to a different internal port.
- **Int Address** - The destination address inside the LAN.

At the bottom right of the page, there are 'Save' and 'Cancel' buttons.

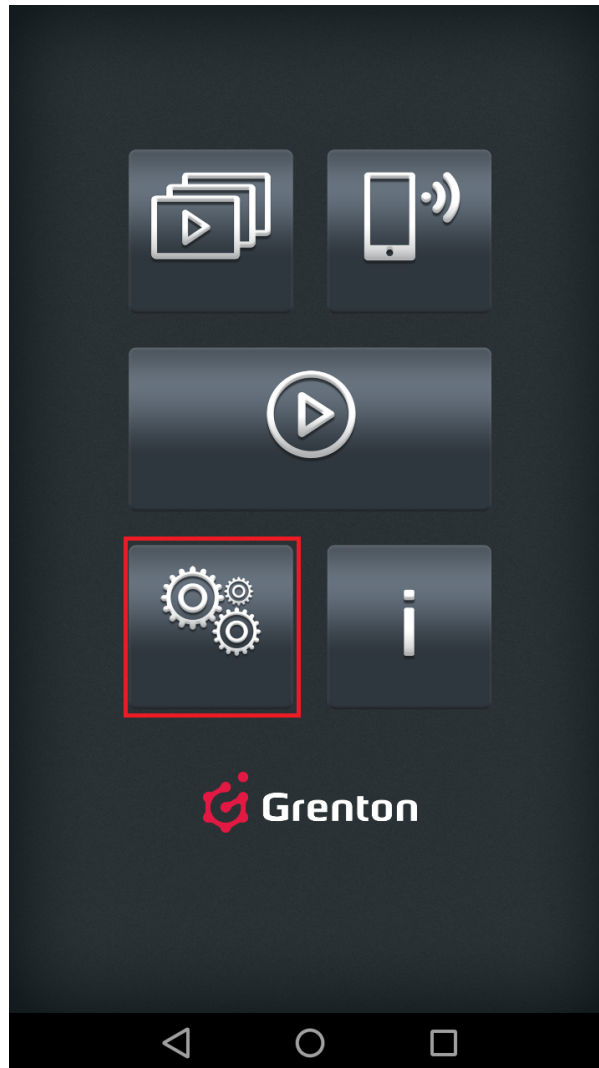
- save router settings - in some cases it may be necessary to restart the device.

**NOTE!** Make sure external communication is not blocked by internal router settings.

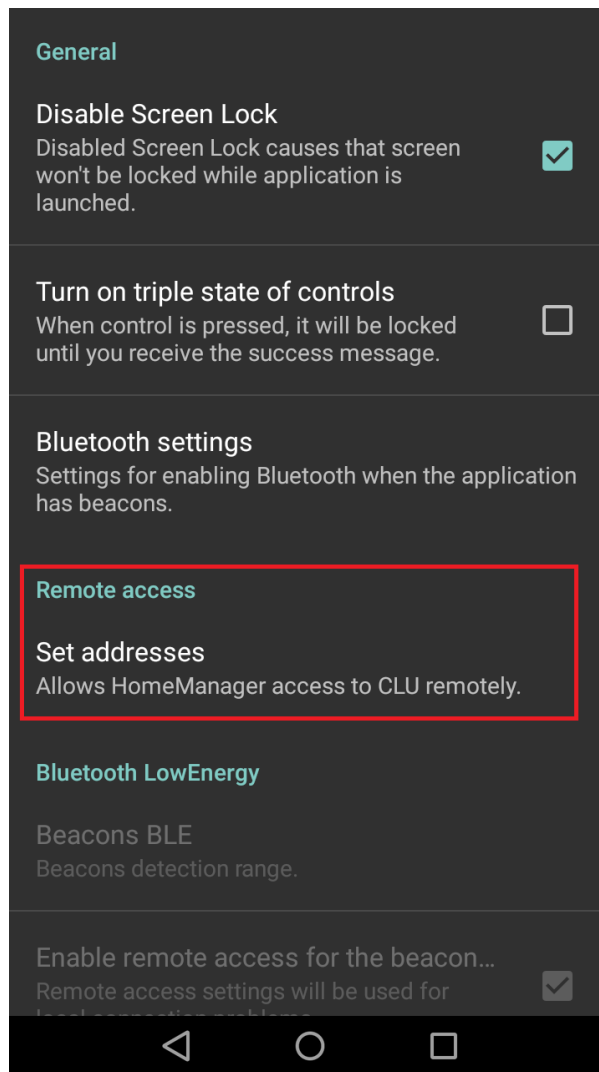
## 8.3. Configuration of the Home Manager mobile application

When creating a configuration, you must:

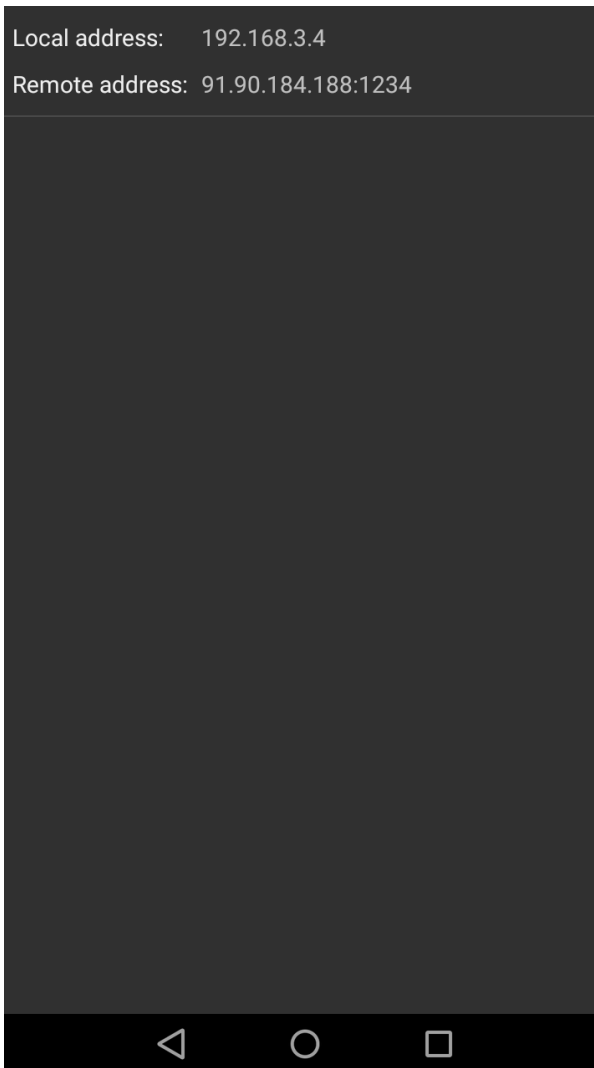
- start the Home Manager mobile application;
- make sure that the interface has been uploaded to the mobile application, by means of which the remote access functionality will be implemented;
- go to the main screen of the mobile application and enter *Settings* (by clicking on the gear icon in the lower left corner of the screen):



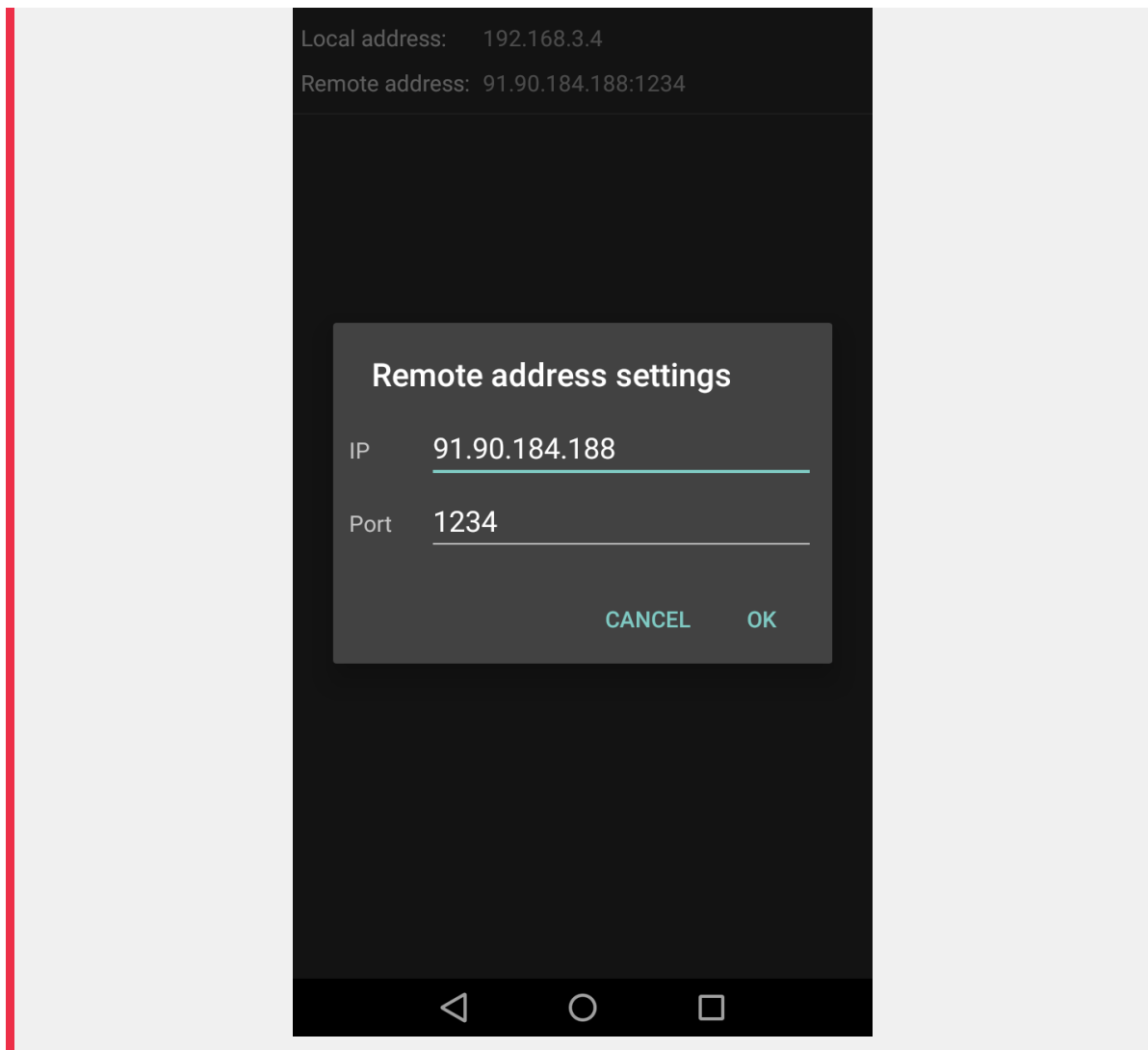
- in the settings, click *Remote access, Set addresses*:



- select the one for which remote access is to be configured from the list of available interfaces;
- then a window will be displayed with the current network configuration of the system with the address information:
  - local (local IP address of the central unit);
  - remote (external IP address of the network to which the central unit is connected together with the port number assigned to it);



**NOTE!** If the specified remote address differs from the actual external IP address, change should be made by clicking on the address window. In the newly opened window, it is necessary to make changes according to the actual IP address of the device. To accept changes, press *OK*.

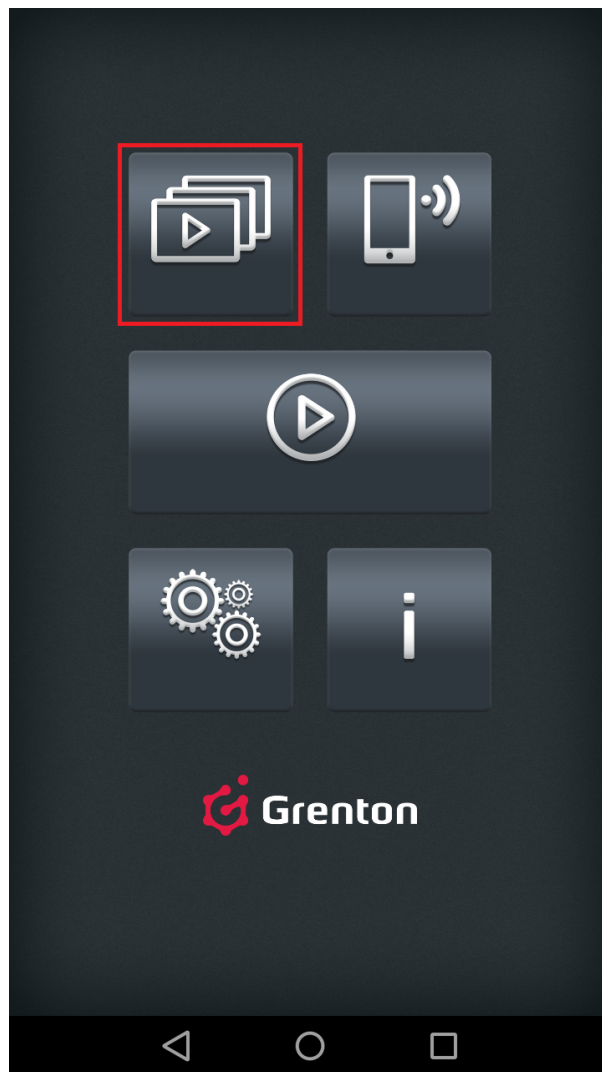


## 8.4. Starting remote access

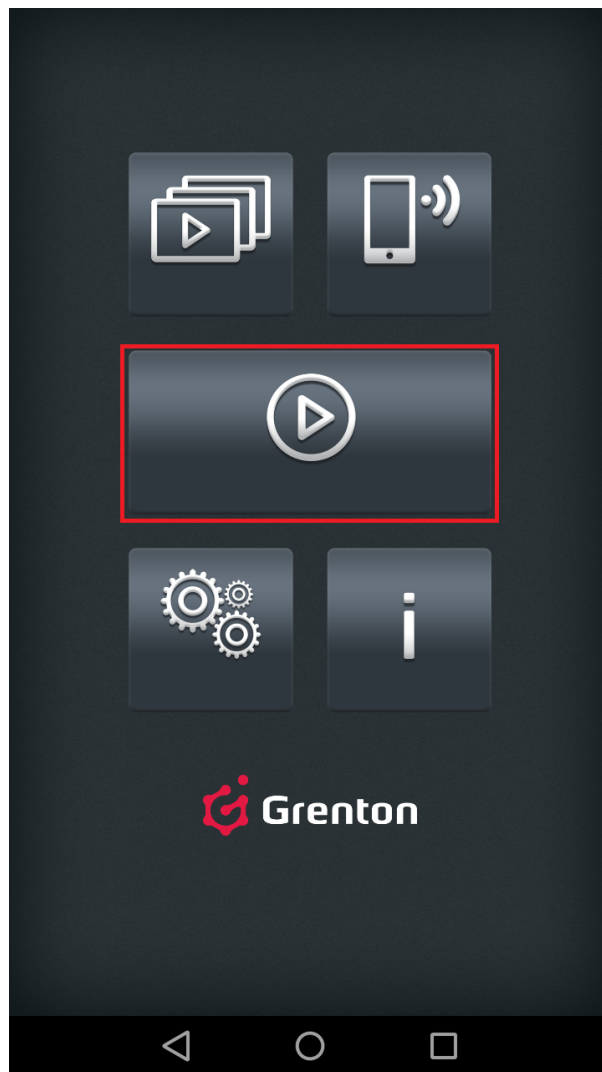
The Grenton Home Manager mobile application automatically switches from local communication to remote communication. For remote access to be possible, the mobile device must meet the following conditions:

- remote access must be configured correctly;
- the device must be connected to a non-local internet network (other than the one to which the system is connected) or it must have cellular network data enabled (*internet in the phone*).

In order to start remote communication with the system, open the interface for which the remote access configuration was performed by selecting it from the list of interfaces:



If the interface was set as the default one, click the button:



Firstly, the Home Manager application will attempt to establish a connection via the local network. When the lack of this possibility is detected, the remote communication will be switched over.



# IX. CLU Objects

## 1. Timers

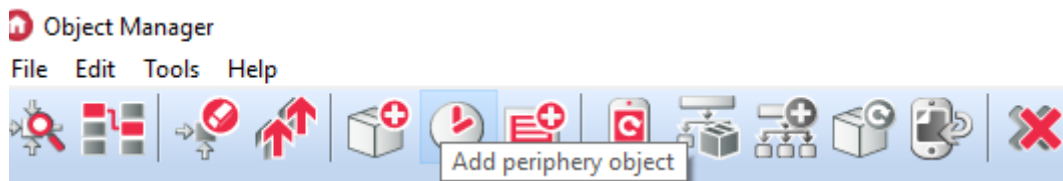
Timers are virtual objects created within specific CLU. Object Manager enables creation of maximum 64 timers. Timers can be used wherever there is need for method invocation after specific time or cyclical method invocation.

The timer itself is also an OM object and as any other object, it has its own features, methods, events, and initial values. Timer can work in two modes:

- **Countdown** After starting, counts down set time. When countdown reaches zero, method connected to `OnTimer` event is invoked, while the timer stops and remained stopped until next usage of `start` method.
- **Interval** Cyclical timer, it counts down set time after starting. When countdown reaches zero, method connected to `OnTimer` event is invoked, while the timer starts the countdown again. The situation repeats until stopping timer using `Stop` method.

### A. Timers creation

To create timer in a specific CLU, mark it, then select `add CLU object` from the upper menu.



After clicking the icon, a list of available objects will appear. Find and select `Timer` option. After selecting the timer, click `OK`, then name the new timer, set its time [in ms] and work mode [countdown or interval]. Selected time will be simultaneously time setting in initial conditions. Created timer will appear on the objects list of the selected CLU.

Created timer is also a CLU object, and as other physical objects, it is controlled by objects configurator [look up V.4.1.](#)

### B. Configuration parameters of timer

#### FEATURES

Name	Description
<code>Time</code>	Countdown time (in ms)
<code>Mode</code>	Timer work mode: 0 - countdown, 1 - interval
<code>State</code>	Current timer work status: 0 - stopped, 1 - counting, 2 - paused
<code>Value</code>	Time left until onTimer event occurs (in ms)

## METHODS

Name	Description
<code>SetTime</code>	Sets time of the timer (in ms)
<code>SetMode</code>	Sets work mode
<code>Start</code>	Starts the timer
<code>Stop</code>	Stops the timer
<code>Pause</code>	Pauses the timer

## EVENTS

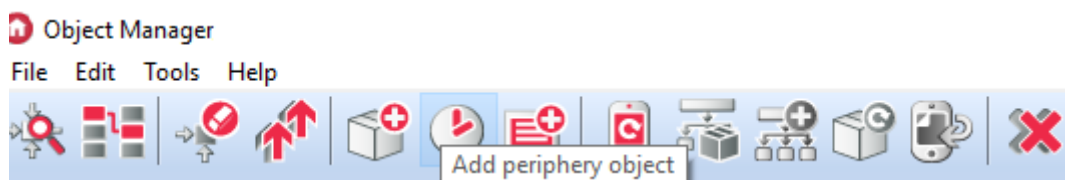
Name	Description
<code>OnTimer</code>	The event is called when the timer is counted
<code>OnStart</code>	The event is triggered when the timer starts
<code>OnStop</code>	Event triggered when the timer stops
<code>OnPause</code>	The event is called when the timer is paused

## 2. Calendar

Calendars, just as timers, are virtual objects created by the user in CLU. It is possible to create up to 64 calendars on one CLU. One calendar created in CLU is a one rule followed on a specific day and time or in daily, monthly, or hourly intervals, with accuracy down to a minute. The rules can be created using graphic interface, or using syntax compliant with CRON rules of the LINUX system.

### A. Calendar creation

To create a calendar, mark CLU in which you want to create it, then launch `Add CLU object` from the upper menu.

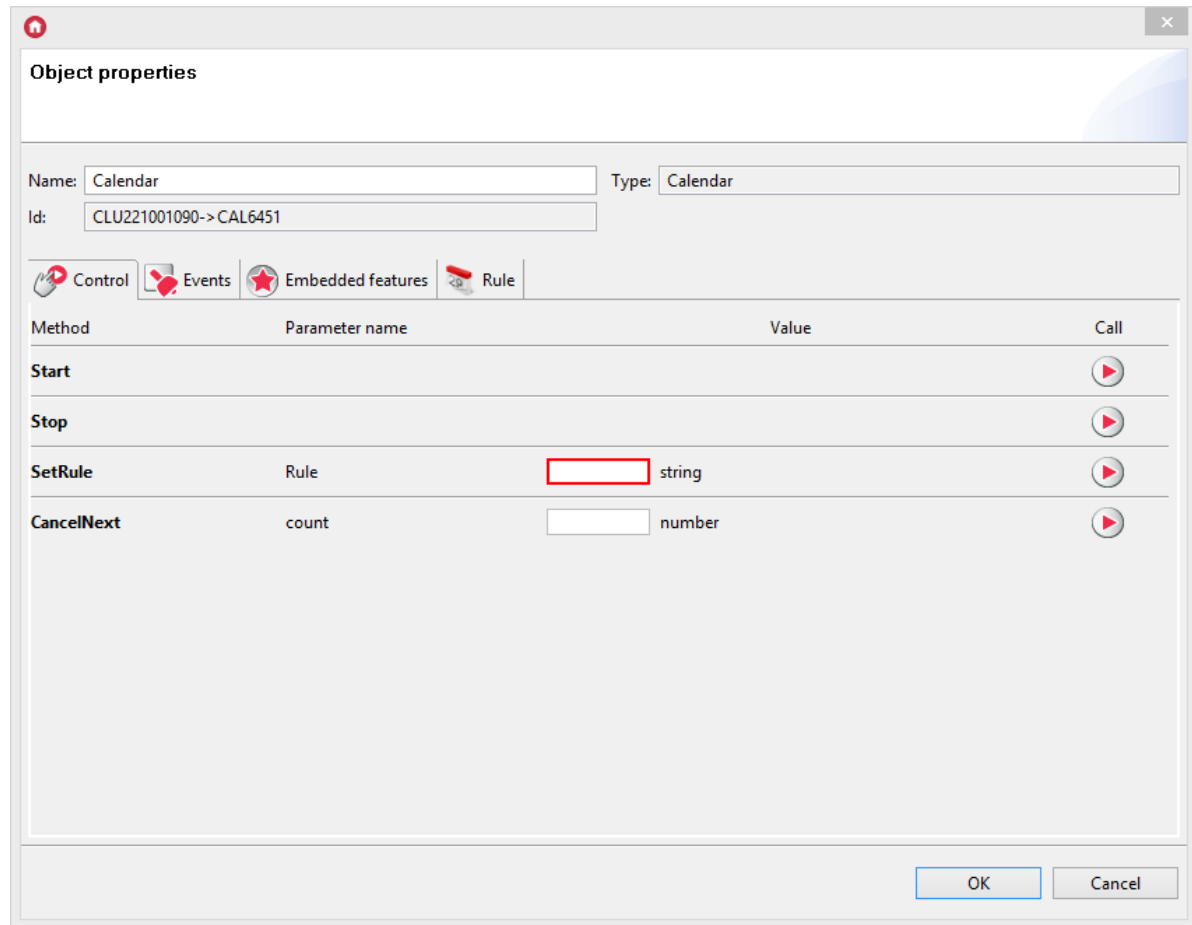


In the window that opens, select `Calendar`. After clicking `OK`, enter the name for the calendar you are creating. The Object Manager will display the properties window of the created object.

**NOTE!** The calendar after creating and sending the configuration to the CLU automatically becomes active - to stop the calendar, call the `STOP` method.

## B. Calendar features

Window calendar features contains four tabs:



- **Control** - contains calendar methods;
- **Events** - contains calendar events;
- **Built-in features** - contains list of calendar features;
- **Rule** - contains interface enabling defining rules easily.

## C. Calendar rules

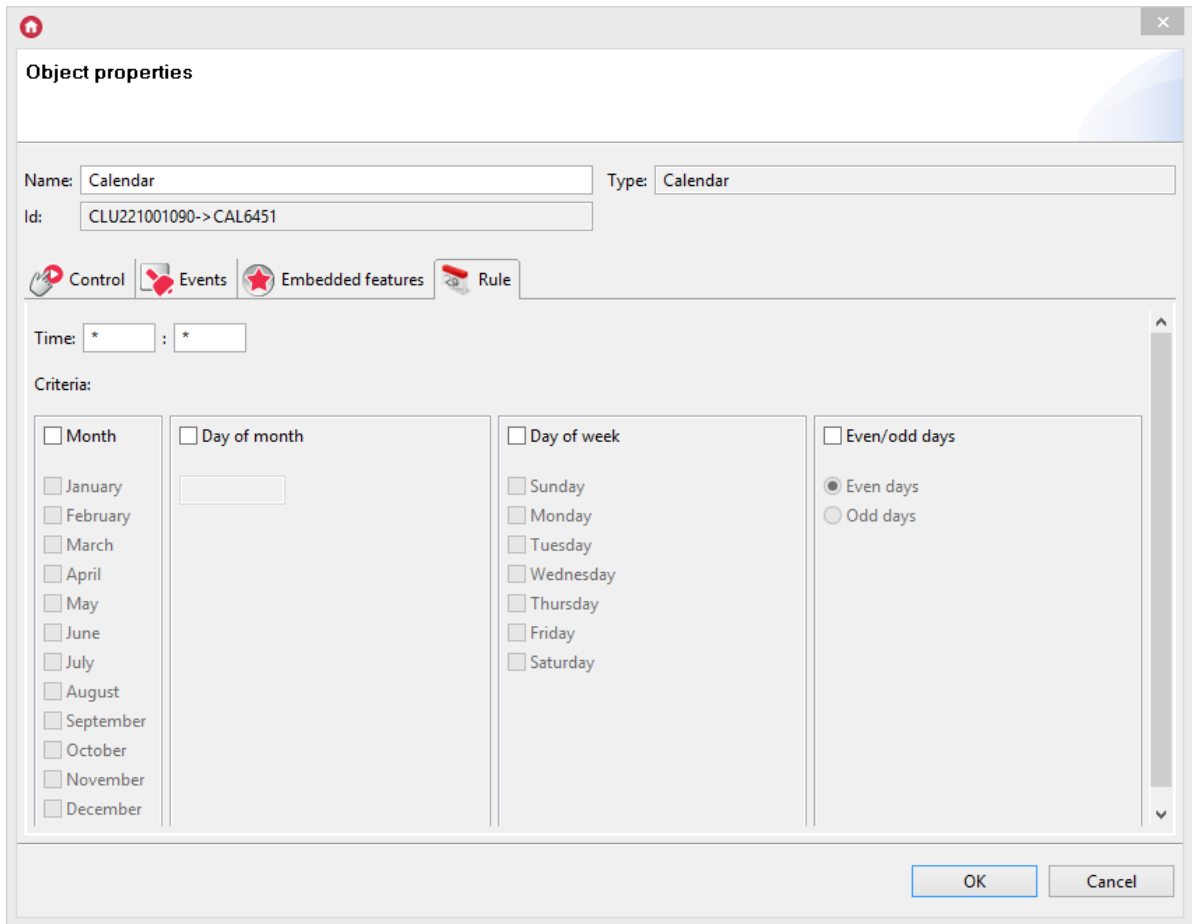
There are two ways of entering rules for the calendar:

- Through the graphic interface using `Rule` tab;
- By entering CRON rule using `SetRule` method in the control tab or `Rule` as a Built-in feature.

## D. Calendar rule creation through graphic interface

There is graphic interface in the `Rule` tab, using which the user can easily set rules parameters of the calendar.

**NOTE!** After entering rule parameters through graphic interface, `Rule` value in the `Built-in features` is entered automatically according to the selected criteria.



There are two sections there, in which the user selects rule parameters:

- **Time** - contains two boxes: the first, in which the hour (or hours range) is entered; the second, in which the minute (or minutes range) is entered. Values in the boxes should be entered according to the CRON rule.
- **Criteria** - contains remaining parameters of the rule. The user makes selection by marking appropriate boxes.

## E. Calendar rules creation in accordance with CRON format

Calendar rules are created by entering them in the `Rule` field in calendar built-in features, or using `SetRule` method. Detailed information on this process can be found in CRON calendar documentation.

## F. Configuration parameters of Calendar

### FEATURES

Name	Description
<code>Rule</code>	Calendar rule in CRON format (or "ERROR" format in the case of entering wrong rule)
<code>SinceLastRun</code>	Time (in minutes) since the condition of the rule was last met
<code>ToNextRun</code>	Time (in minutes) until next calendar action invocation
<code>State</code>	Calendar work status: 1 (active) or 0 (not active)

## METHODS

Name	Description
<code>Start</code>	Switching to active state ( <code>State</code> =1)
<code>Stop</code>	Switching to paused state ( <code>State</code> =0)
<code>SetRule</code>	Setting calendar rule
<code>CancelNext</code>	Cancelling invocation of selected number of the nearest calendar actions

## EVENTS

Name	Description
<code>OnCalendar</code>	Events since calendar action invocation
<code>OnStart</code>	Events since restarting calendar work
<code>OnStop</code>	Events since stopping calendar work
<code>OnCancel</code>	Events since cancelling the nearest actions

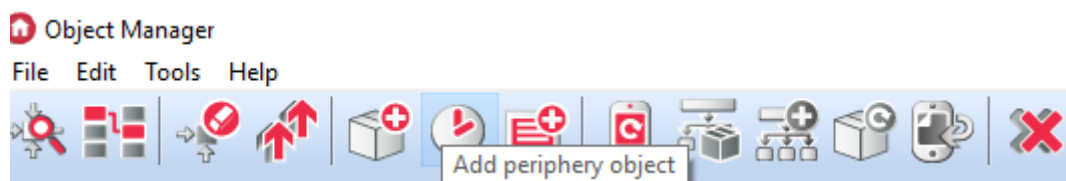
# 3. Schedule

Schedule is a virtual object used for setting value of any feature throughout a week. The values are set using graphic interface for each day and each hour with 15 minutes, 30 minutes, or 1 hour frequency. Up to 64 schedules can be created in one CLU.

**NOTE!** After schedule's creation (after sending new configuration to CLU) it becomes automatically active. To stop schedule work, invoke `Stop` method.

## A. Schedule creation

To create a schedule, mark CLU within which you wish to create it, then launch `Add CLU object` from the upper menu.



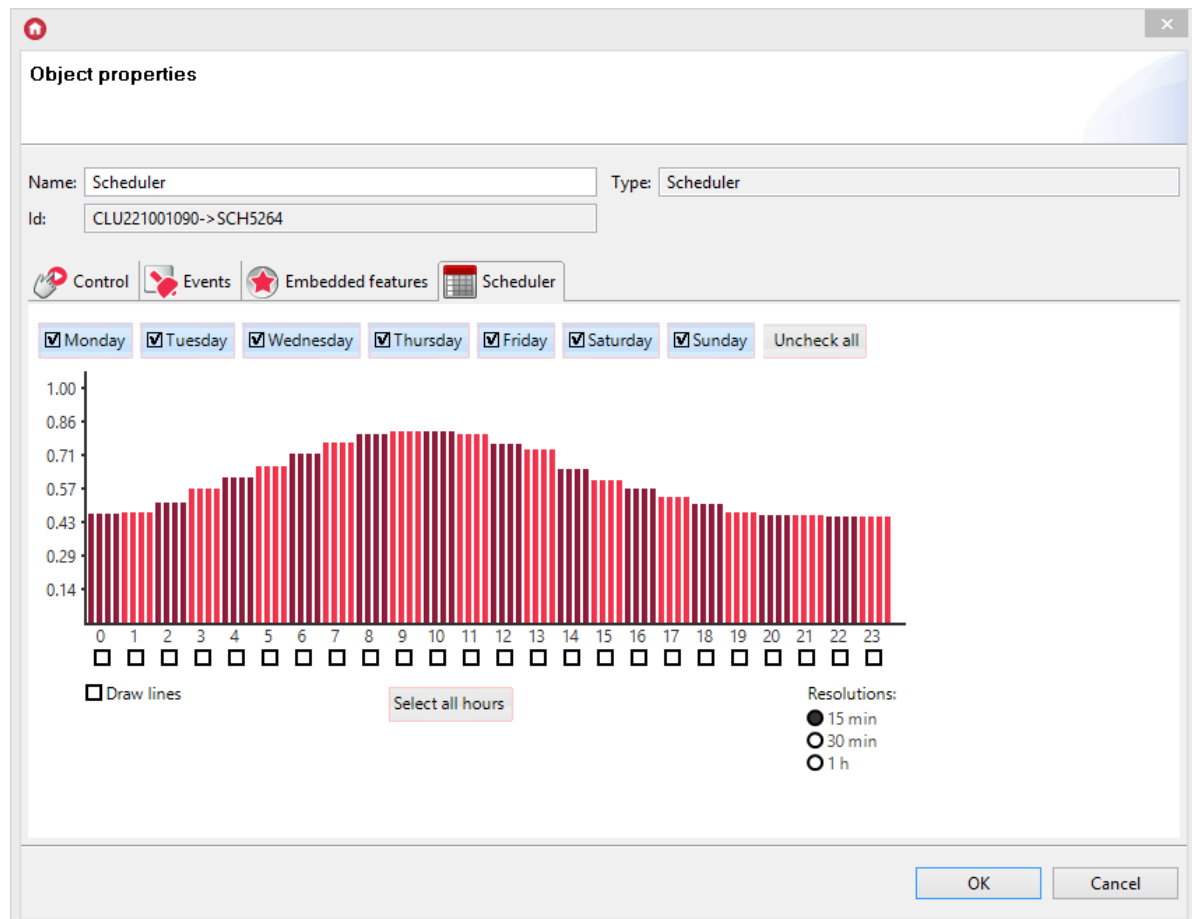
In the opened selection window, it is necessary to find and select the `Scheduler` object. After entering the name, the schedule properties window will open on the screen.

In this window there are four tabs:

- **Control** - includes schedule methods;
- **Events** - contains schedule events;
- **Built-in features** - contains a list of schedule features;
- **Scheduler** - includes a graphical interface allowing simple formulation of values for the entire scope of the schedule.

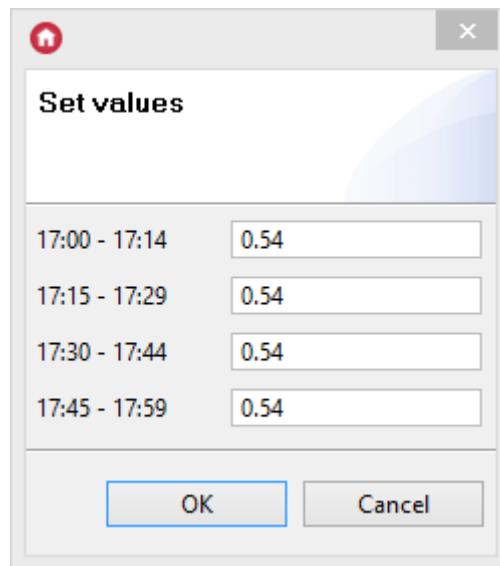
## B. Setting values for the schedule

In the schedule tab of properties window there is graphical interface, thank to which you can define values for specific output.



The schedule allows to enter values for 7 days (within one week) with 15-minutes frequency. You can set values for every day individually or for several days at once. Day of which the values are currently being set is discerned by black mark on the right of its name. Switching to another day happens after clicking its name.

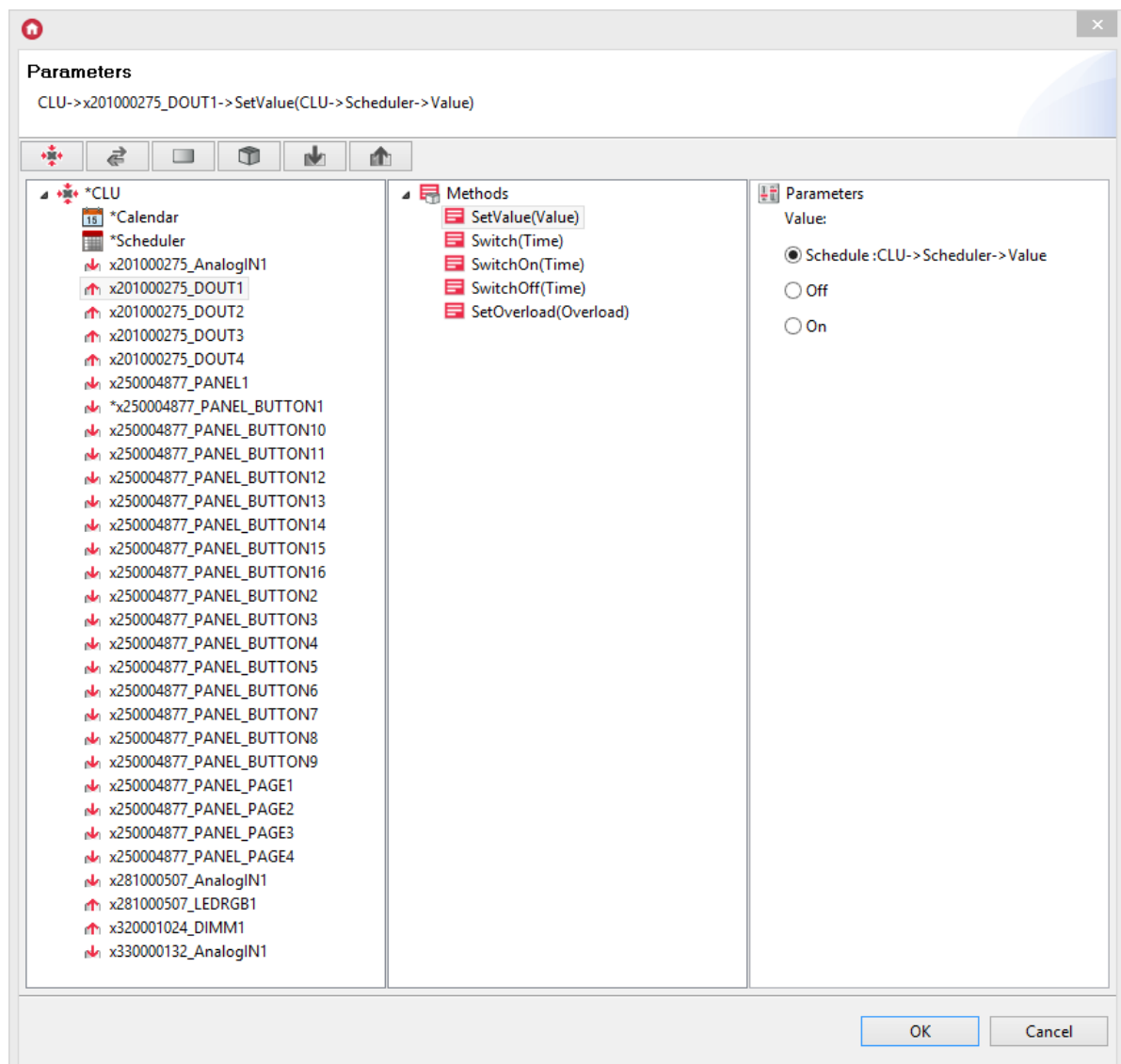
To simultaneously enter values for several days, mark days for which you will set values (by clicking marker), then begin entering the values. You can add values directly on the chart using mouse, or enter them in the values window which appears after clicking a specific hour.



### C. Setting output value using schedule

Changing value in a set schedule triggers `OnHarmonogram` event.

To copy values set in the schedule to the values of selected output, add `SetValue` method of the selected output to the `OnHarmonogram` event. Then, select `Schedule` as parameter of the method.



Every 15 minutes `Value` of this output will be set according to the value saved in the schedule.

**NOTE!** Remember to make sure that the range of values set in the schedule is the same as the range in which the selected output can be controlled. You can change schedule values range using

`SetMax` and `SetMin` methods, and by changing built-in `Min` and `Max` features.

## D. Configuration parameters of schedule

### FEATURES

Name	Description
<code>Data</code>	String defining value changes schedule (see: Data format)
<code>State</code>	Schedule work status: 1 (active) or 0 (not active)
<code>Value</code>	Output value, changed every 15 minutes according to the schedule
<code>Min</code>	Minimum value for setting graphic interface value range
<code>Max</code>	Maximum value for setting graphic interface value range

### METHODS

Name	Description
<code>Start</code>	Switching to active state ( <code>State</code> =1)
<code>Stop</code>	Switching to paused state ( <code>State</code> =0)
<code>SetData</code>	Setting weekly schedule

### EVENTS

Name	Description
<code>OnHarmonogram</code>	Events since change of <code>Value</code> feature
<code>OnStart</code>	Events since restarting work
<code>OnStop</code>	Events since stopping work

## 4. PID controller

CLU enables creation of up to 64 PID controllers (proportional-integratingdifferentiating), used to maintain set output value on the constant level depending on the input value.

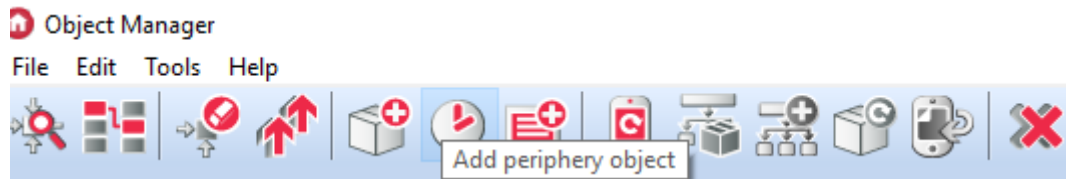
The most popular way of using PID controller is temperature control on the basis of information gathered from temperature sensor.



**NOTE!** PID controller working in AUTO mode after starting work (after first switching on or after CLU reset) performs object calibration procedure, during which the temperature of the controlled object may increase for several percent above the set temperature. Thus, PID controllers are not recommended to use with objects of high thermal inertia, e.g. to control water temperature in an aquarium.

## A. PID controller creation

To create PID controller mark CLU within which you want to create it, then launch `Add CLU object` from the upper menu



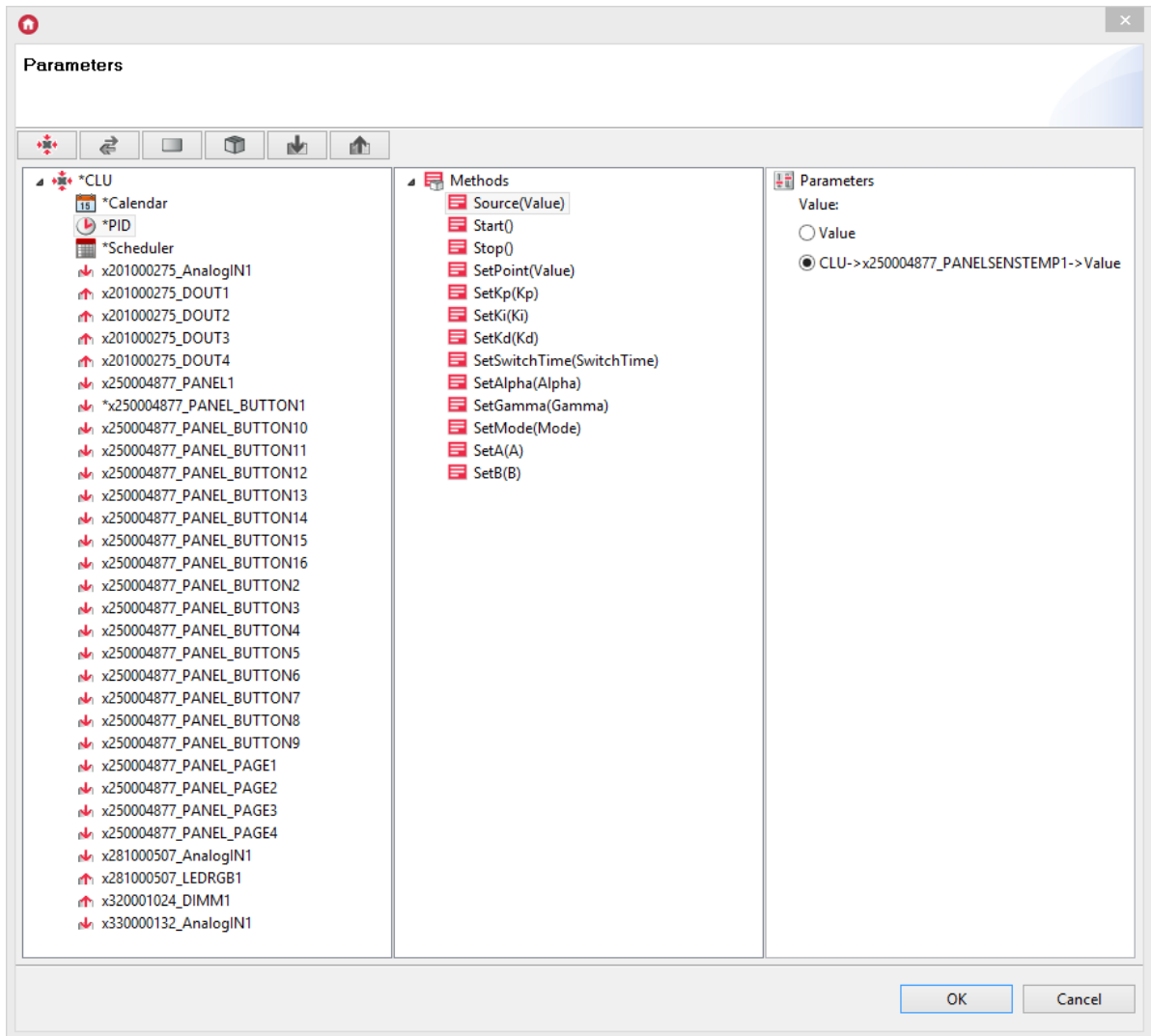
In the open window find and select `PIDcontroler`, then name it. Windows of features of the newly created controller will appear on the screen. It contains three tabs

- **Control** - contains controller methods;
- **Events** - contains controller events;
- **Built-in features** - contains list of controller features.

## B. Control using the controller

To control output values with controller, you must properly connect it to the input and output objects. To do that, follow these steps in order:

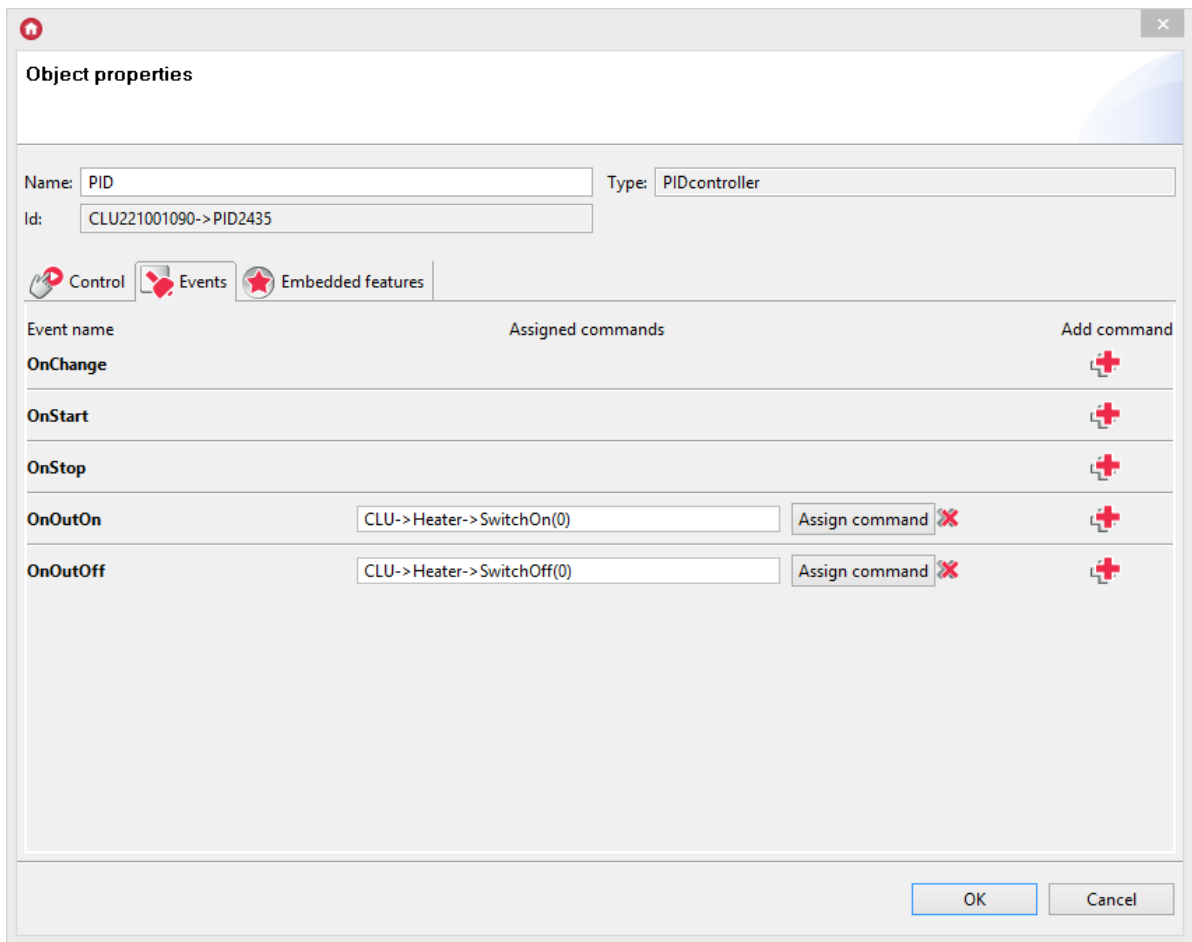
- Add the source value to the Source method, eg the temperature sensor's `Value` feature (in the temperature sensor to the `OnChange` event, select the PID controller, and assign the value from the temperature sensor to the `Source` method as a parameter).



- Associate the output module with the corresponding PID object events.

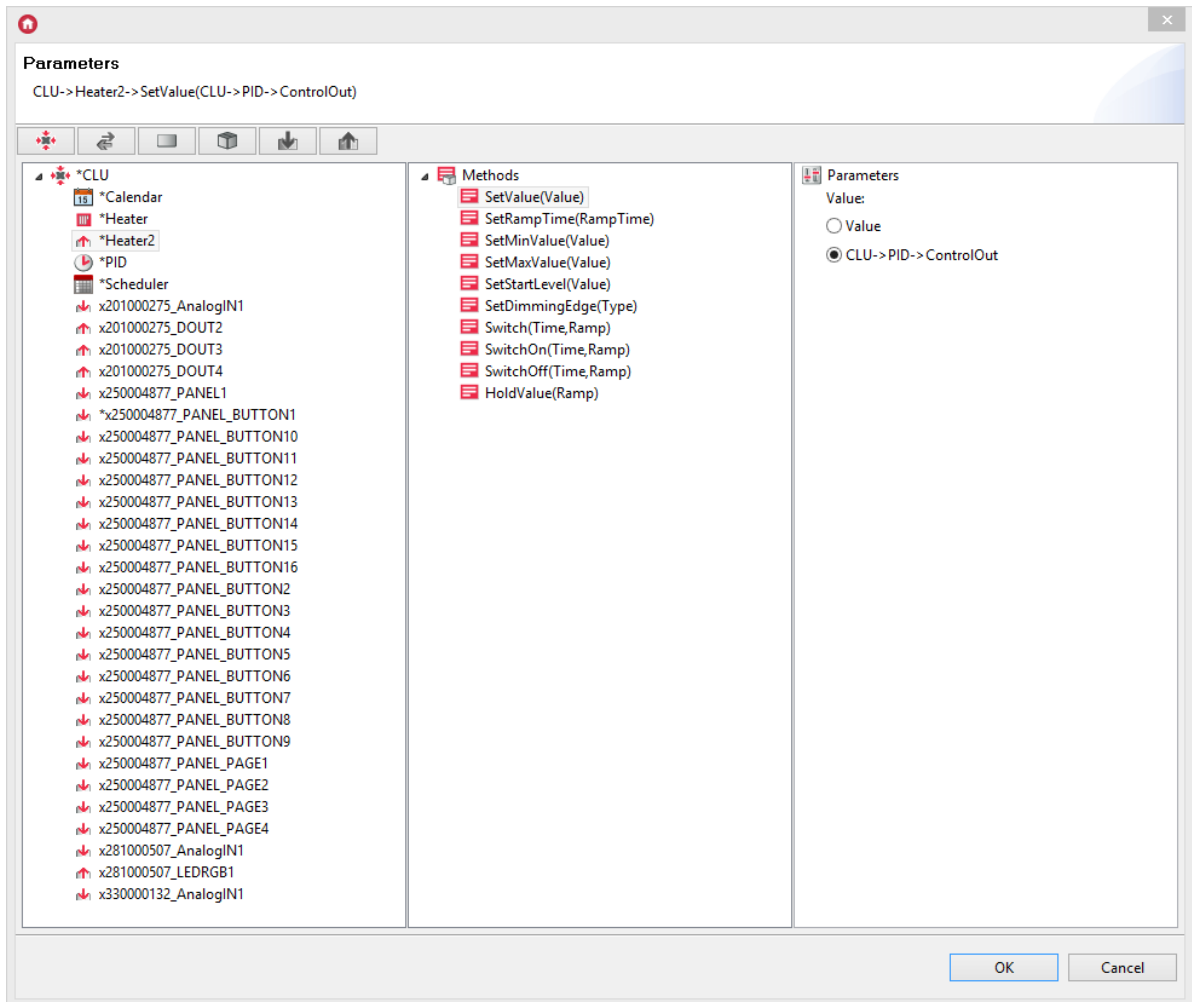
To do this, assign control methods to the object to the `OnOutOn` and `OnOutOff` events.

In the case of temperature control, the `OutOutOn` event in the controller should be assigned the `SwitchOn` method of the output from which the heat source is controlled (to which the radiator, furnace, radiator control valve is connected), whereas the `OutOutOff` event should be assigned a `SwitchOff` of this output.



Alternatively - if the output module's interface allows it, only the `OnChange` event can be used.

To do this, assign the `OnChange` method in the controller to the `SetValue (Value)` method of the output controlling the heat source (the output must have such a method), and then indicate the `ControlOut` feature of the PID controller.



In such configuration, when relay output value in the controller changes, it will be reconnected to the output value.

## C. Work modes

The controller has two possible work modes:

- **Automatic control (  Auto )** Control in the mode is based on automatic control algorithm, in which all the vital parameters are adjusted automatically on the basis of received data.
- **Manual control (  Normal )** In this mode, the user can set all the vital parameters used by PID controller with manual choice of set points (Kp, Ki, Kd parameters). The  Normal mode is intended for advanced users who know the tuning principles of PID controllers.

To set controller in a specific work mode, change value of  feature to:

- - for manual mode;
- - for automatic mode.

Depending on the selected work mode, option of setting values of particular features changes - e.g. parameters A and B are used only for the  algorithm, while parameters Kp, Ki, and Kd are used only in  mode.

**NOTE!** Parameters A and B can't be changed during control since they are continuously updated by the algorithm.

## D. PID Controller operational design

The controller controls `ControlOut` feature by setting its value to 1 or 0 with frequency set by `SwitchTime` feature through duty cycle change.

Before starting control, the controller carries out procedure of controlled object inertia estimation and sets acceptable range of `SwitchTime` values on its basis. After finishing this stage, `SwitchTime` feature value is set automatically in the middle of the selected range.

**NOTE!** If the control is run automatically, manual change of `Switchtime` value is not possible.

## E. PID Controller configuration parameters

### FEATURES

Name	Description
<code>ControlOut</code>	Value of relay output (binary, switched in cycle defined by <code>SwitchTime</code> and <code>DutyCycle</code> )
<code>State</code>	Controller work status: 1 (active) or 0 (not active)
<code>SetPoint</code>	Controller input - target value
<code>Kp</code>	Strengthening of PID controller proportional element
<code>Ki</code>	Strengthening of PID controller integrating element
<code>Kd</code>	Strengthening of PID controller differentiating element
<code>SwitchTime</code> *	Time of switching
<code>Alpha</code>	Parameter $\alpha$ in Kaczmarz algorithm (protection against denominator zeroing)
<code>Gamma</code>	Parameter $\gamma$ in Kaczmarz algorithm (dynamics of a and b estimation changes)
<code>Mode</code>	Controller work mode: 1 - "manual" PID or 2 - automatic Kaczmarz algorithm
<code>A</code> *	Parameter a in Kaczmarz algorithm
<code>B</code> *	Parameter b in Kaczmarz algorithm

- Setting these parameters is not possible in all of controller work modes.

### METHODS

Name	Description
Source	Entering new value of input for driver (feedback loop)
Start	Switching to active mode (State =1)
Stop	Stopping work (State =0)
SetPoint	Setting the target value of the regulator
SetKp	Setting the proportional gain value
SetKi	Setting the gain value of the integrator
SetKd	Setting the gain value of the differentiator
SetSwitchTime	Setting the switching time
SetAlpha	Setting the Alpha parameter in the Kaczmarz algorithm, protecting against zeroing the denominator
SetGamma	Setting the Gamma parameter in the Kaczmarz algorithm
SetMode	Setting the controller's operating mode - manual PID (Normal PID) or automatic algorithm of Kaczmarz (Auto-Kaczmarz)
SetA	Setting parameter a in the Kaczmarz algorithm
SetB	Setting parameter b in the Kaczmarz algorithm

## EVENTS

Name	Description
OnChange	An event dispatched when the value of the Control Out
OnStart	Events since change of ControlOut feature value
OnStop	Events since stopping work
OnOutOn	An event dispatched when the value of the ControlOut property is switched to 1
OnOutOff	An event dispatched when the value of the ControlOut property is changed to 0

## 5. Thermostat

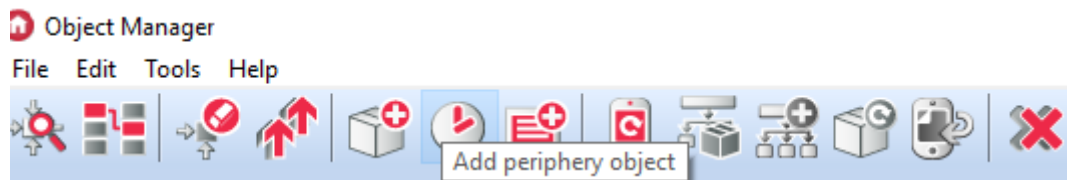
A thermostat is a virtual object that is used to create a heating or cooling control configuration depending on the given temperature sensor and the heating or cooling schedule introduced in the weekly schedule. Temperature values are set using the graphical interface for each day and hour with a 15-minute, 30-minute or hour resolution.

You can create up to 64 thermostats in one CLU.

**NOTE!** After creating the thermostat (after sending a new configuration to the CLU), it becomes active automatically. To stop his work, call the `Stop` method.

## A. Thermostat creation

In order to create a thermostat, select the CLU, under which it is to be placed, and then from the top menu run `Add object CLU`.



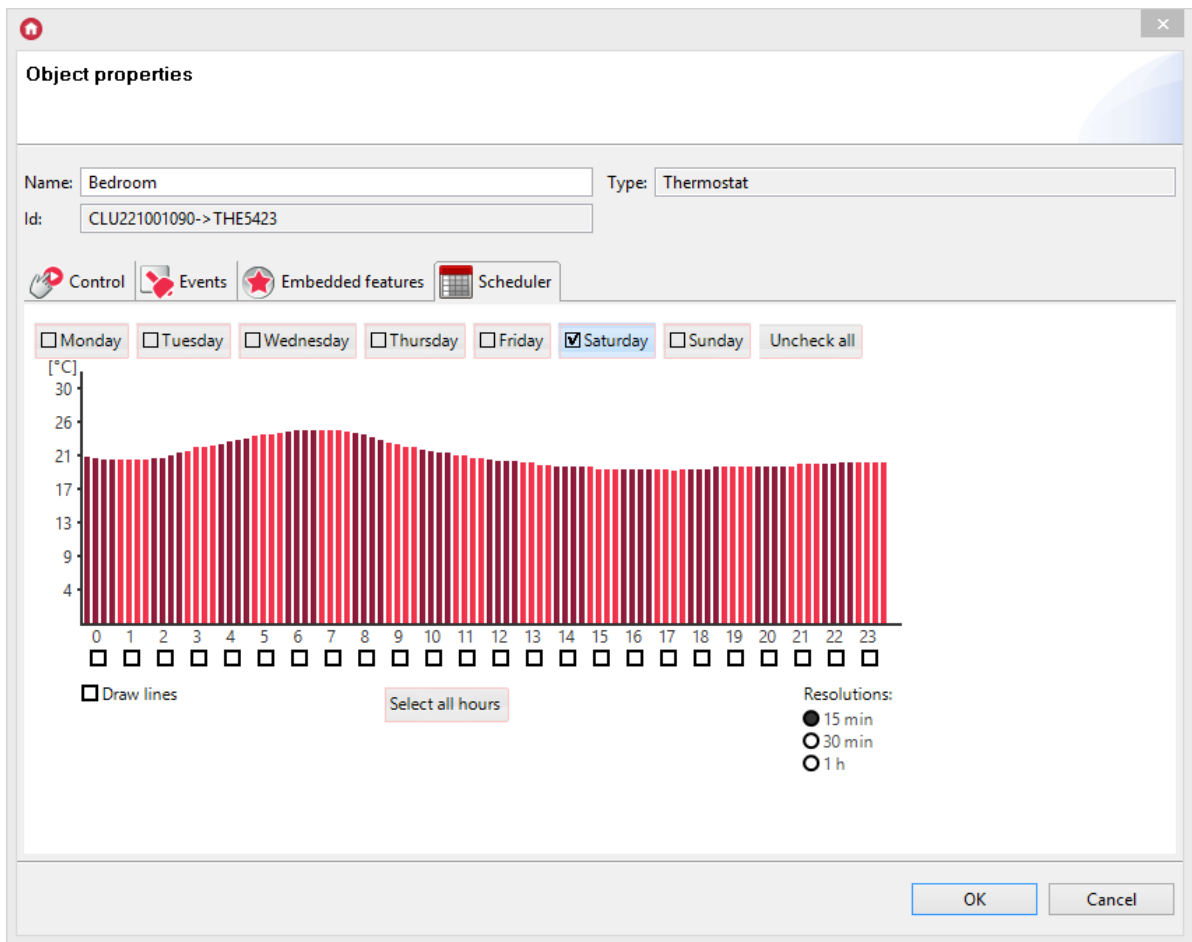
In the opened selection window, search for and select `Thermostat`. After entering the name, entering the source (which should be the temperature sensor responsible for a given heating zone) and selecting the receiver (which is the output to which the device responsible for a given heating zone is connected - eg radiator head, floor heating) for the created object, on the screen will open the schedule properties window.

In this window there are four tabs:

- **Control** - includes schedule methods;
- **Events** - contains schedule events;
- **Built-in features** - contains a list of schedule features;
- **Scheduler** - includes a graphical interface that allows you to easily formulate values for the entire scope of the schedule.

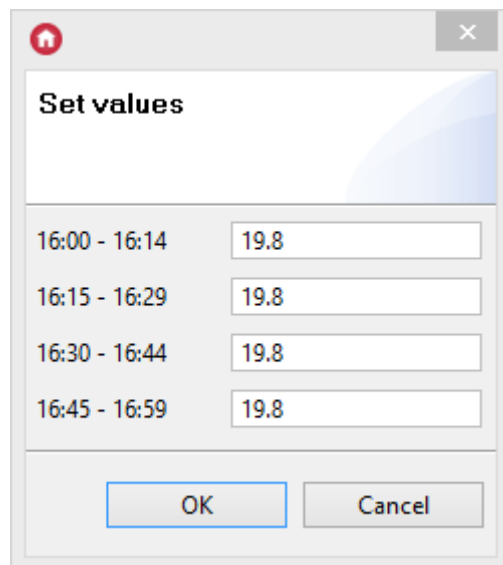
## B. Formulating values for a thermostat

In the tab *Scheduler* (in the properties window) there is a graphical interface, thanks to which it is possible to set values.



The schedule allows you to enter values for 7 days (within one week) with a 15-minute resolution. You can set values for each day separately or for several days at the same time. The day for which the values are currently entered is marked with a black marker on the left side of the name. Switching to another day follows after clicking on its name.

To enter values for several days at the same time, click on the tags next to the names for which the values will be set. The values can be set directly on the graph using the mouse or manually enter values in the window, which opens after clicking on the selected hour



The thermostat responds to the schedule when it is in the `Auto` mode. The choice of the operating mode is made by means of the application or by the methods of the object.



## C. Configuration parameters of the Thermostat object

### FEATURES

Name	Description
Source	Thermostat input, connection to a temperature sensor
Control	Thermostat output, connection with the actuator
OutputType	Determination of the output type (-1 - autodetection, 0 - digital output, 1 - analog output)
PointValue	The value of the temperature set manually
HolidayModeValue	The temperature value for the holiday mode
Hysteresis	Hysteresis value - defining the limits of thermostat activation and deactivation
State	Operation status (1 - active thermostat, 0 - inactive)
ControlDirection	Working direction (0 - normal mode (warming up), 1 - reverse mode (cooling))
Mode	Operating mode (0 - manual mode (using PointValue), 1 - holiday mode (HolidayModeValue), 2 - automatic mode (AutoMode value from the Schedule), 3 - heating mode (HeatUp value))
Data	A string that defines the schedule for changing values
Min	The lower value of the scope of the built-in schedule
Max	The upper value of the scope of the built-in schedule
TargetTemp	The current value of the target temperature
ControlOutValue	The value assigned to the heating control output

### METHODS

Name	Description
Start	Switching thermostat to active state ( <code>State</code> = 1)
Stop	Switching the thermostat to an inactive state ( <code>State</code> = 0)
IncreaseDegree	Increase <code>PointValue</code> by 1 °C
DecreaseDegree	Decrease <code>PointValue</code> by 1 °C
HeatUp	Increasing <code>PointValue</code> by a given value at a specified time
HolidayModeStart	Starting <code>holiday mode</code>
HolidayModeStop	Stopping the <code>holiday mode</code>
AutoModeStart	Starting the <code>AutoMode</code> mode (downloading temperature from the schedule)
AutoModeStop	Stop the <code>AutoMode</code> mode
SetData	Setting the weekly schedule
SetOutputType	Output type setting ( <code>Auto</code> - auto detection, <code>Digital</code> - digital output, <code>Analog</code> - analog output)
SetPointValue	Setting the manually set temperature
SetHolidayModeValue	Setting the temperature value for the holiday mode
SetHysteresis	Setting the hysteresis value
SetControlDirection	Setting the working direction (0 - normal mode (warming up), 1 - reverse mode (cooling))

## EVENTS

Name	Description
OnChange	An event generated when the value of the <code>PointValue</code> property is changed
OnStart	The event is generated when the thermostat is restarted
OnStop	Event generated when the thermostat stops working
OnOutOn	An event dispatched when the value of <code>OutValue</code> is set to a value greater than zero
OnOutOff	An event that is dispatched when the value of <code>OutValue</code> is less than zero
OnHolidayModeOn	An event generated when starting holiday mode
OnHolidayModeOff	An event generated when the holiday mode is turned off

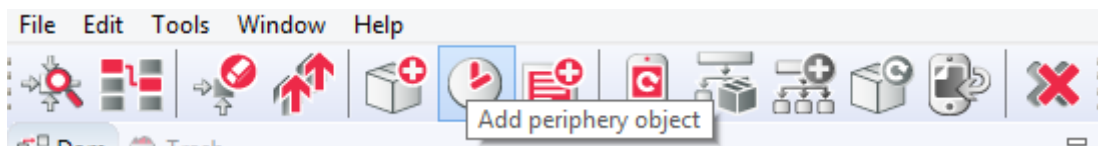
## 6. Push

Push notifications are virtual objects created by the user in the CLU. This object enables sending notifications to the device with MyGrenton app installed. It is possible to create up to 64 calendars on one CLU.

**NOTE!** Push notifications are available for Object Manager version v1.3.3 (or higher) and myGrenton app (Android) version v1.1.9 (or higher) or myGrenton app (iOS) version v1.3.3 (or higher). It is necessary to enable the cloud connection in the CLU (To do this, set the parameter `UseCloud == true` and then send the configuration to the CLU. Correct connection to the cloud will be signaled by the `cloudConnection == true` parameter)

### A. Push creation

To create Push notification, mark the CLU within which you want to create it, then launch `Add CLU object` from the upper menu.



In the open window find and select `Push`, then name it. Windows of features of the newly created Push object will appear on the screen. It contains three tabs

- **Control** - contains push methods;
- **Events** - contains push events;
- **Built-in features** - contains list of push features.

### B. Configuration parameters of the Push object

#### FEATURES

Name	Description
<code>Message</code>	Notification content
<code>Title</code>	Notification title
<code>LastSendTime</code>	The time when the last notification was sent
<code>Interval</code>	Interval between successive notifications (in seconds)

#### METHODS

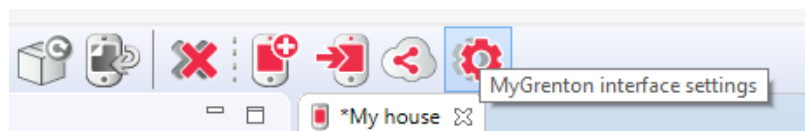
Name	Description
<code>SetMessage</code>	Setting the notification content
<code>ClearMessage</code>	Clearing the notification content
<code>SetTitle</code>	Setting the notification title
<code>ClearTitle</code>	Clearing the notification title
<code>Send</code>	Sending notification to device
<code>SetInterval</code>	Setting the interval between successive notifications (in seconds)

## EVENTS

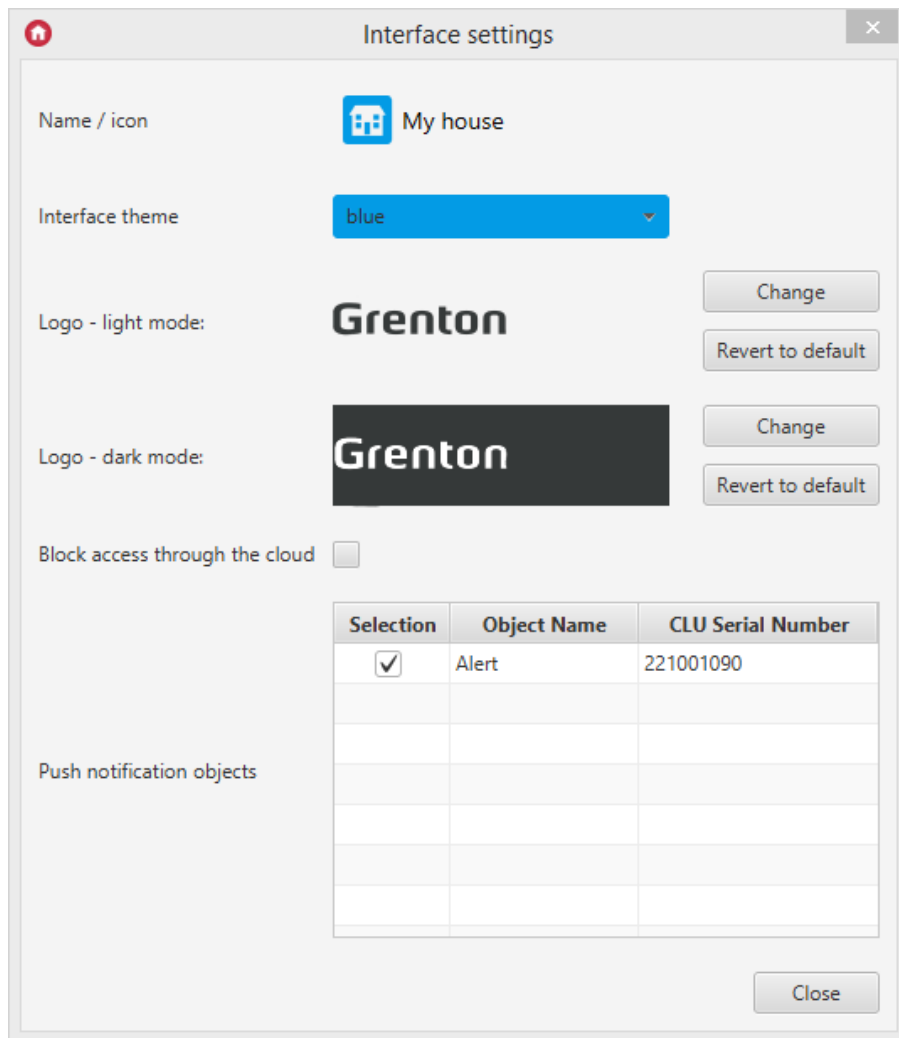
Name	Description
<code>OnSend</code>	An event generated when the message is sent
<code>OnOverflow</code>	An event generated when the queue overflows

## C. MyGrenton configuration

To add a notification to the myGrenton interface, click the `MyGrenton interface settings` link in the toolbar:



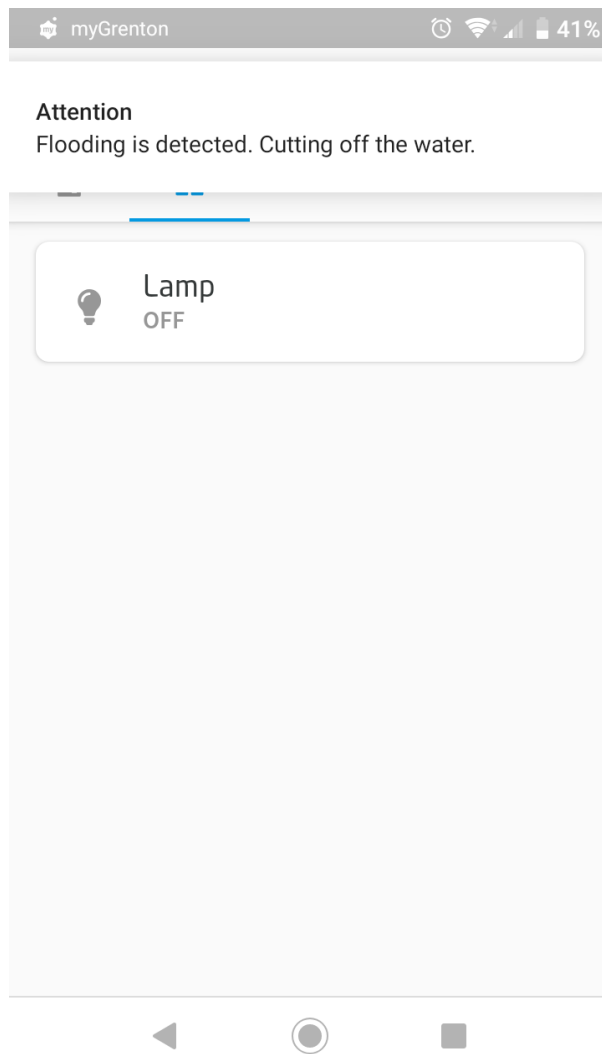
The window with interface settings will appear:



In the `Push notification objects`, select the notifications you want to activate, then send the interface to your mobile device [look up XVII.5.](#)

## D. How push notifications work

After calling the `Send` method, a push notification appears on the device screen.



Sending more notifications from one Push object results in adding them to the queue and appearing on the device at intervals defined by the `Interval` feature.

### QUEUE PROPERTIES

- There can be a maximum of 10 messages in the queue at one time;
- If more than 10 notifications appear in the queue, the `OnOverflow` event will be generated;
- If more than 10 notifications appear in the queue, only the last 10 notifications will be sent to the device.

**NOTE!** Messages sent from different push objects will appear on the device simultaneously.

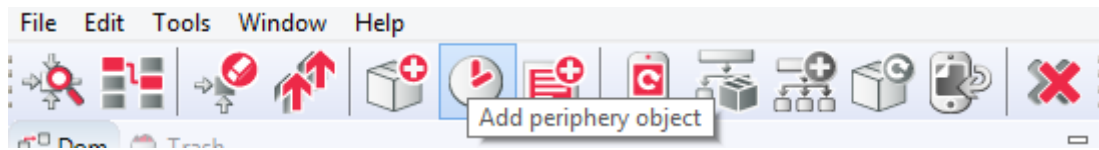
## 7. Presence Sensor

Presence sensor is a virtual object created within a given CLU. Object Manager allows you to create up to 64 objects. The presence detector can be used wherever it is necessary to call the method after a specified time, as well as to count the time from the last call (e.g. after motion detection).

**NOTE!** Functionality of the presence sensor object is available for a CLUZ with firmware 5.07.01 or higher.

## A. Creating a presence sensor

In order to create a presence detector, select the CLU within which it is to be included, and then run `Add CLU object` from the top menu.



After clicking the icon, a selection window with a list of available objects appears, where you need to find and select the `PresenceSensor` object. After selecting, pressing `OK`, it is necessary to give a name to the new presence sensor. The created sensor will appear on the list of objects of the selected CLU.

## B. Presence sensor configuration parameters

### FEATURES

Name	Description
<code>Timeout</code>	Time (in seconds) from last movement detection. When this time elapses <code>PresenceDetected</code> sets on 0
<code>State</code>	Current state of presence sensor, 1 - On, 0 - Off
<code>PresenceDetected</code>	Movement detection state
<code>TimeFromLastPresence</code>	Time (in seconds) from the last movement detection

### METHODS

Name	Description
<code>Start</code>	Start a presence sensor
<code>Stop</code>	Stop a presence sensor
<code>DetectPresence</code>	Method called while presence detection. Sets <code>PresenceDetected</code> value to 1 and resets <code>TimeFromLastPresence</code> timer
<code>SetTimeout</code>	Set a <code>Timeout</code> parameter (in seconds)

### EVENTS

Name	Description
OnStart	A event called when sensor starts
OnStop	A event called when sensor stops
OnPresenceDetected	A event called if a presence detected
OnTimeout	A event called when a timer timeouts

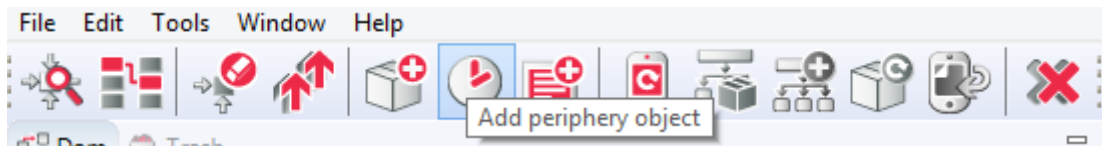
## 8. Sunrise and Sunset Calendar

The sunrise and sunset calendar is a virtual object created within a given CLU. Object Manager allows you to create up to 64 objects. The calendar can be used when certain actions in the system are performed depending on the time of day (sunrise / sunset).

**\*\* NOTE! \*\*** Functionality of Sunrise and Sunset Calendar object is available for a CLUZ with firmware 5.07.01 or higher.

### A. Create a calendar

In order to create a sunrise and sunset calendar, select the CLU within which it is to be included, and then, from the top menu, start `Add CLU object`.



After clicking the icon, a selection window with a list of available objects appears, where you need to find and select the `SunriseSunsetCalendar` object. After selecting, pressing `OK`, it is necessary to name the new calendar. The created calendar will appear on the list of objects of the selected CLU.

### B. Calendar configuration parameters

#### FEATURES



Name	Description
<code>Longitude</code>	Longitude in decimal degrees (DD), range -180 to 180
<code>Latitude</code>	Latitude in decimal degrees (DD), range -90 to 90
<code>State</code>	Current state of the sunrise and sunset calendar: 1 - On, 0 - Off
<code>SunriseUTC</code>	UTC sunrise time for a selected location ( $\pm$ 5 minutes) N/A - Unable to calculate sunrise time
<code>SunsetUTC</code>	UTC sunset time for a selected location ( $\pm$ 5 minutes) N/A - Unable to calculate sunset time
<code>SunriseLocal</code>	Sunrise local time for a selected location ( $\pm$ 5 minutes) N/A - Unable to calculate sunrise time for a selected location
<code>SunsetLocal</code>	Sunset local time for a selected location ( $\pm$ 5 minutes) N/A - Unable to calculate sunset time for a selected location

## METHODS

Name	Description
<code>Start</code>	Starts sunrise and sunset calendar
<code>Stop</code>	Stops sunrise and sunset calendar
<code>SetLongitude</code>	Set a longitude in decimal degrees (DD), range -180 to 180
<code>SetLatitude</code>	Set a latitude in decimal degrees (DD), range -90 to 90

## EVENTS

Name	Description
<code>OnStart</code>	A event called when calendar starts
<code>OnStop</code>	A event called when calendar stops
<code>OnSunrise</code>	A event called when the sun rises
<code>OnSunset</code>	A event called when the sun sets
<code>OnSunriseSunsetChange</code>	A event called when the sun rises or sets

# X. Media measurement

## 1. Virtual media measurement

### 1.1. Launching media measurement on the Object Manager page

The Object Manager allows you to perform a media measurement, which allows the estimated presentation of the energy consumed (based on the time the device is turned on and the receiver power specified in the configuration). The media measurement configuration takes place in OM and it must be run for each input and output separately - so that the CLU collects data on energy consumption. Media measurement is recorded every 15 minutes, starting the countdown from the full hour - based on the CLU clock (*CLU feature -> TIME*).

**NOTE!** Media measurement is available for Object Manager version 1.2.0.180202 and higher, and for CLU with firmware 04.07.29-1802 and higher.

Media measurement can be run for modules:

- Input (Digital IN) - in the continuous mode (counting the working time) or pulse (counting the pulses appearing at the binary input):

**Object properties**

Name: x181000775\_DIN1 Device type: [dropdown]  
Id: CLU221001090->DIN8273 Serial number: 181000775 | 1  
Type: DIN

Control User schemes Events Embedded features Statistics

Feature name	Current value	Initial value	Unit	Range
Inertion	0	0	ms	[0-100]
HoldDelay	500	500	ms	[100-5000]
HoldInterval	100	100	ms	[100-2000]
Value	0		bool	[0-1]
DistributedLogicGroup	0	0		[0-10000]
StatisticState	0	Continuous	number	0,1,2
Load	0	Continuous	number	

Auto refresh Refresh

OK Cancel

- Output (Relay, Led RGB, Dimmer) - in continuous mode (counting working time):

**Object properties**

Name: x201000275\_DOUT1 Device type: ▼

Id: CLU221001090->DOU5138 Serial number: 201000275 1

Type: DOUT

Control User schemes Events **Embedded features** Statistics

Feature name	Current value	Initial value	Unit	Range
Value	0	Off ▼	bool	0,1
StatisticState	1	Continuous ▼	number	0,1
VoltageType	2	Off Continuous		0,1,2
VoltageValue	230	230	V	[0-230]
Power	0		W	[0-3000]
Overload	3000	3000	W	[0-3000]
DistributedLogicGroup	0	0		[0-10000]
Load	60	60	number	

Auto refresh  Refresh

OK Cancel

**NOTE!** The media measurement of the above-mentioned modules applies to modules for DIN rail and flush-mounted Tf-bus! The measurement setting is not available for Z-Wave modules!

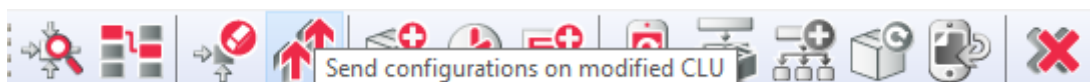
## A. Creating a configuration

To create a configuration:

- Double click on the selected module from the list of modules in the main view of the program (this applies to the above-mentioned modules for media measurement support);
- Go to the tab *Embedded features*;
- Change the selection of the `StatisticState` feature to: `Continuous` or `Pulse` (for binary inputs of the Digital In module);
- The `Load` item will appear below - to its initial value, enter the active power input of the device connected to the input or output in watts per hour (eg 60) - CLU will recalculate the given value continuously (multiplying by time in hours):

<b>StatisticState</b>	1	Continuous ▼	number	0,1,2
<b>Load</b>	60	60	number	

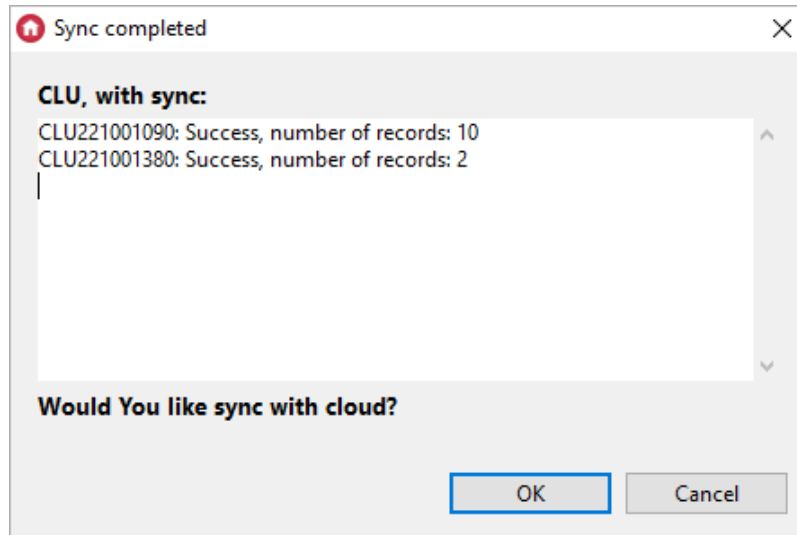
- Confirm with *OK*;
- Add media measurement settings for subsequent modules - repeat the above steps;
- Send the configuration to the CLU.



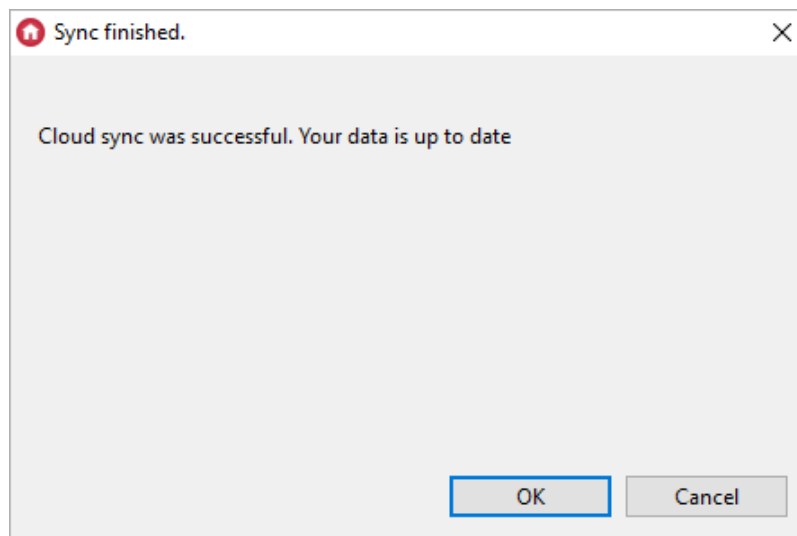
## B. Read media measurement in the Object Manager

In order to read the media measurement in the Object Manager program:

- Wait at least for the first scheduled measurement recording by the CLU (up to XX.00 or XX.15 or XX.30 or XX.45 - where XX is the hour);
- Select **Tools** -> **Download file with measurements**;
- A window will be displayed with information about downloaded records:

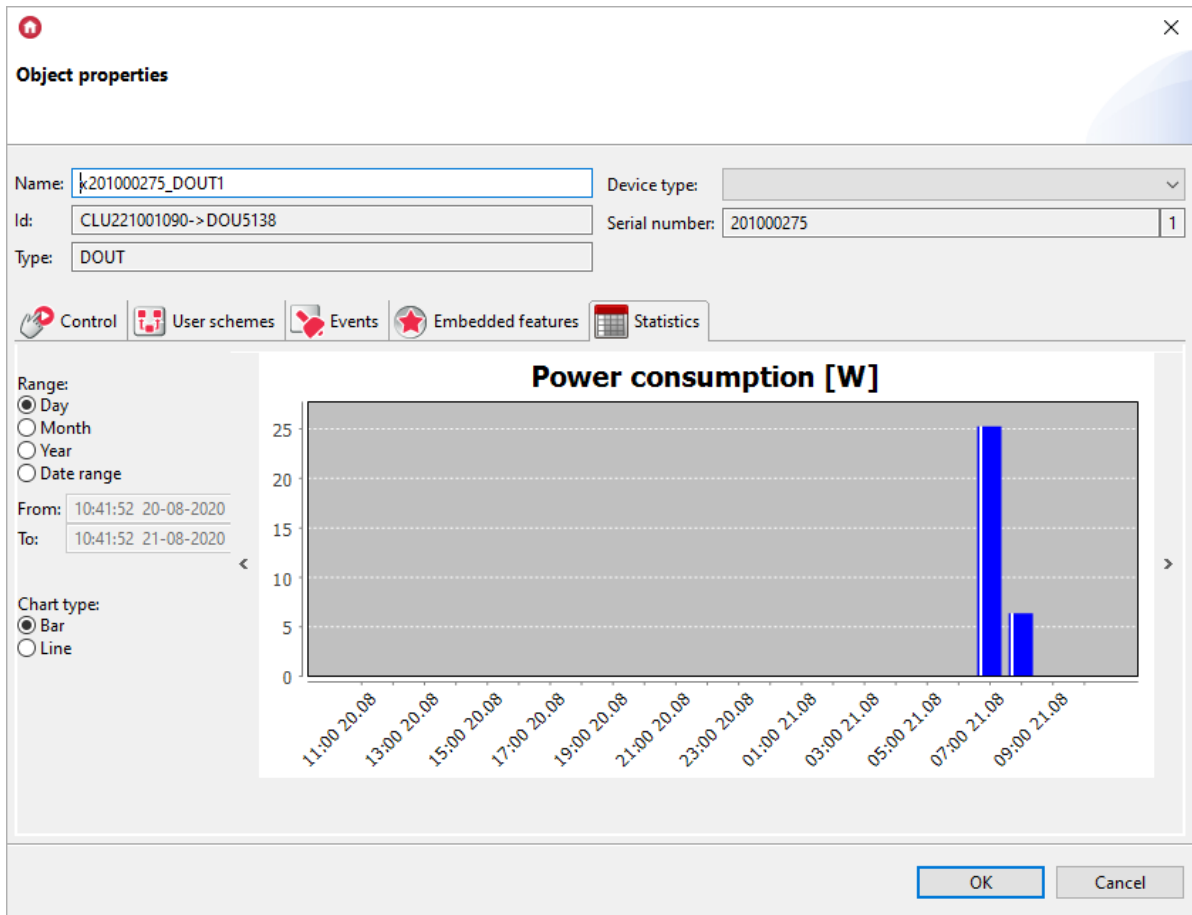


- Click *OK*;
- The Object Manager will then synchronize the downloaded data with the cloud;
- After the synchronization is completed, press *OK*:



**NOTE!** In case of a synchronization error, please contact Support!

- In order to make sure that the media measurement has been registered, double-click on the selected module for which the media measurement has been run;
- Then go to the *Statistics* tab:
  - You can choose the type of graph displayed: bar or line - in both cases, the total amount of energy consumed (in watts) for each hour appears on the chart;
  - You can also select the interval of the media measurement viewed: day, month, year or manually select the date range - depending on the selected interval, the corresponding graph will be displayed.



### C. Configure the media measurement for the Home Manager application interface

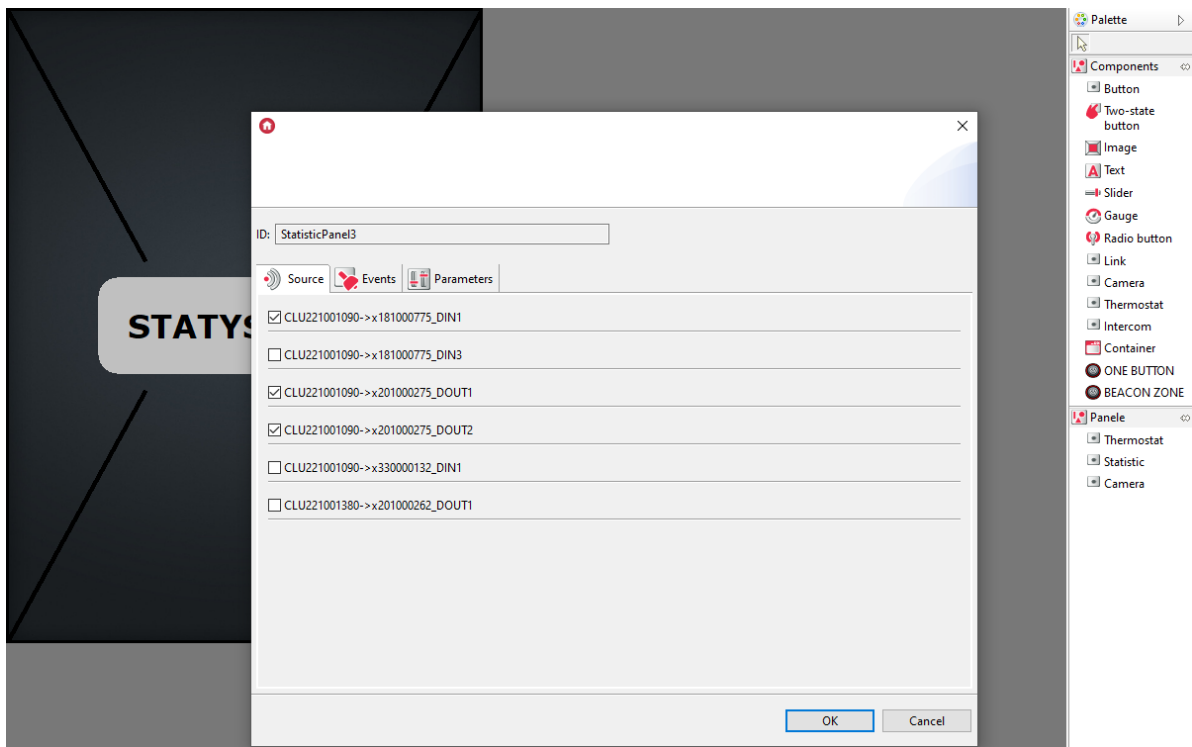
**NOTE!** Support for Visual Builder functionality has been ended. The Home Manager interface creator will be removed in the next versions of Object Manager.

The media measurement configuration for the application interface must follow the following diagram:

- Add a new application interface:



- Enter the name of the application being created;
- Set resolution, skin, add at least one page, click *OK*;
- From the panel tray, drag the panel *Statistics* to the editable area of the application interface;
- In the *Source* tab, select check boxes for modules whose media measurement graphs are to be displayed in the statistics panel in the application:



- Click *OK*;
- Send the interface to the mobile device- [look up VIII.4.7.](#)

## 1.2. Using media measurement on the Home Manager application side

**NOTE!** Media measurement is available for Home Manager version 1.1.110 or higher.

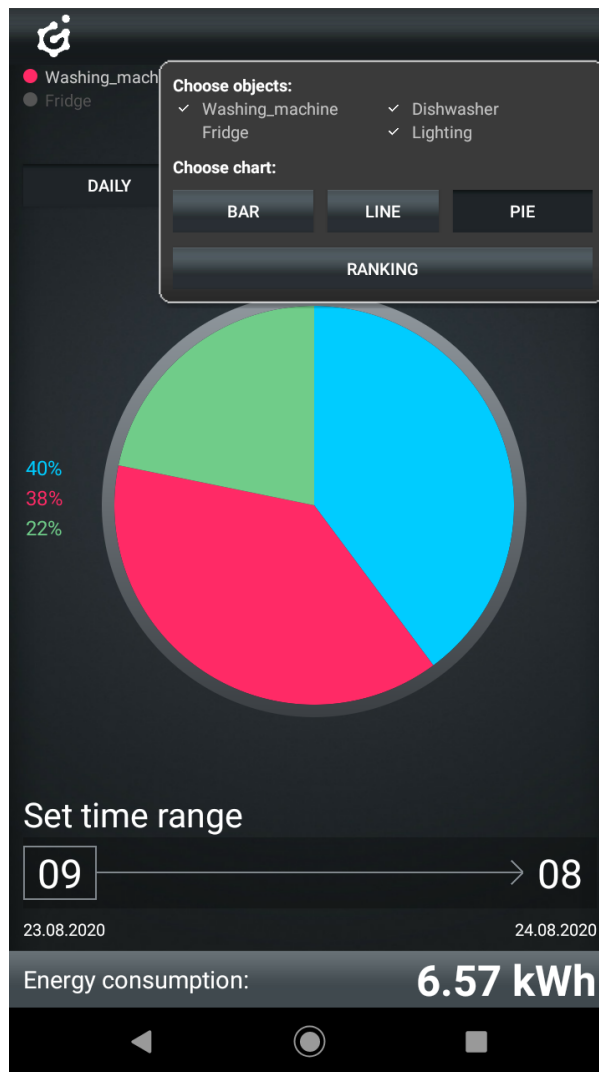
To properly use the media measurement in the mobile application, first take measurements from the CLU and - if necessary - synchronize the measurements with the cloud.

### A. Taking measurements

- Enter the application settings from the main menu (gear icon).
- Select from the list of settings: *Download measurements from the CLU.*
- After a moment, the following message will be displayed: *Success for CLU: X, Y<sup>6</sup>.*
- Launch the application interface - the measurements should be updated and displayed on the chart.

### B. Media panel view options

- Change of displayed data of specific *I / O* - after clicking on the modules listed, the upper bar of the media measurement panel displays a window of available modules added to the panel, which are selected by default - their unchecking results in the lack of showing measured values for specific *I / O*:

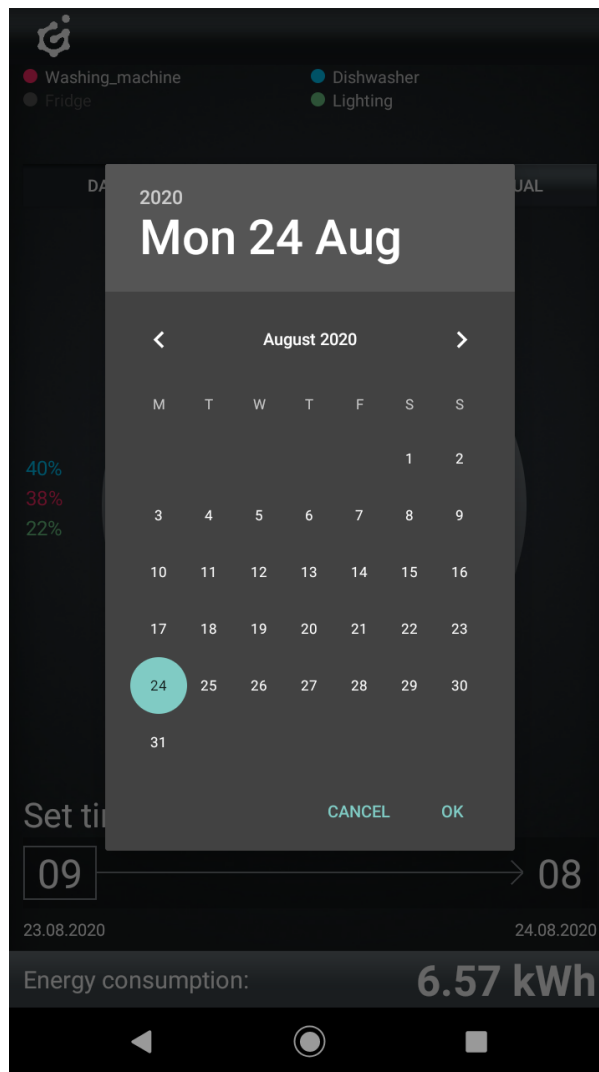


- In the same window where the modules are visible, it is possible to change the graph view - the default is a line graph, but you can also select a bar, pie or ranking;



- Change of the time range of the displayed waveforms - this can be done using the "daily" buttons (summing measurements for each hour of the day), "monthly" (adding up values for each day in the month) and "annual" (adding up the measurements for each month separately);
- It is also possible to choose your own time range - after clicking on a given hour, the window for selecting the start and end day is displayed:





## C. Synchronization and downloading of measurements

- Taking measurements from the CLU, which was done before, took place at the local connection with the CLU. For the measurements to be displayed during remote access, synchronize them with the cloud;
- In order to synchronize the measurements with the cloud, enter the Main menu of the Home Manager application - in the settings and at the bottom select: *Synchronize measurements with the cloud*.

## 2. Real media measurement

**NOTE!** Real media measurement is only available for modules of the Grenton 2.0 series: GRENTON RELAY 2HP (DIN), GRENTON RELAY 4HP (DIN), GRENTON ROLLER SHUTTER (DIN), GRENTON ROLLER SHUTTER (Flush), GRENTON I/O MODULE 2/2(Flush).

## 2.1. Real media measurement settings in Object Manager

Object Manager allows you to carry out media measurement, which allows real presentation of consumed energy (based on the `VoltageValue` and `VoltageType` parameters of the device). The configuration of media measurement is done in OM and should be parameterized for each output separately. In order for the media to be measured correctly, it is necessary to determine the electrical parameters of the network to which the entire system is connected. For this purpose, in the embedded features of the CLU module, define the frequency (`VoltageFrequency`) and the rated voltage (`DefaultVoltageValue`) of the network.

**CLU properties**

Name:  Serial number:   
IP:  FW:

Control  Events  Embedded features  User features

<b>Minute</b>	20	m	[0-59]
<b>LocalTime</b>	1621869627	s	
<b>FirmwareVersion</b>	05.07.02-212		
<b>UseCloud</b>	false	False	bool
<b>CloudConnection</b>	false		bool
<b>VoltageFrequency</b>	50	50Hz	Hz 50,60
<b>DefaultVoltageValue</b>	230	230	V
<b>NTPServer</b>	tempus1.gun	tempus1.gum.gov.pl	
<b>TimeZone</b>	0	Europe/Warsaw	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
<b>QoS</b>	0	QoS0	0,1
<b>PrimaryDNS</b>	8.8.8.8	8.8.8.8	string
<b>SecondaryDNS</b>	8.8.4.4	8.8.4.4	string

Auto refresh

Media measurement is recorded in real time. `Power` features are expressed in watts for relay output modules and `LoadCurrent` are expressed in milliamperes for blind drive control modules.

The `VoltageType` feature takes the values:

- For DOUT objects: 0-AC, 1-DC, 2-Signal

✖

### Object properties

Name:  Device type:

Id:  Serial number:

Type:

Control
  User schemes
  Events
  Embedded features
  Statistics

Feature name	Current value	Initial value	Unit	Range
<b>Value</b>	1	Off <input type="text" value=""/>	bool	0,1
<b>StatisticState</b>	0	Off <input type="text" value=""/>	number	0,1
<b>VoltageType</b>	0	AC <input type="text" value=""/>		0,1,2
<b>VoltageValue</b>	230	<input type="text" value="230"/>	V	[0-230]
<b>Power</b>	10		W	[0-3000]
<b>Overload</b>	3000	<input type="text" value="3000"/>	W	[0-3000]
<b>DistributedLogicGroup</b>	0	<input type="text" value="0"/>		[0-10000]

Auto refresh

- For ROLLER SHUTTER objects: 0-AC, 1-DC

✖

### Object properties

Name:  Device type:

Id:  Serial number:

Type:

Control
  User schemes
  Events
  Embedded features
  Statistics

Feature name	Current value	Initial value	Unit	Range
<b>State</b>	1		-	0,1,2
<b>MaxTime</b>	30000	<input type="text" value="30000"/>	ms	
<b>Up</b>	1			0,1
<b>Down</b>	0			0,1
<b>LoadCurrent</b>	106		mA	
<b>Overcurrent</b>	1600	<input type="text" value="1600"/>	mA	
<b>VoltageType</b>	0	AC <input type="text" value=""/>		0,1

Auto refresh

# XI. CLU service functions

## 1. Restoring factory settings CLU - *Hard Reset*

Activating the *Hard Reset CLU* function results in:

- Removal of the saved configuration;
- Formatting the flash memory partition;
- Removal of all created LUA objects;
- Clearing all data of the Z-Wave controller;
- Removal of information about connected Z-Wave modules.

In order to restore the factory settings of the CLU with the *Hard Reset* function, perform the following steps (in accordance with the order given):

- Disconnect power from the CLU module;
- Press and hold the *Link* button on the module;
- Connect the power supply to the CLU module;
- Keep the *Link* button depressed for at least 10 seconds - both LEDs on the CLU will be permanently illuminated;
- After 10 seconds, release the *Link* button - the correct execution of the reset will be confirmed by a blink of both LEDs 5 times.

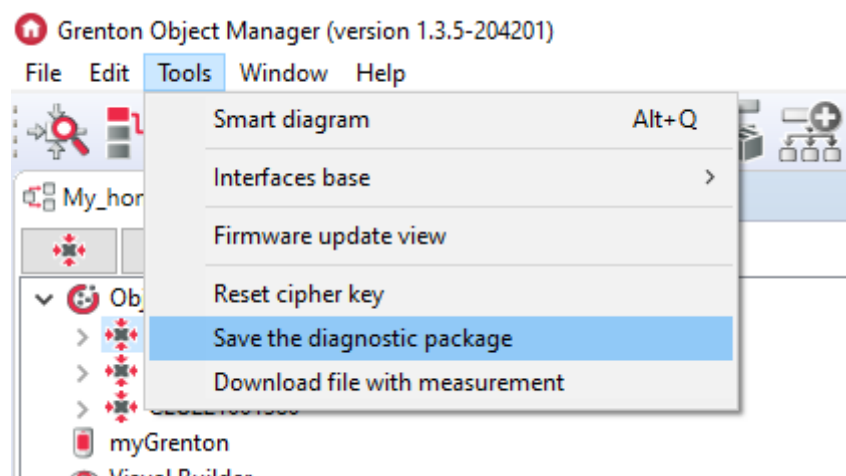
**NOTE!** If Z-Wave modules were added to the CLU before starting the *Hard Reset* function, after performing the reset it will be necessary to perform the procedure of deleting and re-adding each Z-Wave module!

## 2. System diagnostics - *Save the diagnostic package*

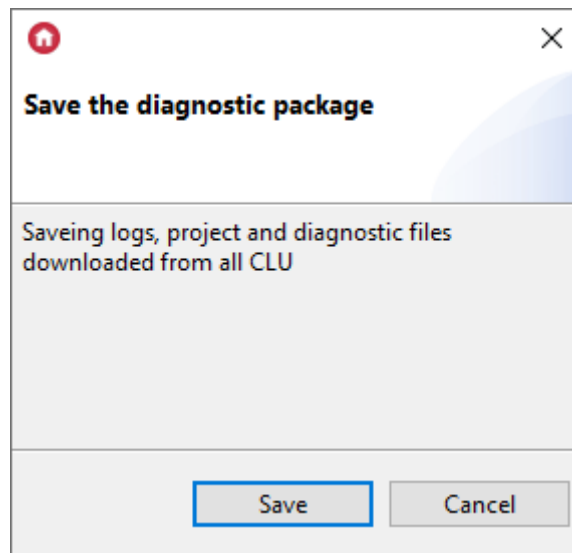
*Diagnostic package* is used for CLU diagnostics and for quick finding of problems in the created project.

In order to diagnose the system, you should:

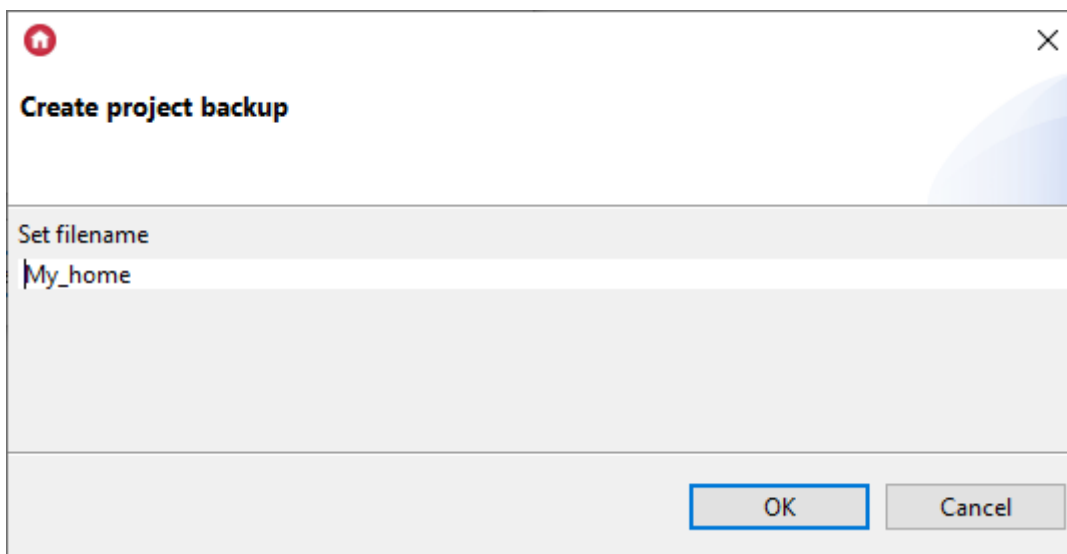
- Open the project in the Object Manager;
- Select **Tools** from the taskbar and then *Save the diagnostic package*:



- In the window that opens, select *Save*:



- Specify where to save the file and give the backup name:



- Then the package will appear in the selected location in \*.zip\* format. The content will be as follows:

CLU220001096	22.10.2020 11:11	
CLU221001090	22.10.2020 11:11	
CLU221001380	22.10.2020 11:11	
logs	22.10.2020 11:11	
interfaces.zip	22.10.2020 11:11	544 KB
My_home.zip	22.10.2020 11:11	217 KB
My_home_backup_20-10-22_11-11.omp	22.10.2020 11:11	215 KB

- The package created in this way contains:
  - folders with configuration files of all CLUs;
  - "logs" folder containing the file with the specified application logs;
  - .zip package containing the interface database used in the project;
  - package .zip containing information about the project;
  - project backup file.

# XII. SMART PANEL

---

## 1. Smart Panel equipment

---

Smart Panel consists of:

- OLED display;
- Four touch buttons;
- A sensor that recognizes four gestures;
- proximity / presence sensor;
- Temperature sensor;
- Light intensity sensor;
- Buzzer.

## 2. Connection of the Smart Panel to the CLU

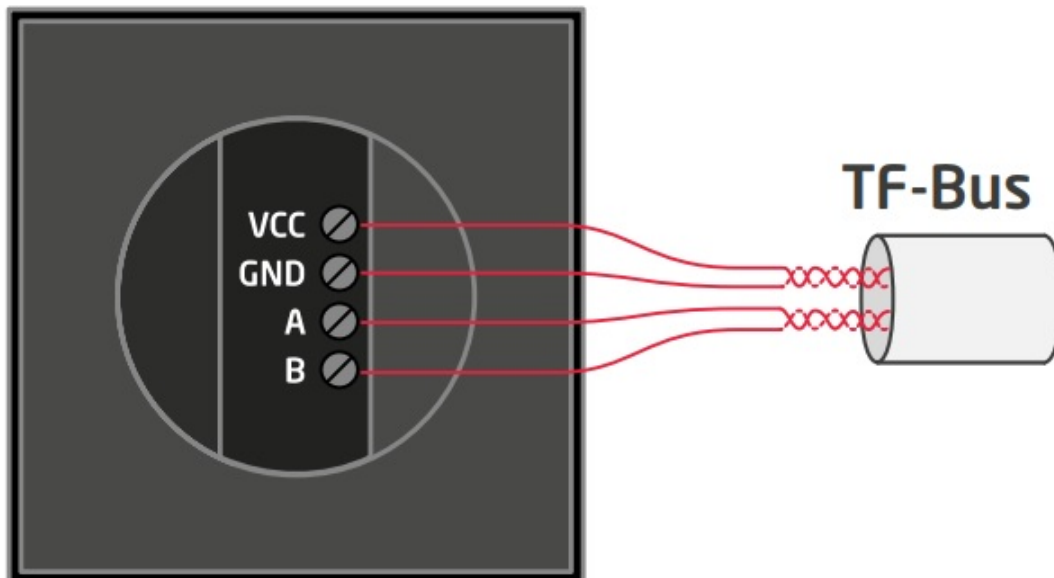
---

**NOTE!** Smart Panel is available for Object Manager version 1.2.0.180202 and higher, and for CLU with firmware 04.07.29-1802 and higher.

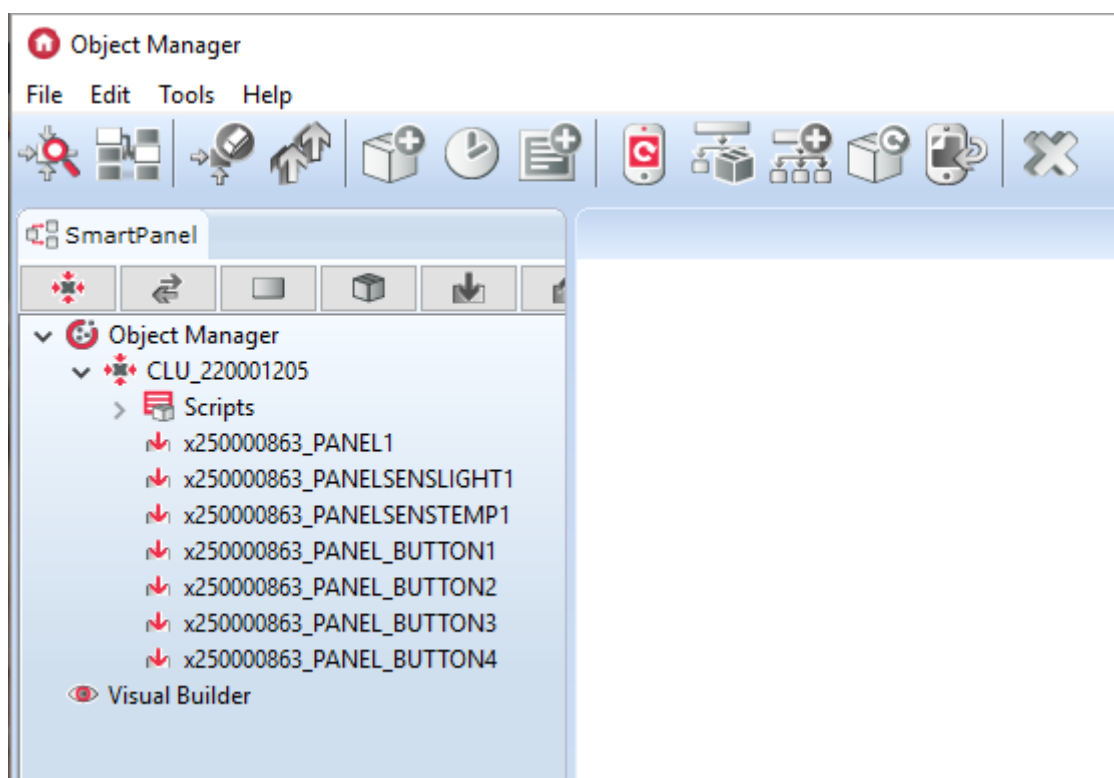
**NOTE!** Smart Panel version v4 is available for Object Manager in version 1.2.1.190201 and higher, and for CLU with firmware 04.07.49-1912 and higher.

Connection of the Smart Panel to the system takes place by means of twisted-pair cable. To the appropriate terminals of the ARK connector, two pairs of twisted wires should be derived from the Smart Panel - the connection diagram is shown in the figure below:

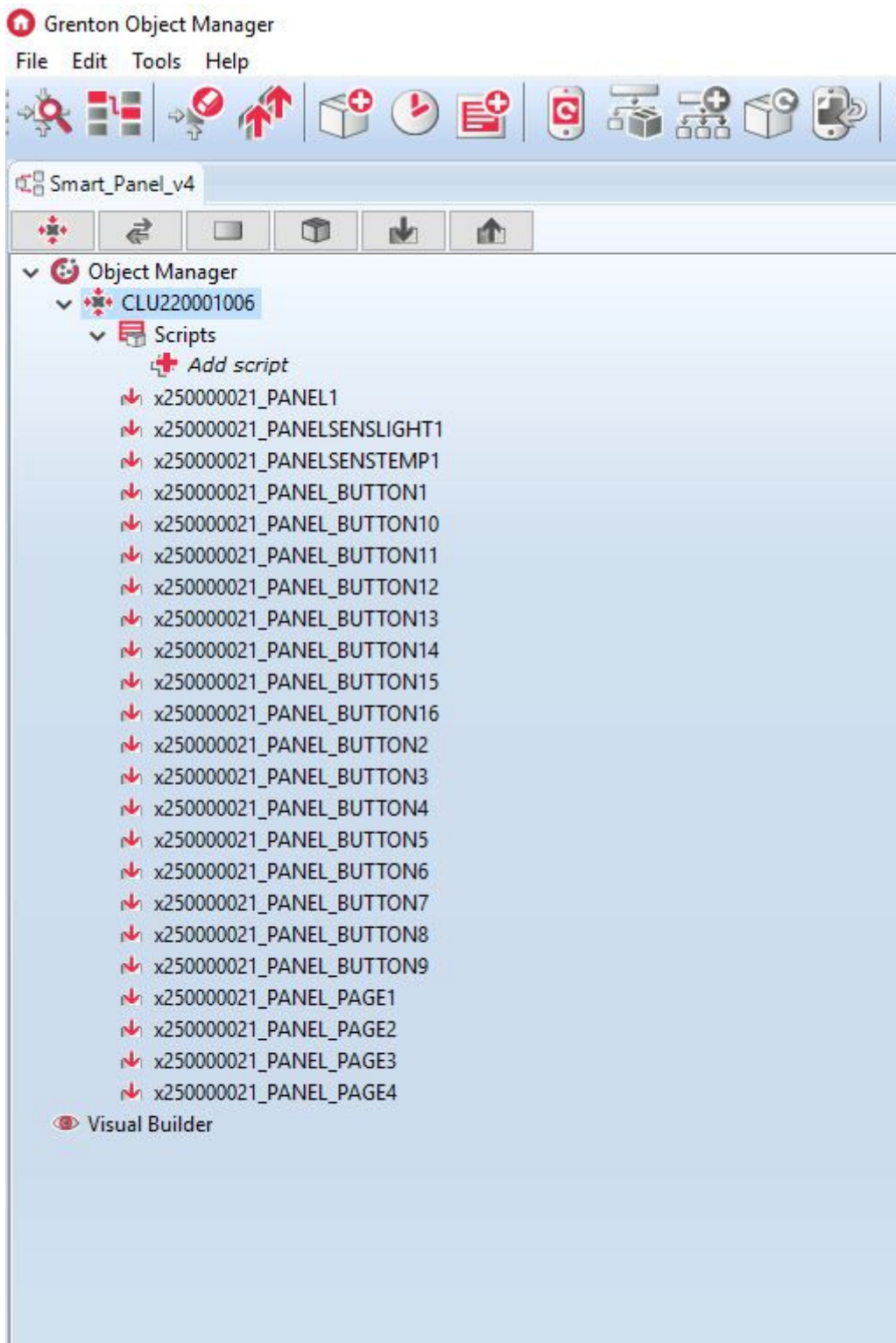
- Connect one lead from the first twisted pair (eg UTP cable) to the Vcc terminal;
- Connect the other wire from the pair to the GND terminal;
- Connect one cable from the other pair to terminals A and B.



After connecting and carrying out the *CLU Discovery* operation in the project, the following Smart Panel v3 elements will appear in the list of modules:



After connecting and carrying out the *CLU Discovery* operation in the project, the following Smart Panel v4 elements will appear in the list of modules:



If you correctly add elements to the project, you can proceed to create a configuration.

**NOTE!** In case of failure, please contact Support!

### 3. Information to help you create a configuration

1. The configuration of the panel with the display differs from the configuration of the classic Grenton touch panel, inter alia, that in addition to: features, methods and events of each button, temperature / light sensor, the user also has: a gesture sensor, as well as features, methods and events for the *Smart Panel* only.



From version 04.03.04.1910 new Smart Panel functionalities are available, such as the PANEL\_PAGE configuration object or new features, methods and events in the PANEL object.

2. The display, in which the touch panel is equipped, has a resolution of 128x64 pixels.
3. Smart Panel v3 can work in two modes: displaying icons (display is divided into 4 fields) or in drawing mode (using the entire display field).

Smart Panel v4 can work in four modes:

1. Backward compatibility mode (default configuration) - `Inactive`,
  2. Icon display mode (display divided into 4 fields) - `Buttons`,
  3. Drawing mode (using the entire display field) - `FreeDraw`,
  4. Operating mode of thermostats - `Thermostats`.
4. The touch panel is equipped with a microSD card slot, which is used to store the default icons displayed on the panel. Files must be placed in the main directory of the card with the extension `.bmp`.
  5. The Smart Panel screen is blank by default. It lights up when the proximity sensor is activated (display time is taken from the `panel -> ProximityTimeout` feature - after this time the panel does not detect presence, the display turns off).
  6. The presence sensor operates depending on the distance set using the sensitivity - the `ProximitySens` feature. Upon detection of presence, the `OnProximityDetect` event is generated.

## 4. Configuration of the Smart Panel module in the version v3

### 4.1. Configuration parameters

#### A. Panel

##### FEATURES

Name	Description
<code>GestureIconUp</code>	The name of the BMP file with the icon for the up gesture (no extension)
<code>GestureIconDown</code>	BMP file name with icon for gesture down (no extension)
<code>GestureIconLeft</code>	BMP file name with icon for gesture left (no extension)
<code>GestureIconRight</code>	The name of the BMP file with the icon for the right gesture (no extension)
<code>ProximitySens</code>	Sensitivity of the proximity sensor
<code>ProximityTimeout</code>	The time after which the display will be turned off
<code>ProximityValue</code>	Proximity sensor signal (non-dimensional value)
<code>BuzzerValue</code>	Control of sound signaling (on / off)

##### METHODS

Name	Description
SwitchOnDisplay	It wakes the display from sleep mode
ShowButtons	Changes the display mode to <i>buttons</i>
ClearScreen	Cleans the display content in <i>freedraw mode</i>
PrintText	Display text in <i>freedraw</i>
PrintFloat	Displays the number in <i>freedraw</i>
DrawLine	Draws the line in <i>freedraw</i>
DrawPoint	Draws a point in <i>freedraw</i>
DrawIcon	Draws the icon (bmp) in <i>freedraw</i>
DisplayContent	Displays the contents of the graphic memory buffer; changes the display mode to <i>freedraw</i>
SetGestureIconUp	Sets the BMP file with the icon for the up gesture
SetGestureIconDown	Sets the BMP file with the icon for the down gesture
SetGestureIconLeft	Sets the BMP file with the icon for the left gesture
SetGestureIconRight	Sets the BMP file with the icon for the right gesture
SetProximitySens	Sets the sensitivity of the proximity sensor
SetProximityTimeout	Sets the time after which the display will be dimmed
SetBuzzerValue	Enables / disables sound signaling

## EVENTS

Name	Description
OnGestureUp	An event related to a gesture up
OnGestureDown	An event related to a gesture down
OnGestureLeft	An event related to a gesture left
OnGestureRight	An event related to a gesture right
OnProximityDetect	An event triggered when a person is detected approaching the panel display

## B. Buttons

### FEATURES

Name	Description
Mode	Returns the set operation mode of the button: 0 - monostable, 1 - bistable, 2 - locked (the diode is red with continuous light)
HoldDelay	The time (in milliseconds) after which the OnHold event will be triggered (when pressing and holding the button)
HoldInterval	Cyclic time interval (in milliseconds), after which when the button is held the OnHold event will be triggered
Value	Returns the input state (0 or 1)
Label	Text describing the button (displayed instead of the icon)
IconA	File name of the icon assigned to the button in monostable and bistable mode in the OFF position; the name preceded by "~" will display the graphic in negative; IconA has priority over the Label feature
IconB	The file name of the icon assigned to the button in bistable mode in the ON position; the name preceded by "~" will display the graphic in negative

### METHODS

Name	Description
SetMode	Sets the mode of the button operation: 0 - monostable, 1 - bistable, 2 - locked (the diode is permanently red)
SetHoldDelay	Sets the value of HoldDelay
SetHoldInterval	Sets the value of HoldInterval
SetLabel	Sets the text describing the button
SetIconA	Sets the A icon file
SetIconB	Sets the B icon file
ShowOK	Blinks the green LED on the button for two seconds (frequency 500ms)
ShowError	Blinks the red LED on the button for two seconds (frequency 500ms)
LedSwitchOn	It turns on the green LED on the button
LedSwitchOff	Turns off the green LED on the button

### EVENTS

Name	Description
<code>OnChange</code>	An event dispatched when the state changes (regardless of the value)
<code>OnSwitchOn</code>	An event that is triggered when the high state on input is set
<code>OnSwitchOff</code>	An event dispatched when the low state on input is set
<code>OnShortPress</code>	The event is triggered after pressing the button for 500 - 2000 ms
<code>OnLongPress</code>	The event is called after pressing the button for the 2000 - 5000 ms
<code>OnHold</code>	An event that is called for the first time after the <code>HoldDelay</code> time has elapsed, and then cyclically every time <code>HoldInterval</code>
<code>OnClick</code>	The event is triggered after pressing the button for less than 500ms

## C. Temperature and lighting sensors

### FEATURES

Name	Description
<code>Threshold</code>	Hysteresis size (accuracy 0.1) specifying the sensitivity at which events are generated: <code>OnChange</code> , <code>OnLowerValue</code> , <code>OnRaiseValue</code>
<code>Sensitivity</code>	Time (in ms) for which the sampled values are averaged
<code>MinValue</code>	The minimum value of the <code>Value</code> property that is triggered by the <code>OnOutOfRange</code> event
<code>MaxValue</code>	The maximum value of the <code>Value</code> property, which is exceeded by the <code>OnOutOfRange</code> event
<code>Value</code>	Input value: for temperature sensor (from 0 to 45 ° C) or for light sensor (0 - 100%)

### EVENTS

Name	Description
<code>OnChange</code>	Event triggered when the input state changes (regardless of value)
<code>OnRaiseValue</code>	An event triggered when the upper hysteresis threshold is exceeded
<code>OnLowerValue</code>	An event triggered when the hysteresis threshold is exceeded
<code>OnOutOfRange</code>	The event is dispatched when the output value is outside the specified range

## 4.2 Creating button and display configurations

In order to create a configuration:

- Open the *PANEL\_BUTTONX* object (where X is the number of one of the 4 buttons) by double clicking on the list of modules;
- Go to the tab *Events*;
- Configure the operation of the button by assigning methods to specific events (by clicking on "+" on the right side of the window):

The screenshot shows a configuration window titled "CLU\_220001205->x240000392\_BUTTON1". The window has a header bar with a home icon and a close button. Below the header, there are fields for "Name" (k240000392\_BUTTON1), "Source/Receiver" (a dropdown menu), "Identification" (240000392), and "Type" (BUTTON). There are five tabs: "Control", "User schemes", "Events" (selected), "Embedded features", and "Statistics". The "Events" tab contains a table with columns "Event name", "Assigned commands", and "Add command". The table has seven rows: "OnChange" with command "CLU\_220001205->x190000558\_DOUT1->Switch(0)", "OnSwitchOn" with command "CLU\_220001205->Heater->SwitchOn(0,500)", "OnSwitchOff" with command "CLU\_220001205->Heater->SwitchOff(0,500)", "OnShortPress", "OnLongPress", "OnHold", and "OnClick". Each row has an "Assign command" button with a red 'X' icon and an "Add command" button with a red '+' icon. At the bottom right, there are "OK" and "Cancel" buttons.

- Select the tab *Embedded features* and define the objects displayed on the screen of a given button:
  - `Label` - a feature defining the text assigned to a given button;
  - `IconA` - a feature that defines the name of the icon assigned to a given button when it is in monostable mode or for bistable mode for the `Value = 0` attribute;
  - `IconB` - a feature that specifies the name of the icon assigned to a given button when it is in bistable mode for the `Value = 1` property. To assign the same icon, but with the inverted colors, the prefix name should be preceded by the "~" character (eg `~ lamp1on`):

CLU\_220001205->x250000863\_PANEL\_BUTTON1

Name:  Source/Receiver:

Identification:   Type:

Control User schemes Events Embedded features Statistics

Feature name	Current value	Initial value	Unit	Range
Mode	1	<input type="text" value="Bistable"/>		0,1,2
HoldDelay	1000	<input type="text" value="1000"/>	ms	[0-5000]
HoldInterval	100	<input type="text" value="50"/>	ms	[0-2000]
Value	0		bool	0,1
Label	-	<input type="text" value=""/>	string	[0-15]
IconA	lamp2off	<input type="text" value="lamp2off"/>	string	[0-9]
IconB	~lamp2on	<input type="text" value="~lamp2on"/>	string	[0-9]

Auto refresh

The above features can be set both in the tab *Built-in features*, as well as via the methods:

`SetLabel`, `SetIconA`, `SetIconB`.

**NOTE!** The `SetIcon` method has a higher priority in the system than the `SetLabel` method!

- Send the configuration to the CLU.

## 4.3 Creating a gesture sensor configuration

To create a configuration for a gesture sensor:

- Open - by double click - object *Panel*;
- Go to the tab *Events*;
- Assign methods to the events `OnGestureUp`, `OnGestureDown`, `OnGestureLeft`, `OnGestureRight` (clicking on the '+' on the right of each method):

CLU\_220001205->x250000863\_PANEL1

Name:  Source/Receiver:

Identification:  5 Type:

Control User schemes Events Embedded features Statistics

Event name	Assigned commands	Add command
OnGestureUp	<input type="text" value="CLU_220001205-&gt;Heater-&gt;SwitchOn(500,500)"/> Assign command ✕	<input type="button" value="Add command"/>
OnGestureDown	<input type="text" value="CLU_220001205-&gt;Heater-&gt;SwitchOff(500,500)"/> Assign command ✕	<input type="button" value="Add command"/>
OnGestureLeft	<input type="text" value="CLU_220001205-&gt;x250000863_PANEL_BUTTON1-&gt;ShowOK()"/> Assign command ✕	<input type="button" value="Add command"/>
OnGestureRight	<input type="text" value="CLU_220001205-&gt;x250000863_PANEL_BUTTON1-&gt;ShowOK()"/> Assign command ✕	<input type="button" value="Add command"/>
OnProximityDetect		<input type="button" value="Add command"/>

It is possible to substitute icons displayed by default when calling gestures - for this purpose go to the tab *Built-in features* and enter the names of the desired icons without the *.bmp* extension:

CLU\_220001205->x250000863\_PANEL1

Name:  Source/Receiver:

Identification:  5 Type:

Control User schemes Events Embedded features Statistics

Feature name	Current value	Initial value	Unit	Range
GestureIconUp	~lamp3on	<input type="text" value="~lamp3on"/>	.bmp	[0-9]
GestureIconDown	lamp3off	<input type="text" value="lamp3off"/>	.bmp	[0-9]
GestureIconLeft	minus	<input type="text" value="minus"/>	.bmp	[0-9]
GestureIconRight	plus	<input type="text" value="plus"/>	.bmp	[0-9]
ProximitySens	3	<input type="text" value="3"/>		[2-100]
ProximityTimeout	5000	<input type="text" value="5000"/>	ms	[1000-60000]
ProximityValue	373		-	
BuzzerValue	1	<input type="text" value="On"/>		0,1

Auto refresh

The use of icons will be possible when they will be loaded onto a microSD card with the extension *.bmp*.

- Confirm the configuration window with *OK*;
- Send the configuration to the CLU.

## 4.4 Configuration of the proximity sensor

To set the proximity sensor parameters:

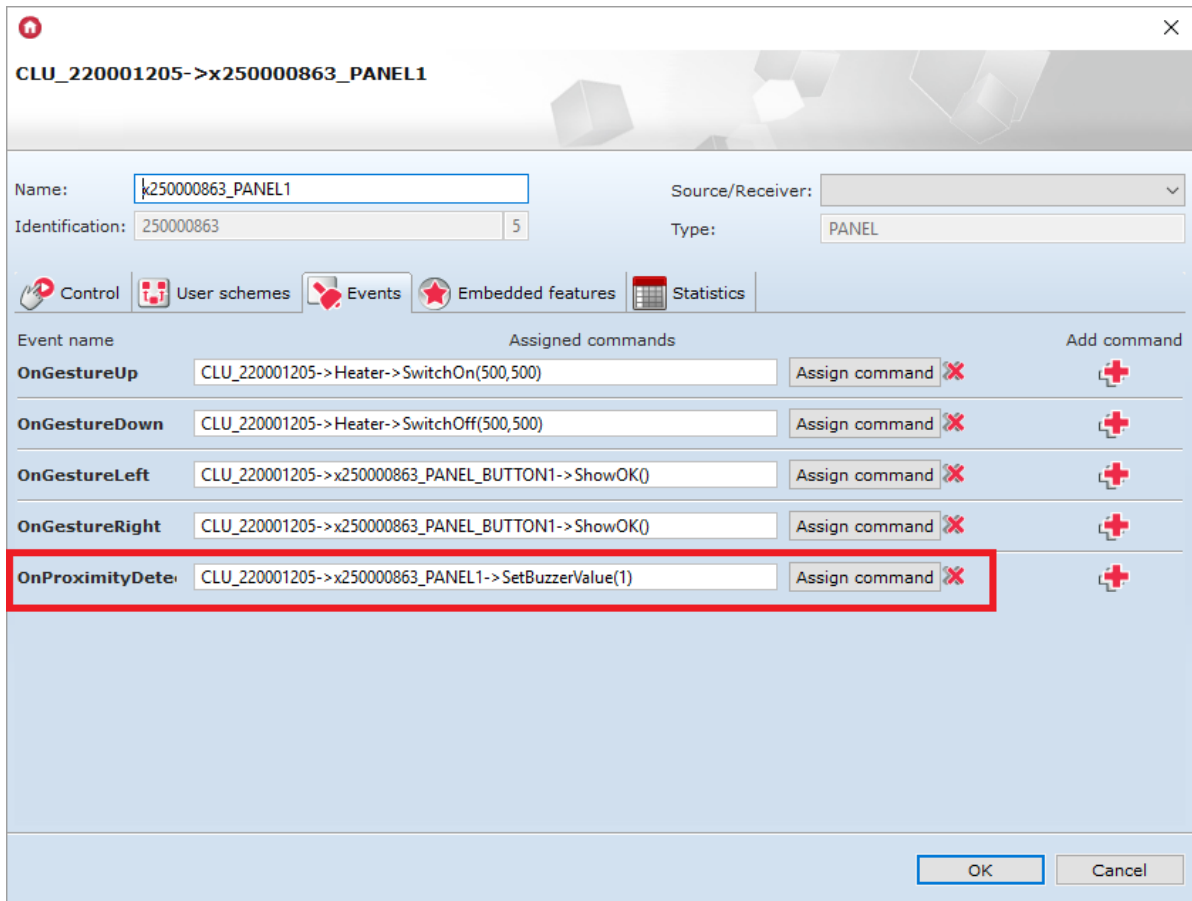
- Open - by double click - object *Panel*;
- Go to the *embedded features* tab, where there are 3 features related to the proximity sensor:
  - `ProximitySens` - defines the sensitivity of the sensor;
  - `ProximityTimeout` - defines the time after which the display is blanked when motion is not detected;
  - `ProximityValue` - returns the approximate distance in centimeters from the panel to the object:

Feature name	Current value	Initial value	Unit	Range
<b>GestureIconUp</b>	~lamp3on	~lamp3on	.bmp	[0-9]
<b>GestureIconDown</b>	lamp3off	lamp3off	.bmp	[0-9]
<b>GestureIconLeft</b>	minus	minus	.bmp	[0-9]
<b>GestureIconRight</b>	plus	plus	.bmp	[0-9]
<b>ProximitySens</b>	3	3		[2-100]
<b>ProximityTimeout</b>	5000	5000	ms	[1000-60000]
<b>ProximityValue</b>	386		-	
<b>BuzzerValue</b>	1	On		0,1

The above features can be set both in the tab *Built-in features*, as well as using the methods: `SetProximitySens` and `SetProximityTimeout` (in the methods of the *Panel* object).

- The proximity sensor reaction generates the `OnProximityDetect` event to which additional methods can be added:

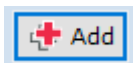




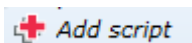
- Send the configuration to the CLU.

## 4.5 Creating a multi-panel configuration of the touch panel

If you want to start creating a multi-page panel configuration, create a *number* (determines the number of the start page) property on the CLU with the sample name *page* - double click on the CLU, go to the *User properties* tab and select the button:



In order for the panel to display the desired content on the screen, it is necessary to create a script (eg *Display*) with several pages - to do this select the button at the left edge of the Object Manager window:



**NOTE!** The name of the script can not contain Polish characters!

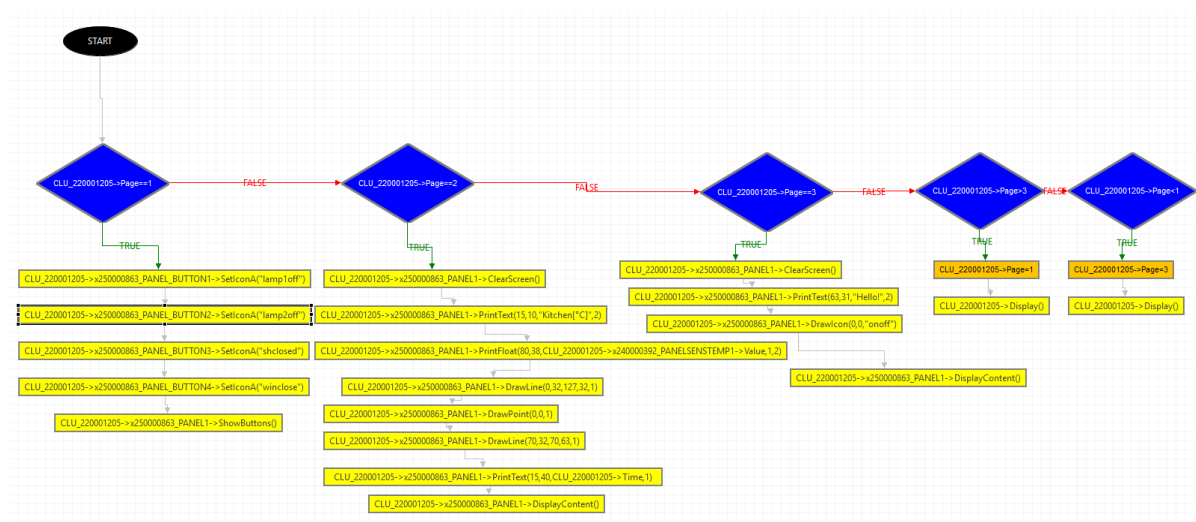
- **PAGE WITH THE BUTTONS** - Add a condition checking the current page number (value *User features: page*) to the script and for the given condition - for a specific page - add the icon allocation action for all 4 buttons (`SetIconA` methods for elements *PANEL\_BUTTON1-4*) and the method `ShowButtons` displaying selected icons on the panel screen;

**NOTE!** In addition to assigning icons to specific buttons, it is necessary to call the `ShowButtons` method, as simply assigning them will not cause them to appear on the display!

**NOTE!** In the case of creating multiple pages, setting the button in the bistable mode - using the feature / method - will not correctly read the state of the relay (due to the different functionality of the buttons when changing pages)!

- **PAGE WITH GRAPHICS AND TEXTS** - When designing a page containing graphics and texts, please add:
  - condition checking the page number (it can not be a page with buttons);
  - **PANEL action** -> `ClearScreen ()` ;
  - text and line setting actions (described below);
  - **PANEL action** -> `DisplayContent ()` .
- Text and line setting actions:
  - **PANEL** -> `PrintText` - method that causes text or feature to be printed - four parameters to call it: initial screen coordinates (x, y), text and font size (where 1 - 10 pts, 2 - 14 pts), 3 - 28 points);
  - **PANEL** -> `PrintFloat` - method working in the same way as `PrintText`, with the difference that it has an additional parameter *Precision*, responsible for the number of decimal places of the *number* parameter;
  - **PANEL** -> `DrawLine` - method drawing a line - it is necessary to enter 5 parameters to call it: initial coordinates (x, y), final coordinates (xe, ye) and line color (where 0 - black, 1 - white) ;
  - **PANEL** -> `DrawPoint` - method drawing a point - you must specify 3 parameters to call it: coordinates (x, y) and color (the parameter works as when calling the `DrawLine` method);
  - **PANEL** -> `DrawIcon` - method drawing the icon - you must enter 3 parameters to call it: initial coordinates (x, y) and the name of the icon from the tray.
- **LOCKING THE SCRIPT** - Add to the script the conditions that will cause that when the gesture is generated to the right on the last page, the panel will return to the first page (and vice versa) - so that the loop works.

The implementation of all the methods described above is presented in the screen shot of the sample script:



The above script is placed at the end of the document in the text version (point 3).

The second page programmed in the script will look like this:



- In the next step - to the panel gestures to the left and to the right - assign operations of increasing the user variable *page* and running the script *Display* as in the drawing below:

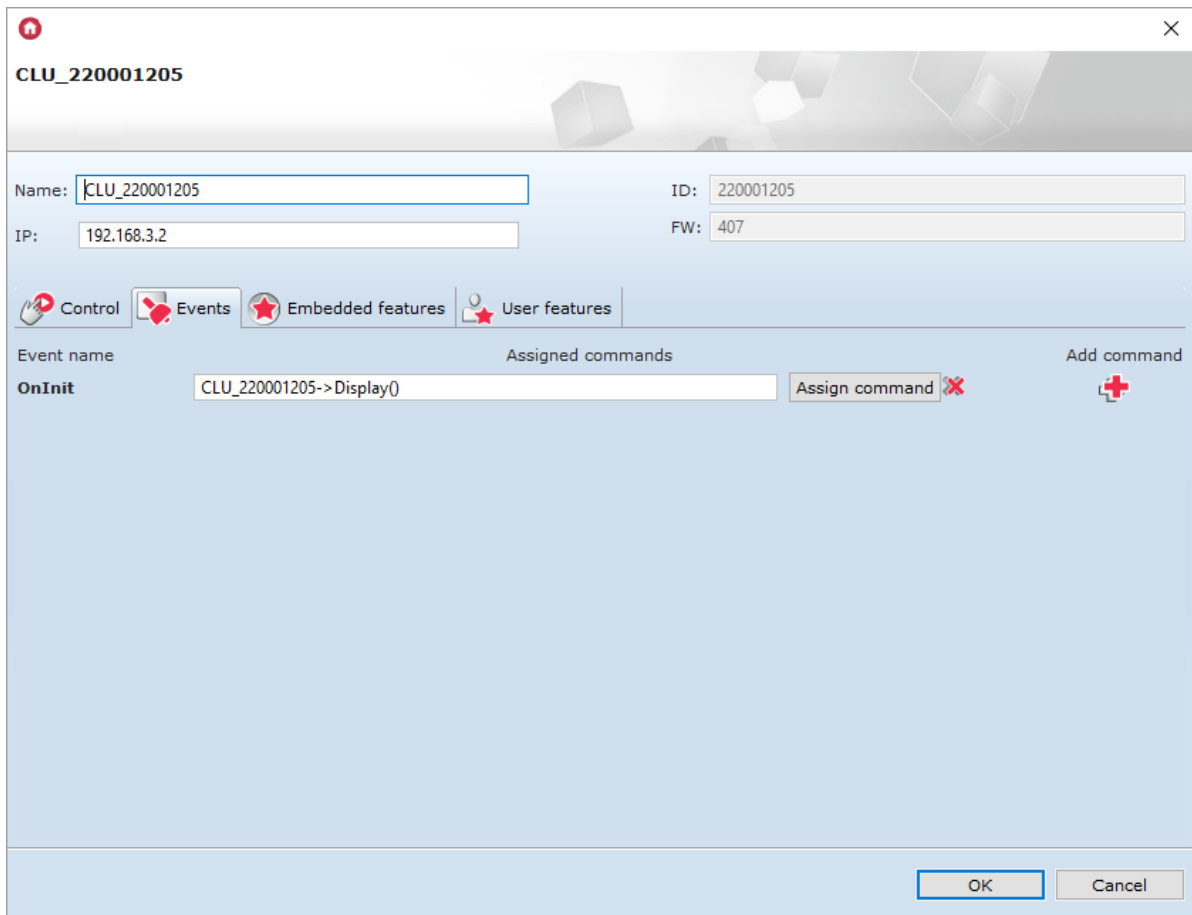
CLU\_220001205->x250000863\_PANEL1

Name:  Source/Receiver:

Identification:   Type:

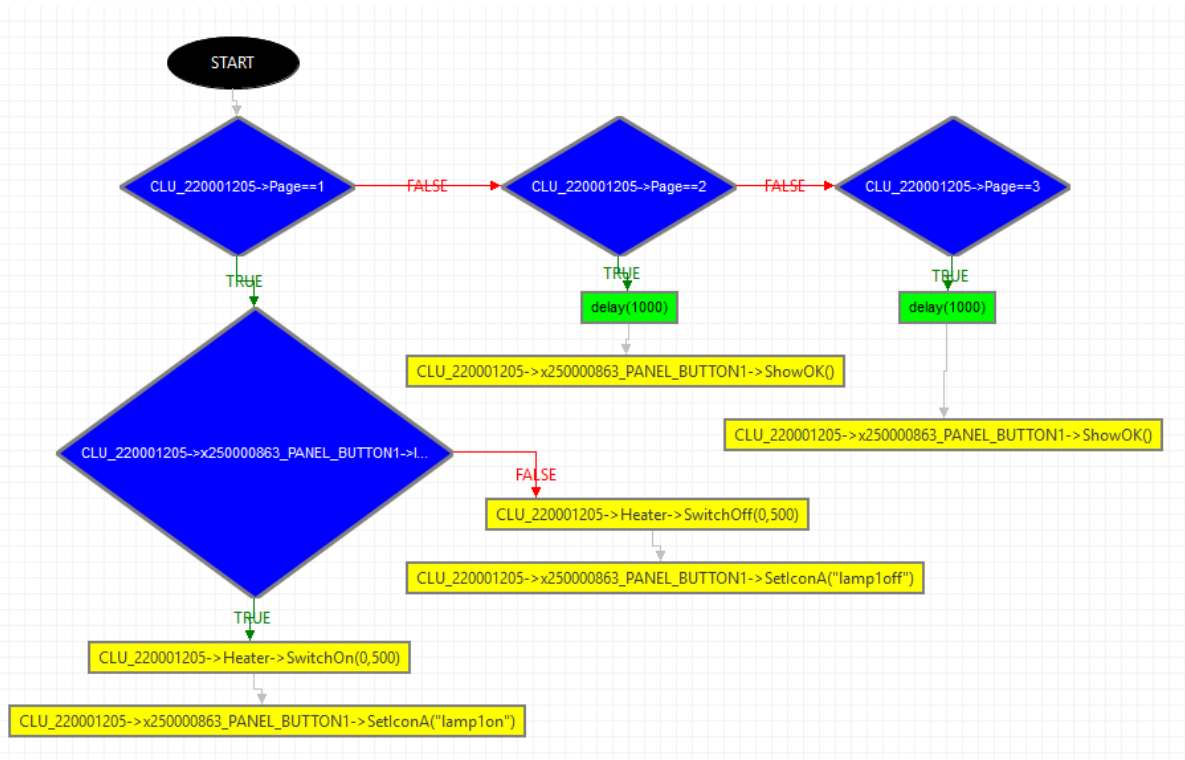
Event name	Assigned commands	Add command
<b>OnGestureUp</b>		<input type="button" value="+"/>
<b>OnGestureDown</b>	<input type="text" value="CLU_220001205-&gt;Display()"/> <input type="button" value="Assign command"/> <input type="button" value="X"/> <input type="text" value="CLU_220001205-&gt;Page=1"/> <input type="button" value="Assign command"/> <input type="button" value="X"/>	<input type="button" value="+"/>
<b>OnGestureLeft</b>	<input type="text" value="CLU_220001205-&gt;Page=CLU_220001205-&gt;Page-1"/> <input type="button" value="Assign command"/> <input type="button" value="X"/> <input type="text" value="CLU_220001205-&gt;Display()"/> <input type="button" value="Assign command"/> <input type="button" value="X"/>	<input type="button" value="+"/>
<b>OnGestureRight</b>	<input type="text" value="CLU_220001205-&gt;Page=CLU_220001205-&gt;Page+1"/> <input type="button" value="Assign command"/> <input type="button" value="X"/> <input type="text" value="CLU_220001205-&gt;Display()"/> <input type="button" value="Assign command"/> <input type="button" value="X"/>	<input type="button" value="+"/>
<b>OnProximityDetect</b>		<input type="button" value="+"/>

- Assign *CLU* -> `OnInit` to the script call *Display*.



- Create a script (e.g. *ClickButton1*) to handle the `OnClick` event of one selected button on each page - create separate scripts for each button:
  - Add a condition checking the page number;
  - In order to implement the bistable mode function for a button, add another condition checking the current status of the icon and undertaking appropriate actions (switching on or off, eg lighting);
  - Add further conditions to check the page number.

The implementation is shown in the following screenshot:



The above script is placed at the end of the document in a text version (point 4)

**NOTE!** The operation on variables used in the graphical mode of the panel does not refresh, therefore the action of re-generating the page was used in the above script!

- Finally, add additional scripts to all buttons and used events - respectively: script *ClickButton1* to event *PANEL\_BUTTON1* ->

OnHold			
OnClick	<input type="text" value="CLU_220001205-&gt;ClickButton1"/>	Assign command	

### 3. Script *Display* in text version:

```

if(not (CLU_220001205->Page==1)) then
if (CLU_220001205->Page==2) then
CLU_220001205->x250000863_PANEL1->ClearScreen ()
CLU_220001205->x250000863_PANEL1->PrintText (15,10, "Kitchen[°C]", 2)
CLU_220001205->x250000863_PANEL1->PrintFloat (80,38,CLU_220001205-
>x240000392_PANELSENSTEMP1->Value, 1,2)
CLU_220001205->x250000863_PANEL1->DrawLine (0,32,127,32,1)
CLU_220001205->x250000863_PANEL1->DrawPoint (0,0,1)
CLU_220001205->x250000863_PANEL1->DrawLine (70,32,70,63,1)
CLU_220001205->x250000863_PANEL1->PrintText (15,40,CLU_220001205->Time, 1)
CLU_220001205->x250000863_PANEL1->DisplayContent ()
else
if (CLU_220001205->Page==3) then
CLU_220001205->x250000863_PANEL1->ClearScreen ()
CLU_220001205->x250000863_PANEL1->PrintText (63,31, "Hello!", 2)
CLU_220001205->x250000863_PANEL1->DrawIcon (0,0, "onoff")

```

```

CLU_220001205->x250000863_PANEL1->DisplayContent()
else
if(CLU_220001205->Page>3) then
CLU_220001205->Page=1
CLU_220001205->Display()
else
if(CLU_220001205->Page<1) then
CLU_220001205->Page=3
CLU_220001205->Display()
end
end
end
end
else
CLU_220001205->x250000863_PANEL_BUTTON1->SetIconA("lamp1off")
CLU_220001205->x250000863_PANEL_BUTTON2->SetIconA("lamp2off")
CLU_220001205->x250000863_PANEL_BUTTON3->SetIconA("shclosed")
CLU_220001205->x250000863_PANEL_BUTTON4->SetIconA("winclose")
CLU_220001205->x250000863_PANEL1->ShowButtons()
end

```

#### 4. ClickButton1 script in text version:

```

if(CLU_220001205->Page==1) then
if(CLU_220001205->x250000863_PANEL_BUTTON1->IconA=="lamp1off") then
CLU_220001205->Heater->SwitchOn(0,500)
CLU_220001205->x250000863_PANEL_BUTTON1->SetIconA("lamp1on")
else
CLU_220001205->Heater->SwitchOff(0,500)
CLU_220001205->x250000863_PANEL_BUTTON1->SetIconA("lamp1off")
end
else
if(CLU_220001205->Page==2) then
SYSTEM.Wait(1000)
CLU_220001205->x250000863_PANEL_BUTTON1->ShowOK()
else
if(CLU_220001205->Page==3) then
SYSTEM.Wait(1000)
CLU_220001205->x250000863_PANEL_BUTTON1->ShowOK()
end
end
end

```

## 5. Configuration of the Smart Panel v4

**NOTE!** Smart Panel in the v4 version is available for Object Manager in version 1.2.1.190201 and higher and for CLU with firmware 04.07.49-1912 and higher.

## **5.1. Configuration parameters**

### **A. Panel**

#### **FEATURES**

Name	Description
GestureIconUp	The name of the BMP file with the icon for the gesture Top (without extension)
GestureIconDown	The name of the BMP file with the icon for the Down gesture (no extension)
GestureIconLeft	The name of the BMP file with the icon for the Left gesture (no extension)
GestureIconRight	The name of the BMP file with the icon for the Right gesture (no extension)
ProximitySens	Sensitivity of the proximity sensor (lower value - higher sensitivity)
ProximityTimeout	The time after which the display will be turned off
ProximityValue	Proximity sensor signal (non-dimensional value)
BuzzerValue	Control of sound signaling: 0 - Off , 1 - On
GestureMode	Gesture orientation: 0 - Off , 1 - Vertical , 2 - Horizontal , 3 - Vert+Horiz
GestureSens	Gesture sensitivity: 1 - Low , 2 - Mid , 3 - High
PageNr	The number of the currently displayed page
PageDisplayMode	Information before changing the page: 0 - ShowImmediately , 1 - ShowIconOrName , 2 - ShowGesture
ButtonsLEDMode	The location of the buttons with low LED light: 0 - LocationLedOFF , 1 - LocationLedON , 2 - LocationLedONforActive
PageControlMode	The source that switches pages: 0 - Command (switching using the SetNextPage and SetPrevPage methods) 1 - Gesture/Command (switching using gestures and SetNextPage and SetPrevPage methods)



Name	Description
<code>GestureDisplayMode</code>	Display information about the currently executed gesture: <code>0 - Off,</code> <code>1 - On</code>

## METHODS

Name	Description
SwitchOnDisplay	Wakes the display from sleep mode
ShowButtons	Changes the display mode to <i>buttons</i> . Clears the display and displays the icons (or text) again for all buttons
ClearScreen	Cleans the display content in <i>freedraw mode</i>
PrintText	Displays the text in <i>freedraw mode</i> using the parameters: <code>x</code> , <code>y</code> , <code>txt</code> , <code>font size</code> , where: <code>x</code> and <code>y</code> are the coordinates expressed in pixels, <code>txt</code> is a string, <code>font size</code> is the font size(1: 10p, 2: 14p, 3: 32p)
PrintFloat	Displays the number in <i>freedraw mode</i> using the parameters: <code>x</code> , <code>y</code> , <code>number</code> , <code>precision</code> , <code>font size</code> , where: <code>x</code> and <code>y</code> are coordinates expressed in pixels, <code>number</code> is the number, <code>precision</code> is the number of decimal places, <code>font size</code> is the font size (1:10p, 2:14p, 3:32p)
DrawLine	Draw lines in <i>freedraw mode</i> using the parameters: <code>x</code> , <code>y</code> , <code>xe</code> , <code>ye</code> , <code>color</code> , where: <code>x</code> and <code>y</code> are initial coordinates, <code>xe</code> and <code>ye</code> are final coordinates, <code>color</code> is the colour line (0 - black, 1 - white). The starting and ending coordinates are expressed in pixels
DrawPoint	Draws a point in the <i>freedraw mode</i> using the parameters: <code>x</code> , <code>y</code> , <code>color</code> , where: <code>x</code> and <code>y</code> are the coordinates expressed in pixels, <code>color</code> is the color of the point (0 - black, 1 - white)
DrawIcon	Draws the icon (bmp) in <i>freedraw mode</i> using the parameters: <code>x</code> , <code>y</code> , <code>Filename</code> , where: <code>x</code> and <code>y</code> are the coordinates expressed in pixels <code>Filename</code> is the name of the icon (without extension)
DisplayContent	Displays the contents of the graphical memory buffer. Changes the display mode to <i>freedraw</i>
SetGestureIconUp	Sets the icon to perform the up gesture
SetGestureIconDown	Sets the icon to perform the down gesture
SetGestureIconLeft	Sets the icon to perform the left gesture
SetGestureIconRight	Sets the icon to perform the right gesture
SetProximitySens	Sets the ProximitySens value
SetProximityTimeout	Sets the time in seconds after which the display goes out
SetBuzzerValue	Control of sound signaling (On / Off)

Name	Description
SetGestureMode	Choice of gesture orientation
SetGestureSens	Choice of gesture sensitivity
SetBeep	Generates sound at a given frequency [Hz], duration [ms] and volume
SetPageNr	Sets the number of the page displayed
SetPageDisplayMode	Sets the information display mode before changing the page
SetButtonsLEDMode	Sets the button location mode using the LEDs
SetPageControlMode	Sets the source that switches pages (commands / pages)
SetGestureDisplayMode	Sets the display mode of the information about the executed gesture
SetNextPage	Displays the next page
SetPrevPage	Displays the previous page
Draw	Triggers an OnDraw event when OLED is active

## EVENTS

Name	Description
OnGestureUp	An event triggered when an up gesture is executed
OnGestureDown	An event triggered when a down gesture is executed
OnGestureLeft	An event triggered when a left gesture is executed
OnGestureRight	An event triggered when a right gesture is executed
OnProximityDetect	An event triggered when a person approaching the display is detected
OnPageChange	An event triggered when the page is changed in the panel

## B. Buttons

### FEATURES

Name	Description
Mode	Returns the set operation mode of the button: 0 - monostable, 1 - bistable, 2 - locked (locked)
HoldDelay	Time in milliseconds, after pressing and holding the button triggers the event OnHold
HoldInterval	The cyclic interval in milliseconds that the OnHold event triggers when the button is held
Value	Returns the state of the button as 0 or 1
Label	The text that describes the button (displayed instead of the icon)
IconA	File name of the icon assigned to the button in monostable and bistable mode in the OFF position; the name preceded by "~" will display the graphic in negative; IconA has priority over the Label feature
IconB	The file name of the icon assigned to the button in bistable mode in the ON position; the name preceded by "~" will display the graphic in negative

## METHODS

Name	Description
SetMode	Sets the button operation mode: 0 - monostable, 1 - bistable, 2 - locked (locked)
SetHoldDelay	Sets the value of HoldDelay
SetHoldInterval	Sets the value of HoldInterval
SetLabel	Sets the value of Label (text describing the button)
SetIconA	Sets the file name of the A icon (without extension)
SetIconB	Sets the file name of the B icon (without extension)
ShowOK	Blinks the green LED on the button for two seconds (frequency 500 ms). The red LED of the button remains off
ShowError	The red LED on the button flashes for two seconds (500 ms frequency). The green LED of the button remains off
LedSwitchOn	It turns on the green LED on the button
RedLedSwitchOn	Activates the red LED on the button
LedSwitchOff	Turns off all LEDs on the button

## EVENTS

Name	Description
OnChange	An event that is triggered when the state changes to the opposite one
OnSwitchOn	An event that is triggered when the high state on input is set
OnSwitchOff	An event triggered when the low state on input is set
OnShortPress	An event triggered after pressing the button for 500 ms - 2000 ms
OnLongPress	An event is triggered after pressing the button for 2000 ms - 5000 ms
OnHold	An event triggered when the input is in the high state, the first time after the <code>holdDelay</code> time has elapsed, and then cyclically every <code>HoldInterval</code> value
OnClick	An event triggered after pressing the button for less than 500ms

### C. Pages configuration (Panel\_Page)

#### FEATURES

Name	Description
<div data-bbox="236 241 360 277" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">PageType</div>	<p>The type of page displayed on the Smart Panel:</p> <div data-bbox="491 203 679 232" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">0 - Inactive ,</div> <div data-bbox="491 248 663 277" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">1 - Buttons ,</div> <div data-bbox="491 293 724 322" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">2 - Thermostats ,</div> <div data-bbox="491 338 667 367" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">3 - FreeDraw</div>
<div data-bbox="236 416 360 452" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">PageName</div>	<p>Page name / name of the icon displayed on the Smart Panel (when switching between pages)</p>
<div data-bbox="236 562 405 598" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">Object_1_Id</div>	<p>Thermostat object ID or button number depending on page type, e.g.:  <i>Thermostats</i> page type: - for thermostat on local CLU: THE1325 - for remote thermostat CLU: CLU220000001-&gt; THE4321 For the PageType feature set to <i>Buttons / FreeDraw</i> enter the number of the button (1..16)</p>
<div data-bbox="236 728 427 763" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">Object_1_Name</div>	<p>The name of the thermostat displayed on the Smart Panel page. Applies only to page <i>Thermostats</i> (no name - inactive thermostat). In the case of the <div data-bbox="491 768 616 804" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">PageType</div> feature set to <i>Buttons / FreeDraw</i>, the feature remains empty</p>
<div data-bbox="236 896 405 931" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">Object_2_Id</div>	<p>Thermostat object ID or button number depending on page type, e.g.:  <i>Thermostats</i> page type: - for thermostat on local CLU: THE1325 - for remote thermostat CLU: CLU220000001-&gt; THE4321 For the PageType feature set to <i>Buttons / FreeDraw</i> enter the number of the button (1..16)</p>
<div data-bbox="236 1061 427 1097" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">Object_2_Name</div>	<p>The name of the thermostat displayed on the Smart Panel page. Applies only to page <i>Thermostats</i> (no name - inactive thermostat). In the case of the <div data-bbox="491 1102 616 1137" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">PageType</div> feature set to <i>Buttons / FreeDraw</i>, the feature remains empty</p>
<div data-bbox="236 1229 405 1265" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">Object_3_Id</div>	<p>Thermostat object ID or button number depending on page type, e.g.:  <i>Thermostats</i> page type: - for thermostat on local CLU: THE1325 - for remote thermostat CLU: CLU220000001-&gt; THE4321 For the PageType feature set to <i>Buttons / FreeDraw</i> enter the number of the button (1..16)</p>
<div data-bbox="236 1395 427 1431" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">Object_3_Name</div>	<p>The name of the thermostat displayed on the Smart Panel page. Applies only to page <i>Thermostats</i> (no name - inactive thermostat). In the case of the <div data-bbox="491 1435 616 1471" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">PageType</div> feature set to <i>Buttons / FreeDraw</i>, the feature remains empty</p>
<div data-bbox="236 1563 405 1599" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">Object_4_Id</div>	<p>Thermostat object ID or button number depending on page type, e.g.:  <i>Thermostats</i> page type: - for thermostat on local CLU: THE1325 - for remote thermostat CLU: CLU220000001-&gt; THE4321 For the PageType feature set to <i>Buttons / FreeDraw</i> enter the number of the button (1..16)</p>
<div data-bbox="236 1729 427 1765" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">Object_4_Name</div>	<p>The name of the thermostat displayed on the Smart Panel page. Applies only to page <i>Thermostats</i> (no name - inactive thermostat). In the case of the <div data-bbox="491 1769 616 1805" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px; display: inline-block;">PageType</div> feature set to <i>Buttons / FreeDraw</i>, the feature remains empty</p>

## METHODS

Name	Description
<code>SetPageType</code>	Sets the type of page displayed on the Smart Panel
<code>SetPageName</code>	Sets the page name / name of the icon displayed on the Smart Panel (when switching between pages)
<code>SetObject_1_Id</code>	Sets <code>Object_1_Id</code> value
<code>SetObject_1_Name</code>	Sets <code>Object_1_Name</code> value
<code>SetObject_2_Id</code>	Sets <code>Object_2_Id</code> value
<code>SetObject_2_Name</code>	Sets <code>Object_2_Name</code> value
<code>SetObject_3_Id</code>	Sets <code>Object_3_Id</code> value
<code>SetObject_3_Name</code>	Sets <code>Object_3_Name</code> value
<code>SetObject_4_Id</code>	Sets <code>Object_4_Id</code> value
<code>SetObject_4_Name</code>	Sets <code>Object_4_Name</code> value

## EVENTS

Name	Description
<code>OnPageOpen</code>	An event triggered when the page is opened
<code>OnPageClose</code>	An event triggered when the page is closed
<code>OnDraw</code>	An event signaling the need for redrawing. Generation only in <i>freedraw</i> mode, after entering the given page or when calling the <code>Draw</code> method and wake up the screen

## D. Temperature and lighting sensors

### FEATURES

Name	Description
<code>Threshold</code>	Hysteresis size (accuracy 0.1 ° C / 0.1%) defining the sensitivity at which events are generated: <code>OnChange</code> , <code>OnLowerValue</code> , <code>OnRaiseValue</code>
<code>Sensitivity</code>	The period (in ms) at which the sampled values are averaged
<code>MinValue</code>	The minimum value of the <code>Value</code> feature, exceeded by the <code>OnOutOfRange</code> event
<code>MaxValue</code>	The maximum value of the <code>Value</code> feature, exceeded by the <code>OnOutOfRange</code> event
<code>Value</code>	Input value: for a temperature sensor from 0.0 to 45.0 ° C or for a light sensor 0 - 100%

## EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the Value attribute changes
<code>OnRaiseValue</code>	An event triggered when the value changes to a higher one (rising edge)
<code>OnLowerValue</code>	An event triggered when the value changes to a lower one (falling edge)
<code>OnOutOfRange</code>	An event triggered when the input value is outside the specified range ( <code>MinValue</code> ; <code>MaxValue</code> )

## 5.2. Creating a gesture sensor configuration

- To create a configuration for a gesture sensor:
  - Open - by double click - object *Panel*;
  - Go to the tab *Events*;
  - Assign methods to the events `OnGestureUp`, `OnGestureDown`, `OnGestureLeft`, `OnGestureRight` (clicking on the `+` on the right of each method):

**Object properties**

Name: `k25000021_PANEL1` Source/Receiver:

Id: `null->PAN4218` Serial number: `250000021`

Type: `PANEL`

Control User schemes **Events** Embedded features Statistics

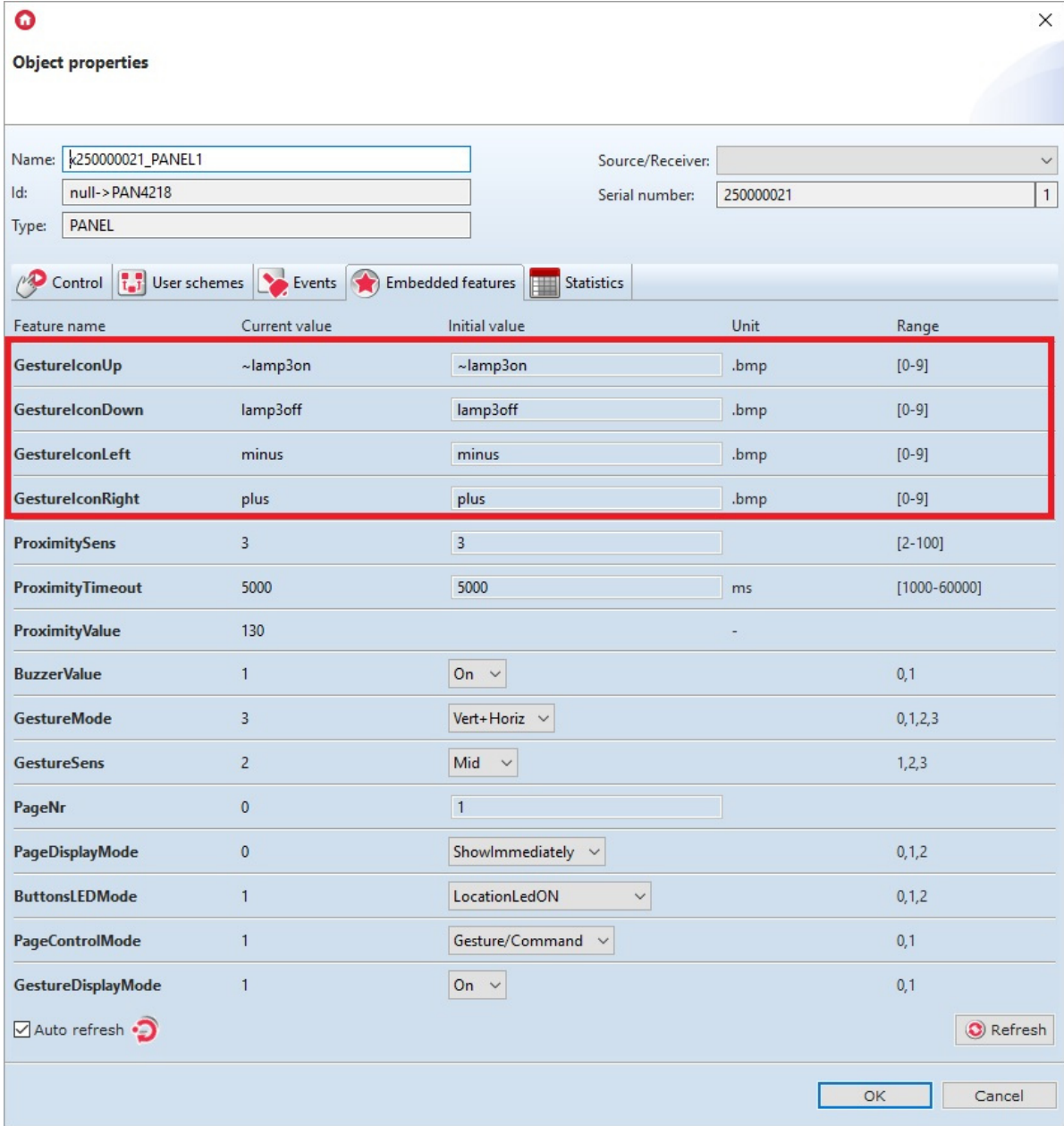
Event name	Assigned commands	Assign command	Add command
<code>OnGestureUp</code>	<code>CLU220001006-&gt;x190000558_DOUT1-&gt;SwitchOn(0)</code>	Assign command <input type="button" value="✘"/>	<input type="button" value="✚"/>
<code>OnGestureDown</code>	<code>CLU220001006-&gt;x190000558_DOUT1-&gt;SwitchOff(0)</code>	Assign command <input type="button" value="✘"/>	<input type="button" value="✚"/>
<code>OnGestureLeft</code>	<code>CLU220001006-&gt;x250000021_PANEL_BUTTON1-&gt;RedLedSwitchOn()</code>	Assign command <input type="button" value="✘"/>	<input type="button" value="✚"/>
<code>OnGestureRight</code>	<code>CLU220001006-&gt;x250000021_PANEL_BUTTON1-&gt;LedSwitchOff()</code>	Assign command <input type="button" value="✘"/>	<input type="button" value="✚"/>
<code>OnProximityDetect</code>			<input type="button" value="✚"/>
<code>OnPageChange</code>			<input type="button" value="✚"/>

OK Cancel

**NOTE!** In the case of configurations containing the configuration of pages (Buttons / FreeDraw / Thermostats), the methods assigned to the `OnGestureLeft` and `OnGestureRight` events will not be executed. This is related to the predefined functionality of switching between pages. You can change the way pages scroll. To do this, change the setting of the `PageControlMode` feature to Command. After doing this, the methods assigned to the events will be executed.



It is also possible to substitute the default icons displayed when gesturing - go to the *Built-in features* and enter the names of the icons you want without a *.bmp* extension:


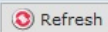


**Object properties**

Name: k250000021\_PANEL1 Source/Receiver:   
 Id: null->PAN4218 Serial number: 250000021 1  
 Type: PANEL

Control User schemes Events **Embedded features** Statistics

Feature name	Current value	Initial value	Unit	Range
GestureIconUp	~lamp3on	~lamp3on	.bmp	[0-9]
GestureIconDown	lamp3off	lamp3off	.bmp	[0-9]
GestureIconLeft	minus	minus	.bmp	[0-9]
GestureIconRight	plus	plus	.bmp	[0-9]
ProximitySens	3	3		[2-100]
ProximityTimeout	5000	5000	ms	[1000-60000]
ProximityValue	130		-	
BuzzerValue	1	On		0,1
GestureMode	3	Vert+ Horiz		0,1,2,3
GestureSens	2	Mid		1,2,3
PageNr	0	1		
PageDisplayMode	0	ShowImmediately		0,1,2
ButtonsLEDMode	1	LocationLedON		0,1,2
PageControlMode	1	Gesture/Command		0,1
GestureDisplayMode	1	On		0,1

Auto refresh  

OK Cancel

The use of icons will be possible when they will be uploaded to the microSD card with the *.bmp* extension.

In addition, from version 04.03.04.1910 there is a possibility to choose the orientation of recognizable gestures and their sensitivity. To do this, go to the *Built-in features* tab and select the desired orientation and sensitivity of gesture recognition:

**Object properties**

Name:  Source/Receiver:

Id:  Serial number:

Type:

Control User schemes Events Embedded features Statistics

Feature name	Current value	Initial value	Unit	Range
GestureIconUp	~lamp3on	<input type="text" value="~lamp3on"/>	.bmp	[0-9]
GestureIconDown	lamp3off	<input type="text" value="lamp3off"/>	.bmp	[0-9]
GestureIconLeft	minus	<input type="text" value="minus"/>	.bmp	[0-9]
GestureIconRight	plus	<input type="text" value="plus"/>	.bmp	[0-9]
ProximitySens	3	<input type="text" value="3"/>		[2-100]
ProximityTimeout	5000	<input type="text" value="5000"/>	ms	[1000-60000]
ProximityValue	130		-	
BuzzerValue	1	<input type="text" value="On"/>		0,1
GestureMode	3	<input type="text" value="Vert+ Horiz"/>		0,1,2,3
GestureSens	2	<input type="text" value="Mid"/>		1,2,3
PageNr	0	<input type="text" value="1"/>		
PageDisplayMode	0	<input type="text" value="ShowImmediately"/>		0,1,2
ButtonsLEDMode	1	<input type="text" value="LocationLedON"/>		0,1,2
PageControlMode	1	<input type="text" value="Gesture/Command"/>		0,1
GestureDisplayMode	1	<input type="text" value="On"/>		0,1

Auto refresh

Embedded features, through which you can choose orientation and sensitivity:

- - possible change in the direction of gesture detection:
  - Off - gestures are not recognized;
  - Vertical - only up and down gestures are recognized;
  - Horizontal - only gestures left and right are recognized;
  - Vert+Horiz - gestures are recognized both up and down, as well as left and right.
- - possible change in gesture detection sensitivity:
  - Low - gesture performed close to the device in an accurate manner;
  - Mid - gesture performed both close to the device as well as from a short distance;
  - High - gesture made from a further distance, it is possible to detect the wrong gesture.

The above features can be set both in the tab *Built-in features*, as well as using methods:

, , , ,  
,  (in the methods of the Panel object).

- Confirm the configuration window with *OK*;

- Send the configuration to the CLU Z-Wave.

## 5.3. Configuration of the proximity sensor

To set the proximity sensor parameters:

- Open - by double-clicking - the Panel object;
- Go to the *Embedded Features* tab, where there are 3 features related to the proximity sensor:
  - `ProximitySens` - determines the sensitivity of the sensor;
  - `ProximityTimeout` - defines the time after which the display is blanked when motion is not detected;
  - `ProximityValue` - returns the approximate distance in centimeters from the panel to the object;

The screenshot shows the 'Object properties' dialog box for a PANEL object. The 'Embedded features' tab is selected, displaying a table of features. The features 'ProximitySens', 'ProximityTimeout', and 'ProximityValue' are highlighted with a red box. The table has columns for Feature name, Current value, Initial value, Unit, and Range.

Feature name	Current value	Initial value	Unit	Range
GestureIconUp	~lamp3on	~lamp3on	.bmp	[0-9]
GestureIconDown	lamp3off	lamp3off	.bmp	[0-9]
GestureIconLeft	minus	minus	.bmp	[0-9]
GestureIconRight	plus	plus	.bmp	[0-9]
<b>ProximitySens</b>	<b>3</b>	<b>3</b>		<b>[2-100]</b>
<b>ProximityTimeout</b>	<b>5000</b>	<b>5000</b>	<b>ms</b>	<b>[1000-60000]</b>
<b>ProximityValue</b>	<b>130</b>		<b>-</b>	
BuzzerValue	1	On		0,1
GestureMode	3	Vert+Horiz		0,1,2,3
GestureSens	2	Mid		1,2,3
PageNr	0	1		
PageDisplayMode	0	ShowImmediately		0,1,2
ButtonsLEDMode	1	LocationLedON		0,1,2
PageControlMode	1	Gesture/Command		0,1
GestureDisplayMode	1	On		0,1

At the bottom of the dialog, there is a checked 'Auto refresh' checkbox, a 'Refresh' button, and 'OK' and 'Cancel' buttons.

The above features can be set both in the tab *Built-in features*, as well as using the methods: `SetProximitySens` and `SetProximityTimeout` (in the methods of the Panel object).

- The proximity sensor reaction generates the `OnProximityDetect` event to which additional methods can be added:

**Object properties**

Name:  Source/Receiver:

Id:  Serial number:

Type:

Control User schemes Events Embedded features Statistics

Event name	Assigned commands		Add command
OnGestureUp	<input type="text" value="CLU220001006-&gt;x190000558_DOUT1-&gt;SwitchOn(0)"/>	<input type="button" value="Assign command"/> ✖	<input type="button" value="+"/>
OnGestureDown	<input type="text" value="CLU220001006-&gt;x190000558_DOUT1-&gt;SwitchOff(0)"/>	<input type="button" value="Assign command"/> ✖	<input type="button" value="+"/>
OnGestureLeft	<input type="text" value="CLU220001006-&gt;x250000021_PANEL_BUTTON1-&gt;RedLedSwitchOn()"/>	<input type="button" value="Assign command"/> ✖	<input type="button" value="+"/>
OnGestureRight	<input type="text" value="CLU220001006-&gt;x250000021_PANEL_BUTTON1-&gt;LedSwitchOff()"/>	<input type="button" value="Assign command"/> ✖	<input type="button" value="+"/>
OnProximityDetect	<input type="text" value="CLU220001006-&gt;x250000021_PANEL_BUTTON1-&gt;ShowError()"/>	<input type="button" value="Assign command"/> ✖	<input type="button" value="+"/>
OnPageChange			<input type="button" value="+"/>

- Send the configuration to the CLU Z-Wave.

## 5.4. Panel object - new functionality

In the latest version of the Smart Panel module (from 04.03.04.1910), the Panel facility has introduced a new functionality enabling, among other things:

- sound generation;
- management of LED button backlight;
- the ability to enable / disable notification of the detected gesture;
- page management mechanism, which will be described in detail in the next section.

The first of the introduced novelties is the ability to generate sound at a given frequency, length and volume. The `SetBeep` method is used for this purpose:

**Object properties**

Name: x250000021\_PANEL1      Source/Receiver:

Id: null->PAN4218      Serial number: 250000021      1

Type: PANEL

Control   User schemes   Events   **Embedded features**   Statistics

<b>SetProximitySens</b>	ProximitySens	<input type="text"/>	number [2-100]	<input type="button" value="▶"/>
<b>SetProximityTimeout</b>	ProximityTimeout	<input type="text"/>	number [1000-60000]	<input type="button" value="▶"/>
<b>SetBuzzerValue</b>	BuzzerValue	Off		<input type="button" value="▶"/>
<b>SetGestureMode</b>	GestureMode	Off		<input type="button" value="▶"/>
<b>SetGestureSens</b>	GestureSens	Low		<input type="button" value="▶"/>
<b>SetBeep</b>	Frequency	500	number [30-5700]	
	Duration	1000	number [1-2000]	<input type="button" value="▶"/>
	Volume	16	number [0-16]	
	Reserved	0	number [0]	
<b>SetPageNr</b>	Nr	<input type="text"/>	number [1-4]	<input type="button" value="▶"/>
<b>SetPageDisplayMode</b>	PageDisplayMode	ShowImmediately		<input type="button" value="▶"/>
<b>SetButtonsLEDMode</b>	ButtonsLEDMode	LocationLedOFF		<input type="button" value="▶"/>
<b>SetPageControlMode</b>	PageControlMode	Command		<input type="button" value="▶"/>

OK   Cancel

Another feature available from the latest version of the software is the ability to locate buttons using low LED light. To do this, go to the *Built-in features* tab and set the desired value of the `ButtonsLEDMode` feature:

- LocationLedOFF - the buttons on the Smart Panel module are not illuminated;
- LocationLedOn - the buttons on the SmartPanel module are slightly illuminated;
- LocationLedforActive - only keys that are in one of the two operating modes *Monostable / Bistable* are highlighted. If the button is in *Locked* mode, its LED remains off.

In addition to the ability to manage the backlight buttons, it is possible to enable / disable informing about the detection of a gesture. To do this, in the *Embedded Features* tab, find the `GestureDisplayMode` feature by setting any value:

- Off - information on the detection of a gesture is not displayed on the module screen;
- On - information on the detection of a gesture is displayed on the module's screen.

The above built-in features can also be set using the methods: `SetButtonsLEDMode` and `SetGestureDisplayMode`.

## 5.5. Panel object - page management mechanism

- Smart Panel v4 introduces a new mechanism for page management. It consists of features, methods and events that were placed in the Panel object:

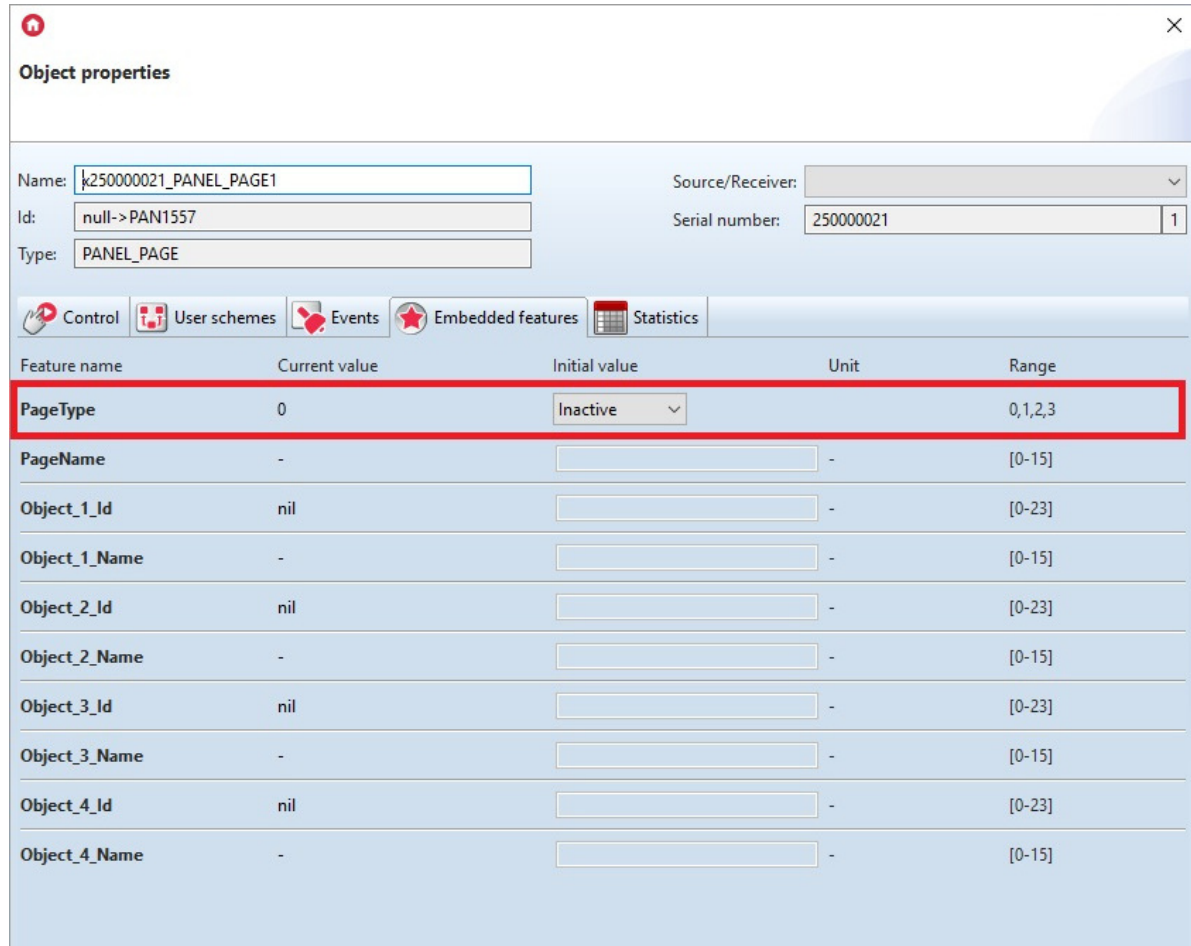
Methods / Features:

- `SetPageNr / PageNr` - using this method / feature it is possible to directly transition between more pages at the same time. By entering the page number in the parameter and then calling the method, the desired page will be displayed on the screen (you may need to wake up the screen);
- `SetPageDisplayMode / PageDisplayMode` - via the method / feature it is possible to set the method of switching between pages. There are three modes to choose from:
  - ShowImmediately (0) - the transition between pages takes place immediately, it is not preceded by displaying a message / icon / name;
  - ShowIconOrName (1) - the transition between pages precedes displaying the icon or name entered in the feature `PageName`;
  - ShowGesture (2) - the transition between the pages is preceded by the display of the icon entered in the feature `GestureIconLeft` or `GestureIconRight`, depending on the gesture made;
- `SetPageControlMode / PageControlMode` - using the method / feature it is possible to change the source with which the page change is made:
  - Command (0) - go to the previous / next page only using the methods `SetPrevPage` and `SetNextPage`. In addition, left and right gestures become active, which means that it is possible to assign `OnGestureLeft` and `OnGestureRight` events to the event;
  - Gesture / Command (1) - the transition to the previous / next page is possible using gestures left and right, as well as using the methods `SetPrevPage` and `SetNextPage`. If this property value is set, the left and right gestures have a predefined functionality that has a higher priority over the actions assigned to the `OnGestureLeft` and `OnGestureRight` events. This means that actions assigned to these events will not be executed;
- `SetNextPage` - the method allows you to go to the next page in the configuration;
- `SetPrevPage` - the method allows you to go to the previous page in the configuration;
- `Draw` - method used to generate the OnDraw event when the OLED is active;
- Happening:
  - `OnPageChange` - an event generated when switching between pages

**NOTE!** The page management mechanism is available only for the configuration of pages made through Panel\_Page objects (Buttons / FreeDraw / Thermostats). In the case of a configuration that was created in the previous manner (section 4.5), the above features, methods and event are ignored.

## 5.6. Backward compatibility

When starting work with the new version of the Smart Panel module, the device is in the default configuration, which is backward compatible. All four Panel\_Page objects have the built-in feature `PageType` set to *Inactive*. This allows you to work with the panel in the same way as before (in version v3). Only the first four buttons on the list of objects are available. Buttons 5 to 16 are inactive, despite the configuration options. The configuration of multiple pages is carried out in accordance with the procedure described in Section 4.5.



**Object properties**

Name:  Source/Receiver:

Id:  Serial number:

Type:

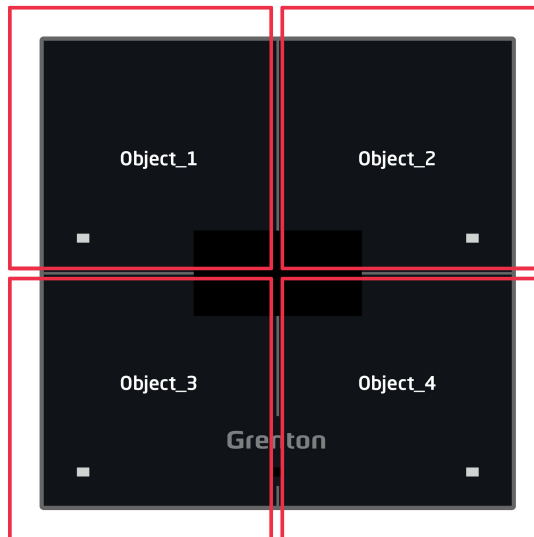
Control User schemes Events Embedded features Statistics

Feature name	Current value	Initial value	Unit	Range
PageType	0	Inactive		0,1,2,3
PageName	-	<input type="text"/>	-	[0-15]
Object_1_Id	nil	<input type="text"/>	-	[0-23]
Object_1_Name	-	<input type="text"/>	-	[0-15]
Object_2_Id	nil	<input type="text"/>	-	[0-23]
Object_2_Name	-	<input type="text"/>	-	[0-15]
Object_3_Id	nil	<input type="text"/>	-	[0-23]
Object_3_Name	-	<input type="text"/>	-	[0-15]
Object_4_Id	nil	<input type="text"/>	-	[0-23]
Object_4_Name	-	<input type="text"/>	-	[0-15]

## 5.7. Creating a configuration using the Buttons page object

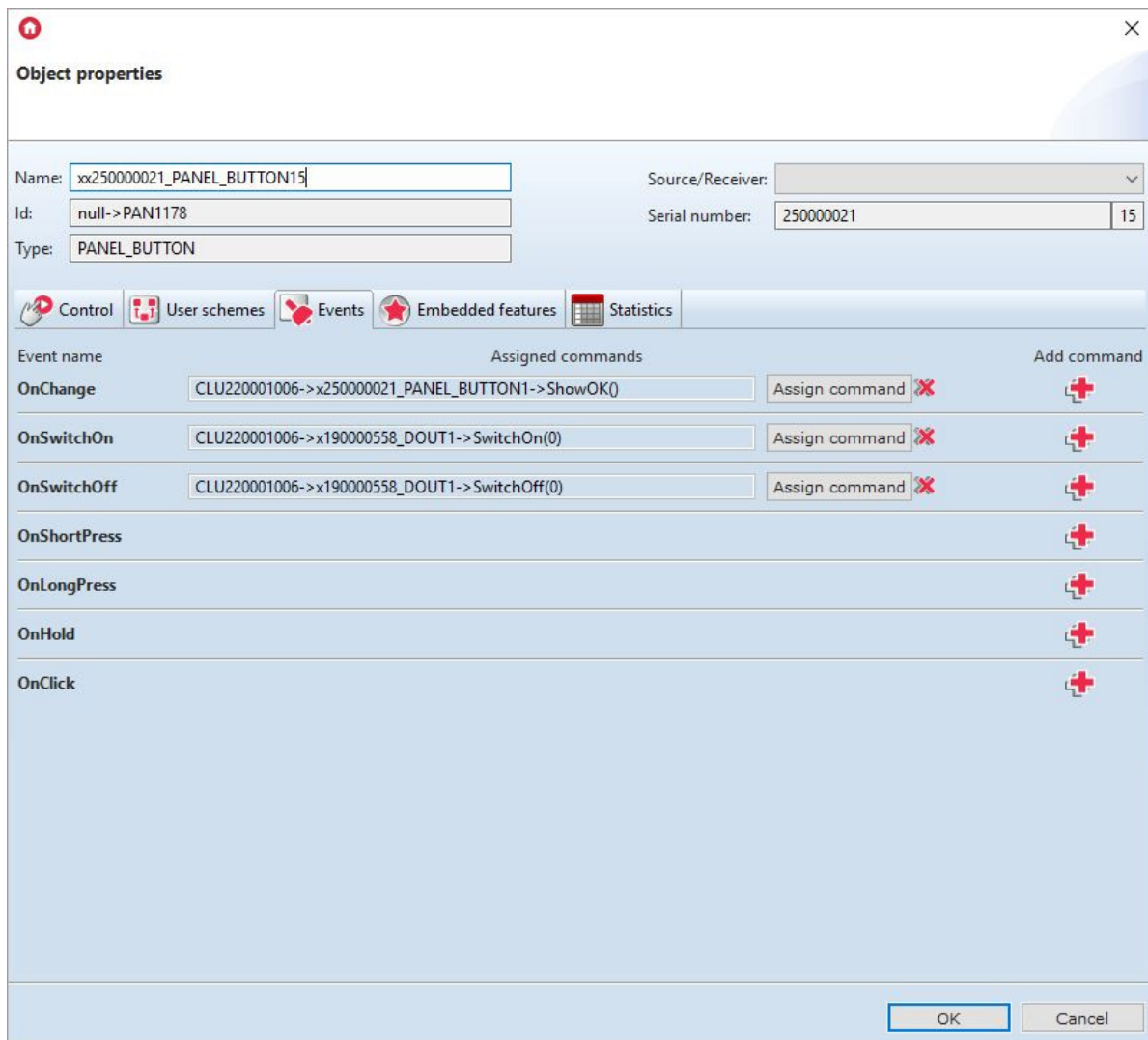
In the *Buttons* operating mode, there are 4 physical touch buttons and up to 16 virtual buttons spread over 4 pages, each of which can perform independent functions. It is also possible to combine / merge 2,3,4 objects into one button (described in more detail in subsection XII.5.10).

**NOTE!** In the *Buttons* mode, drawing content on the display is blocked.



### Page type „Buttons/FreeDraw“

- Creating a panel configuration that supports a page or pages *Buttons* is best to start with the configuration of the buttons to be used. In order to parametrize them:
  - Open the *PANEL\_BUTTONX* object (where X is the number of one of the 16 buttons) by double clicking on the list of modules;
  - Go to the Events tab;
  - Configure the operation of the button by assigning methods to specific events (by clicking “+” on the right side of the window);





- Select the tab *Embedded features* and define the objects displayed on the screen of a given button:
  - `Label` - a feature defining the text assigned to a given button;
  - `IconA` - a feature that defines the name of the icon assigned to a given button when it is found in \* Monostable \* mode or \* Bistable \* mode for OFF position;
  - `IconB` - a feature that defines the name of the icon assigned to a given button when it is in *Bistable* mode in the ON position. To assign the same icon, but with the inverted colors, precede the name of the pictogram with the "~" sign (eg `~ heaton`);

**Object properties**

Name: `k250000021_PANEL_BUTTON15` Source/Receiver:

Id: `null->PAN1178` Serial number: `250000021` `15`

Type: `PANEL_BUTTON`

Control User schemes Events **Embedded features** Statistics

Feature name	Current value	Initial value	Unit	Range
Mode	0	Monostable		0,1,2
HoldDelay	1000	1000	ms	[1-5000]
HoldInterval	100	50	ms	[1-2000]
Value	0		bool	0,1
Label	-	Lamp3	string	[0-15]
IconA	-	lamp3off	string	[0-9]
IconB	-	~lamp3on	string	[0-9]

Auto refresh

The above built-in features can be set both in the tab *Built-in features*, as well as via the methods:

`SetLabel`, `SetIconA`, `SetIconB`.

**NOTE!** The `SetIconA` method has a higher priority in the system than the `SetLabel` method!

- Send the configuration to the CLU Z-Wave.

The next step in creating the configuration is configuring `Panel_Page` objects depending on the number of buttons. One `Panel_Page` object supports up to 4 buttons. To do this:

- Open the object `PANEL_PAGEX` (where X is the number of the next page) by double clicking on the list of modules;
- Go to the tab *Events*;

- Configure the operation of the site by assigning methods to specific events (by clicking "+" on the right side of the window):

The screenshot shows the 'Object properties' dialog box for an object named 'k250000021\_PANEL\_PAGE1'. The object is of type 'PANEL\_PAGE' and has an ID of 'null->PAN1557'. It is associated with source/receiver '250000021' and has a serial number of '1'. The 'Events' tab is selected, showing a table of event assignments:

Event name	Assigned commands	Add command
OnPageOpen	CLU220001006->x250000021_PANEL_BUTTON1->LedSwitchOn()	Assign command ✖ +
OnPageClose	CLU220001006->x250000021_PANEL_BUTTON1->LedSwitchOff()	Assign command ✖ +
OnDraw		+ (disabled)

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

**NOTE!** For page type *Buttons*, the `OnDraw` event is not generated.

- Select the tab *Built-in features* and define the supported page type and link the page objects to the buttons:
  - `PageType` - a feature that specifies the page type, set it to *Buttons (1)*;
  - `PageName` - a feature that specifies the name of the page or icon that will be displayed when switching between pages (works only when the 'PageDisplayMode' feature is set to 1 (ShowIconOrName) in the Panel object);
  - `Object_X_Id` - identifier / button number. In order to read the value in the field *Serial number* of the object *PANEL\_BUTTONX*

The screenshot shows the 'Object properties' dialog box for an object named 'k250000021\_PANEL\_BUTTON15'. The object is of type 'PANEL\_BUTTON' and has an ID of 'null->PAN1178'. It is associated with source/receiver '250000021' and has a serial number of '15'. The 'Serial number' field is highlighted with a red box.

- `Object_X_Name` - name of the thermostat. For the page type *Buttons*, the feature should be left blank;

Object properties
✕

Name:

Id:

Type:

Source/Receiver:

Serial number:

Control User schemes Events Embedded features Statistics

Feature name	Current value	Initial value	Unit	Range
PageType	0	<input type="text" value="Buttons"/>		0,1,2,3
PageName	-	<input type="text" value="Page1"/>	-	[0-15]
Object_1_Id	nil	<input type="text" value="1"/>	-	[0-23]
Object_1_Name	-	<input type="text" value=""/>	-	[0-15]
Object_2_Id	nil	<input type="text" value="2"/>	-	[0-23]
Object_2_Name	-	<input type="text" value=""/>	-	[0-15]
Object_3_Id	nil	<input type="text" value="7"/>	-	[0-23]
Object_3_Name	-	<input type="text" value=""/>	-	[0-15]
Object_4_Id	nil	<input type="text" value="8"/>	-	[0-23]
Object_4_Name	-	<input type="text" value=""/>	-	[0-15]

Auto refresh 
Refresh

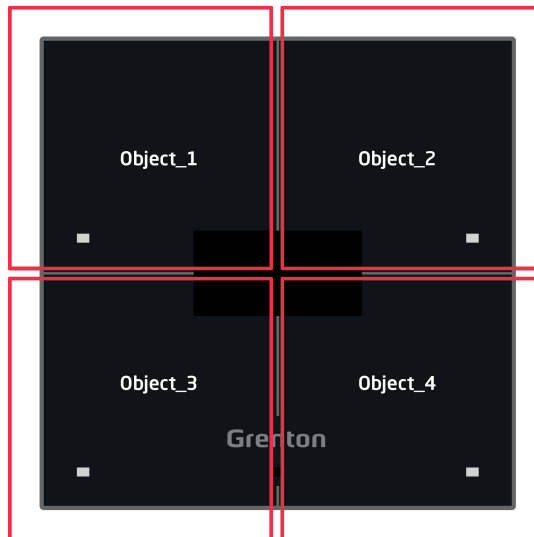
OK
Cancel

**NOTE!** Sending the configuration only with the defined page type, without setting the binding of objects with the buttons is connected with starting the panel operation mode as *Buttons*. However, the buttons on the module will be inactive. This is related to the non-complementing of the Object\_X\_Id features.

- Send the configuration to the CLU Z-wave.

## 5.8. Creating a configuration using the FreeDraw site object

In *FreeDraw* mode, as with *Buttons*, there are 4 physical touch buttons and up to 16 virtual buttons spread over 4 pages, each of which can perform independent functions. You can also combine / merge objects into a single button. The OLED display works in *FreeDraw* mode, i.e. it is fully available for user's LUA scripts. A drawing engine has also been created, in which drawing scripts are called by the `OnDraw` event generated by the panel when it is necessary. The system calls the `Draw` method at the moment when the content drawn on the module has changed.



**Page type „Buttons/FreeDraw“**

## A. General rules for creating configurations

Creating a panel configuration that supports a page or pages *FreeDraw* is best to start with the configuration of the buttons to be used. Their parameterization is described in the previous subsection.

The next step in creating the configuration should be creating scripts that draw the content on the Smart Panel display. Their creation is analogous to the v3 version of the Smart Panel module (see chapter XII.4).

Example of a script that draws content on the display (*Page1*):

Example of a script that draws content on the display (*Page1*):

```

CLU220000260->x250000053_PANEL1->ClearScreen()
CLU220000260->x250000053_PANEL1->PrintText(15,10,"Kitchen [°C]:",2)
CLU220000260->x250000053_PANEL1->PrintFloat(80,38,CLU220000260->
>x240000659_PANELSENSTEMP1->Value,1,2)
CLU220000260->x250000053_PANEL1->DrawLine(0,32,127,32,1)
CLU220000260->x250000053_PANEL1->DrawPoint(0,0,1)
CLU220000260->x250000053_PANEL1->DrawLine(70,32,70,63,1)
CLU220000260->x250000053_PANEL1->PrintText(15,40,CLU220000260->Time,1)
CLU220000260->x250000053_PANEL1->DisplayContent()

```

**NOTE!** A restriction has been introduced in the drawing mechanism. CLU Z-Wave expects 2 seconds to finish drawing with the `DisplayContent` method. Otherwise, the following message will be displayed on the screen:

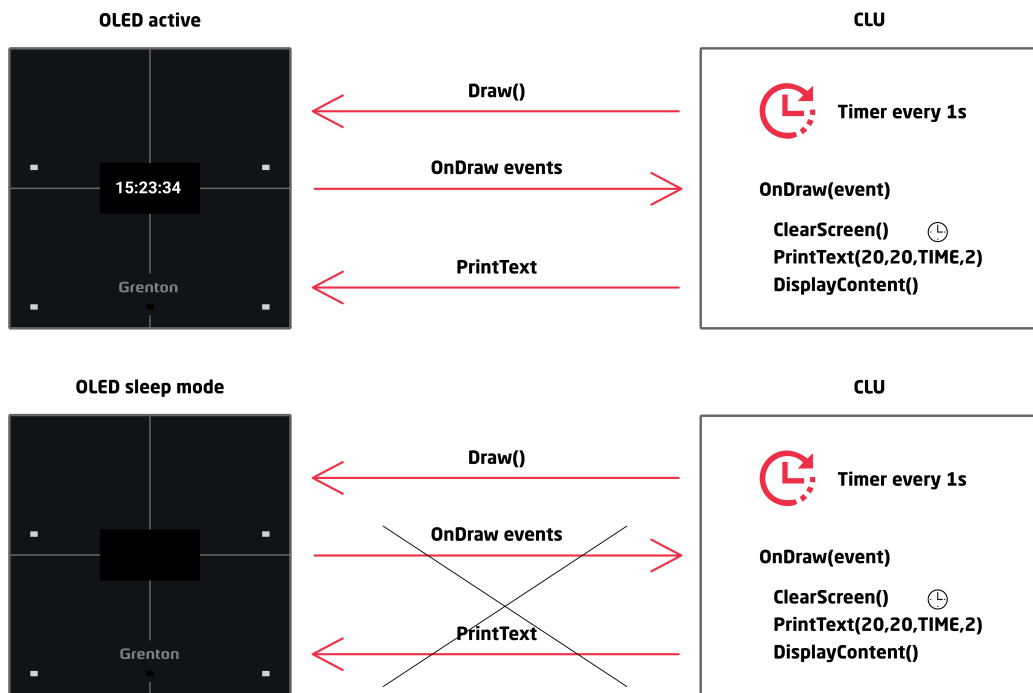
"page:

free draw

! TIMEOUT !"

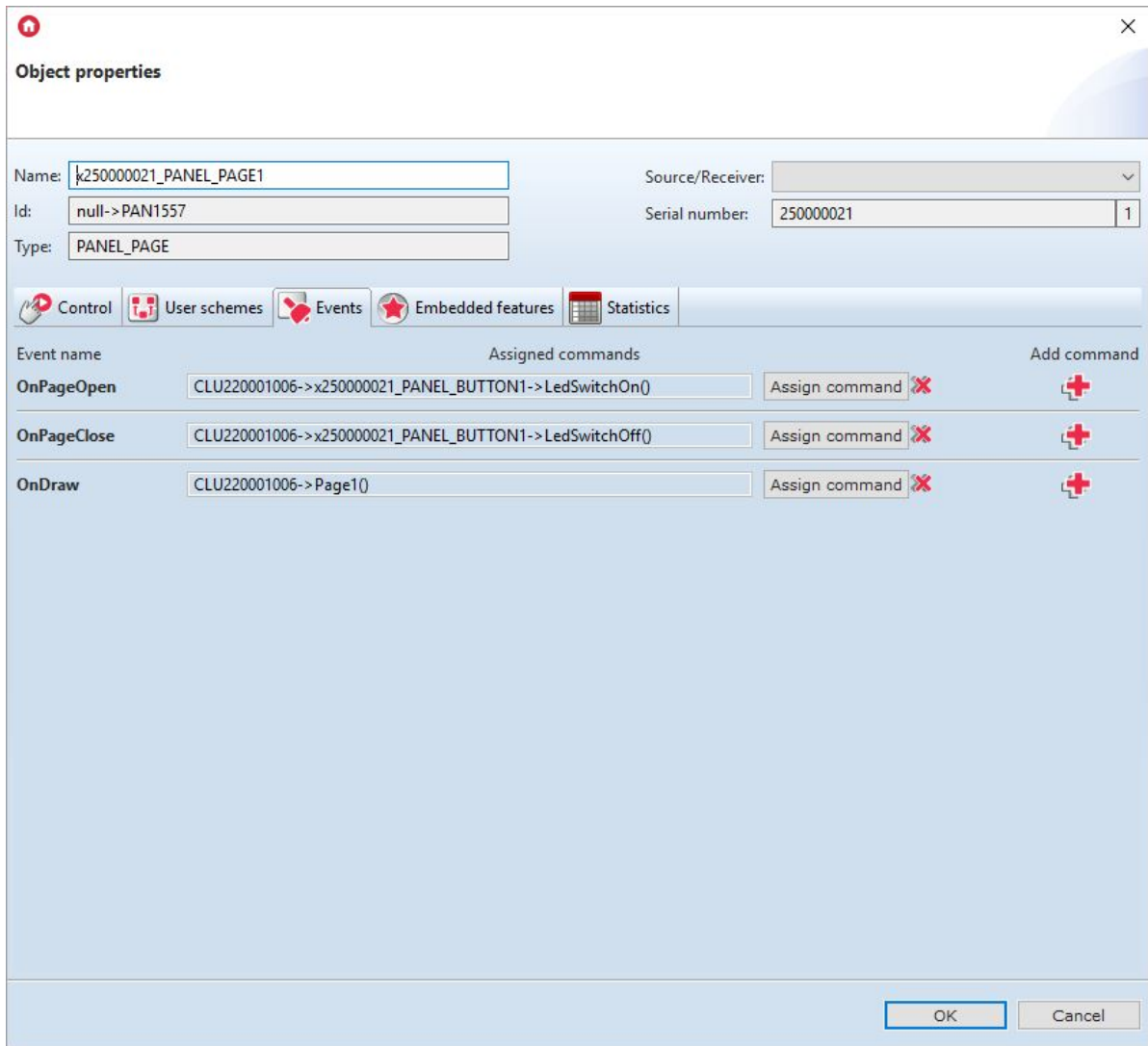
The following figure shows the current drawing mechanism.

## TF-Bus



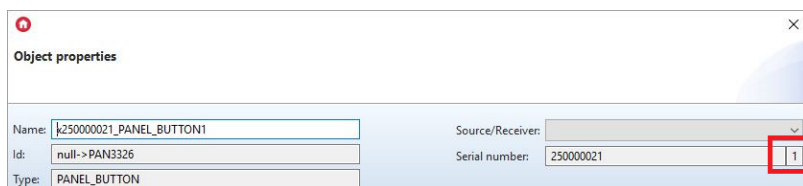
The next step in creating the configuration is configuring `Panel_Page` objects depending on the number of buttons. One `Panel_Page` object supports up to 4 buttons. To do this:

- Open the object `PANEL_PAGEX` (where `X` is the number of the next page) by double clicking on the list of modules;
- Go to the tab `Events`;
- Configure the operation of the site by assigning methods to specific events (by clicking "+" on the right side of the window):



**NOTE!** For the page type *FreeDraw*, complete the `OnDraw` event.

- Select the tab *Built-in features* and define the supported page type and link the page objects to the buttons:
  - `PageType` - a feature that specifies the page type, set it to *FreeDraw (3)*;
  - `PageName` - a feature that specifies the name of the page or icon that will be displayed when switching between pages (works only when the `PageDisplayMode` feature is set to 1 (ShowIconOrName) in the Panel object);
  - `Object_X_Id` - identifier / button number. To do this, read the value in the field *Serial number* of the object *PANEL\_BUTTONX*



- `Object_X_Name` - name of the thermostat. For the page type *FreeDraw*, leave the feature blank;

Object properties
✕

Name:

Id:

Type:

Source/Receiver:

Serial number:

Control User schemes Events Embedded features Statistics

Feature name	Current value	Initial value	Unit	Range
PageType	3	FreeDraw <input type="text" value=""/>		0,1,2,3
PageName	Page1	<input type="text" value="Page1"/>	-	[0-15]
Object_1_Id	1	<input type="text" value="1"/>	-	[0-23]
Object_1_Name	-	<input type="text" value=""/>	-	[0-15]
Object_2_Id	2	<input type="text" value="2"/>	-	[0-23]
Object_2_Name	-	<input type="text" value=""/>	-	[0-15]
Object_3_Id	1	<input type="text" value="1"/>	-	[0-23]
Object_3_Name	-	<input type="text" value=""/>	-	[0-15]
Object_4_Id	1	<input type="text" value="1"/>	-	[0-23]
Object_4_Name	-	<input type="text" value=""/>	-	[0-15]

Auto refresh 
Refresh

OK
Cancel

**NOTE!** Sending the configuration only with the defined page type, without setting the binding of objects with the buttons is connected with starting the panel operation mode as *FreeDraw*. However, the buttons on the module will be inactive. This is related to the non-complementing of the `Object_X_Id` features.

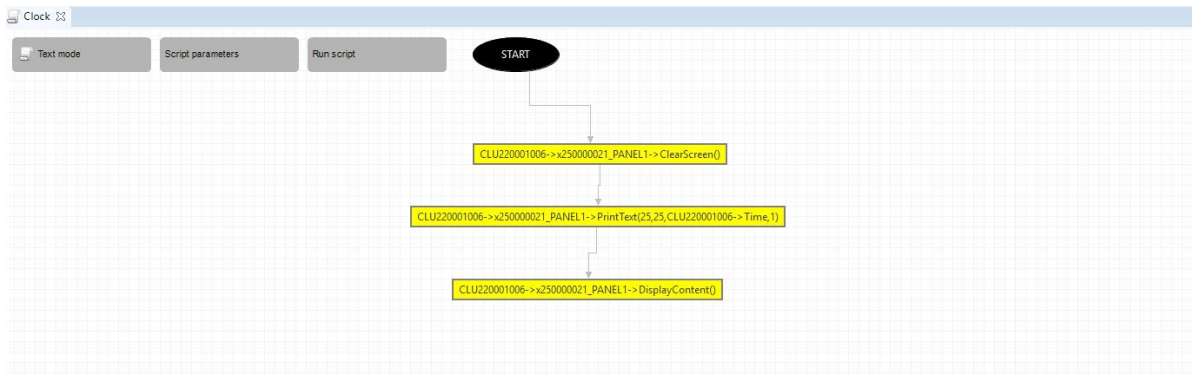
- Send the configuration to the CLU Z-Wave.

**NOTE!** It is possible to overwrite the display content by calling drawing methods from the Object Manager application or through other scripts that are not assigned to the `OnDraw` event. However, the overwritten content will be cleared when you move to another page or call the `Draw` method and wake up the screen.

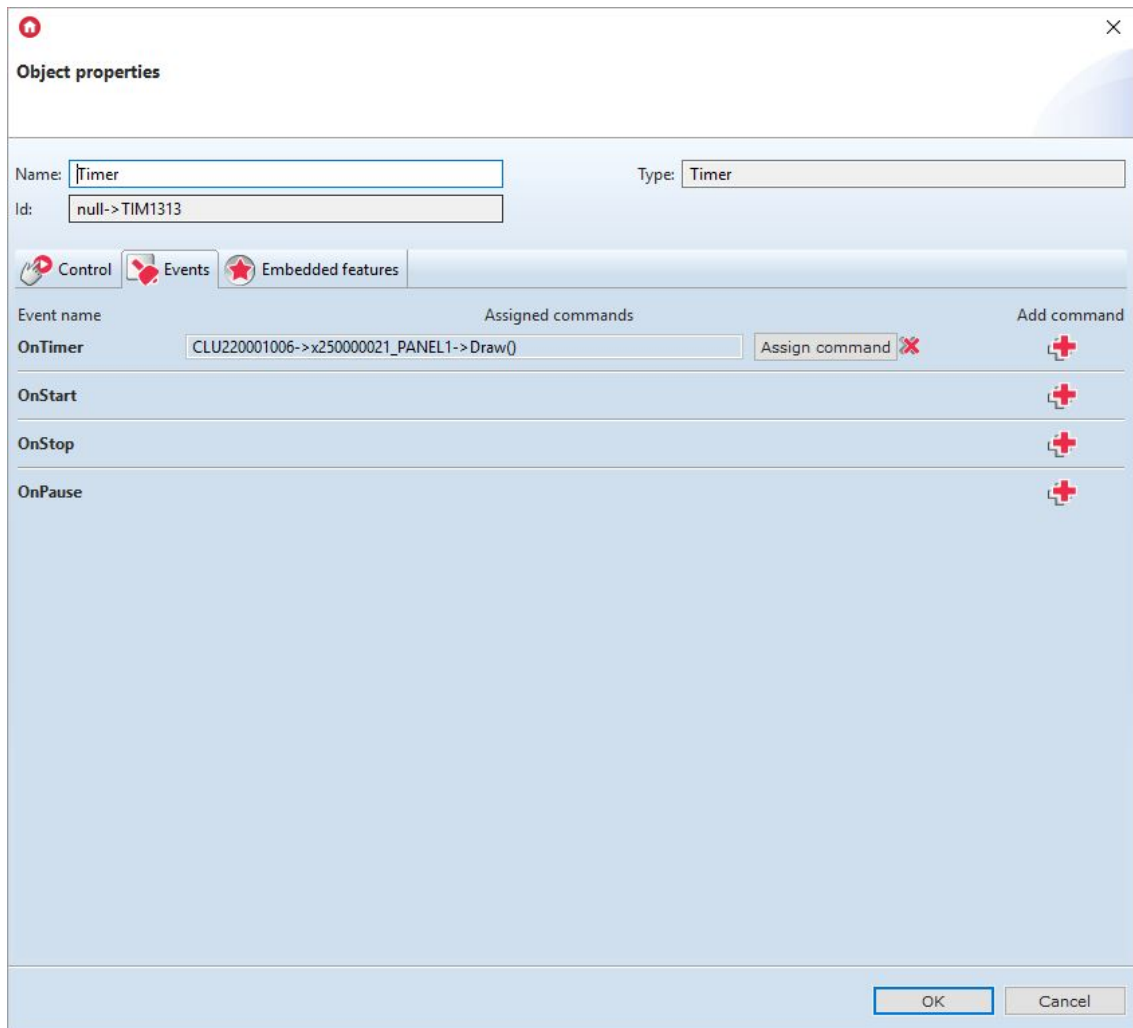
## B. Set up the site as a clock

To configure the site as a clock:

- Create a script displaying the current time (*Clock*);



- ○ Create a virtual object Timer:
  - Go to the tab *Events*;
  - Configure the operation of the virtual object by assigning the `Draw` method of the *Panel* object to the `OnTimer` event:



- Select the *Embedded features* tab and define the configuration parameters of the object:




**Object properties**

Name:  Type:

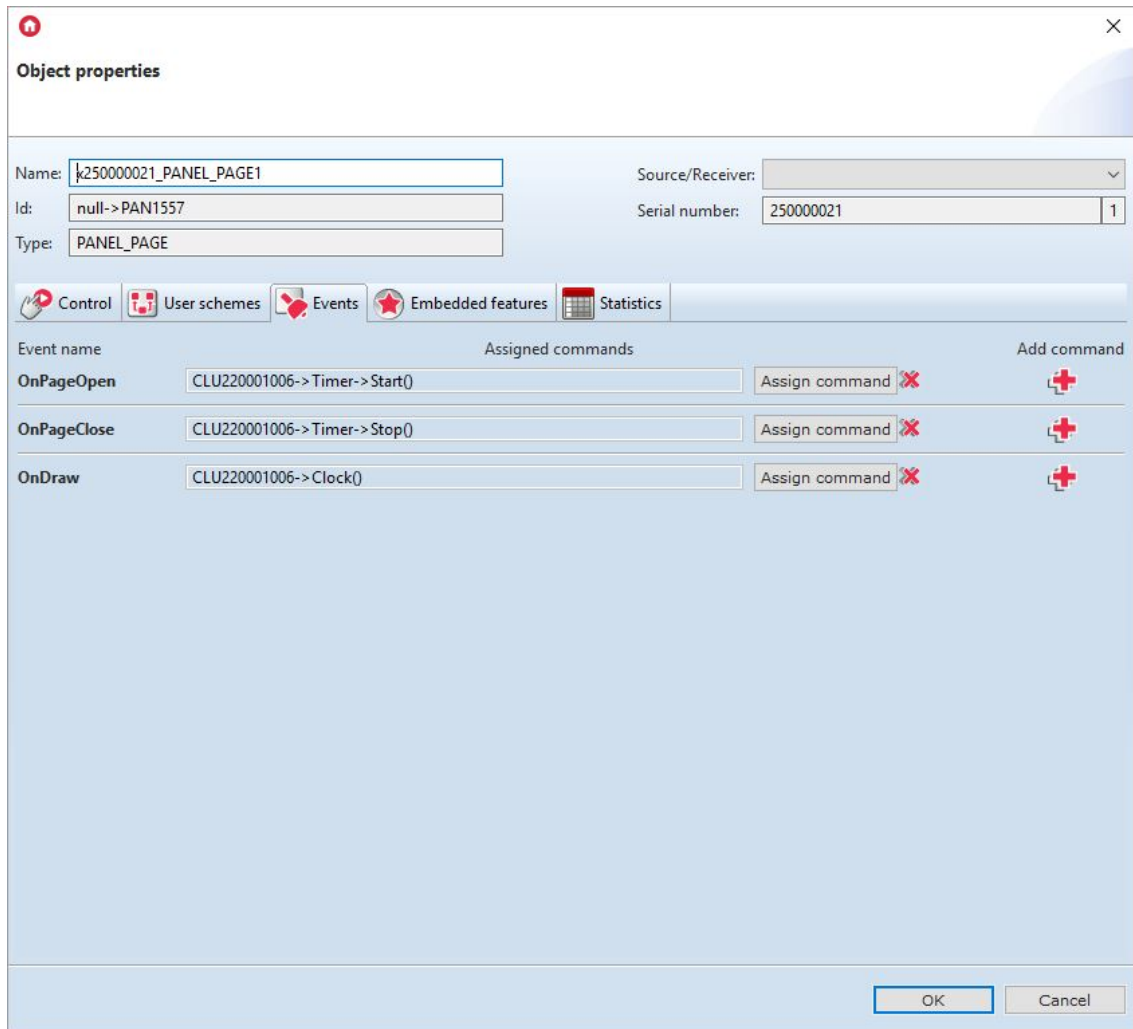
Id:

Control  Events  Embedded features

Feature name	Current value	Initial value	Unit	Range
Time	1000	<input type="text" value="1000"/>	ms	
Mode	1	<input type="text" value="Interval"/>		0,1
State	0			0,1,2
Value	0		ms	

Auto refresh 

- Open the *PANEL\_PAGEX* object (where X is the page number) by double-clicking on the list of objects:
  - Go to the tab *Events*
  - Configure the operation of the website by assigning methods to specific events (by clicking "+" on the right side of the window):



- Select the tab *Built-in features* and define the configuration parameters of the object;
- Send the configuration to the CLU Z-Wave.

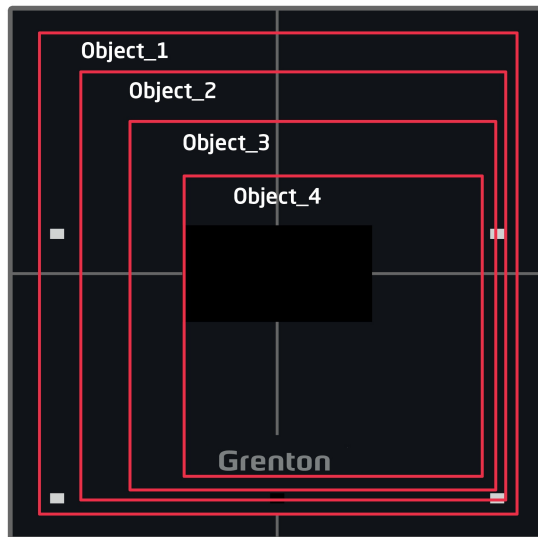
Script *Clock* in text version:

```
CLU220000260->x250000053_PANEL1->ClearScreen()
CLU220000260->x250000053_PANEL1->PrintText(25,25,CLU220000260->Time,1)
CLU220000260->x250000053_PANEL1->DisplayContent()
```

## 5.9. Creating a configuration using the Thermostats page object

In the *Thermostats* mode, a page consisting of 4 objects (including support for up to 16 objects on 4 pages) is available for which thermostat objects defined in the system are assigned. It is possible to change the parameters of thermostats such as set temperature or operating mode. It is also possible to switch the thermostat on or off.

**NOTE!** In the *Thermostats* mode, the buttons as well as the drawing of content on the display are blocked.



## Page type „Thermostats“

Creating a panel configuration that supports a page or pages of type *Thermostats* is best started by creating thermostats to be used in the configuration. Description of creation and operation of the virtual object *Thermostat* is described in subsection IX.5.

The v4 version of the Smart Panel module supports two types of thermostats:

- Local thermostat - it is a virtual object type *Thermostat* created on the CLU Z-Wave module, to which the Smart Panel module is connected with the currently created configuration;
- Remote thermostat - it is a virtual object of type *Thermostat* created on another CLU Z-Wave module;

Through the Smart Panel module it is possible to change such parameters of a virtual object *Thermostat* as:

- `PointValue` - preset temperature, the ability to read the currently set temperature as well as the change to a new value;
- `Mode` - thermostat mode:
  - In automatic mode `Auto (2)` the temperature value is read from the schedule. It is not possible to change this temperature via the Smart Panel module;
  - In manual mode `Manual (0)`, the temperature value is read from the `PointValue` feature. Through the Smart Panel module, it is possible to change this temperature;
- `State` - current thermostat status: off (`Off (0)`) / on (`On (1)`).

### A. Creating a configuration with a local thermostat

To create a configuration using a local thermostat you should:

- Create a thermostat on the Z-Wave CLU, to which the Smart Panel module is connected;
- Configure the virtual object as intended;
- Open object `PANEL_PAGEX` (where X is the number of one of 4 pages) by double clicking on the list of objects
- Select the tab *Embedded features* and define the objects displayed on the screen:
  - `PageType` - a feature that specifies the page type, set it to *Thermostats (2)*;
  - `PageName` - a feature that specifies the name of the page or icon that will be displayed when switching between pages (works only when the `PageDisplayMode`` feature is set to 1

(ShowIconOrName) in the Panel object);

- `Object_X_Id` - thermostat identifier. To do this, read the value in the field `Id` of the virtual object `Thermostat`. The local thermostat ID is not preceded by the CLU ID:

The screenshot shows a dialog titled "Object properties" for a "Thermostat" object. The "Name" field contains "LocalThermostat" and the "Type" field contains "Thermostat". The "Id" field, which is highlighted with a red border, contains the value "CLU220000260->THE3749".

- `Object_X_Name` - name of the thermostat. The lack of a thermostat name in the parameter causes that the thermostat is not displayed;

The screenshot shows a dialog titled "Object properties" for a "PANEL\_PAGE" object. The "Name" field contains "k250000021\_PANEL\_PAGE1", "Id" contains "null->PAN1557", and "Type" contains "PANEL\_PAGE". The "Source/Receiver" dropdown is set to "Thermostats" and the "Serial number" is "250000021". Below the fields is a tabbed interface with "Control", "User schemes", "Events", "Embedded features", and "Statistics". The "Embedded features" tab is active, showing a table of features.

Feature name	Current value	Initial value	Unit	Range
PageType	2	Thermostats		0,1,2,3
PageName	Page1	Page1	-	[0-15]
Object_1_Id	THE3749	THE3749	-	[0-23]
Object_1_Name	Kitchen	Kitchen	-	[0-15]
Object_2_Id	THE5081	THE5081	-	[0-23]
Object_2_Name	LivingRoom	LivingRoom	-	[0-15]
Object_3_Id	THE4059	THE4059	-	[0-23]
Object_3_Name	Hall	Hall	-	[0-15]
Object_4_Id	THE2718	THE2718	-	[0-23]
Object_4_Name	Bathroom	Bathroom	-	[0-15]

At the bottom, there is a checked "Auto refresh" checkbox and a "Refresh" button. "OK" and "Cancel" buttons are at the very bottom.

**NOTE!** Sending the configuration only with the defined page type, without setting the binding of objects with the buttons, is connected with starting the panel operation mode as *Thermostats*. The display will show dots ("..."). This is related to the non-complementing of the features `Object_X_Id` and `Object_X_Name`.

- Send the configuration to the CLU Z-Wave.

## B. Creating a configuration with a remote thermostat

To create a configuration using a remote thermostat you should:

- Create a thermostat on the Z-Wave CLU, to which the Smart Panel module with the current configuration is not connected;
- Configure the virtual object as intended;
- Open object *PANEL\_PAGEX* (where *X* is the number of one of 4 pages) by double clicking on the list of objects
- Select the tab *Embedded features* and define the objects displayed on the screen:
  - `PageType` - a feature that specifies the page type, set it to *Thermostats (2)*;
  - `PageName` - a feature that specifies the name of the page or icon that will be displayed when switching between pages (works only when the `PageDisplayMode` feature is set to 1 (ShowIconOrName) in the Panel object);
  - `Object_X_Id` - thermostat identifier. To do this, read the value in the field *Id* of the virtual object *Thermostat*. The remote thermostat ID must be preceded by the CLU ID:

Name:	<input type="text" value="RemoteThermostat"/>	Type:	<input type="text" value="Thermostat"/>
Id:	<input type="text" value="CLU220000331-&gt;THE5372"/>		

- `Object_X_Name` - name of the thermostat. The lack of a thermostat name in the parameter causes that the thermostat is not displayed;

**Object properties**

Name:  Source/Receiver:

Id:  Serial number:

Type:

Feature name	Current value	Initial value	Unit	Range
<b>PageType</b>	2	<input type="text" value="Thermostats"/>		0,1,2,3
<b>PageName</b>	Page1	<input type="text" value="Page1"/>	-	[0-15]
<b>Object_1_Id</b>	CLU220000331->THE5372	<input type="text" value="CLU220000331-&gt;THE5372"/>	-	[0-23]
<b>Object_1_Name</b>	Kitchen	<input type="text" value="Kitchen"/>	-	[0-15]
<b>Object_2_Id</b>	CLU220000331->THE6721	<input type="text" value="CLU220000331-&gt;THE6721"/>	-	[0-23]
<b>Object_2_Name</b>	LivingRoom	<input type="text" value="LivingRoom"/>	-	[0-15]
<b>Object_3_Id</b>	CLU220000331->THE9021	<input type="text" value="CLU220000331-&gt;THE9021"/>	-	[0-23]
<b>Object_3_Name</b>	Hall	<input type="text" value="Hall"/>	-	[0-15]
<b>Object_4_Id</b>	CLU220000331->THE5542	<input type="text" value="CLU220000331-&gt;THE5542"/>	-	[0-23]
<b>Object_4_Name</b>	Bathroom	<input type="text" value="Bathroom"/>	-	[0-15]

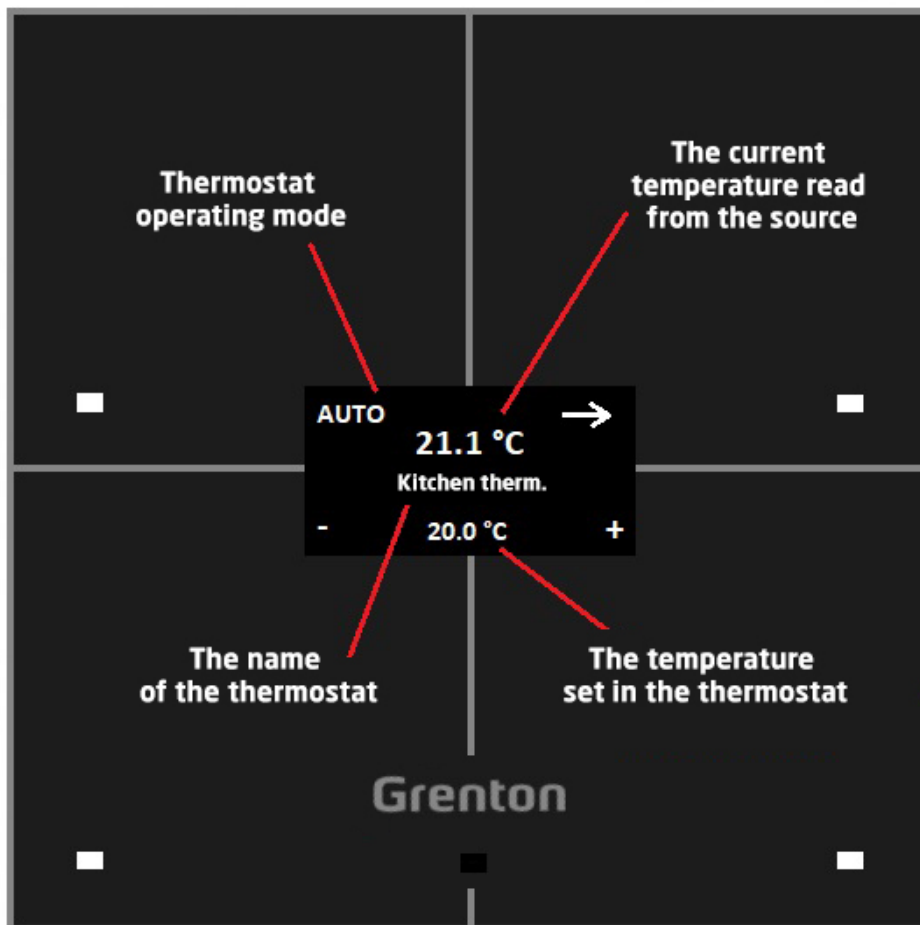
Auto refresh

**NOTE!** Sending the configuration only with the defined page type, without setting the binding of objects with the buttons, is connected with starting the panel operation mode as *Thermostats*. The display will show dots ("..."). This is related to the non-complementing of the features

`Object_X_Id` and `Object_X_Name`.

- Send the configuration to the CLU Z-Wave.

The diagram below shows an overview of the thermostat on the Smart Panel screen. Via the arrow, the user can go to the next thermostat on the page. However, you can change the set temperature using "-" / "+".



### C. Predefined button behavior

Button	Short / long press	Description of behavior
Top left	Short press (click)	Changing the thermostat operating mode: Manual / Auto
Top left	Long press (hold)	Change the state of thermostat: Off/On
Top right	Short press (click)	Go to the next thermostat on the page
Top right	Long press (hold)	No defined functionality
Lower left	Short press (click)	Reduction of the set temperature ( <code>PointValue</code> ) by 0.1 ° C
Lower left	Long press (hold)	Reduction of the set temperature ( <code>PointValue</code> ) - as long as the button is held down
Lower right	Short press (click)	Increase of the set temperature ( <code>PointValue</code> ) by 0.1 ° C
Lower right	Long press (hold)	Increase of the set temperature ( <code>PointValue</code> ) - as long as the button is held down

## 5.10. Connecting objects to larger buttons

The new version of Smart Panel also introduces the ability to combine / merge 2, 3 or 4 objects into one larger button. The functionality is available only in the *Buttons* and *FreeDraw* page mode. In order to create a bigger button you should:

- Configure `PANEL_BUTTOX` objects (where X is the button number):
  - In the *Events* tab configure the operation of the button by assigning methods to specific events;
  - In the *Built-in features*, define objects displayed on the screen of a given button;
- Open object `PANEL_PAGEX` (where X is page number);
- Go to the tab *Events*;
- Set up the website by assigning methods to specific events;
- Go to the tab *Embedded features*;
- Set the `PageType` feature to *Buttons* or *FreeDraw*;
- Set the `Object_X_Id` features according to any version of the join:
  - Merging 2 objects into one horizontal button - the icon set for the button is displayed in the middle at the top of the screen (for objects `Object_1_Id` and `Object_2_Id`) or lower part of the screen (for objects `Object_3_Id` and `Object_4_Id`);
  - merging 2 objects into one button vertically - the icon set for the button is displayed in the middle on the left part of the screen (for objects `Object_1_Id` and `Object_3_Id`) or on the right part of the screen (for objects `Object_2_Id` and `Object_4_Id`);
  - Merging 3 objects into one button - two identical icons are displayed, depending on how the objects are connected;

- Merging 4 objects into one button - the icon set for the button is displayed in the center of the screen

## 6. Configuration of the Smart Panel v6

**NOTE!** Smart Panel in the v6 version is available for Object Manager in version 1.4.1 and higher and for CLU with firmware 5.08.01 and higher.

### 6.1. Configuration parameters

In the latest version of the Smart Panel v6 module, new configuration parameters have been introduced for such objects as:

- PANEL,
- PANEL\_PAGE,
- PANELSENSTEMP.

The full list of changes introduced in the V6 version can be found in the release notes of the given version: [Release Notes - Smart Panel module](#)

### 6.2. New functionality

#### A. The mechanism of informing about incorrect configuration / entering to the Distributed Logic mode

A new functionality introduced with the v6 version is the mechanism of informing the user about a wrong configuration or entering to the Distributed Logic mode. This mechanism is based on the fact that the module waits about 10 seconds for receiving the configuration after sending it or restarting the system. After this time, the waiting period for configuration ends and the user will be informed about a misconfiguration or switching to Distributed Logic mode by one short and low beep.

#### B. Distributed Logic mode

Another functionality added to the latest version of the Smart Panel module is the Distributed Logic mode. It is available from version 6.1.8-2115 and higher. Detailed information on the configuration and operation of Distributed Logic - [see XIX](#).

### 6.3. Changing the UI and the mechanism of operation of Thermostats pages

The Smart Panel module in version v6 offers a refreshed UI of Thermostats websites, as well as new functions. The method of creating and configuring a page with the use of Thermostats type objects is the same as for the previous versions - [see XII.5.9](#).

#### A. Thermostat UI change

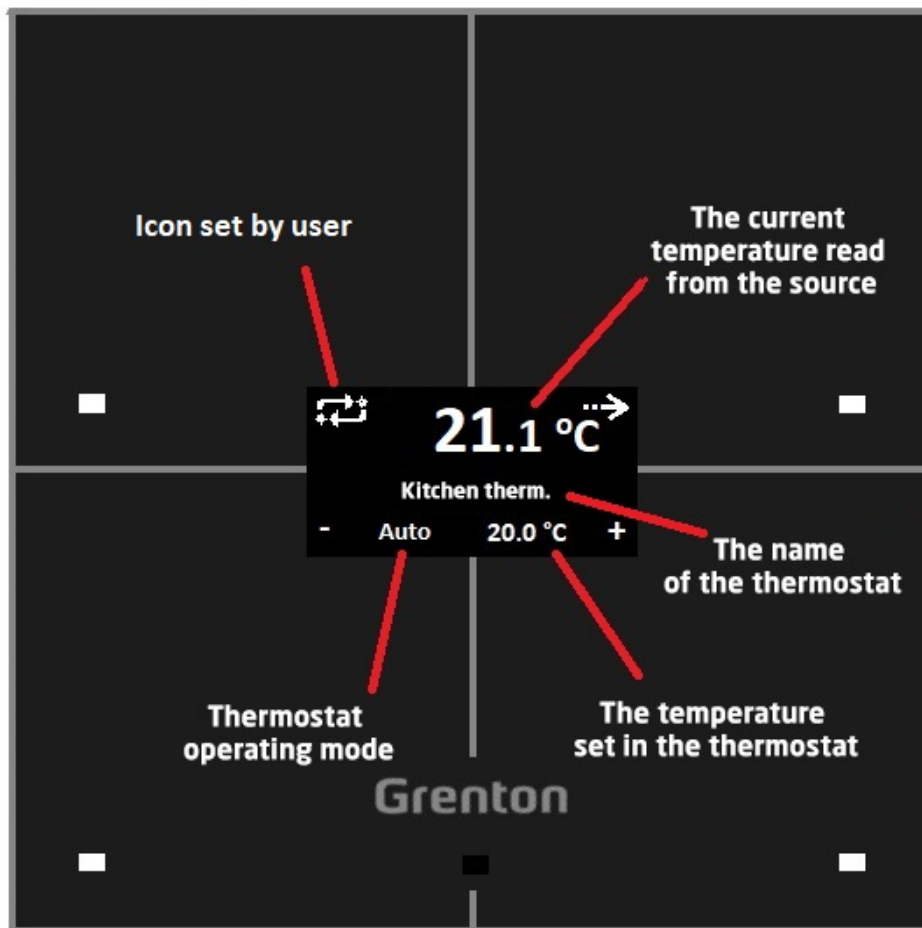
The diagram below shows an overview the appearance of the refreshed thermostat on the Smart Panel screen. Several elements have changed:

- entering the icon in the upper left corner - by default, the "chmode" icon on the SD card is displayed (the icon is shown in the diagram). However, if this icon is not present on the SD card, the word "mode" will be displayed. Additionally, the user can enter his own icon using the

`SetObject_X_CustomIcon` method or the feature `Object_X_CustomIcon`,



- using the arrow, the user can go to the next thermostat on the page (short press of the button in case of more than one thermostat on the page) or go to the next page (short press in case of one thermostat on the page, longer press of the button in case of more than one thermostat on the page ),
- dots have been introduced next to the arrow, which indicate the number of the currently displayed thermostat (one dot - Object\_1\_Id, two dots - Object\_2\_Id, etc.). If there is only one thermostat on the website, the dots are not displayed,
- using "-" / "+" it is possible to change the set temperature and the operating mode of the thermostat from Auto to Manual,
- a longer pressing of the upper left button (hereinafter referred to as 'mode') turns the thermostat off / on or changing from Manual mode to Auto mode,
- when the thermostat is turned off, the set temperature disappears and the word "Off" appears, which is located centrally,
- the display of temperature read from the source has also changed - now the temperature value before the decimal point is displayed in larger font, while the value after the decimal point and the unit are displayed in a smaller font. Additionally, the display of this temperature depends on the size of the entered icon - more on this subject in the next section.



## B. New features on the Thermostats page

From version v6 of the Smart Panel module, new functionalities are available on the Thermostats page. It is related to the changes made to the UI of the thermostat.

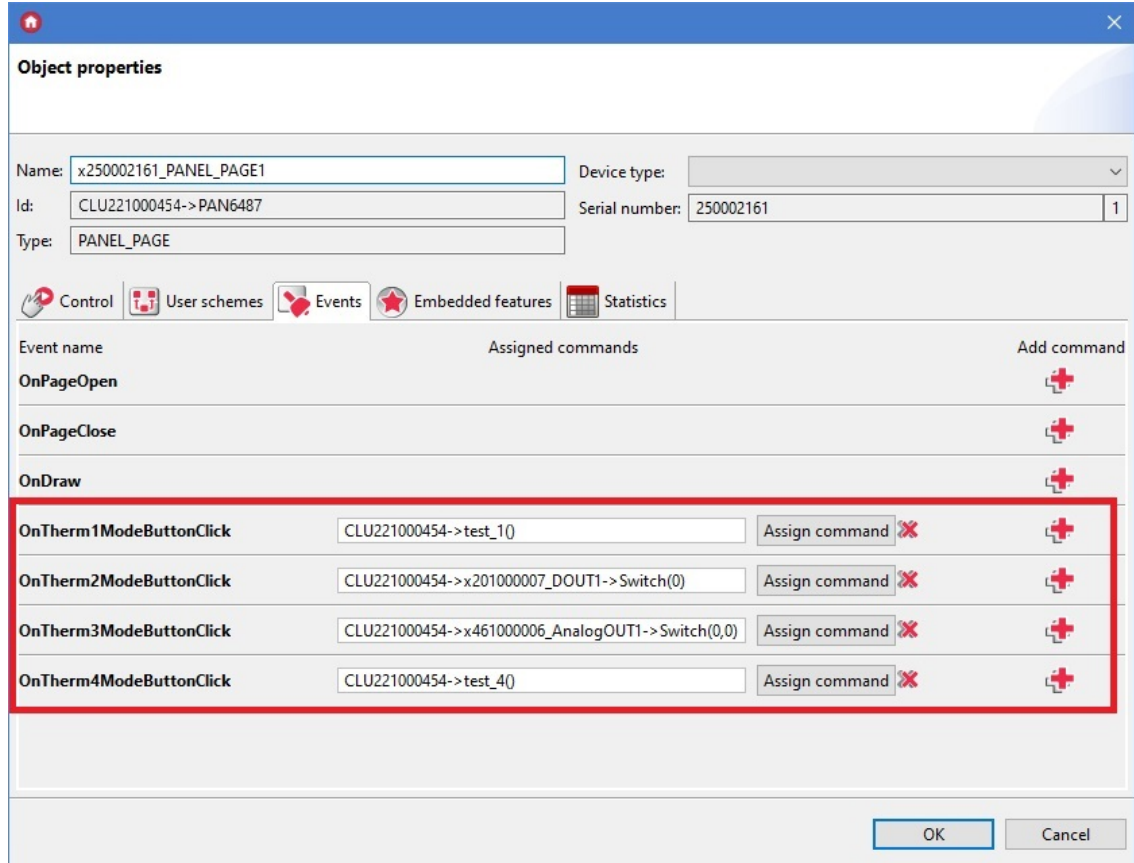
### 1. Possibility to set your own icons

The first new functionality is the aforementioned possibility for the user to set their own icons in the upper left corner of the display. Their change is possible both through the `SetObject_X_CustomIcon` method as well as through the feature `Object_X_CustomIcon`. The width of the entered icon affects the "x" coordinate of the current temperature. The 64 x 32 pixel

icons are allowed. If the icon width  $x > 64$  is exceeded, only the icon itself will be drawn on the display - the UI of the thermostat will not be displayed - it is the so-called "big icon" mode. To go back to the thermostat interface, set an icon whose width does not exceed 64 pixels.

## 2. Ability to assign actions to new events

Another functionality introduced in the new version of the module is the ability to assign actions to new events `OnThermXModeButtonClick`, where X is the number of the thermostat on the page. This event is generated when the 'mode' button (upper left button) is clicked.



## 3. An example of configuration of new functionalities

### 1. Local thermostat configuration with operation change (heating / cooling)

The following objects are used to create this configuration:

- Thermostat virtual object,
- DOUT1 object (e.g. Relay module) - responsible for heating / cooling activation - used in the Thermostat virtual object,
- DOUT2 object (e.g. Relay module) - selection of what is to be switched on: heating or cooling - used in scripts.

The following screenshot shows the configuration of the PANEL\_PAGE object and the Thermostat virtual object in the default setting, which is heating.

**Object properties**

Name:  Device type: 
 Id:  Serial number:  
 Type:

Control  User schemes  Events  Embedded features  Statistics

Feature name	Current value	Initial value	Unit	Range
PageType	2	Thermostats <input type="text" value="Thermostats"/>		0,1,2,3
PageName	Thermostats	Thermostats <input type="text" value="Thermostats"/>	-	[0-15]
Object_1_Id	THE5235	THE5235 <input type="text" value="THE5235"/>	-	[0-23]
Object_1_Name	Living room	Living room <input type="text" value="Living room"/>	-	[0-15]
Object_1_CustomIcon	sun	sun <input type="text" value="sun"/>	-	[0-9]
DistributedLogicGroup_1	0	0 <input type="text" value="0"/>		[0-10000]
Object_2_Id	-	<input type="text" value=""/>	-	[0-23]
Object_2_Name	-	<input type="text" value=""/>	-	[0-15]

Auto refresh

**Object properties**

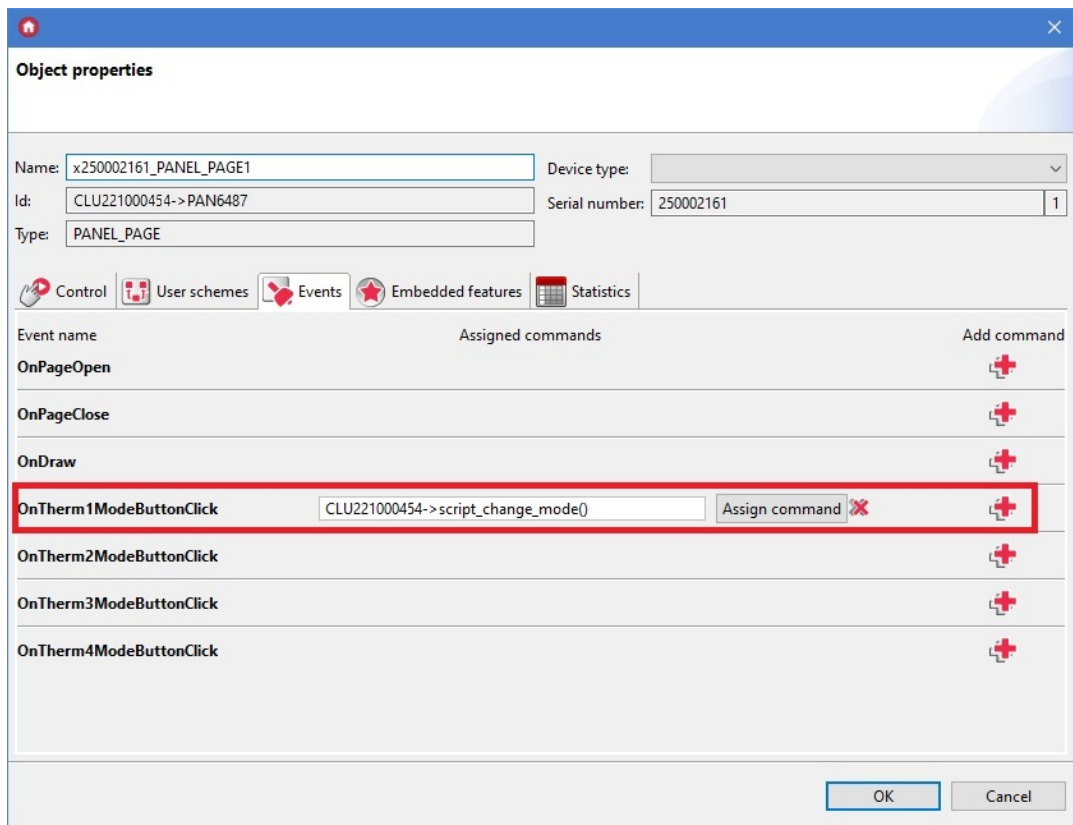
Name:  Type: 
 Id:

Control  Events  Embedded features  Scheduler

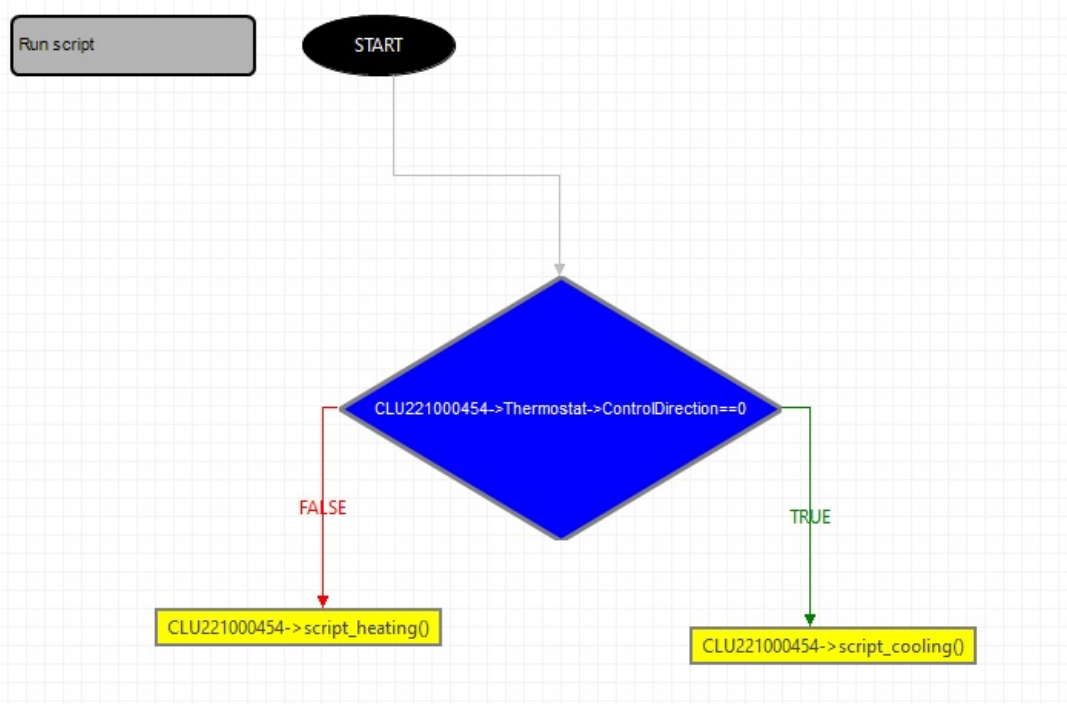
Feature name	Current value	Initial value	Unit	Range
Source	27.50	<a href="#">CLU221000454-&gt;x250002161_PANELSENSTEMP1</a>		
Control	0	<a href="#">CLU221000454-&gt;x191000012_DOUT1</a>		
OutputType	0			-1,0,1
PointValue	20.0	20.0 <input type="text" value="20.0"/>		
HolidayModeValue	17	17 <input type="text" value="17"/>		
Hysteresis	2	2 <input type="text" value="2"/>		
State	1			0,1
ControlDirection	0	Normal <input type="text" value="Normal"/>		0,1
Mode	0			0,1,2,3

Auto refresh

Configuration of the OnTherm1ModeButtonClick event:



Script *script\_change\_mode()* that changes the operating mode of the thermostat from heating to cooling and vice versa:



The above script in the text version:

```

if (CLU221000454->Thermostat->ControlDirection==0) then
  CLU221000454->script_cooling()
else
  CLU221000454->script_heating()
end
  
```

Script *script\_heating()* for changing the ControlDirection feature to a value responsible for heating, as well as changing the icon and selecting heating / cooling:

```
CLU221000454->Termostat->SetControlDirection(0)
CLU221000454->x201000007_DOUT2->SwitchOff(0)
CLU221000454->x250002161_PANEL_PAGE1->SetObject_1_CustomIcon("sun")
```



Script *script\_cooling()* for changing the ControlDirection feature to the value responsible for cooling, as well as changing the icon and selecting heating / cooling:

```
CLU221000454->Termostat->SetControlDirection(1)
CLU221000454->x201000007_DOUT2->SwitchOff(1)
CLU221000454->x250002161_PANEL_PAGE1->SetObject_1_CustomIcon("cold")
```



### C. Predefined button behavior

Button	Short / long press	Description of behavior
Top left	Short press (click)	Generating an OnThermXModeButtonClick event, where X is the number of the thermostat on the page
Top left	Long press (hold)	Change of the thermostat status: Off/On. In addition, it allows to switch from Manual mode to Auto mode
Top right	Short press (click)	Go to the next thermostat on the page if there is more than one thermostat on the page Go to the next page if there is only one thermostat on the page
Top right	Long press (hold)	Go to the next page
Lower left	Short press (click)	Reduction of the set temperature ( <input type="text" value="PointValue"/> ) by 0.1 ° C, as well as changing the operating mode from Auto to Manual
Lower left	Long press (hold)	Reduction of the set temperature ( <input type="text" value="PointValue"/> ) - as long as the button is held down, as well as changing the operating mode from Auto to Manual
Lower right	Short press (click)	Increasement of the set temperature ( <input type="text" value="PointValue"/> ) by 0.1 ° C, as well as changing the operating mode from Auto to Manual
Lower right	Long press (hold)	Increasement of the set temperature ( <input type="text" value="PointValue"/> ) - as long as the button is held down, as well as changing the operating mode from Auto to Manual

# XIII. GATE ALARM Module

---

**NOTE!** The described functionality and integration with the mentioned alarm control panels is available for **GRENTON GATE ALARM, DIN, Eth (INT-221-E-01)** with **firmware 1.1.0-2034C or higher!**

## 1. General information

---

The module is used to integrate the Grenton Smart Home system with 3rd party alarm systems. Grenton GATE ALARM allows integration with Satel and Jablotron alarm systems.

## 2. Module configuration

---

**NOTE!** Before starting any work with the GATE ALARM module, it is necessary to update the interface database!

### Time setting via NTP server

The GATE Modbus module allows you to set the time using the NTP server, taking into account the time zone and changing the time (winter / summer). The time is taken automatically from the NTP server (*pool.ntp.org*).

There are three features for configuration:

- `UseNTP` - determines whether GATE uses NTP,
- `NTPTimeout` - waiting time for a response from the NTP server,
- `TimeZone` - setting the GATE time zone - 22 zones are available.

**NOTE!** Getting the time from an NTP server requires that GATE be in a network that has an internet connection.

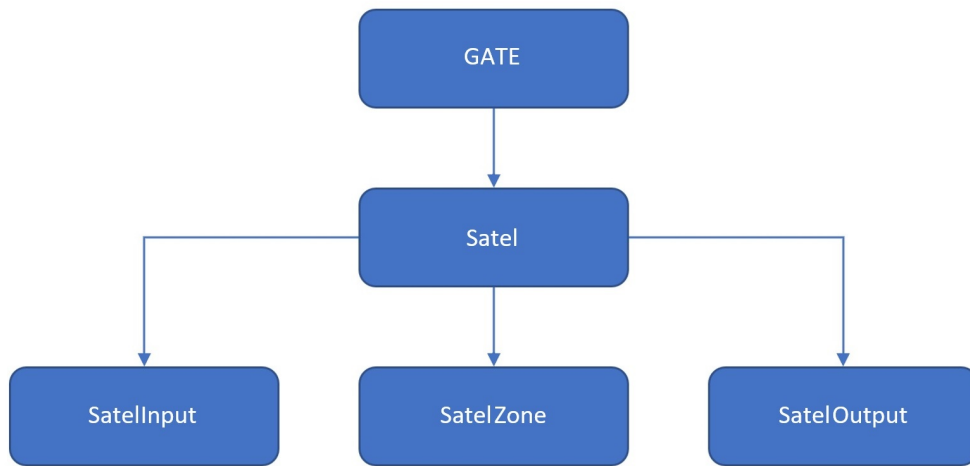
## 3. Integration with the Satel alarm control panel

---

### 3.1. General information

Integration of the Grenton system with the Satel alarm control panel is possible via the ETHM-1 module. You can create up to 64 virtual objects of the type: ``SatelZone, SatelInput, SatelOutput`. It is also possible to use the integration coding offered by Satel.

The configuration structure is as follows:



Virtual objects:

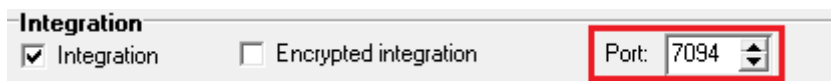
- **Satel** - allows you to perform a configuration that allows you to integrate the system with the Grenton alarm panel;
- **SatelZone** - allows to create a zone to which access will be possible after entering the password of one of the users or the password of the administrator himself;
- **SatelInput** - gives the ability to monitor the status of the selected input;
- **SatelOutput** - allows you to monitor and set the status of the selected output after entering the user or administrator password.

## 3.2. Configuration for the Satel system

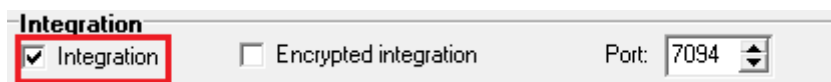
**NOTE!** All required information can be found in the ETHM module configuration - using the keypad connected to the Satel panel or using a dedicated DLOADX program.

Before starting the configuration you should have information about the Satel central unit and the ETHM-1 module:

- IP address of the ETHM module (Satel) - available in the Satel configuration (*DLOADX -> Data -> Structure and Hardware -> Equipment tab -> Keypads -> ETHM-1 -> section Server IP address*);
- ETHM integration port - available in the Satel configuration (*DLOADX -> Data -> Structure and Hardware -> Equipment tab -> Keypads -> ETHM-1 -> Integration section*);

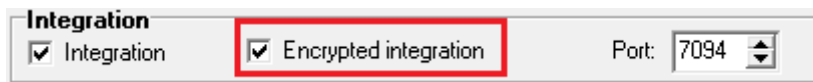


- Administrator / users password - the default password in the Satel configuration for the administrator is: 1111 (*DLOADX -> Users -> Users*);
- Integration on the side of the ETHM **module must be enabled** (*DLOADX -> Data -> Structure and Hardware -> Equipment tab -> Keypads -> ETHM-1 -> Integration section*);

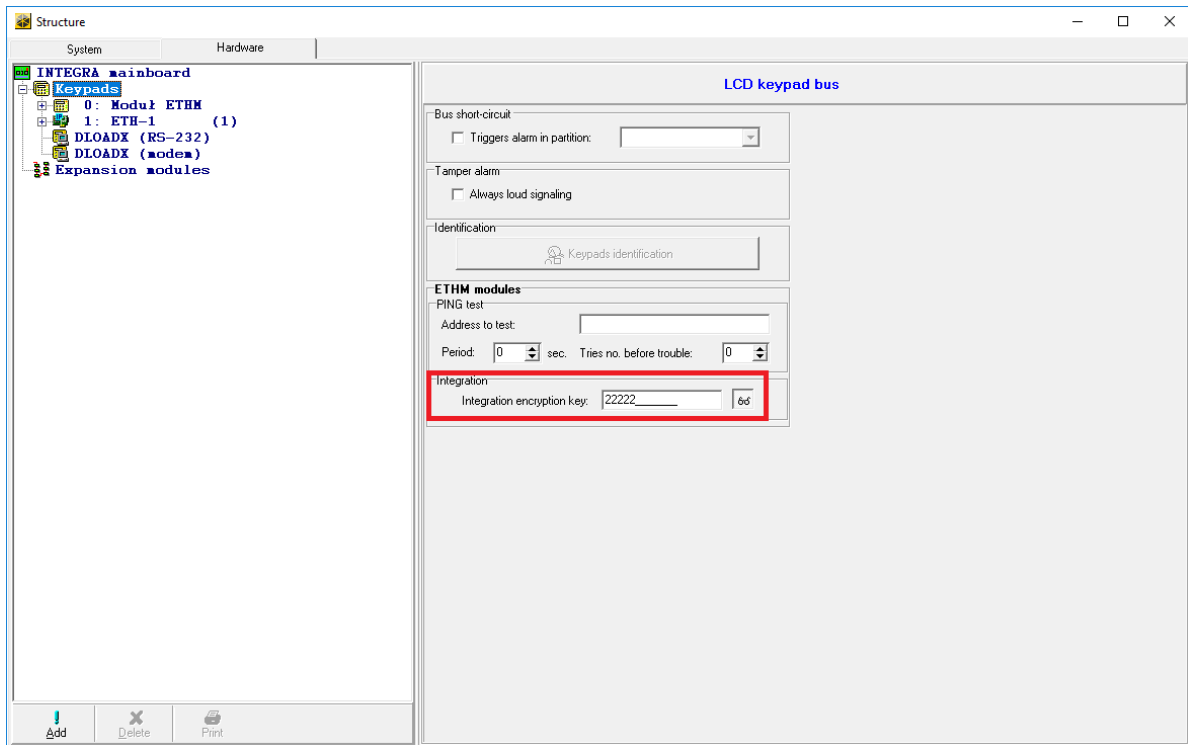


- In case when encryption - *Integration Encoding* is enabled, you should also know the encryption key (*DLOADX -> Data -> Structure and Hardware -> Equipment tab -> Keypads -> ETHM-1 -> Integration section*);





- The coding key can be found in the Satel configuration (*DLOADX -> Data -> Structure and Hardware -> Equipment tab -> Keypads*) or read it using the keypad (*Keypad -> Service mode -> Options -> Key integration*).

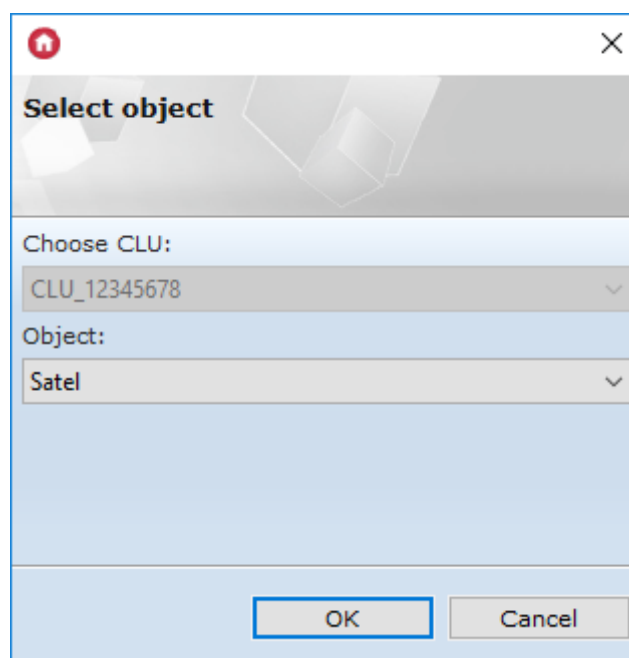


### 3.3. Virtual Objects

#### A. Satel

To perform the correct configuration of the GATE Alarm module it is necessary to:

- Create a virtual object *Satel*:



- Go to configuration - tab *Embedded features* and enter:

- **IP** - IP address of the ETHM module (Satel);
- **Port** - ETHM integration port;
- **AdminPassword** - administrator password;
- **EncryptionEnabled** - enable encoding - set if the integration on the ETHM module is marked with *Integration Encoding*;
- **Encryption Key** - integration key (for attached encoding);

CLU\_12345678->Satel1

Name:  Type:

Control Events **Embedded features**

Feature name	Current value	Initial value	Unit	Range
<b>State</b>	0		bool	0,1
<b>LastError</b>	0			
<b>IP</b>	192.168.1.10	<input type="text" value="192.168.1.10"/>	string	
<b>Port</b>	7094	<input type="text" value="7094"/>		[1-65535]
<b>AdminPassword</b>	1111	<input type="text" value="1111"/>	string	
<b>UpdateTime</b>	1000	<input type="text" value="1000"/>		[1000-20000]
<b>EncryptionEnabled</b>	true	<input type="text" value="True"/>	bool	0,1
<b>EncryptionKey</b>	22222	<input type="text" value="22222"/>	string	

Auto refresh

Information on where to find the information you need can be found in the second section - [look up XIII.1.2.](#)

- Send configuration and verify connection - tab *Embedded features*, 'State' feature (1 - correctly connected to the control panel, 0 - no connection):

CLU\_12345678->Satel1

Name:  Type:

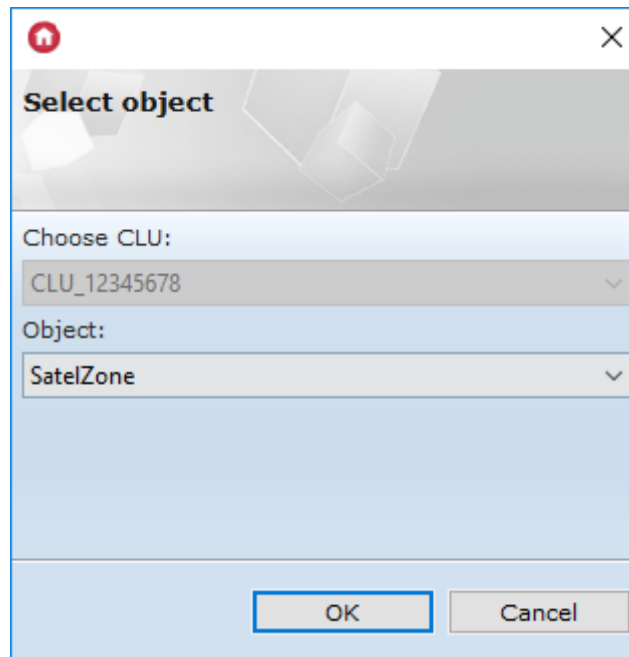
Control Events **Embedded features**

Feature name	Current value	Initial value	Unit	Range
<b>State</b>	1		bool	0,1
<b>LastError</b>	0			

## B. Zone

The GATE Alarm module allows you to add a virtual object *Zone*:

- Create an object *SatelZone*:



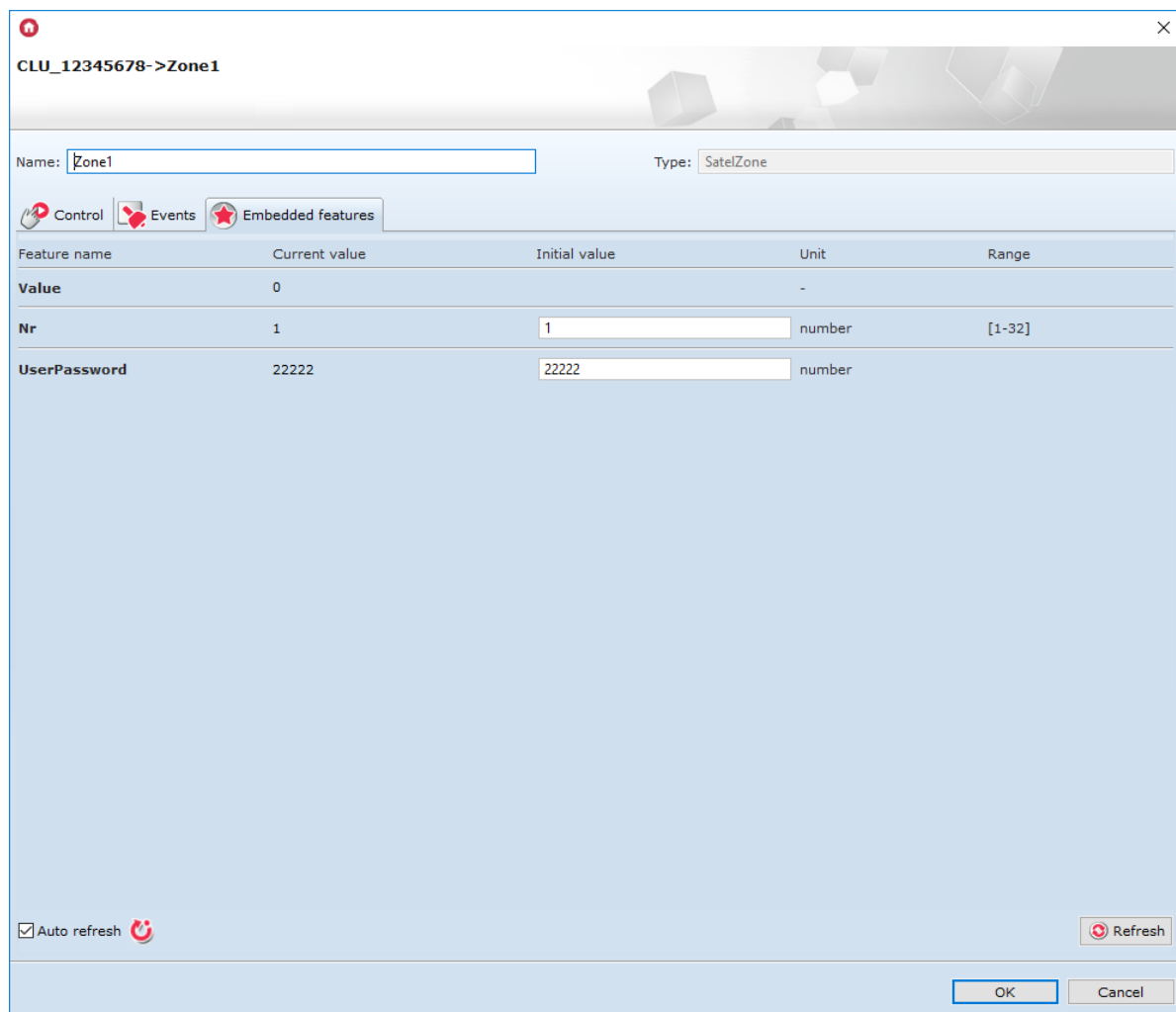
**Select object**

Choose CLU:  
CLU\_12345678

Object:  
SatelZone

OK Cancel

- Define the No. (number of the selected zone) and enter the user's password:



**CLU\_12345678->Zone1**

Name: Zone1 Type: SatelZone

Control Events Embedded features

Feature name	Current value	Initial value	Unit	Range
<b>Value</b>	0		-	
<b>Nr</b>	1	1	number	[1-32]
<b>UserPassword</b>	22222	22222	number	

Auto refresh Refresh

OK Cancel

- Send configuration and verify the connection - tab *Built-in features*, the `Value` feature (-1 means no connection to the control panel, the others mean a correct connection and the state of the zone

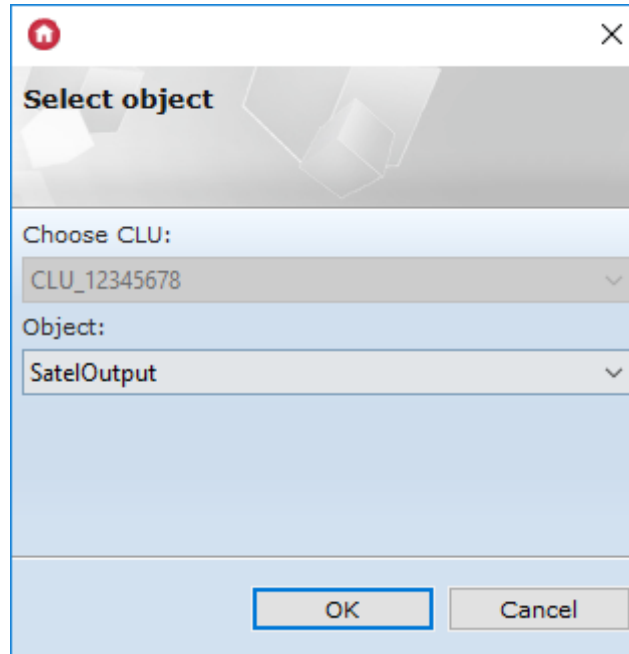
is returned: 0 or 1);

- Arm / disarm the zone - the `ArmZone` and `DisarmZone` methods.

## C. Output

GATE Alarm allows adding a virtual object *Output*:

- Create an *SatelOutput* object:



The screenshot shows a dialog box titled "Select object" with a red home icon and a close button (X) in the top right corner. The dialog contains two dropdown menus. The first is labeled "Choose CLU:" and has "CLU\_12345678" selected. The second is labeled "Object:" and has "SatelOutput" selected. At the bottom of the dialog, there are two buttons: "OK" and "Cancel".

- Define the No. (number of the selected output on the Satel board) and enter the user's password:

CLU\_12345678->Output1

Name:  Type:

Control Events **Embedded features**

Feature name	Current value	Initial value	Unit	Range
<b>Value</b>	0		bool	[0-1]
<b>Nr</b>	1	<input type="text" value="1"/>	number	[1-256]
<b>UserPassword</b>	1234	<input type="text" value="1234"/>	number	[0-99999]

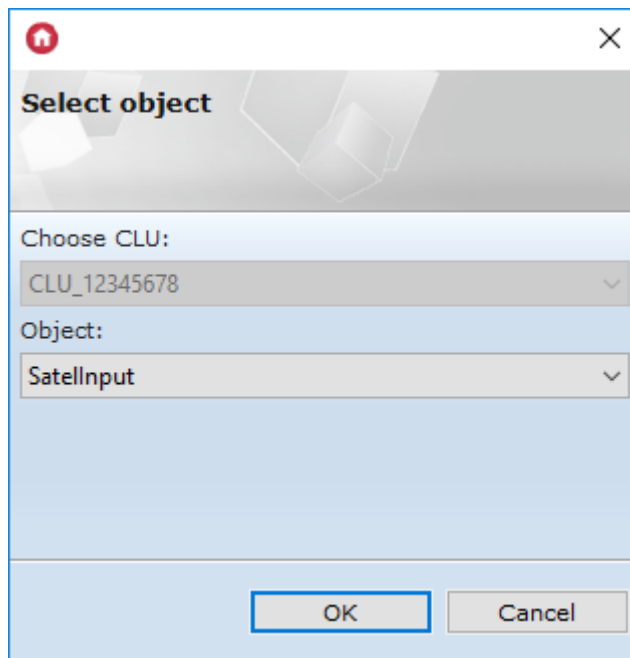
Auto refresh

- Send configuration and verify the connection - tab *Embedded features*, the  feature (-1 means no connection to the central unit, the others mean a correct connection and the status of the zone is returned: 0 or 1);
- Switch on / off the output - methods  and .

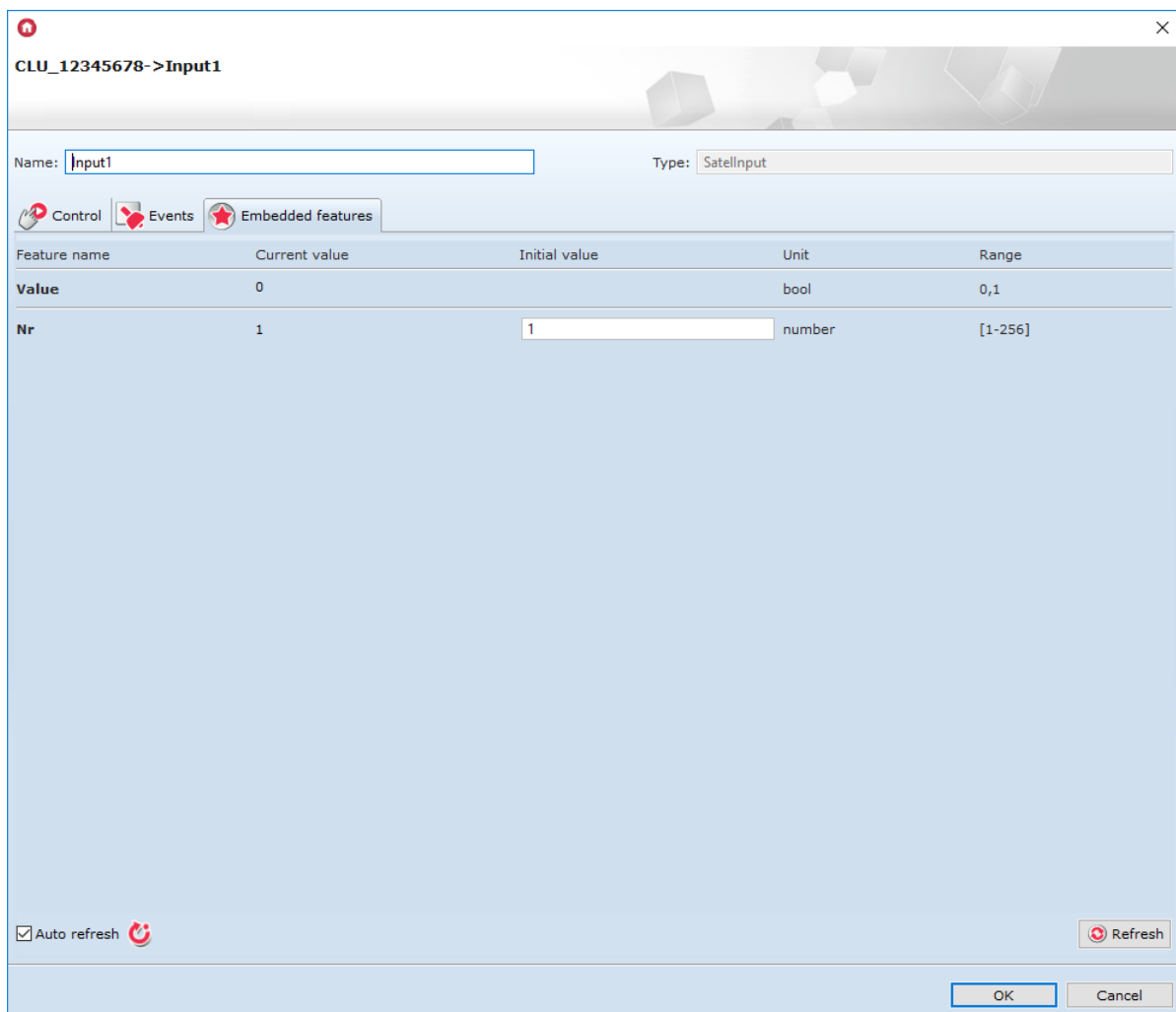
## D. Input

GATE Alarm allows adding a virtual object *Input*:

- Create an *SatellInput* object:



- Define No. (number of the selected input on the SateI):



- Send configuration and verify the connection - tab *Embedded features*, the `Value` feature (-1 means no connection to the control panel, the others mean a correct connection and the status of the zone is returned: 0 or 1).

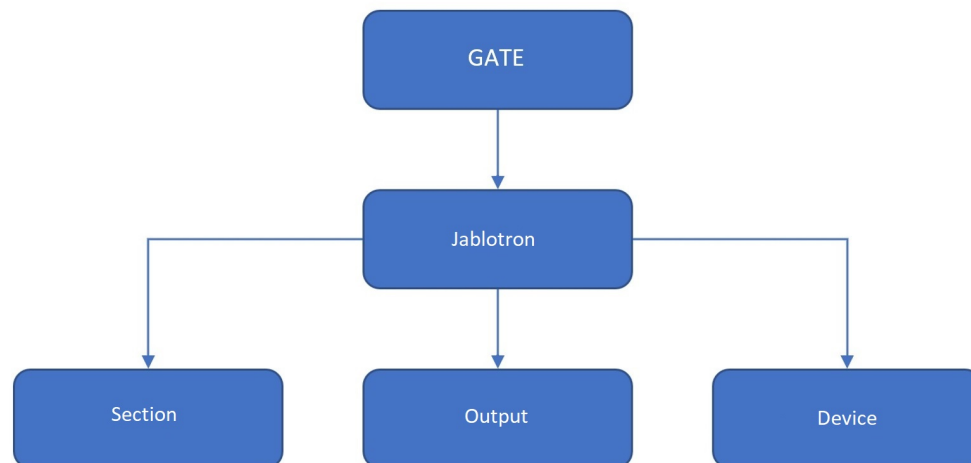
## 4. Integration with the Jablotron control panel

## 4.1. General information

**NOTE!** The described functionality and integration with the mentioned alarm control panels is available for **GRENTON GATE ALARM, DIN, Eth (INT-221-E-01)** with **firmware 1.0.2-194202 or higher!**

**NOTE!**  
Grenton system integration concerns the **Jablotron JA-101K-LAN** control panel.

Integration of the Grenton system with the Jablotron control panel is possible via the JA-121T module. It is possible to create virtual objects of the type: *JablotronSection*, *JablotronOutput*, *JablotronDevice*. The configuration structure looks like this:



- **Jablotron** - allows configuration to be made to integrate the system with the Jablotron alarm control panel;
- **JablotronSection** - allows creating a zone to which access will be possible after entering the password of one of the users or the password of the administrator himself;
- **JablotronDevice** - gives the ability to monitor the status of the selected input / device;
- **JablotronOutput** - allows to monitor and set the status of the selected output after entering the user or administrator password.

## 4.2. Configuration for the Jablotron system

**NOTE!** An interface database update is required before any work with the GATE Alarm module!

For integration between the Gate module and the Jablotron control panel, the **JA-121T** module is used. The JA-121T module must be connected to the Jablotron control panel system and added to the system list. Information on adding / using individual Jablotron modules can be found in the manufacturer's documentation.

Communication between the Gate and the JA-121T module takes place via the RS485 interface - connection between the screw terminals A (D +), B (D-) on both modules.

**NOTE!** The JA-121T module must be enabled in the Passive mode.

RS-485 BUS internal settings (JA-121T)

Terminal console	Device basic mode
Disabled	TMP terminal
Enrolling	Motherboard button
0	Period [min] after which a fault is triggered
every 10 s	Connection confirmed
<input checked="" type="checkbox"/> Passive mode	

**NOTE!** Information on the remaining settings and their application can be found in the documentation of the JA-121T module on the manufacturer's website.

After connecting the modules and correctly configuring the JA-121T device, you can proceed to creating and configuring virtual objects of the GATE Alarm module.

## 4.3. Virtual objects

### A. Jablotron

In order to properly configure the GATE Alarm module, you must create the virtual **Jablotron** object:

**Select object**

Choose CLU:

Object:

Then go to configuration - Built-in Features tab and enter

- **AdminCode** - administrator access code;
- **UpdatePeriod** - Control panel status update period;



**Object properties**

Name:  Type:

Id:

Control
  Events
  Embedded features

Feature name	Current value	Initial value	Unit	Range
AdminCode	1*1234	<input type="text" value="1*1234"/>	string	
UpdatePeriod	1000	<input type="text" value="1000"/>		[1000-25000]

Auto refresh

## B. Section / zone

GATE Alarm allows you to add a virtual **JablotronSection** object:

Creating the *JablotronSection* object:

**Select object**

Choose CLU:

Object:

Then go to configuration - Built-in Features tab and enter:

- **Nr** - parameter defining to which section the object refers;
- **UpdatePeriod** - The access code will use the admin code for '\*';

**Object properties**

Name:  Type:

Id:

Control  Events  Embedded features

Feature name	Current value	Initial value	Unit	Range
State	1.00		number	
Nr	1	<input type="text" value="1"/>	number	[1-15]
AccessCode	*	<input type="text" value="*"/>	string	

Auto refresh

## C. Output

GATE Alarm allows you to add a virtual object **JablotronOutput**

Creating the *JablotronOutput* object:

**Select object**

Choose CLU:

Object:

Then go to configuration - Built-in Features tab and enter:

- **Nr** - parameter defining to which output the object relates;
- **AccessCode** - The access code will use the admin code for '\*';

**Object properties**

Name:  Type:

Id:

Control
  Events
  Embedded features

Feature name	Current value	Initial value	Unit	Range
State	0		bool	[0-1]
Nr	1	<input type="text" value="1"/>	number	[1-32]
AccessCode	*	<input type="text" value="*"/>	string	

Auto refresh

## D. Input / Device

GATE Alarm allows you to add a virtual object **JablotronDevice**

Creating the *JablotronDevice* object:

**Select object**

Choose CLU:

Object:

Then go to configuration - Built-in Features tab and enter:

- **Nr** - parameter defining to which input the object refers;

Object properties
✕

Name:

Id:

Type:

Control
 Events
 Embedded features

Feature name	Current value	Initial value	Unit	Range
Value	0		bool	0,1
Nr	1	<input type="text" value="1"/>	number	[1-256]

Auto refresh

Refresh

OK

Cancel

## 5. Virtual object - Timer

Timers are virtual objects created as part of a given GATE module. Timers can be used wherever it is necessary to call a method after a specified time or also to call it cyclically.

The timer can operate in two modes:

- **Countdown** After starting, it counts down the set time. At the end of the countdown, the method associated with the `OnTimer` event is started, and the timer stops and does not count down until the next start using the `Start` method.
- **Interval** Cyclic timer - after the start, it starts counting down the set time. After its expiry, the timer calls the method associated with the `OnTimer` event, and the timer itself again begins to count down the set time. The situation is repeated until it is stopped by the `Stop` method.

## 6. Restoring factory settings - *Hard Reset*

Running the *Hard Reset* function on the GATE Alarm module causes:

- Removal of the saved configuration;
- Formatting the flash memory partition;
- Removal of all created LUA objects;
- Loss of communication between OM / HM and Gate module.

In order to restore the factory settings with the *Hard Reset* function, perform the following steps (in accordance with the given order):

- Disconnect power from the Gate module

- Press and hold the *Reset* button on the module (the button is located under the bottom end of the module)
- Connect the power supply to the Gate module
- Keep the *Reset* button pressed for at least 10 seconds - during the reset the green LED will be on steady. Correct completion of the reset will be confirmed by three blinks of the green LED
- Release the *Reset* button after 10 seconds
- Wait about 60 seconds until the LED module - green and red - blink alternately (*Emergency* mode).

After the procedure the module will be cleared, but the module will no longer be visible (no response to *Keep-Alive*) in the project from the Object Manager level. To restore the module again, perform CLU Discovery and then send the configuration.

## 7. Configuration parameters

**NOTE!** The described functionality and integration with these alarm control panels is available for **GRENTON GATE ALARM, DIN, Eth (INT-221-E-01)** with **firmware 1.1.0-2034C or higher!**

### A. GATE

#### FEATURES

Name	Description
Uptime	Operation time of the device since the last reset (in seconds)
ClientReportInterval	Reporting period about changes in features
Date	Current date
Time	Current time (hh: mm: ss)
Local Time	Current local time stamp
Time Zone	Time zone
UnixTime	Current Unix time stamp
FirmwareVersion	Gate software version
UseCloud	Specifies whether GATE connects to the cloud
CloudConnection	Specifies the status of the GATE connection to the cloud
NTPTimeout	Waiting time for response from NTP server
UseNTP	Specifies whether GATE uses NTP
PrimaryDNS	Preferred DNS server
SecondaryDNS	Alternate DNS server

#### METHODS

Name	Description
<code>SetDateTime</code>	Sets the date and time
<code>StartConsole</code>	Launches the Lua console
<code>StartConsoleOnReboot</code>	Starts the Lua console on restart
<code>SetClientReportInterval</code>	Sets the reporting period for feature changes
<code>SetPrimaryDNS</code>	Sets the PrimaryDNS feature
<code>SetSecondaryDNS</code>	Sets the SecondaryDNS feature

## EVENTS

Name	Description
<code>OnInit</code>	An event dispatched when the device initializes

## B. Satel

### FEATURES

Name	Description
<code>State</code>	Central status (0 - no connection to the control panel, 1 - connected to the control panel)
<code>LastError</code>	Last ETHM module error code (0 - ok, 1 - incorrect password)
<code>IP</code>	ETHM module IP address (Satel)
<code>Port</code>	ETHM module port (Satel)
<code>AdminPassword</code>	Satel administrator password
<code>UpdateTime</code>	Control panel status update period
<code>EncryptionEnabled</code>	Encryption status ( <i>true</i> - enabled, <i>false</i> - disabled)
<code>EncryptionKey</code>	Satel encryption key
<code>Value</code>	Returns the current status (1 - for armed zone, violated input, output enabled; 0 - for disarmed zone, normal input, output disabled; -1 - no information about the state due to no connection)
<code>Nr</code>	Parameter defining the zone, input or output to which the object refers
<code>UserPassword</code>	User password (will use the administrator password for "_")

### METHODS

Name	Description
SetIP	Sets the IP address of the ETHM module (Satel)
SetPort	Sets the ETHM module port (Satel)
SetAdminPassword	Sets an administrator password
SetEncryptionEnabled	Enables / disables encryption
SetEncryptionKey	Sets the Satel encryption key
ArmZone	Arms the zone
DisarmZone	Disarms the zone
SetNr	Sets a parameter that defines which zone, zone or output the object refers to
SetUserPassword	Sets the user password (will use the administrator password for _)
SwitchOn	Turns on the output
SwitchOff	Turns off the output

## EVENTS

Name	Description
OnConnected	Event triggered after establishing connection with the control panel
OnDisconnected	Event triggered when the connection with the control panel is lost
OnError	Event triggered after a control panel error ( <code>LastError</code> )
OnChange	Event dispatched when the status changes (regardless of the value)
OnSwitchOn	Event triggered when the output is switched on or the zone is violated
OnSwitchOff	Event triggered when the output is turned off or the input is set to normal
OnArm	An event dispatched when the zone is armed
OnDisarm	An event dispatched when the area was disarmed

## C. Jablotron

### FEATURES

Name	Description
AdminCode	Administrator access code
State	<p><i>JablotronSection:</i></p> <p>1 - READY - normal operation mode,  2 - ARMED_PART - partial arming of the section,  3 - ARMED - armed section,  4 - SERVICE - service mode included ,  5 - BLOCKED - section blocked,  6 - OFF - section switched off</p> <p><i>JablotronOutput:</i></p> <p>0 - Output switched on  1 - Output switched off</p>
Nr	Parameter defining which input / output / section the object refers to
AccessCode	The access code will use the admin code for '*'
Value	<p>Returns the current status:</p> <p>1 - for armed zone, violated input, output enabled;  0 - for disarmed zone, normal input, output disabled;  -1 - no status information due to lack of connection</p>

## METHODS

Name	Description
SetAccessCode	Sets the access code, uses 'admin' for '*'
Arm	Arms the zone / section
ArmPartially	Partially arm section (Jablotron)
Disarm	Disarms the zone / section
SetNr	Sets a parameter that defines which zone, zone or output the object refers to
SwitchOn	Turns on the output
SwitchOff	Turns off the output

## EVENTS



Name	Description
OnStateChange	Event dispatched when the status changes (regardless of the value)
OnArm	An event dispatched when the section is armed
OnDisarm	An event dispatched when the section is disarmed
OnChange	Event dispatched when the status changes (regardless of the value)
OnSwitchOn	Event triggered when the output is switched on or the zone is violated
OnSwitchOff	Event triggered when the output is turned off or the input is set to normal
OnArm	An event dispatched when the section is armed
OnDisarm	An event dispatched when the section is disarmed

## D. Timer

### FEATURES

Name	Description
Time	Counted time (in ms)
Mode	Timer mode: 0 - countdown, 1 - cyclical (interval)
State	Current timer status: 0 - stopped, 1 - counting

### METHODS

Name	Description
SetTime	Sets the timer time (in ms)
SetMode	Sets the operating mode: 0 - count down (countdown), 1 - cyclical (interval)
Start	Starts the timer
Stop	Stops the timer

### EVENTS

Name	Description
OnTimer	An event triggered when the timer is counted
OnStart	An event triggered when the timer is started
OnStop	An event triggered when the timer is stopped

# XIV. GATE MODBUS module

**NOTE!** The described functionality and integration is available for **GRENTON GATE MODBUS MASTER, DIN, Eth (INT-201-E-01)** with **firmware 1.1.0-2034C or higher!**

## 1. General information

The GATE Modbus module enables the integration of the Grenton system with all devices supporting the MODBUS RTU standard.

Before starting the configuration one must obtain information about the selected Slave device supporting the MODBUS RTU standard - it will be necessary to know, among other things: device address, type and address of registers, as well as transmission speed.

## 2. Module configuration

**NOTE!** Before starting any work with the GATE MODBUS module, it is necessary to update the interface database!

### Time setting via NTP server

The GATE Modbus module allows you to set the time using the NTP server, taking into account the time zone and changing the time (winter / summer). The time is taken automatically from the NTP server (*pool.ntp.org*).

There are three features for configuration:

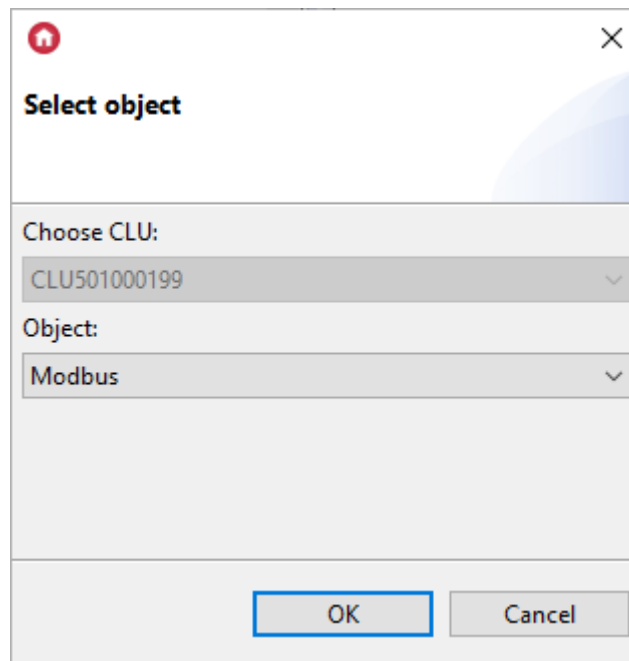
- `UseNTP` - determines whether GATE uses NTP,
- `NTPTimeout` - waiting time for a response from the NTP server,
- `TimeZone` - setting the GATE time zone - 22 zones are available.

**NOTE!** Getting the time from an NTP server requires that GATE be in a network that has an internet connection.

### Modbus virtual object configuration

To perform the correct configuration of the GATE Modbus module it is necessary to:

- Create a virtual object *Modbus* and name it:



- Go to *Embedded features* tab and enter:
  - **DeviceAddress** - address of the slave device;
  - **AccessRights** - operating mode (*Read* - reading the value from the register; *ReadWrite* - allows saving the value to the set register);
  - **RegisterAddress** - address of the supported registry;
  - **TransmissionSpeed** - transmission speed;
  - **RefreshInterval** - the period of polling the slave device register by GATE Modbus;
  - **ResponseTimeout** - Slave device time for response (if it is exceeded, `ErrorCode` = - 2 is returned);
  - **Divisor** - divisor (for `ValueType` = *Number / Float*);
  - parameters appropriate for the selected slave device type - [look up XIV.3.](#)
  - **StopBits** - sets the number of stop bits: 0 - 1 stop bit 1 - 1.5 stop bits 2 - 2 stop bits
  - **Parity** - sets the parity check: 0 - None 1 - Odd 2 - Even



**Object properties**

Name:  Type:

Id:

Control
  Events
  Embedded features

Feature name	Current value	Initial value	Unit	Range
DeviceAddress	111	<input type="text" value="111"/>	number	[0-255]
AccessRights	0	<input type="text" value="Read"/>	-	0,1
RegisterAddress	141	<input type="text" value="141"/>	number	[0-65535]
TransmissionSpeed	9600	<input type="text" value="9600"/>	bps	1200,2400,4800,9600,19200,38400,57600,115200
ValueType	1	<input type="text" value="Number"/>		1,2,3
BitPosition	0	<input type="text" value="0"/>	number	[0-15]
BitCount	16	<input type="text" value="16"/>	number	[1-32]
RefreshInterval	1000	<input type="text" value="1000"/>	number	[0-65535]
ResponseTimeout	100	<input type="text" value="100"/>	number	[10-65535]
Divisor	1	<input type="text" value="1"/>	number	[1-65535]
Endianess	3	<input type="text" value="SwapWords"/>	-	0,1,2,3
RegisterType	2	<input type="text" value="HoldingRegisters"/>	-	0,1,2,3
ErrorCode	0		number	
Value	639	<input type="text" value="0"/>	number	
RegisterValue	639		number	
StopBits	0	<input type="text" value="1"/>	-	0,1,2
Parity	0	<input type="text" value="None"/>	-	0,1,2

Auto refresh 
 Refresh

- Send configuration and verify connection - tab *Embedded features*, feature  = 0 (correct read / write):

Object properties
✕

Name:  Type:   
Id:

Control Events Embedded features

Feature name	Current value	Initial value	Unit	Range
DeviceAddress	111	<input type="text" value="111"/>	number	[0-255]
AccessRights	0	<input type="text" value="Read"/>	-	0,1
RegisterAddress	141	<input type="text" value="141"/>	number	[0-65535]
TransmissionSpeed	9600	<input type="text" value="9600"/>	bps	1200,2400,4800,9600,19200,38400,57600,115200
ValueType	1	<input type="text" value="Number"/>		1,2,3
BitPosition	0	<input type="text" value="0"/>	number	[0-15]
BitCount	16	<input type="text" value="16"/>	number	[1-32]
RefreshInterval	1000	<input type="text" value="1000"/>	number	[0-65535]
ResponseTimeout	100	<input type="text" value="100"/>	number	[10-65535]
Divisor	1	<input type="text" value="1"/>	number	[1-65535]
Endianness	3	<input type="text" value="SwapWords"/>	-	0,1,2,3
RegisterType	2	<input type="text" value="HoldingRegisters"/>	-	0,1,2,3
ErrorCode	0		number	
Value	639	<input type="text" value="0"/>	number	
RegisterValue	639		number	
StopBits	0	<input type="text" value="1"/>	-	0,1,2
Parity	0	<input type="text" value="None"/>	-	0,1,2

Auto refresh

### 3. Parameters of registers

Depending on the type of Slave register, the next available parameters must be set accordingly.

#### A. 16-bit registers

Reading 16-bit holding registers (  , *FunctionCode = 03*);

- : *Read*;
- : *Number*;
- : default value;
- : **16**;
- : default value;
- : *HoldingRegisters*;

Object properties
✕

Name: 
Type:

Id:

Control
 Events
 Embedded features

Feature name	Current value	Initial value	Unit	Range
DeviceAddress	1	<input type="text" value="1"/>	number	[0-255]
AccessRights	0	<input type="text" value="Read"/>	-	0,1
RegisterAddress	1	<input type="text" value="1"/>	number	[0-65535]
TransmissionSpeed	115200	<input type="text" value="115200"/>	bps	1200,2400,4800,9600,19200,38400,57600,115200
ValueType	1	<input type="text" value="Number"/>		1,2,3
BitPosition	0	<input type="text" value="0"/>	number	[0-15]
BitCount	16	<input type="text" value="16"/>	number	[1-32]
RefreshInterval	5000	<input type="text" value="5000"/>	number	[0-65535]
ResponseTimeout	1000	<input type="text" value="1000"/>	number	[10-65535]
Divisor	1	<input type="text" value="1"/>	number	[1-65535]
Endianess	0	<input type="text" value="NoSwap"/>	-	0,1,2,3
RegisterType	2	<input type="text" value="HoldingRegisters"/>	-	0,1,2,3
ErrorCode	0		number	
Value	55555	<input type="text" value="0"/>	number	
RegisterValue	55555		number	
StopBits	0	<input type="text" value="1"/>	-	0,1,2
Parity	0	<input type="text" value="None"/>	-	0,1,2

Auto refresh

Reading 16-bit input registers ( , *FunctionCode = 04*):

- : *Read*;
- : *Number*;
- : default value;
- : **16**;
- : default value;
- : *InputRegisters*.

Records of 16-bit holding registers ( , *FunctionCode = 06*):

- : *ReadWrite*;
- : *Number*;
- : default value;
- : **16**;
- : default value;
- : *HoldingRegisters*.

## B. Fields in 16-bit registers

Reading of bit fields in a 16-bit remembering register ( `Read Holding Registers`, `FunctionCode = 03`):

- `AccessRights` : *Read*;
- `ValueType` : *Bit*;
- `BitPosition` : 0-15 (position of the first interesting bit);
- `BitCount` : 1-16 (number of bits read sequentially);
- `Endianness` : default value;
- `RegisterType` : *HoldingRegisters*;

**Object properties**

Name:  Type:

Id:

Control  Events  Embedded features

Feature name	Current value	Initial value	Unit	Range
DeviceAddress	2	<input type="text" value="2"/>	number	[0-255]
AccessRights	0	<input type="text" value="Read"/>	-	0,1
RegisterAddress	1	<input type="text" value="1"/>	number	[0-65535]
TransmissionSpeed	115200	<input type="text" value="115200"/>	bps	1200,2400,4800,9600,19200,38400,57600,115200
ValueType	3	<input type="text" value="Bit"/>	-	1,2,3
BitPosition	2	<input type="text" value="2"/>	number	[0-15]
BitCount	1	<input type="text" value="1"/>	number	[1-32]
RefreshInterval	10000	<input type="text" value="10000"/>	number	[0-65535]
ResponseTimeout	1000	<input type="text" value="1000"/>	number	[10-65535]
Divisor	1	<input type="text" value="1"/>	number	[1-65535]
Endianness	0	<input type="text" value="NoSwap"/>	-	0,1,2,3
RegisterType	2	<input type="text" value="HoldingRegisters"/>	-	0,1,2,3
ErrorCode	0		number	
Value	1	<input type="text" value="0"/>	number	
RegisterValue	4		number	
StopBits	0	<input type="text" value="1"/>	-	0,1,2
Parity	0	<input type="text" value="None"/>	-	0,1,2

Auto refresh

Reading of bit fields in a 16-bit input register ( `Read Input Registers`, `FunctionCode = 04`):

- `AccessRights` : *Read*;
- `ValueType` : *Bit*;
- `BitPosition` : 0-15 (position of the first interesting bit);
- `BitCount` : 1-16 (number of bits read sequentially);
- `Endianness` : default value;

- `RegisterType` : *InputRegisters*.

Writing bit fields in a 16-bit reminder register ( `Preset / Write Single Holding Register`, `FunctionCode = 06`):

- `AccessRights` : *ReadWrite*;
- `ValueType` : *Bit*;
- `BitPosition` : 0-15 (position of the first interesting bit);
- `BitCount` : 1-16 (number of bits read sequentially);
- `Endianness` : default value;
- `RegisterType` : *HoldingRegisters*.

### C. 32-bit integer values of registers

Reading 32-bit integer values of the retaining register ( `Read Holding Registers`, `FunctionCode = 03`):

- `AccessRights` : read;
- `ValueType` : number;
- `BitPosition` : default value;
- `BitCount` : 32;
- `Endianness` : in the case of 32-bit registers, Slaves usually require reordering of bytes and words - *SwapBytesAndWords*; detailed information should be found in the slave device card;
- `RegisterType` : *HoldingRegisters*:



Object properties
✕

Name:  Type:   
Id:

Control
 Events
 Embedded features

Feature name	Current value	Initial value	Unit	Range
DeviceAddress	3	<input type="text" value="3"/>	number	[0-255]
AccessRights	0	<input type="text" value="Read"/>	-	0,1
RegisterAddress	1	<input type="text" value="1"/>	number	[0-65535]
TransmissionSpeed	115200	<input type="text" value="115200"/>	bps	1200,2400,4800,9600,19200,38400,57600,115200
ValueType	1	<input type="text" value="Number"/>		1,2,3
BitPosition	0	<input type="text" value="0"/>	number	[0-15]
BitCount	32	<input type="text" value="32"/>	number	[1-32]
RefreshInterval	10000	<input type="text" value="10000"/>	number	[0-65535]
ResponseTimeout	1000	<input type="text" value="1000"/>	number	[10-65535]
Divisor	1	<input type="text" value="1"/>	number	[1-65535]
Endianness	1	<input type="text" value="SwapBytesAndWords"/>	-	0,1,2,3
RegisterType	2	<input type="text" value="HoldingRegisters"/>	-	0,1,2,3
ErrorCode	0		number	
Value	200000000	<input type="text" value="0"/>	number	
RegisterValue	200000000		number	
StopBits	0	<input type="text" value="1"/>	-	0,1,2
Parity	0	<input type="text" value="None"/>	-	0,1,2

Auto refresh

Reading 32-bit integer values of the input register (  , *FunctionCode = 04*):

- : *Read*;
- : *Number*;
- : default value;
- : 32;
- : in the case of 32-bit registers, Slaves usually require reordering of bytes and words - *SwapBytesAndWords*; detailed information should be found in the slave device card;
- : *InputRegisters*.

Writing of 32-bit integers in the remembering register (  , *FunctionCode = 16*):

- : *ReadWrite*;
- : *Number*;
- : default value;
- : 32;

- `Endianness`: in the case of 32-bit registers, Slaves usually require reordering of bytes and words  
- *SwapBytesAndWords*; detailed information should be found in the slave device card;
- `RegisterType`: *HoldingRegisters*.

## D. 32-bit floating point values of registers

Reading of the 32-bit floating point values of the remembering register (`Read Holding Registers`, *FunctionCode = 03*):

- `AccessRights`: *Read*;
- `ValueType`: *Float*;
- `BitPosition`: default value;
- `BitCount`: *32*;
- `Endianness`: in the case of 32-bit registers, Slaves usually require reordering of bytes and words  
- *SwapBytesAndWords*; detailed information should be found in the slave device card;
- `RegisterType`: *HoldingRegisters*:

**Object properties**

Name:  Type:

Id:

Control  Events  Embedded features

Feature name	Current value	Initial value	Unit	Range
DeviceAddress	4	<input type="text" value="4"/>	number	[0-255]
AccessRights	0	<input type="text" value="Read"/>	-	0,1
RegisterAddress	1	<input type="text" value="1"/>	number	[0-65535]
TransmissionSpeed	115200	<input type="text" value="115200"/>	bps	1200,2400,4800,9600,19200,38400,57600,115200
ValueType	2	<input type="text" value="Float"/>	-	1,2,3
BitPosition	0	<input type="text" value="0"/>	number	[0-15]
BitCount	32	<input type="text" value="32"/>	number	[1-32]
RefreshInterval	10000	<input type="text" value="10000"/>	number	[0-65535]
ResponseTimeout	1000	<input type="text" value="1000"/>	number	[10-65535]
Divisor	1	<input type="text" value="1"/>	number	[1-65535]
Endianness	1	<input type="text" value="SwapBytesAndWords"/>	-	0,1,2,3
RegisterType	2	<input type="text" value="HoldingRegisters"/>	-	0,1,2,3
ErrorCode	0		number	
Value	100.00	<input type="text" value="0"/>	number	
RegisterValue	1120403453		number	
StopBits	0	<input type="text" value="1"/>	-	0,1,2
Parity	0	<input type="text" value="None"/>	-	0,1,2

Auto refresh

Reading of 32-bit floating point register values (`Read Input Registers`, *FunctionCode = 04*):

- `AccessRights` : *Read*;
- `ValueType` : *Float*;
- `BitPosition` : default value;
- `BitCount` : *32*;
- `Endianness` : in the case of 32-bit registers, slaves usually require reordering of bytes and words  
- *SwapBytesAndWords*; detailed information should be found in the slave device card;
- `RegisterType` : *InputRegisters*.

Record of 32-bit floating point values in the reminder register ( `Preset / Write Multiple Holding Registers` , *FunctionCode = 16*);

- `AccessRights` : *ReadWrite*;
- `ValueType` : *Float*;
- `BitPosition` : default value;
- `BitCount` : *32*;
- `Endianness` : in the case of 32-bit registers, slaves usually require reordering of bytes and words  
- *SwapBytesAndWords*; detailed information should be found in the slave device card;
- `RegisterType` : *HoldingRegisters*.

## E. Discrete inputs / outputs

Readout of discrete outputs / bit inputs ( `Read Coil Status` , *FunctionCode = 01*);

- `AccessRights` : *Read*;
- `ValueType` : *Number*;
- `BitPosition` : default value;
- `BitCount` : *1-32*;
- `Endianness` : default value;
- `RegisterType` : *BinaryInputs*;

**Object properties**

Name:  Type:

Id:

Control  Events  Embedded features

Feature name	Current value	Initial value	Unit	Range
DeviceAddress	5	<input type="text" value="5"/>	number	[0-255]
AccessRights	0	<input type="text" value="Read"/>	-	0,1
RegisterAddress	1	<input type="text" value="1"/>	number	[0-65535]
TransmissionSpeed	115200	<input type="text" value="115200"/>	bps	1200,2400,4800,9600,19200,38400,57600,115200
ValueType	1	<input type="text" value="Number"/>		1,2,3
BitPosition	0	<input type="text" value="0"/>	number	[0-15]
BitCount	8	<input type="text" value="8"/>	number	[1-32]
RefreshInterval	10000	<input type="text" value="10000"/>	number	[0-65535]
ResponseTimeout	1000	<input type="text" value="1000"/>	number	[10-65535]
Divisor	1	<input type="text" value="1"/>	number	[1-65535]
Endianness	0	<input type="text" value="NoSwap"/>	-	0,1,2,3
RegisterType	0	<input type="text" value="BitOutputsInputs"/>	-	0,1,2,3
ErrorCode	0		number	
Value	30	<input type="text" value="0"/>	number	
RegisterValue	30		number	
StopBits	0	<input type="text" value="1"/>	-	0,1,2
Parity	0	<input type="text" value="None"/>	-	0,1,2

Auto refresh

Readout of discrete binary inputs (  , *FunctionCode = 02*):

- : *Read*;
- : *Number*;
- : default value;
- : *1-32*;
- : default value;
- : *BinaryInputs*.

Writing of discrete outputs / bit inputs (  , *FunctionCode = 05*;  , *FunctionCode = 15*):

- : *ReadWrite*;
- : *Number*;
- : default value;
- : *1-32*;
- : default value;
- : *BitOutputsInputs*.

## 4. Restoring factory settings - *Hard Reset*

**NOTE!** The steps of the procedure apply to the GATE Modbus module fw. **18.21.4.1453** or lower.

Running the *Hard Reset* function on the GATE Modbus module causes:

- Removal of the saved configuration;
- Formatting the flash memory partition;
- Removal of all created LUA objects;
- Loss of communication between OM / HM and Gate module.

In order to restore the factory settings with the *Hard Reset* function, perform the following steps (in accordance with the given order):

- Disconnect power from the Gate module;
- Press and hold the *Reset* button on the module (the button is located under the bottom end of the module);
- Connect the power supply to the Gate module;
- Keep the *Reset* button pressed for at least 10 seconds - during the reset the green LED will be on steady. Correct completion of the reset will be confirmed by three blinks of the green LED.
- Release the *Reset* button after 10 seconds
- Wait about 60 seconds until the LED module - green and red - blink alternately (*Emergency mode*)

After the procedure the module will be cleared, but the module will no longer be visible (no response to *Keep-Alive*) in the project from the Object Manager level. To restore the module again, perform CLU Discovery and then send the configuration.

## 5. Configuration parameters

**NOTE!** The described functionality and integration is available for **GRENTON GATE MODBUS MASTER, DIN, Eth (INT-201-E-01)** with **firmware 1.1.0-2034C** or higher!

### A. GATE

#### FEATURES

Name	Description
<code>Uptime</code>	Operation time of the device since the last reset (in seconds)
<code>FirmwareVersion</code>	Gate software version
<code>PrimaryDNS</code>	Preferred DNS server
<code>SecondaryDNS</code>	Alternate DNS server
<code>ClientReportInterval</code>	Reporting period about changes in features
<code>Date</code>	Current date
<code>Time</code>	Current time (hh: mm: ss)
<code>Local Time</code>	Current local time stamp
<code>Time Zone</code>	Time zone
<code>UnixTime</code>	Current Unix time stamp
<code>UseCloud</code>	Specifies whether GATE connects to the cloud
<code>CloudConnection</code>	Specifies the status of the GATE connection to the cloud
<code>NTPTimeout</code>	Waiting time for response from NTP server
<code>UseNTP</code>	Specifies whether GATE uses NTP

## METHODS

Name	Description
<code>StartConsole</code>	Launches the Lua console
<code>StartConsoleOnReboot</code>	Starts the Lua console on restart
<code>SetClientReportInterval</code>	Sets the reporting period for feature changes
<code>SetPrimaryDNS</code>	Sets the PrimaryDNS feature
<code>SetSecondaryDNS</code>	Sets the SecondaryDNS feature
<code>SetDateTime</code>	Sets the date and time

## EVENTS

Name	Description
<code>OnInit</code>	An event dispatched when the device initializes

## **B. Modbus**

### **FEATURES**

Name	Description
DeviceAddress	Address of the Slave Modbus device
AccessRights	Operating mode: <i>read</i> (0 - reading); <i>read / write</i> (1 - read / write)
RegisterAddress	Address of the supported registry
TransmissionSpeed	Transmission speed
ValueType	Variable type (1 - <i>number</i> ; 2 - <i>float</i> ; 3 - <i>bit</i> )
BitPosition	Bit position (for bit access to 16-bit registers)
BitCount	The number of registry bits to read
RefreshInterval	Refresh time
ResponseTimeout	Response time
Divisor	Divisor
Endianness	The order of bytes and words[^XIV.3_1]: <i>No swap</i> (0 - no exchange); <i>Swap bytes and words</i> (1 - change the order of bytes and words); <i>Swap bytes</i> (2 - changing the order of bytes within each word); <i>Swap words</i> (3 - exchange of words)
RegisterType	Modbus register type (0 - bit inputs / outputs, 1 - binary inputs, 2 - holding registers, 3 - input registers)
ErrorCode	Error code: (- <b>3</b> - frame error; - <b>2</b> - exceeding the response time; - <b>1</b> - out of date value of the last read out register; <b>0</b> - correct reading / writing of the register; <b>1</b> - not allowed function; <b>2</b> - not allowed register number; <b>3</b> - unauthorized data value; <b>4</b> - damage to the connected device; <b>5</b> - positive confirmation; <b>6</b> - no readiness / message removed; <b>7</b> - negative confirmation; <b>8</b> - memory parity error)
Value	Read / write value
RegisterValue	Unscaled register value
StopBits	Stop bits: 0 - 1 stop bit 1 - 1.5 stop bits 2 - 2 stop bits
Parity	Parity bit: 0 - None 1 - Odd 2 - Even
[^XIV.3_1]: Within the meaning of the instructions, the word consists of two bytes.	

## METHODS



Name	Description
<code>SetDeviceAddress</code>	Sets the address of the Slave Modbus device
<code>SetAccessRights</code>	Sets the operating mode: reading or reading / writing
<code>SetRegisterAddress</code>	Sets the address of the supported registry
<code>SetTransmissionSpeed</code>	Sets the transmission speed
<code>SetValueType</code>	Sets the type of variable
<code>SetBitPosition</code>	Sets the position of the bit
<code>SetBitCount</code>	Sets the number of registry bits to read
<code>SetRefreshInterval</code>	Sets the refresh time
<code>SetReadWriteTimeout</code>	Sets the response timeout
<code>SetDivisor</code>	Sets the divisor
<code>SetEndianness</code>	Sets the byte order type
<code>SetRegisterType</code>	Sets the Modbus register type
<code>SetValue</code>	Sets the read / write value
<code>SetStopBits</code>	Sets the number of stop bits
<code>SetParity</code>	Sets the parity check

## EVENTS

Name	Description
<code>OnChange</code>	An event dispatched when the state changes (regardless of the value)
<code>OnError</code>	An event dispatched when the slave device reports an error

# XV. GATE HTTP Module

**NOTE!** The described functionality and integration is available for **GRENTON GATE HTTP, DIN, Eth (INT-211-E-01)** with **firmware 1.1.0-2034C or higher!**

## 1. General information

The GATE Http module is a device enabling system integration with external sites using the HTTP protocol, as well as a wide group of devices and external / third-party systems - eg AV devices with HTTP interfaces.

## 2. Module configuration

**NOTE!** Before starting any work with the GATE HTTP module, the interface database update is required!

### Time setting via NTP server

The GATE Modbus module allows you to set the time using the NTP server, taking into account the time zone and changing the time (winter / summer). The time is taken automatically from the NTP server (*pool.ntp.org*).

There are three features for configuration:

- `UseNTP` - determines whether GATE uses NTP,
- `NTPTimeout` - waiting time for a response from the NTP server,
- `TimeZone` - setting the GATE time zone - 22 zones are available.

**NOTE!** Getting the time from an NTP server requires that GATE be in a network that has an internet connection.

## 2.1. Virtual objects

### 2.1.1. HTTP Request

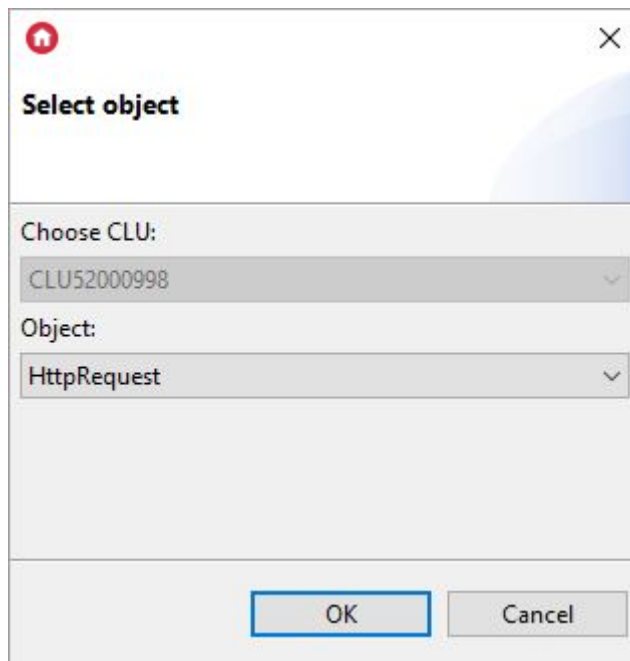
For the HttpRequest, for example, the weather service <http://api.openweathermap.org> is used

According to the example on the openweathermap.org site, the API query looks like this:

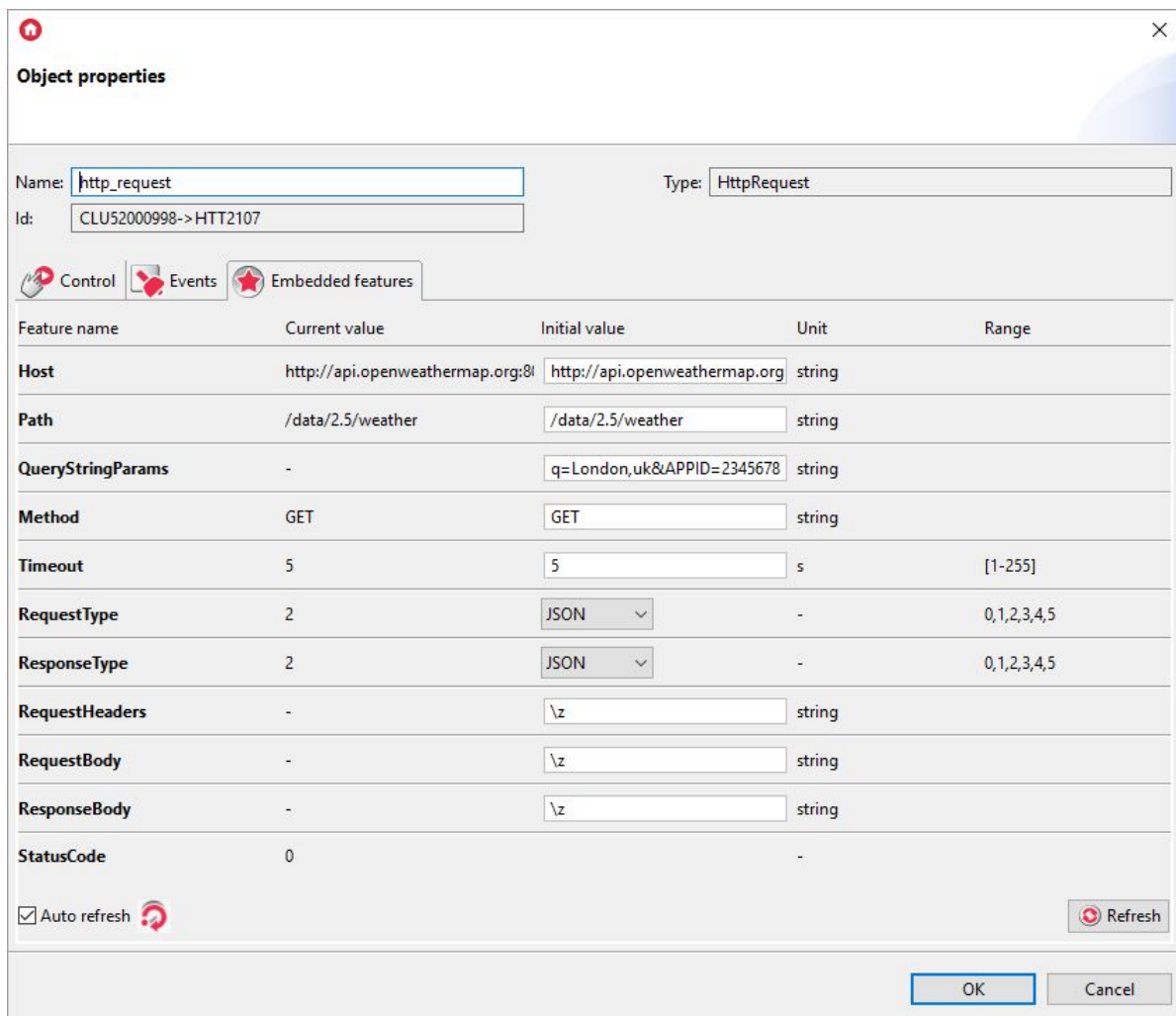
API call: <http://api.openweathermap.org/data/2.5/weather?q=London&APPID={APIKEY}>

HttpRequest - is used to send HTTP (GET, POST) requests to a specific host. Standard content types are supported, e.g. JSON, XML.

To use the Gate module to receive queries, create an HttpRequest virtual object



- The following parameters must be set in the HttpRequest object:



- **Host:** api.openweathermap.org
- **Path:** /data/2.5/weather
- **QueryStringParams:** q=London&APPID={APIKEY}
- **Method:** GET
- **RequestType:** JSON

- **ResponseType:** JSON

**NOTE!** The Gate Http object enables TLS encrypted connections. If such a connection is required, the 'https: //' field should be entered in the Host field at the beginning of the value. If the value is not specified, the standard http connection will be used.

**NOTE!** Gate Http does not support all TLS encrypted connections, so we recommend testing the connection with the given host.

**NOTE!** During the https connection, the time to establish a connection and receive a response from the host is longer than for the http connection, therefore the value for the Timeout parameter should be increased.

**NOTE!** Features described as not settable are features containing answers. The initial values of these features should be left unchanged. All operations on these variables should be performed on scripts (and local variables).

After sending the configuration and calling the SendRequest Method, the StatusCode takes the value 200 (OK).

The received response to the query is kept in ResponseBody. For the JSON ResponseType set, the response is parsed from json to the table. The feature value is invisible from the OM level. The response values should be drawn from the response from the script.

## 2.1.2. Downloading certain values from the received response (XML, JSON)

**NOTE!** The response Response obtained should be assigned to the local variable (in the script).

For example:

```
local resp = GATE-> http_openweather_json-> ResponseBody
```

Then, in the scripts, you must perform the operation on the variable resp!

**NOTE!** Scripts reading the content stored in the ResponseBody must be created in the GATE HTTP module.

The received responses depending on their type (ResponseType) are properly parsed to the table.

Exemplary value readings are written to local variables (inside the script).

In order to be able to use a variable (eg to display in an application), it should be assigned to global variables (user's features).

Below are examples of answers in XML and JSON format as well as the method of reading a given value (in the presented examples the answers from the openweathermap.org weather service were used)

## A. JSON:

Example answer (openweathermap.org):

```
resp = [[
  {"coord":
    {"lon":145.77,"lat":-16.92},
    "weather":[{"id":803,"main":"Clouds","description":"broken
clouds","icon":"04n"}],
    "base":"cmc stations",
    "main":
    {"temp":293.25,"pressure":1019,"humidity":83,"temp_min":289.82,"temp_max":295.3
7},
    "wind":{"speed":5.1,"deg":150},
    "clouds":{"all":75},
    "rain":{"3h":3},
    "dt":1435658272,
    "sys":
    {"type":1,"id":8166,"message":0.0166,"country":"AU","sunrise":1435610796,"sunse
t":1435650870},
    "id":2172797,
    "name":"Cairns",
    "cod":200}
  ]]
```

How to read :

- Parameter value **lon**

```
{"coord":
{"lon":145.77,"lat":-16.92},
"weather":[{"id":803,"main":"Clouds","description":"broken
clouds","icon":"04n"}],
"base":"cmc stations",
"main":
{"temp":293.25,"pressure":1019,"humidity":83,"temp_min":289.82,"temp_max":295.3
7},
```

In a script:

```
local lon = resp.coord.lon
```

After calling the script 145.77 will be assigned to the local variable (script variable).

- Parameter value **description**

```
{ "coord":
  { "lon":145.77, "lat":-16.92},
  "weather":[{"id":803, "main":"Clouds", "description":"broken
clouds", "icon":"04n"}],
  "base":"cmc stations",
  "main":
  { "temp":293.25, "pressure":1019, "humidity":83, "temp_min":289.82, "temp_max":295.3
7},
```

In a script:

```
local description = resp.weather[1].description
```

After calling the script *"broken clouds"* will be assigned to the local variable (script variable).

## B. XML:

Example answer (openweathermap):

```
resp= [[
<current>
  <city id="2643741" name="City of London">
    <coord lon="-0.09" lat="51.51">
    <country>GB</country>
    <sun rise="2015-06-30T03:46:57" set="2015-06-30T20:21:12">
  </city>
  <temperature value="72.34" min="66.2" max="79.88" unit="fahrenheit"/>
  <humidity value="43" unit="%">
  <pressure value="1020" unit="hPa">
  <wind>
    <speed value="7.78" name="Moderate breeze">
    <direction value="140" code="SE" name="SouthEast">
  </wind>
  <clouds value="0" name="clear sky">
  <visibility value="10000">
  <precipitation mode="no">
  <weather number="800" value="Sky is Clear" icon="01d">
  <lastupdate value="2015-06-30T08:36:14">
</current>
]]
```

How to read:

- The value of the id attribute in the tag **city**

```
<current>
  <city id="2643741" name="City of London">
    <coord lon="-0.09" lat="51.51">
    <country>GB</country>
  <sun rise="2015-06-30T03:46:57" set="2015-06-30T20:21:12">
</city>
```

In a script:

```
local city_id = resp[1].id
```

After calling the script, *2643741* will be assigned to the local variable (script variable).

- The value between the **country** tag:

```
<current>
  <city id="2643741" name="City of London">
    <coord lon="-0.09" lat="51.51">
      <country>GB</country>
      <sun rise="2015-06-30T03:46:57" set="2015-06-30T20:21:12">
    </city>
```

In a script:

```
local country = resp[1][2][1]
```

After calling the script, *"GB"* will be assigned to the local variable (script variable).

- Tag name **country**

```
<current>
  <city id="2643741" name="City of London">
    <coord lon="-0.09" lat="51.51">
      <country>GB</country>
      <sun rise="2015-06-30T03:46:57" set="2015-06-30T20:21:12">
    </city>
```

In a script:

```
local nameTag = resp[1][2].xmlTag
```

After calling the script, the value of *"country"* will be assigned to the local variable (script variable).

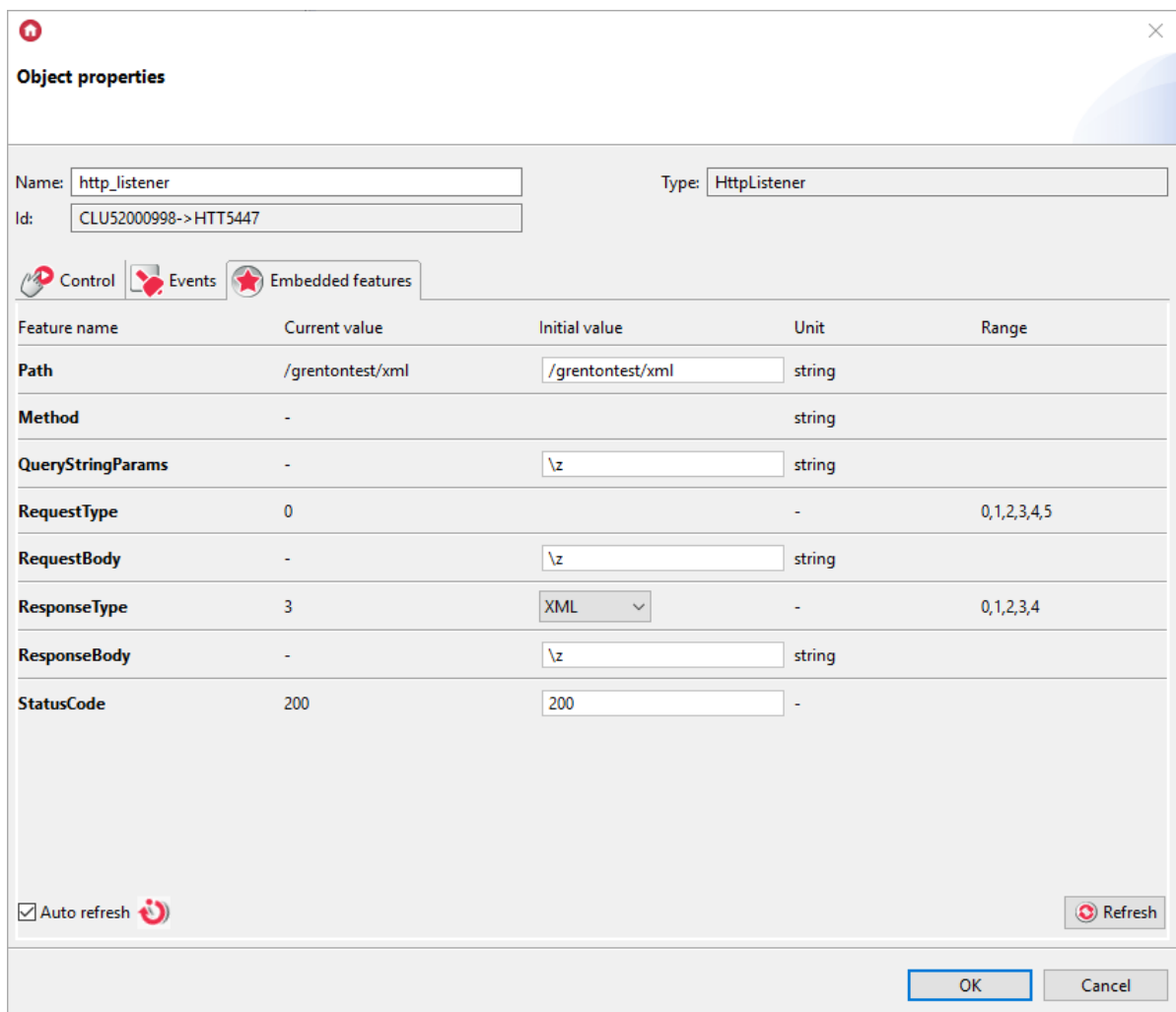
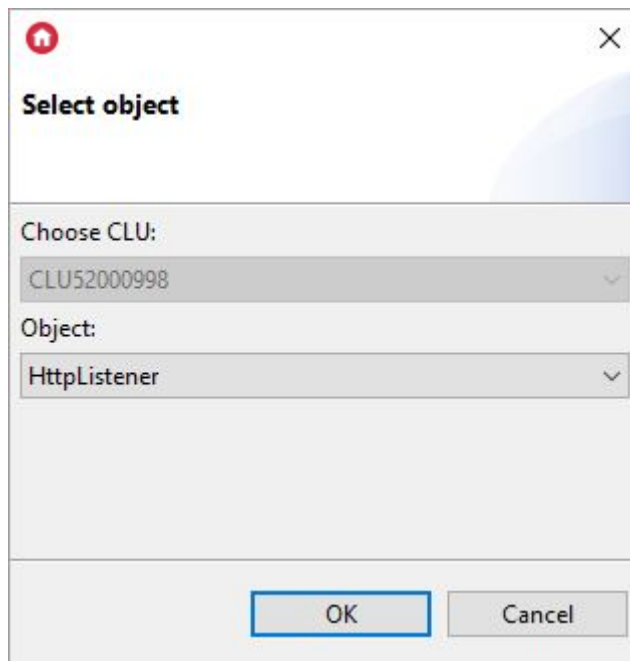
## 2.2.1. HttpListener

The HttpListener object is used for receiving HTTP (GET, POST) requests. The returned response can be serialized to one of the standard types including JSON, XML. In the HttpListener object, it is important to return the response to every incoming Request.

In the case of listening to Request from the Gate module on the query - for example (using an internet browser):

***GET 192.168.4.12/grentontest/xml***

You must create the HttpListener virtual object.



The following parameters must be set in the HttpListener object:

- **Path:** /grentontest/xml
- **ResponseType:** XML
- **StatusCode:** 200



**NOTE!** Features described as not settable are features containing answers. The initial values of these features should be left unchanged. All operations on these variables should be performed on scripts (and local variables)

You must create a script for the OnRequest event that will create the correct answer and send it back.

## 2.2.2. Preparation of the response sent to the server

The response is created in the local resp variable.

After preparing the response, set it for ResponseBody (resp) and then send it using the SendResponse () method.

### A. XML:

To send the value of a given attribute in response:

```
local resp = "<clu><temperature>" .. CLUZ->x103478262_ONEW_SENSOR1->Value .. "  
</temperature></clu>"  
GATE_2->Listener_XML->SetResponseBody (resp)  
GATE_2->Listener_XML->SendResponse ()
```

The answer you've provided is as follows:

```
<clu>  
  <temperature>22.5</temperature>  
</clu>
```

### B. JSON:

```
local resp = {  
  Temp = CLUZ->x103478262_ONEW_SENSOR1->Value  
}  
GATE_2->Listener_JSON->SetResponseBody (resp)  
GATE_2-> Listener_JSON->SendResponse ()
```

The answer you've provided is as follows:

```
{"Temp": 22.6}
```

## 2.2.3. Reading key values from the querystringparams parameter

According to the description of the QueryStringParams feature, its value is not settable, it can be read in the script. If querystring with keys (keys) is sent in the query, the given value can be read from the script level - it is saved in the form of a table.

Individual key values can be obtained on the basis of:

```
value1 = qs.klucz1
```

For the query received:

**192.168.1.12/grentontest/query?light1=on&light2=off&light3=on**

You must create a script:

```
local qs = HTTP_L->grentontest_query_listener->QueryStringParams

local test0 = qs.light1
local test1 = qs.light2
local test2 = qs.light3

HTTP_L->grentontest_query_listener->SetResponseBody()
HTTP_L->grentontest_query_listener->SendResponse()
```

All key values will be saved in local variables (test0, test1, test2).

## 2.3.1. Timer

Timers are virtual objects created as part of a given GATE module. Timers can be used wherever it is necessary to call a method after a specified time or also to call it cyclically.

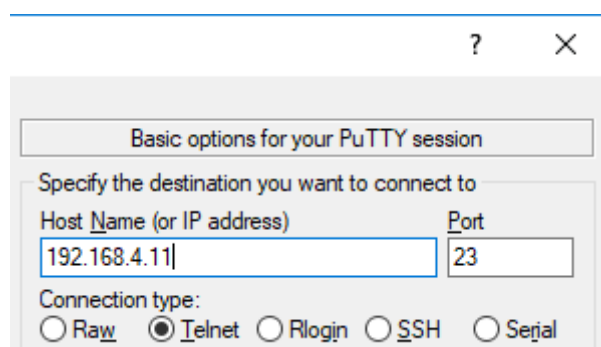
**NOTE!** It is recommended to use the Timer object when sending queries periodically using the HttpRequest object.

The timer can operate in two modes:

- **Countdown** After starting, it counts down the set time. At the end of the countdown, the method associated with the **OnTimer** event is started, and the timer stops and does not count down until the next start using the **Start** method.
- **Interval** Cyclic timer - after the start, it starts counting down the set time. After its expiry, the timer calls the method associated with the **OnTimer** event, and the timer itself again begins to count down the set time. The situation is repeated until it is stopped by the 'Stop' method.

## 3. The ability to connect to the Gate using TELNET

For the Gate Http module it is possible to view Lua scripts. In case of configuration error (emergency mode), it is possible to view the error location in the LUA configuration created. The connection is established using the Telnet protocol - for this purpose, for example, the PuTTY program can be used. Examples of parameters to establish a connection:



Two methods can be used to call a connection on the Gate side:

- `StartConsole` - Launches the Lua console. When the method is called, the user has 10s to set the connection to Gate. If the connection is correct, the information about the correct connection will be returned on the terminal (client):

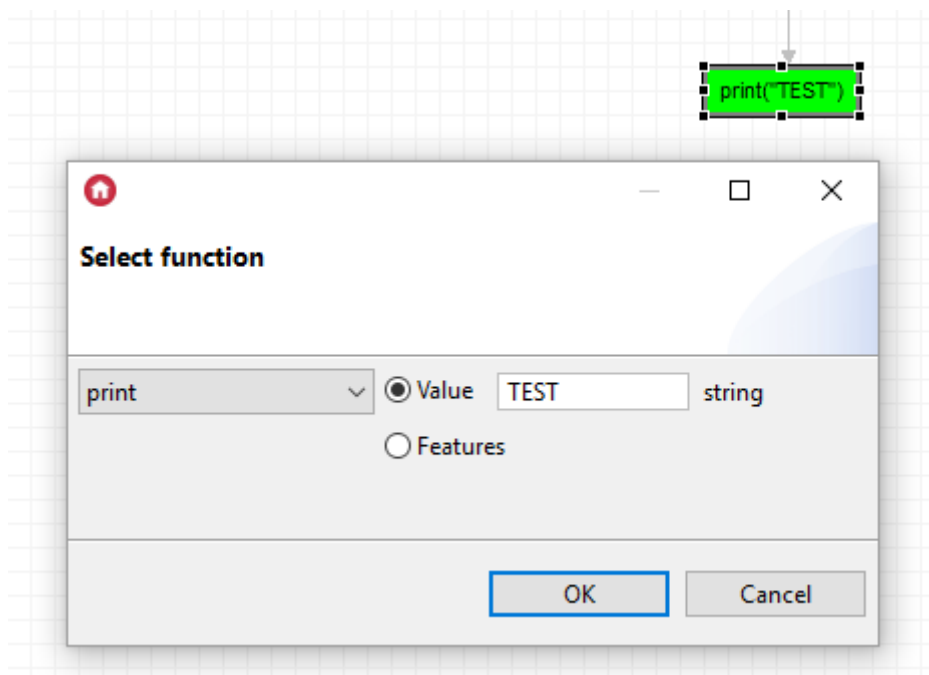
```
CLU SN Telnet session started.
```

- `StartConsoleOnReboot` - allows you to establish a connection the next time Gate reboots. After reboot, the user has 10s to set the connection to Gate. If the connection is correct, the information about the correct connection will be returned on the terminal (client)

```
CLU SN initializing...
CLU: running user.lua...
CLU: running om.lua...
CLU: running OnInit...
CLU: Project loaded.
```

**NOTE!** It is not recommended to assign the `StartConsole` and `StartConsoleOnReboot` methods to the `OnInit` event of the GATE Http module.

To display eg the value of a given feature on the console, use the *Function block* component and select the *Print* method, and then select the desired feature.



**NOTE!** If a Telnet connection between the client <--> GATE has been established, calling the `StartConsole` method again will disconnect the current session.

## 4. Comprehensive integration with external systems using the GATE Http device

Step-by-step configuration description on the example of a relay output.

## 4.1. System

Let's say we have a simple system consisting of the following elements:

- CLU Z-Wave - named (Name) "CluZ"
- Relay module - for the purpose of using one output called "Relay"
- Gate Http - name "GateHttp"

## 4.2. Output control

In order to enable control of the relay output from an external system, we create a new `HttpListener` object on `GateHttp` and configure it as follows:

- **Name:** `RelayControlListener`
- **Path:** `/relaycontrol`

We leave the other parameters unchanged for now.

### Script

For the `RelayControlListener` object to work, create a script that will handle the incoming Http queries.

Here it is worth noting that from this script we have access to the entire system and all its functionalities. This opens up virtually unlimited possibilities but also raises some risks, especially if Gate's functionality is not well thought out. Therefore, we pay special attention that when implementing Gate's functionality, we should think carefully about the way we want to achieve and how Gate's operation may depend on or affect other elements of the system. Examples of this approach will also be discussed further.

Returning to the script controlling Relay. We want to be able to switch Relay on or off by sending him the expected status (On / Off) or calling the `Switch` method. This approach to implementation makes it possible to connect both bistable and monostable switch type control to it.

Going to action, we create a script on `GateHttp` called `RelayControlOnRequest`, and in the code editing mode we put the following:

```
-- RelayControlOnRequest()
local data = GateHttp->RelayControlListener->QueryStringParams
if data == nil then
    CluZ->Relay->Switch(0)
else
    if data.cmd == "setValue" then
        local val = tonumber(data.val)
        if(val == 1) then
            CluZ->Relay->SwitchOn(0)
        elseif(val == 0) then
            CluZ->Relay->SwitchOff(0)
        end
    end
end
end

GateHttp->RelayControlListener->StatusCode = 200
GateHttp->RelayControlListener->ResponseBody = "OK"
GateHttp->RelayControlListener->SendResponse()
```

Next, we assign the script to the `OnRequest` event of the `RelayControlListener` object and send the configuration to the system.

The above script retrieves the query parameter values from the `RelayControlListener` object and performs the appropriate actions depending on what is in them. Then sends the status of the operation back to the client - in this case `200, OK`.

Operation can be easily tested using a regular web browser by entering the following URLs ( the IP address should be changed to the actual address of your Http Gateway):

```
http://192.168.88.4/relaycontrol?cmd=setValue&val=1 - Turns Relay on
```

```
http://192.168.88.4/relaycontrol?cmd=setValue&val=0 - Turns Relay Off
```

```
http://192.168.88.4/relaycontrol - Switches Relay state
```

As you can see in the examples, we can use Listener in two ways. If the parameters `cmd` (command to be executed) and `val` (value to be set) are defined properly, they set the specific Relay state. If we omit these parameters in the URL, the object works like a Switch.

The above example can be further expanded with further commands if other commands are needed. You can also add further parameters identifying the object on which these commands should be performed.

## 4.3. Status download

In the previous step, we enabled controlling the object in the system from the outside. Very often, in the next step, there is a need to provide access to the current state of the object.

One of the faster and most intuitive methods (not necessarily the best) is the definition of another Listener that will take the value `Value` from the Relay object and send it to the client. The simplest script with such functionality may look like this.

```
-- RelayStateOnRequest ()
GateHttp->RelayState->StatusCode = 200
GateHttp->RelayState->ResponseBody = "Relay State: "..CluZ->Relay->Value
GateHttp->RelayState->SendResponse ()
```

By entering the URL below into the browser, we can see that we get a response with the state of the Relay object (in a simple text form, but it is not the format of sending data that is the subject of this example).

```
http://192.168.88.4/relaystate - returns Relay State: 0 or Relay State: 1 depending on the state of object.
```

The above example works fine at first glance but let's try to take a closer look.

## 4.4. Event order

We have just constructed the Http interface (API) having two methods:

- **/relaycontrol** - allows controlling the Relay object
- **/relaystate** - returns the current state (value) of the Relay object

After a quick test, everything works well, but as we wrote above, you still need to consider how such methods will be used. Namely, it is easy to imagine that in the external system these two methods will be used right after each other: calling the switching action and after receiving the response, reading the status to confirm that the action has occurred and synchronizing the status.

And here unexpected system operation can occur - Relay turns on but the returned status is 0, i.e. invalid. The reason for this is that these operations are performed asynchronously on two different devices. There is no guarantee that the Relay state change operation will be done before asking about its state. Calling the state change action in the script `RelayControlOnRequest()` is invoked asynchronously, which means that the script does not wait for CluZ to complete the task.

The considered case is very simple and practically always works, but in the case of more complicated operations (when different target objects are involved, the operation requires the exchange of data, sending features, etc.) the risk that the status will be retrieved before the actual state of the object(object) changes is real and in complex systems we often observe such effects.

## 4.5. Event synchronization

The above problem can be solved by forcing the script `RelayControlOnRequest()` to wait for CluZ to actually execute the state change action on the target device. This can be easily done with the `clu.await()` function. E.g. calling:

```
CluZ->Relay->Switch(0)
```

we replace with:

```
clu.await(CluZ->Relay->Switch(0))
```

(We change the other calls to CluZ in the same way).

From now on, our Listener will not send the confirmation `200, OK` until the action on CluZ is actually done, so the client using this interface will not be misled by too quickly confirming the task.

However, the `clu.await()` function has a limit. The time limit for making a call is 800ms, and if the task cannot be completed in that time, the script will end in timeout and the Http client will receive a Http error: 500 Internal Server Error in response.

In most cases, this timeout is not a problem and the system will work properly but in the case of complex operations and / or when CluZ will be charged with other tasks it can happen. The way to solve the problem in this case is described in the next section.

## 4.6. Feedback confirmation

In complex systems and where we want high reliability and stability of integration, you should delay Listener Http's response until you receive confirmation from CluZ that the task has been completed.

For clarity, we define a new script and assign it to Listener:

- Event `OnRequest`: `GateHttp->SplitSyncOnRequest()`

The script `SplitSyncOnRequest()` looks like this:

```
-- SplitSyncOnRequest()
local data = GateHttp->RelayControlListener->QueryStringParams
if data == nil then
```

```

CluZ->SplitSyncCluzTask("Switch")
return
else
if data.cmd == "setValue" then
local val = tonumber(data.val)
if(val == 1) then
CluZ->SplitSyncCluzTask("On")
return
elseif(val == 0) then
CluZ->SplitSyncCluzTask("Off")
return
end
end
end
end

GateHttp->RelayControlListener->StatusCode = 400
GateHttp->RelayControlListener->ResponseBody = "Bad request"
GateHttp->RelayControlListener->SendResponse()

```

In each place of the script, when we delegate the task to CluZ (this time through an additional script, about which in a moment) we end our script without sending an Http response to the client. If the script execution reaches the final lines, it means that the query could not be interpreted correctly, which means that it is incorrect and we send back the error `400, Bad request`. By the way, we've added another level of protection against invalid call parameters.

The task is not now as previously performed directly on the target Relay object but delegated to a script on CluZ named `SplitSyncCluzTask (action: string)`. The notation used means that the script is called with a parameter called `action`, which is of type `string` - not to be confused with LUA notation where we do not define the type of the function call parameter. The `action` parameter defines the specific action to be called on the Relay object. The operation is identical to the previous case.

```

-- SplitSyncCluzTask(action: string)
if(action == "On") then
CluZ->Relay->SwitchOn(0)
elseif (action == "Off") then
CluZ->Relay->SwitchOff(0)
elseif (action == "Switch") then
CluZ->Relay->Switch(0)
else
-- Unknown action
GateHttp->SplitSyncRequestCompleted(false)
-- Return to avoid double completion
return
end
end
GateHttp->SplitSyncRequestCompleted(true)

```

Depending on the action defined, the appropriate method is performed on the Relay object. Finally, we inform GateHttp that we have completed the task and send the response to the client. A gateHttp method called `SplitSyncRequestCompleted (success: boolean)` was created for this purpose, which takes the boolean parameter: `true` if the action was successful, `false` otherwise.

```

-- SplitSyncRequestCompleted(success: boolean)
if success then
  GateHttp->RelayControlListener->StatusCode = 200
  GateHttp->RelayControlListener->ResponseBody = "OK"
else
  GateHttp->RelayControlListener->StatusCode = 405
  GateHttp->RelayControlListener->ResponseBody = "Not allowed"
end
GateHttp->RelayControlListener->SendResponse()

```

GateHttp through the above method sends a response to the client informing about success or error depending on the received parameter. In this way, we implemented the fully synchronous Http method, which has no time limit for operation. In more advanced cases, you can further improve system performance by calling the function `SplitSyncRequestCompleted (success: boolean)` in response to events informing about a change in the value of a particular object. This ensures that the change has occurred and further increases the stability of the system.

**NOTE!** For data received from external systems, always use the limited trust method as to their correctness. We recommend not passing directly the values to methods and scripts inside the system, but using specific actions depending on the method values as seen in the above scripts. If it is necessary to directly use variables received from outside, they should be transferred via user variables (which are addressable throughout the Grenton system and can be freely transferred between CLU devices). In addition, each variable received from the outside should be validated in the script for correctness, value and scope. Lack of proper verification of the received values may cause unexpected operation of the system, open access to unwanted functionalities and even cause errors and the CLU entering Emergency mode.

## 4.7. Timeout

The created Listener works almost reliably. Why almost? Let's think about what happens if CluZ for some reason never calls the `SplitSyncRequestCompleted (success: boolean)` method. GateHttp is then waiting for the current query to be terminated and stops responding to subsequent queries.

This should not happen on a well-configured system. However, an unexpected situation can always occur and therefore every element of the system should be configured to operate as independently as possible and to be resistant to errors in other areas. Therefore, our Listener should also be fully resistant to such situations.

For this purpose,, we will define a Timer object on GateHttp that will ensure that waiting for CluZ responses does not last indefinitely. Parameters of the new object:

- **Name:** SplitSyncTimeout
- **Event OnTimer:** GateHttp->SplitSyncTimeoutOnTimer()
- **Time:** 3000 - In this case you should choose the time according to the specific situation, for the purposes of the example we take 3s (3000ms)
- **Mode:** Countdown

The script executed after the specified time has passed looks like this:



```

-- SplitSyncTimeoutOnTimer()
GateHttp->RelayControlListener->StatusCode = 408
GateHttp->RelayControlListener->ResponseBody = "Timeout"
GateHttp->RelayControlListener->SendResponse()

```

The script works fairly simply, it returns the error `408, Timeout`.

In order for everything to work, the `SplitSyncOnRequest ()` and `SplitSync Request Completed (success: boolean)` scripts must be modified accordingly.

```

-- SplitSyncOnRequest()
local data = GateHttp->RelayControlListener->QueryStringParams
if data == nil then
    CluZ->SplitSyncCluzTask("Switch")
    GateHttp->SplitSyncTimeout->Start()
    return
else
    if data.cmd == "setValue" then
        local val = tonumber(data.val)
        if(val == 1) then
            CluZ->SplitSyncCluzTask("On")
            GateHttp->SplitSyncTimeout->Start()
            return
        elseif(val == 0) then
            CluZ->SplitSyncCluzTask("Off")
            GateHttp->SplitSyncTimeout->Start()
            return
        end
    end
end
end

GateHttp->RelayControlListener->StatusCode = 400
GateHttp->RelayControlListener->ResponseBody = "Bad request"
GateHttp->RelayControlListener->SendResponse()

```

Every time we delegate a task to CluZ we start the Timer `SplitSyncTimeout`.

```

-- SplitSyncRequestCompleted(success: boolean)
if(GateHttp->SplitSyncTimeout->State == 1) then
    GateHttp->SplitSyncTimeout->Stop()
    if success then
        GateHttp->RelayControlListener->StatusCode = 200
        GateHttp->RelayControlListener->ResponseBody = "OK"
    else
        GateHttp->RelayControlListener->StatusCode = 405
        GateHttp->RelayControlListener->ResponseBody = "Not allowed"
    end
end
GateHttp->RelayControlListener->SendResponse()
end

```

However, in the script `SplitSyncRequestCompleted (success: boolean)` we first check if the Timer is still in the state `1` (on). It prevents from an unnecessary attempt to send a response when the timeout has already occurred - the time for response has expired and a response was sent informing about the occurrence of the error `408, Timeout`. If the Timer still works (normal situation, the time for the answer has not run out), we stop the Timer and continue as before.

## 4.8. A lot of objects

Let's go back to the method of retrieving Relay's state for a moment. In particular, let's look at the following line again:

```
GateHttp->RelayState->ResponseBody = "Relay State: "..CluZ->Relay->Value
```

The key here is to get the value of the Value property of the Relay object:

```
CluZ->Relay->Value
```

This method works well but be aware that the value of this feature is retrieved when the script is executed. It results in communication between GateHttp and CluZ via the network. This is a synchronous call, i.e. the method waits until the response with the value of the `Value` feature of the Relay objects is delivered. We already know about some of the limitations of such a call. There are even more threats in this particular case. Namely, the value of this feature is downloaded every time the client asks about its value through the Http interface which generates unnecessary traffic in the system. Additionally, it introduces an unnecessary delay in the system. If there are a lot of such queries, it can affect system performance. In some especially simple cases this is acceptable and the system will handle it well. But not always.

Let's imagine that there are many objects in the system and we need to provide the status of all of them (in the form of JSON or CSV) in response. If in this case we use the same method, then the script that performs this task may look something like the following:

```
GateHttp->RelayState->StatusCode = 200

local response = CluZ->Relay01->Value
response = response .. "," .. CluZ->Relay02->Value
response = response .. "," .. CluZ->Relay03->Value
response = response .. "," .. CluZ->Relay04->Value
response = response .. "," .. CluZ->Relay05->Value
response = response .. "," .. CluZ->Relay06->Value
response = response .. "," .. CluZ->Relay07->Value
response = response .. "," .. CluZ->Relay08->Value
response = response .. "," .. CluZ->Relay09->Value
response = response .. "," .. CluZ->Relay10->Value
response = response .. "," .. CluZ->Relay11->Value

GateHttp->RelayState->ResponseBody = "System State: ".. response
GateHttp->RelayState->SendResponse()
```

There can be much more Relay objects in real system. Each line is used to send a request to CluZ for the Value feature over the network. Collecting the status of all objects may take a lot of time. This delays the response significantly and blocks GateHttp during the operation.

A series of inquiries occurs whenever the client asks about the state of the system. In most cases, the value of the feature between queries only changes for one object, the one that has just been changed. All this causes a lot of unnecessary traffic and negatively affects the speed of the system. From the end user's point of view, the system may be unstable in such cases, have unexpected delays, hang for short or long periods and even lose some events.

## 4.9. Status for the complex system

In order to solve the above problem, you should approach the task of downloading the device status a bit differently. Later in this section, for simplicity of examples, we will return to a single Relay object, but the given method will work for virtually any number of objects.

Let's say that instead of asking remotely CluZ about the state of Relay objects, every time the client asks for it, we could keep its value locally in the GateHttp user variable. Thanks to this, when the client asks without any delay, we return its value immediately without any delay, let's call it `RelayValueOnGateHttp`. What's more, we would like to eliminate all queries that synchronize its value and receive information only when it is needed, i.e. when the value of the `CluZ-> Relay-> Value` feature changes. To achieve this, we assign the following command to the `OnValueChanged` event of the Relay object:

```
GateHttp->RelayValueOnGateHttp=CluZ->Relay->Value
```

Which more or less means: Every time the value of the Value attribute changes, assign to the user attribute `RelayValueOnGateHttp` on GateHttp this new value. From now on, we will always have the current value of the Relay object on the GateHttp side. At the time of inquiry, we simply send this value to the client. To accomplish this, we modify the `RelayStateOnRequest ()` script as follows:

```
-- RelayStateOnRequest ()
GateHttp->RelayState->StatusCode = 200
GateHttp->RelayState->ResponseBody = "Relay State: "..GateHttp-
>RelayValueOnGateHttp
GateHttp->RelayState->SendResponse ()
```

As mentioned earlier, it can be used for any number of objects and does not cause any negative impact on system performance because only changes of individual values are communicated when they occur.

## 4.10. Push Notifications

Going one step further on the road to perfect integration, let's implement one more improvement. So far, the client himself had to ask every now and then if something did not change in the system. If the system is to be responsive then such queries must be frequent. Frequent queries generate unnecessary traffic and increase the risk of delays, especially in handling events very sensitive to delays, such as switching on the lighting, where the user immediately feels that the action did not take place immediately after touching the button.

In addition, the customer is not notified immediately about a change in the system, but only when he asks himself if anything will change.

The solution is the Push state method where the system itself actively sends a notification and changes the state of the device in the system. In order to implement such a mechanism, we create a new object on the GateHttp type HttpRequest:

- **Name:** StatePushNotification
- **Host: IP:** The http server port listening for status changes
- **Path:** /statechanged
- **Method:** PUT

Other settings unchanged.

Next we add new script `SendStatePushNotification(newValue: number)`:

```
-- SendStatePushNotification(newValue: number)
GateHttp->StatePushNotification->SetQueryStringParams ("val"..newValue)
GateHttp->StatePushNotification->SendRequest ()
```

To inform the client about the new status, call the script, specifying as a parameter the new value of the value attribute. This is best done in the `OnValueChanged` event of the Relay object. Because we assigned the value to the user variable `GateHttp-> RelayValueOnGateHttp` a step earlier, we can use it to avoid unnecessary retransmission of this value. So the assignment will look like this:

```
GateHttp->SendStatePushNotification (GateHttp->RelayValueOnGateHttp)
```

Note that copying values to the `RelayValueOnGateHttp` features must come out earlier.

From now on, whenever the value of the `Value` feature of a Relay object changes, a notification will be automatically sent with the new value.

The chosen method sends the new value as the URL parameter, but you can of course format the answer in any way and send messages in the body by setting the value using the `SetRequestBody (value)` method.

**NOTE!** Please note that Gate Http opens up unlimited possibilities of cooperation with the system and you can use it to perform any operation, even malicious. Therefore, it is important that the Gate Http configuration is carefully thought out and made with the utmost care.

## 5. Restoring factory settings - *Hard Reset*

Running the *Hard Reset* function on the GATE Http module results in:

- Removal of the saved configuration;
- Formatting the flash memory partition;
- Removal of all created LUA objects;
- Loss of communication between OM / HM and Gate module.

In order to restore the factory settings with the *Hard Reset* function, perform the following steps (in accordance with the given order):

- Disconnect power from the Gate module;
- Press and hold the *Reset* button on the module (the button is located under the bottom end of the module);
- Connect the power supply to the Gate module;
- Keep the *Reset* button pressed for at least 10 seconds - during the reset, the green LED will be permanently illuminated. The correct execution of the reset will be confirmed by a 3-blink green diode.
- Release the *Reset* button after 10 seconds
- Wait about 60 seconds until the LED - green and red - blink alternately (*Emergency mode*)

After the procedure the module will be cleared, but the module will no longer be visible (no response to *Keep-Alive*) in the project from the Object Manager level. To restore the module again, perform CLU Discovery and then send the configuration.

## 6. Configuration parameters

**NOTE!** The described functionality and integration is available for **GRENTON GATE HTTP, DIN, Eth (INT-221-E-01)** with **firmware 1.1.0-2034C or higher!**

### A. GATE

#### FEATURES

Name	Description
Uptime	Operation time of the device since the last reset (in seconds)
ClientReportInterval	Reporting period about changes in features
Date	Current date
Time	Current time (hh: mm: ss)
Local Time	Current local time stamp
Time Zone	Time zone
UnixTime	Current Unix time stamp
FirmwareVersion	Gate software version
UseCloud	Specifies whether GATE connects to the cloud
CloudConnection	Specifies the status of the GATE connection to the cloud
NTPTimeout	Waiting time for response from NTP server
UseNTP	Specifies whether GATE uses NTP
PrimaryDNS	Preferred DNS server
SecondaryDNS	Alternate DNS server

#### METHODS

Name	Description
<code>StartConsole</code>	Launches the Lua console
<code>StartConsoleOnReboot</code>	Starts the Lua console on restart
<code>SetClientReportInterval</code>	Sets the reporting period for feature changes
<code>SetPrimaryDNS</code>	Sets the PrimaryDNS feature
<code>SetSecondaryDNS</code>	Sets the SecondaryDNS feature
<code>SetDateTime</code>	Sets the date and time

## EVENTS

Name	Description
<code>OnInit</code>	An event dispatched when the device initializes

## B. HttpRequest Object

**NOTE!** Features described as not settable are features containing answers. The initial values of these features should be left unchanged. All operations on these variables should be performed on scripts (and local variables).

## FEATURES

Name	Description
Host	Host address
Path	Query Path
QueryStringParams	Query Parameters. \ z means none
Method	The type of method sent in the query, e.g. GET, POST
Timeout	Permitted response time
RequestType	<p>The type of content of the query being sent. Defines the <i>content-type</i> parameter in the query header. Depending on the type selected, the contents of the <code>RequestBody</code> feature are appropriately serialized:</p> <p>0 - None - undefined. The content-type is not sent in the header. The content of the <code>RequestBody</code> feature is not serialized.</p> <p>1 - Text - <i>content-type: text / plain</i>. The content of the <code>RequestBody</code> feature is not serialized.</p> <p>2 - JSON - <i>content-type: application / json</i>. The contents of the <code>RequestBody</code> feature are serialized to JSON format.</p> <p>3 - XML - <i>content-type: text / xml</i>. The contents of the <code>RequestBody</code> feature are serialized to XML format.</p> <p>4 - FormData - <i>content-type: application / x-www-form-urlencoded</i>. The contents of the <code>RequestBody</code> feature are serialized to the table.</p> <p>5 - Other - the content type (<i>content-type</i>) is different from the built-in one. The type can be defined by placing it in the header (the <code>RequestHeaders</code> attribute). The content is not serialized.</p>
ResponseType	<p>The type of expected answer. Defines the <i>Accept</i> parameter in the query header. Depending on the type chosen, the content of the received response (<code>ResponseBody</code> features) is properly parsed into the table:</p> <p>0 - None - <i>Accept</i> is not sent in the header of the query being sent. The answer (feature <code>ResponseBody</code>) is not parsed.</p> <p>1 - Text - <i>Accept: text / plain</i>. The answer (feature <code>ResponseBody</code>) is not parsed.</p> <p>2 - JSON - <i>Accept: application / json</i>. The answer (feature <code>ResponseBody</code>) is parsed with JSON.</p> <p>3 - XML - <i>Accept: text / xml</i>. The response (feature <code>ResponseBody</code>) is parsed from XML.</p> <p>4 - FormData - <i>Accept: application / x-www-form-urlencoded</i>. The answer (<code>ResponseBode</code> feature) is parsed.</p> <p>5 - Other - the <i>Accept</i> parameter of the header is different from the built-in one. The parameter can be defined by placing it in the header (the <code>RequestHeaders</code> attribute).</p>
RequestHeaders	Additional HTTP query headers. \ z means no content.
RequestBody	The content of the message sent in the query. \ z means no content
ResponseBody	The content of the message received after sending the query. (feature used for reading in scripts - not settable)

Name	Description
<code>StatusCode</code>	HTTP response status

## METHODS

Name	Description
<code>SendRequest</code>	Sends the query
<code>AbortRequest</code>	Aborts query handling
<code>Clear</code>	Removes the content of the query
<code>SetHost</code>	Sets the host address
<code>SetPath</code>	Sets the query path
<code>SetQueryStringParams</code>	Sets the query parameters
<code>SetMethod</code>	Sets the type of variable
<code>SetTimeout</code>	Sets the position of the bit
<code>SetResponseType</code>	Sets the number of registry bits to read
<code>SetResponseBody</code>	Sets the refresh time
<code>SetRequestHeaders</code>	Sets the waiting time for a response
<code>SetRequestBody</code>	Sets the divisor

## EVENTS

Name	Description
<code>OnRequestSent</code>	An event triggered when the query is sent
<code>OnResponse</code>	The event is triggered when the response is received

## C. HttpListener Object

**NOTE!** Features described as not settable are features containing answers. The initial values of these features should be left unchanged. All operations on these variables should be performed on scripts (and local variables).

## FEATURES



Name	Description
Path	Query path
Method	The type of method obtained in the query, e.g. GET, POST
QueryStringParams	Returns HTTP query parameters (feature used for reading in scripts - not settable)
RequestType	<p>The type of inquiry received. Depending on the type chosen, the content of the query received (the <code>RequestBody</code> attribute) is properly parsed into the table:</p> <ul style="list-style-type: none"> <li>0 - None - The answer is not parsed.</li> <li>1 - Text - The answer is not parsed.</li> <li>2 - JSON - The answer is parsed with JSON.</li> <li>3 - XML - The answer is parsed from XML.</li> <li>4 - FormData - The answer is parsed.</li> <li>5 - Other - The answer is not parsed. The <code>RequestBody</code> feature returns the contents of an HTTP query (a feature used to read in scripts - not settable).</li> </ul>
RequestBody	Returns the content of the HTTP request (feature used for reading in scripts - non-persistent)
ResponseType	<p>The content type of the sent response to the query. Defines the <i>content-type</i> parameter in the response header. Depending on the type selected, the contents of the <code>ResponseBody</code> feature are appropriately serialized:</p> <ul style="list-style-type: none"> <li>0 - None - undefined. <i>Content-type</i> is not sent in the header. The content is not serialized.</li> <li>1 - Text - <i>content-type: text / plain</i>. The content is not serialized.</li> <li>2 - JSON - <i>content-type: application / json</i>. The <code>RequestBody</code> content is serialized to JSON format.</li> <li>3 - XML - <i>content-type: text / xml</i>. The <code>RequestBody</code> content is serialized to XML format.</li> <li>4 - FormData - <i>content-type: application / x-www-form-urlencoded</i>. The <code>RequestBody</code> content is serialized.</li> <li>5 - Other - the <i>Accept</i> parameter of the header is different from the built-in one. The parameter can be defined by placing it in the header (the <code>RequestHeaders</code> attribute).</li> </ul>
ResponseBody	Returns the contents of the HTTP response (a feature used to read in scripts).

Name	Description
<div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">StatusCode</div>	Status of the HTTP response being sent. Supported statuses: <div style="display: flex; flex-direction: column; gap: 5px;"> <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">200</div> - OK  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">201</div> - Created  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">202</div> - Accepted  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">204</div> - No content  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">205</div> - Reset content  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">400</div> - Bad request  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">403</div> - Forbidden  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">404</div> - Not found  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">405</div> - Method not allowed  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">406</div> - Not acceptable  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">408</div> - Request timeout  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">409</div> - Conflict  <div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">410</div> - Gone         </div>

## METHODS

Name	Description
<div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">SendResponse</div>	Sends a response to the query
<div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">Clear</div>	Removes the contents of the answer
<div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">SetPath</div>	Sets the query path
<div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">SetResponseType</div>	Sets the response type
<div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">SetResponseBody</div>	Sets the content of the response
<div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">SetStatusCode</div>	Sets the status of the response

## EVENTS

Name	Description
<div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">OnRequest</div>	The event is triggered when the request is received

## D. Timer

### FEATURES

Name	Description
<div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">Time</div>	Counted time (in ms)
<div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">Mode</div>	Timer mode: 0 - count down (countdown), 1 - cyclical (interval)
<div style="border: 1px solid #ccc; border-radius: 5px; padding: 2px; display: inline-block;">State</div>	Current timer status: 0 - stopped, 1 - counting

### METHODS

Name	Description
<code>SetTime</code>	Sets the timer time (in ms)
<code>SetMode</code>	Sets the operating mode: 0 - count down (countdown), 1 - cyclical (interval)
<code>Start</code>	Starts the timer
<code>Stop</code>	Stops the timer

## EVENTS

Name	Description
<code>OnTimer</code>	An event triggered when the timer is counted
<code>OnStart</code>	An event triggered when the timer is started
<code>OnStop</code>	An event triggered when the timer is stopped

# XVI. DALI Controller Module

**NOTE!** The described functionality and integration is available for **GRENTON DALI Controller DIN, Eth (INT-202-D-01)** with **1.1.11 (build 2048)** or higher.

**NOTE!** DALI Controller is available for Object Manager in version **1.3.5 (build 204201)** and higher, and for CLU with firmware **5.06.04 (build 2050)** and higher.

## 1. General information

The DALI Controller module acts as a master device, in accordance with the DALI standard, it enables the operation of 64 ballasts - Control Gears, connected to the DALI bus.

**NOTE!** The maximum number of ballasts (DALI\_GEAR objects) assigned to one CLU Z-Wave is 128.

DALI Controller allows you to control all light control devices within the scope defined by the PN-EN 62386-102 standard, and the DT8 extension.

The module allows you to control single ballasts, as well as control by groups, each ballast can be assigned to 16 groups. Thanks to this, it is much easier to organize the lighting control and create advanced control scenarios.

## 2. Module configuration

**NOTE!** Before starting any work with the DALI Controller module, it is necessary to update the interface database!

### LED signaling

- The blue diode indicates the voltage on the DALI bus,
- The green diode indicates the current state of the module:
  - ON - no ballast configuration on module, DALI Discovery must be performed,
  - Flashes at 200 ms interval - DALI Discovery, the ballasts connected to the DALI bus are searched and local addresses assigned to them,
  - Flashes at 1 second interval - ballast configuration is on the module.

### Adding a module to the project

After the CLU Discovery process has been executed, two objects appear in the project:

- DALI\_MASTER - main object used to manage the module configuration,
- AnalogIN - object for monitoring the voltage on the system bus.

### A. Ballast addressing

The module configuration should start with addressing the DALI ballasts connected to the bus. The DALI Controller enables two types of addressing: fully automatic or manual.

**Automatic addressing** allows you to address the entire installation with one click, using the DALI Discovery process.

- In the DALI\_MASTER object in the `Control` tab, call the `ResetGear (Broadcast)` method and then the `DALI_Discovery` method,
- The method call initiates the automatic addressing of all ballasts on the bus, which will receive local addresses in the range 0 to 63. The assignment of an address will be confirmed by lighting the given luminaire for 300 ms. Please note that all existing addresses will be deleted when addressing is started. During DALI Discovery, addresses are assigned to the ballasts randomly,
- During DALI Discovery:
  - The green LED on the DALI Controller flashes at 200 ms interval,
  - The embedded feature `State` of the DALI\_MASTER object takes the value 1.

The duration of the DALI Discovery depends on the number of ballasts (it can take up to several minutes for the maximum number of devices).

**NOTE!** Do not perform any operations on the DALI Controller during DALI Discovery!

**Manual addressing** allows you to address individual ballasts using the `SetLocalAddress` method. It is helpful in the event that the ballast is not found after DALI Discovery, the address is doubled or we want a specific sequence of addresses in accordance with the assembly order.

In the DALI\_MASTER object in the `Control` tab, call the `SetLocalAddress` method with the `FindGear` parameter set:

- `WithoutLocalAddress` - addressing process for a device without an address,
  - `Address` - new unoccupied address that will be given to the device,
- `WithLocalAddress` - addressing process for a device with a given address,
  - `Address` - new unoccupied address that will be given to the device,
- In both cases, the address assignment will be confirmed by lighting the given luminaire for 300 ms,
- During `SetLocalAddress` :
  - The green LED on the DALI Controller flashes at 200 ms interval,
  - The embedded feature `State` of the DALI\_MASTER object takes the value 1.

**NOTE!** Do not perform any operations on the DALI Controller during `SetLocalAddress` !

### After the DALI Discovery

- The green LED on the DALI Controller flashes every 1 s (ballasts found) or is on continuously (no ballasts found),
- The embedded feature `State` of the DALI\_MASTER object takes the value:
  - 3 - ballasts found,
  - 0 - no ballasts found,
- The embedded feature `NumberOfGear` of the DALI\_MASTER returns the number of correctly found and addressed devices,
- The event `OnDALI_DiscoveryCompleted` is generated.

### Operations possible on devices after DALI Discovery has ended

Using the methods of the DALI\_MASTER object we can:

- Verify the device reporting to the given address - the `Identify` method,
- Restart the device at the given address - the `ResetGear` method,
- Set the value of the luminaire for the device at the given address - the `SetDAPCValue` method.

## B. Adding ballasts to the project

After the ballast addressing process is completed with the `DALI_Discovery` and `SetLocalAddress` methods, CLU Discovery should be performed:

- New GEAR objects are added to the project to represent each DALI device (address) correctly found and added during the addressing process,
- The embedded `GearAddresses` feature of the DALI\_MASTER object returns address numbers in the range 0 - 63, occupied by DALI devices
- GEAR objects are in the DALI\_GEAR and DALI\_GEAR\_DT8 - Device Type 8 versions:
  - DALI\_GEAR - all ballasts with basic control methods,
  - DALI\_GEAR\_DT8 - ballasts for color control (RGBWA control mode) or color temperature (Tc control mode).

**NOTE!** For correct operation of GEAR configuration and objects, CLU Discovery should be performed after each change in ballast addressing!

## C. Ballast control

**The control of a single ballast** is carried out using a given DALI\_GEAR / DALI\_GEAR\_DT8 object using available methods or using the methods of the DALI\_MASTER object (detailed functionalities can be found in the description of individual objects).

The ballast groups are controlled by the DALI\_MASTER object using the `SetGroupDAPCValue`, `GroupSwitchOn`, `GroupSwitchOff` methods. In order to be able to control a given group of devices, it is necessary to:

- For the desired GEAR objects, set the value of the embedded feature `Group`. Each object can be assigned to 16 groups in the range 1 - 16, the next groups are given after a decimal point,
- After assigning objects to groups, send the configuration to CLUZ,
- After sending the configuration, the groups are sent by the DALI Controller. Embedded feature `State` of the DALI\_MASTER object takes the value 4. The duration of the process depends on the number of devices for which the value of the `Group` feature has been changed, it can last up to 60 seconds,
- After correct grouping, the embedded feature of the DALI\_MASTER object takes the value 3.

**NOTE!** When assigning groups (after CLUZ restart / configuration sending) it is not possible to control the objects!

## D. RampTime

The DALI Controller supports the smooth change of the `DAPCValue` value using the `RampTime` parameter, in a logarithmic manner:

RampTime	Minimum fade time [s]	Nominal fade time [s]	Maximum fade time [s]
1	0,6	0,7	0,8
2	0,9	1,0	1,1
3	1,3	1,4	1,6
4	1,8	2,0	2,2
5	2,5	2,8	3,1
6	3,6	4,0	4,4
7	5,1	5,7	6,2
8	7,2	8,0	8,8
9	10,2	11,3	12,4
10	14,4	16,0	17,6
11	20,4	22,6	24,9
12	28,8	32,0	35,2
13	40,7	45,3	49,8
14	57,6	64,0	70,4
15	81,5	90,5	99,6

## 3. Objects

### A. DALI\_MASTER

#### FEATURES

Name	Description
State	<ul style="list-style-type: none"> <li>0 - no ballast configuration</li> <li>1 - DALI Discovery</li> <li>3 - ballast configuration is on the device</li> <li>4 - saving information about groups</li> </ul>
NumberOfGear	Number of ballasts in the device configuration
GearAddresses	Ballast addresses given during DALI_Discovery. The feature value is refreshed after restart system

#### METHODS

Name	Description
<code>Identify</code>	Turns on the luminaire for 2 seconds
<code>ResetGear</code>	Resets the ballast
<code>SetLocalAddress</code>	Sets the local address of the ballast
<code>DALI_Discovery</code>	Searching for ballasts connected to the DALI bus and assigning them local addresses. At the time of addressing, the ballast is turned on for 300 ms. No device operations should be performed during <code>DALI_Discovery</code> !
<code>SetDAPCValue</code>	Sets the value of the power with which the luminaire shines. <code>RampTime</code> parameter set on a logarithmic scale 0.8 - 90 [s]
<code>SetGroupDAPCValue</code>	Sets the value of the power with which the luminaire shines for a given group. <code>RampTime</code> parameter set on a logarithmic scale 0.8 - 90 [s]
<code>GroupSwitchOn</code>	Turns on the luminaire for a given group. <code>RampTime</code> parameter set on a logarithmic scale 0.8 - 90 [s]
<code>GroupSwitchOff</code>	Turns off the luminaire for a given group. <code>RampTime</code> parameter set on a logarithmic scale 0.8 - 90 [s]

## EVENTS

Name	Description
<code>OnDALI_DiscoveryCompleted</code>	Event occurring after the ballasts have been found and given local addresses

## B. DALI\_GEAR

### FEATURES

Name	Description
<code>Address</code>	Ballast address
<code>Group</code>	Ballast group numbers, subsequent groups from the 1-16 range are given after the decimal point. 0 - no belonging to any group
<code>DAPCValue</code>	The value of the power with which the luminaire shines

### METHODS



Name	Description
Identify	Turns on the luminaire for 2 seconds
SetDAPCValue	Sets the value of the power with which the luminaire shines. RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
Switch	Changes the luminaire state to the opposite (0 / 254). RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
SwitchOn	Turns on the luminaire. RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
SwitchOff	Turns off the luminaire. RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
Hold	Executes the function of illuminating / dimming the luminaire
HoldUp	Executes the function of illuminating the luminaire
HoldDown	Executes the function of dimming the luminaire

#### EVENTS:

Name	Description
OnDAPCValueChange	Event occurring when changing the DAPCValue
OnSwitchOn	Event occurring when the DAPCValue value is changed from 0 to the greater value
OnSwitchOff	Event occurring when the DAPCValue value is changed to 0

### C. DALI\_GEAR\_DT8

#### FEATURES

Name	Description
Address	Ballast address
Group	Ballast group numbers, subsequent groups from the 1-16 range are given after the decimal point. 0 - no belonging to any group
DAPCValue	The value of the power with which the luminaire shines
HSVValue	Brightness value as per the HSV model (range: 0.00-1.00). The feature does not get the actual brightness of the luminaire! Set according to called <code>SetHSVValue</code> method.
HSVSaturation	Colour saturation value as per the HSV model (0.00-1.00). This feature does not get the actual color saturation of the luminaire! Set according to called <code>SetHSVSaturation</code> method.
HSVHue	Colour hue value as per the HSV model (0-360). The feature does not get the actual color of the luminaire! Set according to called <code>SetHSVHue</code> method.

## METHODS

Name	Description
Identify	Turns on the luminaire for 2 seconds
SetDAPCValue	Sets the value of the power with which the luminaire shines
Switch	Changes the luminaire state to the opposite (0 / 254). RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
SwitchOn	Turns on the luminaire. RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
SwitchOff	Turns off the luminaire. RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
Hold	Executes the function of illuminating / dimming the luminaire
HoldUp	Executes the function of illuminating the luminaire
HoldDown	Executes the function of dimming the luminaire
SetHSVValue	Sets brightness value (0.00-1.00). RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
SetHSVSaturation	Sets saturation value (0.00-1.00). RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
SetHSVHue	Sets hue value (0-360). RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
SetRGBValue	Sets the value of the R (Red), G (Green), B (Blue) channels. RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
SetWAFValue	Sets the value of the W (White) channel, and the A (Amber) and F (Freecolor) parameters. RampTime parameter set on a logarithmic scale 0.8 - 90 [s]
SetColourTemperature	Sets the color temperature value, where 0 - physical minimum, 100 - physical maximum. RampTime parameter set on a logarithmic scale 0.8 - 90 [s]

#### EVENTS:

Name	Description
OnDAPCValueChange	Event occurring when changing the DAPCValue
OnSwitchOn	Event occurring when the DAPCValue value is changed from 0 to the greater value
OnSwitchOff	Event occurring when the DAPCValue value is changed to 0

## D. AnalogIN

### FEATURES

Name	Description
Value	Current output value taking into account the scalar
Value%	Current percentage input value of the maximum value ( <code>MaxValue</code> characteristic)
Sensitivity	Minimum change of input state when the <code>OnValueChanged</code> , <code>OnValueLower</code> or <code>OnValueRise</code> event is generated
MinValue	Minimum value of the Value characteristic after exceeding which the <code>OnOutOfRange</code> event is generated
MaxValue	Maximum value of the Value characteristic after exceeding which the <code>OnOutOfRange</code> event is generated

### METHODS

Name	Description
SetSensitivity	Sets input sensitivity value
SetMinValue	Sets <code>MinValue</code>
SetMaxValue	Sets <code>MaxValue</code>

### EVENTS:

Name	Description
OnValueChanged	Event resulting from changing input state
OnValueLower	Event occurs when a value lower than the value from the last reading appears at input
OnValueRise	Event occurs when a value higher than the value from the last reading appears at input
OnOutOfRange	Event resulting from exceeding the permissible range ( <code>MinValue</code> : <code>MaxValue</code> )
OnInRange	Event occurs when value returns to <code>MinValue</code> / <code>MaxValue</code> range

# XVII. Z-Wave modules

This chapter presents a description of the scope of support for other manufacturers' Z-Wave modules, which are available in the Grenton system.

**NOTE!** A full list of devices is available at <https://support.grenton.pl/pl/support/solutions> in the article 'Which wireless Z-Wave modules are supported?'

## 1. Fibaro UBS

**Module version: FGBS-001 v2.1.**

### 1.1. General information

The Fibaro UBS Z-Wave module has two potential-free inputs. It allows reading of values from up to four 1-Wire sensors. In addition, it allows you to change the configuration parameters (Fibaro configuration interface).

**NOTE!** Addition / removal is done by clicking the button in the module three times during inclusion / exclusion.

### 1.2. Objects

#### A. ZWAVE\_DIN

Potential-free inputs

#### FEATURES

Name	Description
<code>Value</code>	Returns the input state
<code>HoldDelay</code>	The time after which pressing and holding the button will trigger the <code>OnHold</code> event
<code>HoldInterval</code>	The cyclic interval (in ms), after which the next <code>OnHold</code> events are triggered while holding the button

#### METHODS

Name	Description
<code>SetHoldDelay</code>	Sets <code>HoldDelay</code> value
<code>SetHoldInterval</code>	Sets <code>HoldInterval</code> value

#### EVENTS

Name	Description
<code>OnChange</code>	The cyclic interval (in ms), after which the next <code>OnHold</code> events are triggered while holding the button
<code>OnSwitchOn</code>	An event triggered when the high state is set on input
<code>OnSwitchOff</code>	An event triggered when the low state is set on input
<code>OnShortPress</code>	The event is triggered after pressing the button for 500-2000ms
<code>OnLongPress</code>	The event is triggered after pressing the button for 2000-5000ms
<code>OnHold</code>	Event triggered when the input is in the high state, the first time after the <code>holdDelay</code> time has elapsed, and then cyclically every <code>HoldInterval</code> value
<code>OnClick</code>	Event triggered after pressing the button for less than 500ms

## B. ZWAVE\_1W\_SENSOR

The object is responsible for the 1-Wire sensor. A separate object is created for each sensor. Up to 4 1-Wire sensors (DS18B20) can be connected to the UBS Fibaro module.

ZWAVE\_1W\_SENSOR objects are always added with the addition of the Fibaro UBS module to the CLU / project in the OM, regardless of the number of connected sensors. The Discovered feature - informing whether the Discovery 1-Wire sensor has arrived at Discovery and connected to the UBS module - informs about whether the sensor is connected.

When connecting or disconnecting the 1-Wire sensors, you must remove and then add the UBS module to the CLU Z-Wave module. Fibaro UBS module will report the new serial number - it is possible to rewrite the object configuration (automatic or manual). After adding sensors again, the order of sensors can be re-indexed to ZW\_1W\_SENSOR objects.

The Fibaro UBS module for the 1-Wire sensor does not return information if during the system operation the sensor has been disconnected - the last value collected is stored, therefore it is not recommended to use these sensors as a source of temperature control.

At the moment of short-circuit on the 1-Wire, all sensors connected to the Fibaro UBS module (available / visible in OM) return 0.00 - therefore, with a longer (unplanned) occurrence of this value, check the correctness of the 1-Wire connection.

## FEATURES

Name	Description
<code>Value</code>	The value of the input
<code>MinValue</code>	The minimum value of the input
<code>MaxValue</code>	The maximum value of the input
<code>Discovered</code>	Information returned during CLU Discovery about connecting the sensor to the module

## EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the output value is changed
<code>OnRise</code>	Event triggered when the upper hysteresis threshold is exceeded (rising edge)
<code>OnLower</code>	Event triggered when the lower hysteresis threshold is exceeded (falling edge)
<code>OnOutOfRange</code>	Event triggered when the output value is outside the specified range ( <code>MinValue</code> : <code>MaxValue</code> )
<code>OnInRange</code>	An event triggered when the value returns to the interval within the threshold values ( <code>MinValue</code> : <code>MaxValue</code> )

## C. ZWAVE\_CONFIG

The object displays information about parameters and communication with the module in the Z-Wave network. It allows setting advanced configuration parameters of a given module (specified individually in the manual).

### FEATURES

Name	Description
<code>NodeID</code>	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
<code>Banned</code>	<p>Information on blocking Z-Wave communication with the module:</p> <ul style="list-style-type: none"> <li><code>0</code> - communication with the module is not blocked,</li> <li><code>1</code> - communication with the module is blocked (banned module).</li> </ul> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the <code>FailCount</code> attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
<code>FailCount</code>	The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the <code>FailCount</code> attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module.
<code>Register</code>	The register number (parameter) of the configuration that has been read / set recently using the available methods
<code>Value</code>	The value of the configuration register (parameter)

### METHODS

Name	Description
<code>RemoveBan</code>	It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module. <b>Note!</b> <i>The RemoveBan feature is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In case of failure, the entire blocking process is restarted!</i>
<code>ClearFailCount</code>	Clears the number of unsuccessful communication attempts
<code>Set</code>	Sets the value of a given configuration register (parameter): <code>1</code> - <code>Register</code> (register or parameter number), <code>2</code> - <code>Value</code> (the value of the register or parameter), <code>3</code> - <code>Size</code> (size of the sent register or parameter value - in bytes)
<code>Get</code>	Gets the value of a given configuration (parameter) register
<code>SetDefault</code>	Sets the default value for a given configuration (parameter) register

## EVENTS

Name	Description
<code>OnBanned</code>	An event that is triggered when the device is banned

## 2. NEO Coolcam Motion Sensor (PIR)

Module version: NAS-PD01ZE HW: 66 FW: 3.80

### 2.1. General information

The Z-Wave Neo Coolcam Motion Sensor module allows you to read: motion sensor status (PIR), light level and battery level. In addition, it gives you the option of setting / reading the module's wake-up time.

**NOTE!** Addition / removal is done by clicking the button three times in the Neo module during inclusion / exclusion. Correctly carried out process will be confirmed by a five-fold blink of the diode.

### 2.2. Objects

#### A. BINARY\_SENSOR

An object that allows reading the status of the motion sensor.

#### FEATURES

Name	Description
<code>Value</code>	Returns the input status: <b>0</b> - no violation, <b>1</b> - violation



## EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the status changes to the opposite
<code>OnSwitchOn</code>	An event triggered when the high state is set on input
<code>OnSwitchOff</code>	An event triggered when the low state is set on input

## B. ANALOG\_SENSOR

The object allows reading the illumination measured in luxes.

### FEATURES

Name	Description
<code>Value</code>	The current value of the sensor
<code>MinValue</code>	The value below which the <code>OnOutOfRange</code> event is generated
<code>MaxValue</code>	The value above which the <code>OnOutOfRange</code> event is generated

### METHODS

Name	Description
<code>SetMinValue</code>	Sets the low threshold value of the <code>OnOutOfRange</code> event
<code>SetMaxValue</code>	Sets the upper threshold value of the <code>OnOutOfRange</code> event

## EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the sensor value is changed
<code>OnValueRaise</code>	An event is triggered when the sensor value changes to a higher one than the previous one
<code>OnValueDrop</code>	An event triggered when the sensor value is changed to a lower one than the previous one
<code>OnOutOfRange</code>	An event triggered when one of the threshold values <code>MinValue</code> / <code>MaxValue</code> is exceeded
<code>OnInRange</code>	An event triggered when the value returns to the interval within the threshold values ( <code>MinValue</code> : <code>MaxValue</code> )

## C. ZWAVE\_BATTERY

The object allows reading the battery status. The reading takes place cyclically, every time set, for the `Interval` feature of the ZWAVE\_WAKEUP object (3600s by default).

### FEATURES

Name	Description
<code>BatteryLevel</code>	Battery level of the Z-Wave module (in percent)
<code>WarningLevel</code>	Battery level below which warning events are generated

### METHODS

Name	Description
<code>SetWarningLevel</code>	Sets the warning level of the Z-Wave module battery

### EVENTS

Name	Description
<code>OnChange</code>	An event is triggered when the battery level changes
<code>OnLowBattery</code>	An event triggered when a battery drop is detected below the warning level
<code>OnBatteryGood</code>	An event triggered when a battery level returns to a value above the warning level

## D. ZWAVE\_WAKEUP

The object enables setting and reading the battery-awakening time of the Z-Wave module. The default value set by the CLU is 3600s (60 minutes). The minimum value is 300s (5 minutes); maximum 16777200s (about 194 days). It is possible to set values in step 60s (360s, 420s, 480s, etc.)

### FEATURES

Name	Description
<code>Interval</code>	Time of self-awakening of the Z-Wave module from sleep mode (in seconds)
<code>LastWakeUp</code>	Time of the last awakening of the Z-Wave module from sleep mode

### METHODS

Name	Description
<code>SetInterval</code>	Sets the time of automatic wake-up of the Z-Wave module from the sleep mode

### EVENTS

Name	Description
OnWakeUp	An event triggered when the Z-Wave module wakes up from sleep mode

## E. ZWAVE\_CONFIG

The object displays information about parameters and communication with the module in the Z-Wave network. It allows setting advanced configuration parameters of a given module (specified individually in the manual).

### FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:</p> <ul style="list-style-type: none"> <li>0 - communication with the module is not blocked,</li> <li>1 - blocked communication with the module (module banned).</li> </ul> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the <code>FailCount</code> attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked ( <code>Banned</code> = 1)
Register	The register number (parameter) of the configuration that has been read / set recently using the available methods
Value	<p>The value of the configuration register (parameter)</p> <p><b>NOTE!</b> Parameter 2, 3, 5 and 8 refer to the association of modules that is not supported by the Grenton system!</p> <p><b>NOTE!</b> Parameter 3 - changing the parameter value does not cause sending it during motion detection!</p> <p><b>NOTE!</b> Parameter 4 - correct setting of the parameter value, however the module itself does not change the operating mode !</p> <p><b>NOTE!</b> Parameter 7 and 9 - correct setting of the parameter value, however the set value has not been tested due to the faulty sensor!</p> <p><b>NOTE!</b> Parameter 1, 6 - no noticeable changes in module work after the change of value!</p> <p><b>NOTE!</b> Parameter 9 - smaller range of set values (up to 100 lux)!</p> <p><b>NOTE!</b> There is no information on the register number 11 (Motion Event Report One Time Enable) in the documentation!</p>

### METHODS

Name	Description
<code>RemoveBan</code>	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module..</p> <p><b>NOTE!</b> <code>RemoveBan</code> is not synonymous with the correct communication with the module - it allows re-sending the command / query to the module! In case of failure, the entire blocking process is restarted!</p>
<code>ClearFailCount</code>	Clears the number of unsuccessful communication attempts
<code>Set</code>	<p>Sets the value of a given configuration register (parameter):</p> <p><code>Register</code> (register or parameter number),</p> <p><code>Value</code> (the value of the register or parameter),</p> <p><code>Size</code> (size of the sent register or parameter value - in bytes)</p> <p><b>NOTE!</b> Calling the <code>Set</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after wake up the red LED will blink!</p>
<code>Get</code>	<p>Gets the value of a given configuration (parameter) register</p> <p><b>NOTE!</b> Calling the <code>Get</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after wake up the red LED will blink!</p>
<code>SetDefault</code>	<p>Sets the default value for a given configuration (parameter) register</p> <p><b>NOTE!</b> Calling the <code>SetDefault</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after wake up the red LED will blink!</p>

## EVENTS

Name	Description
<code>OnBanned</code>	An event that is triggered when the device is banned

## 3. NEO Coolcam Door / Window Sensor

Module version: NAS-DS01Z

### 3.1. General information

The Z-Wave Neo Coolcam Door / Window Sensor module allows reading the status of the reed (NC) and the battery level. In addition, it gives you the option of setting / reading the module's wake-up time.

**NOTE!** Addition / removal is done by clicking the button three times in the Neo module during inclusion / exclusion. Correctly carried out process will be confirmed by a five-fold blink of the diode.

## 3.2. Objects

### A. BINARY\_SENSOR

The object allows reading the reed open / close status.

#### FEATURES

Name	Description
<code>Value</code>	Returns the input status: <b>0</b> - closing, <b>1</b> - opening

#### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the status changes to the opposite
<code>OnSwitchOn</code>	An event triggered when the high state is set on input
<code>OnSwitchOff</code>	An event triggered when the low state is set on input

### B. ZWAVE\_BATTERY

The object allows reading the battery status. The reading takes place cyclically, every set time, for the `Interval` feature of the object `ZWAVE_WAKEUP`.

#### FEATURES

Name	Description
<code>BatteryLevel</code>	Battery level of the Z-Wave module (in percent)
<code>WarningLevel</code>	Battery level below which warning events are generated

#### METHODS

Name	Description
<code>SetWarningLevel</code>	Sets the warning level of the Z-Wave module battery

#### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the battery level changes
<code>OnLowBattery</code>	An event triggered when a battery drop is detected below the warning level
<code>OnBatteryGood</code>	An event triggered when the battery level returns to a value above the warning level

## C. ZWAVE\_WAKEUP

The object enables setting and reading the battery-awakening time of the Z-Wave module. The default value set by the CLU is 3600s (60 minutes). The minimum value is 300s (5 minutes); maximum 16777200s (about 194 days). It is possible to set values in step 60s (360s, 420s, 480s, etc.)

### FEATURES

Name	Description
<code>Interval</code>	The period of automatic awakening of the Z-Wave module from the sleep mode (in seconds)
<code>LastWakeUp</code>	Time of the last awakening of the Z-Wave module from sleep mode

### METHODS

Name	Description
<code>SetInterval</code>	Sets the period of automatic awakening of the Z-Wave module from the sleep mode

### EVENTS

Name	Description
<code>OnWakeUp</code>	An event triggered when the Z-Wave module wakes up from sleep mode

## D. ZWAVE\_CONFIG

The object displays information about parameters and communication with the module in the Z-Wave network. It allows setting advanced configuration parameters of a given module (specified individually in the manual).

### FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:</p> <p>0 - communication with the module is not blocked,  1 - blocked communication with the module (module banned).</p> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the FailCount attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked (Banned = 1)
Register	The register number (parameter) of the configuration that has been read / set recently using the available methods
Value	<p>The value of the configuration register (parameter)</p> <p><b>NOTE!</b> Parameters 1 and 2 refer to the association of modules, which is not supported by the Grenton system!</p>

## METHODS

Name	Description
<code>RemoveBan</code>	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> <code>RemoveBan</code> is not synonymous with re-communication with the module - it allows re-sending an order / query to the module! In case of failure, the entire blocking process is restarted!</p>
<code>ClearFailCount</code>	Clears the number of unsuccessful communication attempts
<code>Set</code>	<p>Sets the value of a given configuration register (parameter):</p> <p><code>Register</code> (register or parameter number),</p> <p><code>Value</code> (register or parameter value),</p> <p><code>Size</code> (size of the sent register or parameter value - in bytes)</p> <p><b>NOTE!</b> Calling the <code>Set</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after wake up the red LED will blink!</p>
<code>Get</code>	<p>Gets the value of a given configuration (parameter) register</p> <p><b>NOTE!</b> Calling the <code>Get</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after wake up the red LED will blink!**</p>
<code>SetDefault</code>	<p>Sets the default value for a given configuration (parameter) register</p> <p><b>NOTE!</b> Calling the <code>SetDefault</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after waking up the red LED will blink!</p>

## EVENTS

Name	Description
<code>OnBanned</code>	An event triggered when the device is banned

## 4. INFIBITY Motion Sensor (PIR) [NEO Coolcam]

Module version: NAS-PD01ZE HW: 66 FW: 3.80

### 4.1. General information

The Z-Wave Infibity Motion Sensor module enables reading of: motion sensor status (PIR), lighting level, temperature and battery level. In addition, it gives you the option of setting / reading the module's wake-up time.

**NOTE!** Addition / removal is done by clicking the button three times in the Infibity module during inclusion / exclusion. Correctly carried out process will be confirmed by a five-fold blink of the diode.



## 4.2. Objects

### A. BINARY\_SENSOR

The object allows reading the status of the motion sensor.

#### FEATURES

Name	Description
<code>Value</code>	Returns the input status: <b>0</b> - no violation, <b>1</b> - violation

#### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the status changes to the opposite
<code>OnSwitchOn</code>	An event triggered when the high state is set on input
<code>OnSwitchOff</code>	An event dispatched when the low state is set on input

### B. ANALOG\_SENSOR

The object allows reading the illumination measured in luxes (ANALOG\_SENSOR1) and temperature (ANALOG\_SENSOR2).

#### FEATURES

Name	Description
<code>Value</code>	The current value of the sensor
<code>MinValue</code>	The value below which the <code>OnOutOfRange</code> event is generated
<code>MaxValue</code>	The value above which the <code>OnOutOfRange</code> event is generated

#### METHODS

Name	Description
<code>SetMinValue</code>	Sets the low threshold value of the <code>OnOutOfRange</code> event
<code>SetMaxValue</code>	Sets the upper threshold value of the <code>OnOutOfRange</code> event

#### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the sensor value is changed
<code>OnValueRaise</code>	An event triggered when the sensor value changes to a higher one than the previous one
<code>OnValueDrop</code>	An event triggered when the sensor value is changed to a lower one than the previous one
<code>OnOutOfRange</code>	An event triggered when one of the threshold values <code>MinValue</code> / <code>MaxValue</code> is exceeded
<code>OnInRange</code>	An event triggered when the value returns to the interval within the threshold values ( <code>MinValue</code> : <code>MaxValue</code> )

## C. ZWAVE\_BATTERY

The object allows reading the battery status. The reading takes place cyclically, every set time, for the `Interval` feature of the object `ZWAVE_WAKEUP` (3600s by default).

### FEATURES

Name	Description
<code>BatteryLevel</code>	Battery level of the Z-Wave module (in percent)
<code>WarningLevel</code>	Battery level below which warning events are generated

### METHODS

Name	Description
<code>SetWarningLevel</code>	Sets the warning level of the Z-Wave module battery

### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the battery level changes
<code>OnLowBattery</code>	An event triggered when a battery drop is detected below the warning level
<code>OnBatteryGood</code>	An event triggered when the battery level returns to a value above the warning level

## D. ZWAVE\_WAKEUP

The object enables setting and reading the battery-awakening time of the Z-Wave module. The default value set by the CLU is 3600s (60 minutes). The minimum value is 300s (5 minutes); maximum 16777200s (about 194 days). It is possible to set values in step 60s (360s, 420s, 480s, etc.)

### FEATURES

Name	Description
<code>Interval</code>	The period of automatic awakening of the Z-Wave module from the sleep mode (in seconds)
<code>LastWakeUp</code>	Time of the last awakening of the Z-Wave module from sleep mode

## METHODS

Name	Description
<code>SetInterval</code>	Sets the period of automatic awakening of the Z-Wave module from the sleep mode

## EVENTS

Name	Description
<code>OnWakeUp</code>	An event triggered when the Z-Wave module wakes up from sleep mode

## E. ZWAVE\_CONFIG

The object displays information about parameters and communication with the module in the Z-Wave network. It allows setting advanced configuration parameters of a given module (specified individually in the manual).

## FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:  <input type="checkbox"/> 0 - communication with the module is not blocked,  <input type="checkbox"/> 1 - blocked communication with the module (module banned).</p> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the <input type="checkbox"/> FailCount attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked ( <input type="checkbox"/> Banned = 1)
Register	The register number (parameter) of the configuration that has been read / set recently using the available methods
Value	<p>The value of the configuration register (parameter)</p> <p><b>NOTE!</b> Parameter 2, 3, 5 and 8 refer to the association of modules that is not supported by the Grenton!</p> <p><b>NOTE!</b> Parameter 1, 6 and 7 - no noticeable changes in the module's work after the change of value!</p> <p><b>NOTE!</b> Parameter 9 - smaller range of set values (up to 100 lux)!</p>

## METHODS

Name	Description
<code>RemoveBan</code>	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> <code>RemoveBan</code> is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In case of failure, the entire blocking process is restarted!</p>
<code>ClearFailCount</code>	Clears the number of unsuccessful communication attempts
<code>Set</code>	<p>Sets the value of a given configuration (parameter) register:</p> <p><code>Register</code> (register or parameter number),  <code>Value</code> (register or parameter value),  <code>Size</code> (Size of the registry value sent or parameter - in bytes)</p> <p><b>NOTE!</b> Calling the <code>Set</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after wake up the red LED will blink!</p>
<code>Get</code>	<p>Gets the value of a given register (parameter) configuration</p> <p><b>NOTE!</b> Calling the <code>Get</code> method must be made after waking up the battery module! In order to wake up the module, please click the button in the module three times - after waking up the red LED will blink!</p>
<code>SetDefault</code>	<p>Sets the default value for a given register (parameter) configuration</p> <p><b>NOTE!</b> Calling the <code>SetDefault</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after waking up the red LED will blink!</p>

## EVENTS

Name	Description
<code>OnBanned</code>	An event triggered when the device is banned

## 5. INFIBITY Door/Window Sensor [NEO Coolcam]

Module version: NAS-DS01Z HW: 65 FW: 3.61

### 5.1. General information

The Z-Wave Infibity Door / Window Sensor module allows reading of the status of the reed (NC) and the battery level. In addition, it gives you the option of setting / reading the module's wake-up time.

**NOTE!** Addition / removal is done by clicking the button three times in the Infibity module during inclusion / exclusion. Correctly carried out process will be confirmed by a five-fold blink of the diode.

## 5.2. Objects

### A. BINARY\_SENSOR

The object allows reading the reed open / close status.

#### FEATURES

Name	Description
<code>Value</code>	Returns the input state: <code>0</code> - closing, <code>1</code> - opening

#### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the status changes to the opposite
<code>OnSwitchOn</code>	An event triggered when the high state is set on input
<code>OnSwitchOff</code>	An event triggered when the low state is set on input

### B. ZWAVE\_BATTERY

The object allows reading the battery status. The reading takes place cyclically, every time set, for the `Interval` feature of the ZWAVE\_WAKEUP object.

#### FEATURES

Name	Description
<code>BatteryLevel</code>	Battery level of the Z-Wave module (in percent)
<code>WarningLevel</code>	Battery level below which warning events are generated

#### METHODS

Name	Description
<code>SetWarningLevel</code>	Sets the warning level of the Z-Wave module battery

#### EVENTS

Name	Description
<code>OnChange</code>	The event triggered when the battery level changes
<code>OnLowBattery</code>	An event triggered when a battery drop is detected below the warning level
<code>OnBatteryGood</code>	An event triggered when the battery level returns to a value above the warning level

## C. ZWAVE\_WAKEUP

The facility enables setting and reading the battery-awakening time of the Z-Wave module. The default value set by the CLU is 3600s (60 minutes). The minimum value is 300s (5 minutes); maximum 16777200s (about 194 days). It is possible to set values in step 60s (360s, 420s, 480s, etc.)

### FEATURES

Name	Description
<code>Interval</code>	The period of automatic awakening of the Z-Wave module from the sleep mode (in seconds)
<code>LastWakeUp</code>	Time of the last awakening of the Z-Wave module from sleep mode

### METHODS

Name	Description
<code>SetInterval</code>	Sets the period of automatic awakening of the Z-Wave module from the sleep mode

### EVENTS

Name	Description
<code>OnWakeUp</code>	An event that is triggered when the Z-Wave module wakes up from sleep mode

## D. ZWAVE\_CONFIG

The object displays information about parameters and communication with the module in the Z-Wave network. It allows setting advanced configuration parameters of a given module (specified individually in the manual).

### FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:</p> <p>0 - communication with the module is not blocked,  1 - blocked communication with the module (module banned).</p> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the FailCount attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked (Banned = 1)
Register	The register number (parameter) of the configuration that has been read / set recently using the available methods
Value	<p>The value of the configuration register (parameter)</p> <p><b>NOTE!</b> Parameters 1 and 2 refer to the association of modules, which is not supported by the Grenton system!</p>

## METHODS



Name	Description
<code>RemoveBan</code>	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> <code>RemoveBan</code> is not synonymous with the correct communication with the module again - it allows re-sending the command / inquiry to the module! In case of failure, the entire blocking process is restarted!</p>
<code>ClearFailCount</code>	Clears the number of unsuccessful communication attempts
<code>Set</code>	<p>Sets the value of a given configuration register (parameter):</p> <p><code>Register</code> (register or parameter number),</p> <p><code>Value</code> (the value of the register or parameter),</p> <p><code>Size</code> (size of the sent register or parameter value - in bytes)</p> <p><b>NOTE!</b> Calling the <code>Set</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after wake up the red LED will blink!</p>
<code>Get</code>	<p>Gets the value of a given register (parameter) configuration</p> <p><b>NOTE!</b> Calling the <code>Get</code> method must be made after waking up the battery module! In order to wake up the module, please click the button in the module three times - after waking up the red LED will blink!</p>
<code>SetDefault</code>	<p>Sets the default value for a given register (parameter) configuration</p> <p><b>NOTE!</b> Calling the <code>SetDefault</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after waking up the red LED will blink!</p>

## EVENTS

Name	Description
<code>OnBanned</code>	An event triggered when the device is banned

## 6. INFIBITY Water Sensor [NEO Coolcam]

Module version: NAS-WS02ZU HW: 32 FW: 2.133

### 6.1. General information

The Z-Wave Infibity Water Sensor module enables reading of the status of the flood sensor and the battery level. In addition, it gives you the option of setting / reading the module's wake-up time.

**NOTE!** Addition / removal is done by clicking the button three times in the Infibity module during inclusion / exclusion. Correctly carried out process will be confirmed by a five-fold blink of the diode.

**NOTE!** The module in the Object Manager reports as NEO COOLCAM!

## 6.2. Objects

### A. BINARY\_SENSOR

The object allows reading the state of the flood sensor.

#### FEATURES

Name	Description
<code>Value</code>	Returns the input status: <b>0</b> - dry, <b>1</b> - flooded

#### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the status changes to the opposite
<code>OnSwitchOn</code>	An event triggered when the high state is set on input
<code>OnSwitchOff</code>	An event triggered when the low state is set on input

### B. ZWAVE\_BATTERY

The object allows reading the battery status. The reading takes place cyclically, every set time, for the `Interval` feature of the object `ZWAVE_WAKEUP`.

#### FEATURES

Name	Description
<code>BatteryLevel</code>	Battery level of the Z-Wave module (in percent)
<code>WarningLevel</code>	Battery level below which warning events are generated

#### METHODS

Name	Description
<code>SetWarningLevel</code>	Sets the warning level of the Z-Wave module battery

#### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the battery level changes
<code>OnLowBattery</code>	An event triggered when a battery drop is detected below the warning level
<code>OnBatteryGood</code>	An event triggered when the battery level returns to a value above the warning level

## C. ZWAVE\_WAKEUP

The object enables setting and reading the battery-awakening time of the Z-Wave module. The default value set by the CLU is 3600s (60 minutes). The minimum value is 300s (5 minutes); maximum 16777200s (about 194 days). It is possible to set values in step 60s (360s, 420s, 480s, etc.)

### FEATURES

Name	Description
<code>Interval</code>	Okres samoczynnego wybudzania modułu Z-Wave z trybu uśpienia (w sekundach)
<code>LastWakeUp</code>	Czas ostatniego wybudzenia modułu Z-Wave z trybu uśpienia

### METHODS

Name	Description
<code>SetInterval</code>	Sets the period of automatic awakening of the Z-Wave module from the sleep mode

### EVENTS

Name	Description
<code>OnWakeUp</code>	An event that is triggered when the Z-Wave module wakes up from sleep mode

## D. ZWAVE\_CONFIG

The object displays information about parameters and communication with the module in the Z-Wave network. It allows setting advanced configuration parameters of a given module (specified individually in the manual).

### FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:</p> <p>0 - communication with the module is not blocked,  1 - blocked communication with the module (module banned).</p> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the FailCount attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked (Banned = 1)
Register	The register number (parameter) of the configuration that has been read / set recently using the available methods
Value	<p>The value of the configuration register (parameter)</p> <p><b>NOTE!</b> Parameter 7 refers to the association of modules that is not supported by the Grenton system!</p>

## METHODS

Name	Description
RemoveBan	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> <code>RemoveBan</code> is not synonymous with the correct communication with the module again - it allows re-sending the command / inquiry to the module! In case of failure, the entire blocking process is restarted!</p>
ClearFailCount	Clears the number of unsuccessful communication attempts
Set	<p>Sets the value of a given configuration (parameter) register</p> <p><code>Register</code> (register or parameter number),</p> <p><code>Value</code> (the value of the register or parameter),</p> <p><code>Size</code> (size of the sent register or parameter value - in bytes)</p> <p><b>NOTE!</b> Calling the <code>Set</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after wake up the red LED will blink!</p>
Get	<p>Gets the value of a given configuration (parameter) register</p> <p><b>NOTE!</b> Calling the <code>Get</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after waking up the red LED will blink!</p>
SetDefault	<p>Sets the default value for a given configuration (parameter) register</p> <p><b>NOTE!</b> Calling the <code>SetDefault</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after wake up the red LED will blink!</p>

## EVENTS

Name	Description
OnBanned	An event triggered when the device is banned

## 7. Heiman Smart Smoke Sensor

Module version: **HS1SA-Z (HS1SA-Z HW: 255 FW: 1.10)**

### 7.1. General information

The Z-Wave Heiman Smart Smoke Sensor module allows reading: status of the smoke sensor and battery level. In addition, it gives you the option of setting / reading the module's wake-up time.

**NOTE!** Addition / removal is done by clicking the button three times in the HEIMAN module during inclusion / exclusion. Correctly carried out process will be confirmed by a five-fold blink of the diode.

**NOTE!** Module support available on CLU with firmware 04.07.41 (Build 183201) and newer.

## 7.2. Objects

### A. BINARY\_SENSOR

The object allows reading the status of the smoke sensor.

#### FEATURES

Name	Description
<code>Value</code>	Returns the input status: <b>0</b> - no violation, <b>1</b> - violation (smoke)

#### METHODS

-

#### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the status changes to the opposite
<code>OnSwitchOn</code>	An event triggered when the high state is set on input
<code>OnSwitchOff</code>	An event triggered when the low state is set on input

### B. ZWAVE\_BATTERY

The object allows reading the battery status. The reading takes place cyclically, every set time, for the `Interval` feature of the `ZWAVE_WAKEUP` object.

#### FEATURES

Name	Description
<code>BatteryLevel</code>	Battery level of the Z-Wave module (in percent)
<code>WarningLevel</code>	Battery level below which warning events are generated

#### METHODS

Name	Description
<code>SetWarningLevel</code>	Sets the warning level of the Z-Wave module battery

#### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the battery level changes
<code>OnLowBattery</code>	An event triggered when a battery drop is detected below the warning level
<code>OnBatteryGood</code>	An event triggered when the battery level returns to a value above the warning level

## C. ZWAVE\_WAKEUP

The object enables setting and reading the battery-awakening time of the Z-Wave module. The default value set by the CLU is 3600s (60 minutes). The minimum value is 300s (5 minutes); maximum 16777200s (about 194 days). It is possible to set values in step 60s (360s, 420s, 480s, etc.)

### FEATURES

Name	Description
<code>Interval</code>	The period of automatic awakening of the Z-Wave module from the sleep mode (in seconds)
<code>LastWakeUp</code>	Time of the last awakening of the Z-Wave module from sleep mode

### METHODS

Name	Description
<code>SetInterval</code>	Sets the period of automatic awakening of the Z-Wave module from the sleep mode

### EVENTS

Name	Description
<code>OnWakeUp</code>	An event triggered when the Z-Wave module wakes up from sleep mode

## D. ZWAVE\_CONFIG

The object displays information regarding communication parameters with the module in the Z-Wave network.

### FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:</p> <p>0 - communication with the module is not blocked,  1 - blocked communication with the module (module banned).</p> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the FailCount attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked (Banned = 1)

## METHODS

Name	Description
RemoveBan	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature Banned = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> RemoveBan is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</p>
ClearFailCount	Clears the number of unsuccessful communication attempts

## EVENTS

Name	Description
OnBanned	An event triggered when the device is banned

# 8. INFIBITY Siren Alarm [NEO Coolcam]

Module version: **NAS-AB01Z HW:48 FW: 2.90**



## 8.1. General information

Operation of the Infibity Siren Alarm module includes the option of switching on / off the siren signal, reading the battery level, as well as setting and reading of the module wake up. Additionally, it is possible to change the configuration parameters.

**How to add / remove:** Addition / removal is done by clicking the button three times in the INFIBITY module during inclusion / exclusion. Correctly carried out process will be confirmed by a five-fold blink of the diode.

**NOTE!** After CLU reboot (sending configuration), wait 10s before the first attempt to turn on the Siren Alarm module.

## 8.2. Objects

### A. ZWAVE\_DOUT

The object enables / disables and reads the current state of the siren.

#### FEATURES

Name	Description
<code>Value</code>	Returns the output state (0 - low, 1 - high)

#### METHODS

Name	Description
<code>SetValue</code>	Sets the output state as 1 or 0
<code>Switch</code>	Switches the output. The Time parameter determines how long the state change takes place, for 0 it is constant
<code>SwitchOn</code>	Turns on the output. The Time parameter determines how long the state change takes place, for 0 it is constant
<code>SwitchOff</code>	Turns off the output. The Time parameter determines how long the state change takes place, for 0 it is constant

#### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the status changes to the opposite
<code>OnSwitchOn</code>	An event triggered when the high state is set to output
<code>OnSwitchOff</code>	An event triggered when the low state is set to the output

## B. ZWAVE\_BATTERY

The object allows reading the battery status. The status read is done cyclically every set time for the Interval feature of the ZWAVE\_WAKEUP object

### FEATURES

Name	Description
BatteryLevel	Battery level of the Z-Wave module in percent
WarningLevel	Battery level below which warning events are generated

### METHODS

Name	Description
SetWarningLevel	Sets the warning level of the Z-Wave module battery

### EVENTS

Name	Description
OnChange	An event triggered when the device is banned
OnLowBattery	An event triggered when a battery drop is detected below the warning level
OnBatteryGood	An event triggered when the battery level returns to a value above the warning level

## C. ZWAVE\_WAKEUP

An object enabling setting and reading of the battery Wake Up time of the Z-Wave module. The default setting value for the CLU is 3600s (5 minutes). The minimum value is 60s (1 minute); maximum 16777200s (about 194 days).

### FEATURES

Name	Description
Interval	The period of self-awakening of the Z-Wave module from the sleep mode in seconds
LastWakeUp	Time of the last awakening of the Z-Wave module from sleep mode

### METHODS

Name	Description
SetInterval	Sets the period of automatic awakening of the Z-Wave module from the sleep mode

### EVENTS

Name	Description
OnWakeUp	An event triggered when the Z-Wave module wakes up from sleep mode

## D. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network. It also allows setting advanced configuration parameters of a given module (specified individually in the manual).

Setting register 7 changes the siren mode:

- As an **Alarm** - the siren operates according to the parameter settings: 1,2,5,8
- As a **DoorBell** - the siren operates according to the parameter settings: 3,4,6,9

## FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:</p> <ul style="list-style-type: none"> <li>0 - communication with the module is not blocked,</li> <li>1 - blocked communication with the module (module banned).</li> </ul> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the <code>FailCount</code> attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	<p>The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked (<code>Banned</code> = 1).</p> <p><b>NOTE!</b> After restarting the CLU, the <i>Switch Binary Switch</i> command is sent to the module, to which the module does not respond, so that <code>FailCount</code> is increased by 1.</p>
Register	The register number (parameter) of the configuration that has been read / set recently using the available methods
Value	The value of the configuration register (parameter)

## METHODS

Name	Description
RemoveBan	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> <code>RemoveBan</code> is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</p>
ClearFailCount	Clears the number of unsuccessful communication attempts
Set	<p>Sets the value of a given configuration register (parameter):</p> <p><code>Register</code> (register or parameter number),</p> <p><code>Value</code> (the value of the register or parameter),</p> <p><code>Size</code> (size of the sent register or parameter value - in bytes)</p> <p><b>NOTE!</b> <i>Calling the <code>Set</code> method must be done after waking up the battery module! In order to wake up the module, please click the button in the module three times - after wake up the red LED will blink!</i></p>
Get	Gets the value of a given configuration (parameter) register
SetDefault	Sets the default value for a given configuration (parameter) register

## EVENTS

Name	Description
OnBanned	An event triggered when the device is banned

## 9. Danfoss Living Connect

Module version: EU HW: 00 FW: 1.1

### 9.1. General information

The use of the Danfoss Living Connect module includes the possibility of setting the set temperature on the head, as well as switching on / off the key lock. It is also possible to read the battery level of the device and to define the module's wake-up period.

**How to add / remove:** To add / remove a device, 1x click the middle button on the module during inclusion / exclusion (called on the CLU) - the display backlight will blink quickly and then will turn on continuously. If after a long time of fast blinking the display backlight starts to blink slower, it means that the adding process has failed. Before adding the device, one must leave the assembly mode indicated by "M" in the display.

## 9.2. Objects

### A. ZWAVE\_THERMOSTAT

An object that allows setting the temperature on the head as well as switching on/off the key lock.

**NOTE!** Operation does not include reading the set temperature using the buttons on the head.

#### FEATURES

Name	Description
PointValue	Returns the set temperature value (4°C ÷ 28°C)
ProtectionState	Returns the key lock status: 0 - off, 2 - on

#### METHODS

Name	Description
SetPointValue	Sets the temperature (PointValue feature)
SetProtectionState	Sets the key lock status

#### EVENTS

Name	Description
OnPointValueChange	An event triggered when the temperature setpoint is changed
OnProtectionChange	An event triggered when the key lock state changes
OnProtectionOn	An event triggered when the key lock is activated
OnProtectionOff	An event triggered when the key lock is turned off

### B. ZWAVE\_BATTERY

The object allows reading the battery status. The status read is done cyclically every set time for the Interval feature of the ZWAVE\_WAKEUP object

#### FEATURES

Name	Description
BatteryLevel	Battery level of the Z-Wave module in percent
WarningLevel	Battery level below which warning events are generated

#### METHODS

Name	Description
<code>SetWarningLevel</code>	Sets the warning level of the Z-Wave module battery

## EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the device is banned
<code>OnLowBattery</code>	An event triggered when a battery drop is detected below the warning level
<code>OnBatteryGood</code>	An event triggered when a battery level returns to a value above the warning level

## C. ZWAVE\_WAKEUP

An object enabling setting and reading of the battery Wake Up time of the Z-Wave module. The default setting for the CLU is 300s (5 minutes). The minimum value is 60s (1 minute); maximum 1800s (30 minutes). It is possible to set the value in step 60s (60s, 120s, 180s, etc.)

## FEATURES

Name	Description
<code>Interval</code>	The period of self-awakening of the Z-Wave module from the sleep mode in seconds
<code>LastWakeUp</code>	Time of the last awakening of the Z-Wave module from sleep mode

## METHODS

Name	Description
<code>SetInterval</code>	Sets the period of automatic awakening of the Z-Wave module from the sleep mode

## EVENTS

Name	Description
<code>OnWakeUp</code>	An event that is triggered when the Z-Wave module wakes up from sleep mode

## D. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network.

## FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:</p> <ul style="list-style-type: none"> <li>0 - communication with the module is not blocked,</li> <li>1 - blocked communication with the module (module banned).</li> </ul> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the FailCount attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked (Banned = 1)

## METHODS

Name	Description
RemoveBan	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature Banned = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> RemoveBan is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</p>
ClearFailCount	Clears the number of unsuccessful communication attempts

## EVENTS

Name	Description
OnBanned	An event triggered when the device is banned

# 10. POPP Z-Weather

Module version: EU HW: 01 FW: 1.0

## 10.1. General information

Handling for the POPP Z-Weather module includes the ability to read climate parameters from the weather station. It is also possible to read the battery level of the device, as well as to define the module wake-up period.

**How to add / remove:** To add / remove the device, 3x click the button on the module within 1.5s during inclusion / exclusion (called on the CLU) - the red LED on the module will blink 3x when adding or 1x during deletion.

**How to wake up:** To wake up the device, click 1x on the device.

## 10.2. Objects

### A. ZWAVE\_WEATHER

An object enabling the reading of climatic parameters - temperature, luminance, relative humidity, wind speed, barometric pressure and dew point temperature.

#### FEATURES

Name	Description
Temperature	Returns the value of the measured air temperature (-10°C ÷ 60°C)
Luminance	Returns the value of the measured luminance (0% ÷ 100%)
Humidity	Returns the value of the measured relative humidity (0% ÷ 100%)
WindSpeed	Returns the value of the measured wind speed (0m/s ÷ 31m/s)
Pressure	Returns the value of the measured barometric pressure (600hPa ÷ 1200hPa)
DewPoint	Returns the value of the measured dew point temperature (-56,4°C ÷ 60°C)

#### EVENTS

Name	Description
OnTemperatureChange	An event triggered when the air temperature changes
OnLuminanceChange	An event triggered when the luminance value changes
OnHumidityChange	An event triggered when the relative humidity value changes
OnWindSpeedChange	An event triggered when the wind speed value changes
OnPressureChange	An event triggered when the barometric pressure value changes
OnDewPointChange	An event triggered when the dew point value changes

### B. ZWAVE\_BATTERY

The object allows reading the battery status. The status read is done cyclically every set time for the Interval feature of the ZWAVE\_WAKEUP object

#### FEATURES

Name	Description
BatteryLevel	Battery level of the Z-Wave module in percent
WarningLevel	Battery level below which warning events are generated



## METHODS

Name	Description
<code>SetWarningLevel</code>	Ustawia poziom ostrzegawczy baterii modułu Z-Wave

## EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the device is banned
<code>OnLowBattery</code>	An event triggered when a battery drop is detected below the warning level
<code>OnBatteryGood</code>	An event triggered when the battery level returns to a value above the warning level

## C. ZWAVE\_WAKEUP

An object enabling setting and reading of the battery Wake Up time of the Z-Wave module. The default setting for the CLU is 600s (about 10 minutes). The minimum value is 600s (about 10 minutes), maximum 17180s (about 286 minutes). It is possible to set the value in step 1s (600s, 601s, 602s, etc.)

## FEATURES

Name	Description
<code>Interval</code>	The period of self-awakening of the Z-Wave module from the sleep mode in seconds
<code>LastWakeUp</code>	Time of the last awakening of the Z-Wave module from sleep mode

## METHODS

Name	Description
<code>SetInterval</code>	Sets the period of automatic awakening of the Z-Wave module from the sleep mode

## EVENTS

Name	Description
<code>OnWakeUp</code>	An event triggered when the Z-Wave module wakes up from sleep mode

## D. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network.

## FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:</p> <p>0 - communication with the module is not blocked,  1 - blocked communication with the module (module banned).</p> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the FailCount attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked (Banned = 1)

## METHODS

Name	Description
RemoveBan	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature Banned = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> RemoveBan is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</p>
ClearFailCount	Clears the number of unsuccessful communication attempts

## EVENTS

Name	Description
OnBanned	An event triggered when the device is banned

# 11. FAKRO AMZ Solar

Module version: HW: 31 FW: 1.01:01.01

## 11.1. General information

Handling of the FAKRO AMZ Solar module includes the possibility of window control - both through the maximum opening / closing, as well as setting the window opening percentage, changing the operating mode (also seasonal mode), and defining the parameters operating in a given mode. In addition, it allows you to change the configuration parameters (Fakro configuration interface).

**How to add / remove:** Adding / removing the device is done by pressing the 'P' button on the device during inclusion / exclusion (called on the CLU).

## 11.2. Objects

### ZWAVE\_FAKRO

The object enables controlling the opening of the awning and reading the set opening percentage. It is possible to set the maximum value (opening / closing) as well as the percentage of the awning opening (0-100%). In addition, it is possible to set the device operating modes and parameters related to individual modes of operation.

**NOTE!** Information on specific modes of operation can be found in the device documentation provided by the manufacturer.

#### FEATURES

Name	Description
State	Device state: <code>0</code> - lack of movement, <code>1</code> - upward movement, <code>2</code> - downward movement
Percent	Percentage value of the awning opening, where: 0% - window closed, 100% - window opened  <b>NOTE!</b> The value of the <code>Percent</code> feature is refreshed when the awning controller completes the work - it should be taken into account when using this feature eg for the Slider component.
Mode	Device operation mode: <code>0 - Manual</code> - Manual, <code>1 - Semiauto</code> - Semiautomatic, <code>2 - Auto</code> - Automatic
SeasonMode	Seasonal mode of the device <code>0 - Summer</code> - Summer, <code>1 - Winter</code> - Winter  <b>NOTE!</b> Parameter does not apply to manual mode <code>Mode = 0</code>
OpeningTime	The awning opening time in semi-automatic mode
Sensitivity	The sensitivity of the sun exposure level for the awning in automatic mode

**NOTE!** The value of the set configuration parameters is refreshed at the time of `WakeUp` of the given device (values are taken from the Z-Wave device).

#### METHODS

Name	Description
Up	Awning up
Down	Awning down
Stop	Stop if the awning is in motion
Start	Awning up if previously move down, awning down if previously move up
SetPercent	Sets the percentage, where 100% - awning opened
SetMode	Sets the device's operating mode
SetSeasonMode	Sets the seasonal mode
SetOpeningTime	Sets the awning opening time
SetSensitivity	Sets the sensitivity of the sun exposure level

## EVENTS

Name	Description
OnChange	An event triggered when the window controller state changes
OnUp	An event will trigger at the time of changing from Stop to Up
OnDown	An event triggered when the state changes from Stop to Down
OnStart	An event triggered when the Start command is called
OnStop	An event triggered when the Stop command is issued

## ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network. It allows setting advanced configuration parameters of a given module (specified individually in the manual).

## FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:</p> <ul style="list-style-type: none"> <li>0 - communication with the module is not blocked,</li> <li>1 - blocked communication with the module (module banned).</li> </ul> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the FailCount attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked (Banned = 1)
Register	The register number (parameter) of the configuration that has been read / set recently using the available methods
Value	The value of the configuration register (parameter)

## METHODS

Name	Description
RemoveBan	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature Banned = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> RemoveBan it is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</p>
ClearFailCount	Clears the number of unsuccessful communication attempts
Set	<p>Sets the value of a given configuration register (parameter):</p> <ul style="list-style-type: none"> <li>Register (register or parameter number),</li> <li>Value (the value of the register or parameter),</li> <li>Size (size of the sent register or parameter value - in bytes)</li> </ul>
Get	Gets the value of a given configuration (parameter) register
SetDefault	Sets the default value for a given configuration (parameter) register

## EVENTS

Name	Description
OnBanned	An event triggered when the device is banned

# 12. FAKRO ARF

## 12.1. General information

Operation of the FAKRO ARF module includes the option of controlling the roller - both the maximum opening / closing and the setting of the opening percentage of the roller.

**How to add / remove:** Adding / removing the device is done by pressing the 'P' button on the device during inclusion / exclusion (called on the CLU).

## 12.2. Objects

### A. ZWAVE\_FAKRO

An object that allows you to control the roller and read the set percentage of opening. It is possible to set the maximum value (opening / closing) as well as giving the percentage of the roller opening (0-100%).

#### FEATURES

Name	Description
State	Roller state: 0 - Lack of movement 1 - upward movement 2 - downward movement
Percent	The opening percentage of the roller, where: 0% - roller closed, 100% - roller opened  <b>NOTE!</b> The value of the <code>Percent</code> feature is refreshed when the roller completes the work - it should be taken into account when using this feature eg for the Slider component. <b>NOTE!</b> Calling the <code>Stop</code> method while roller is in movement does not refresh the <code>Percent</code> feature

#### METHODS

Name	Description
Up	Roller upward
Down	Roller downward
Stop	Stop, if roller is in movement
Start	Roller up If previously move down, roller down If previously move up
SetPercent	Sets the percentage, where 100% - roller opened

#### EVENTS

Name	Description
<code>OnChange</code>	An event triggered when the roller state is changed
<code>OnUp</code>	An event triggered when the state changes from Stop to Up
<code>OnDown</code>	An event triggered when the state changes from Stop to Down
<code>OnStart</code>	An event triggered when the Start command is called
<code>OnStop</code>	An event triggered when the Stop command is issued

## B. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network.

### FEATURES

Name	Description
<code>NodeID</code>	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
<code>Banned</code>	<p>Information on blocking Z-Wave communication with the module:</p> <ul style="list-style-type: none"> <li><code>0</code> - communication with the module is not blocked,</li> <li><code>1</code> - blocked communication with the module (module banned).</li> </ul> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the <code>FailCount</code> attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
<code>FailCount</code>	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked ( <code>Banned</code> = 1)

### METHODS

Name	Description
<code>RemoveBan</code>	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> <code>RemoveBan</code> is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</p>
<code>ClearFailCount</code>	Clears the number of unsuccessful communication attempts

### EVENTS

Name	Description
OnBanned	An event triggered when the device is banned

## 13. FAKRO FTP\_V

Module version: HW: 25 FW: 1.01:01.01\*\*

### 13.1. General information

FAKRO FTP\_V module support includes window control - both through maximum opening / closing and setting the percentage of window opening.

**How to add / remove:** Adding / removing the device is done by pressing the 'P' button on the device during inclusion / exclusion (called on the CLU).

### 13.2. Objects

#### A. ZWAVE\_FAKRO

An object that allows you to control the opening of the window and read the set percentage of opening. It is possible to set the maximum value (opening / closing), and also to give the window's opening percentage (0-100%).

#### FEATURES

Name	Description
State	Device state: <input type="text" value="0"/> - Lack of movement, <input type="text" value="1"/> - opening, <input type="text" value="2"/> - closing
Percent	The percentage of window opening where: <input type="text" value="0%"/> - window closed, <input type="text" value="100%"/> - window opened  <b>NOTE!</b> The value of the <input type="text" value="Percent"/> feature is refreshed when the window controller finishes the work - it should be taken into account when using this feature eg for the Slider component.
WaterSensor	Value from the rain sensor

#### METHODS



Name	Description
Open	Opening the window
Close	Closing the window
Stop	Stop if the window is being opened or closed
Start	Closing the window if it was previously opened, opening the window if it was previously closed
SetPercent	Sets the percentage, where 100% - the window is open

## EVENTS

Name	Description
OnChange	An event triggered when the window controller state changes
OnOpen	An event triggered when the state changes from Stop to Open
OnClose	An event triggered when the state changes from Stop to Close
OnStart	An event triggered when the Start command is called
OnStop	An event triggered when the Stop command is called
OnRainChange	An event triggered when the sensor state changes to the opposite one
OnRainOn	An event triggered when the high state is set on the sensor
OnRainOff	An event triggered when the low state is set on the sensor

## B. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network.

## FEATURES

Name	Description
<code>NodeID</code>	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
<code>Banned</code>	Information on blocking Z-Wave communication with the module: <code>0</code> - communication with the module is not blocked, <code>1</code> - blocked communication with the module (module banned).  The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the <code>FailCount</code> attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module
<code>FailCount</code>	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked ( <code>Banned</code> = 1)
<code>Register</code>	The register number (parameter) of the configuration that has been read / set recently using the available methods
<code>Value</code>	The value of the configuration register (parameter)

## METHODS

Name	Description
<code>RemoveBan</code>	It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module. <b>NOTE!</b> <code>RemoveBan</code> is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!
<code>ClearFailCount</code>	Clears the number of unsuccessful communication attempts
<code>Set</code>	Sets the value of a given configuration register (parameter): <code>Register</code> (register or parameter number), <code>Value</code> (the value of the register or parameter), <code>Size</code> (size of the sent register or parameter value - in bytes)
<code>Get</code>	Gets the value of a given configuration (parameter) register
<code>SetDefault</code>	Sets the default value for a given configuration (parameter) register

## EVENTS

Name	Description
<code>OnBanned</code>	An event triggered when the device is banned

# 14. Fibaro RGBW

Module version: *FGRGBWM-441 v2/5 EU*

## 14.1. General information

The Z-Wave Fibaro RGBW module allows you to read and set the status of individual R, G, B, W output channels in the range from 0 to 255. In addition, it allows you to change the configuration parameters (Fibaro configuration interface).

## 14.2. Objects

### A. ZWAVE\_RGBW\_LED

The object enables setting values (0-255) for individual output channels R, G, B, W. It is also possible to read these values - e.g. set directly from the button connected to the module.

**NOTE!** The value from the attached button is sent when it is released or brought to the minimum / maximum value!

### FEATURES

Name	Description
Red	The value of the R component (0-255) - red
Green	The value of the G component (0-255) - green
Blue	The value of the B component (0-255) - blue
White	The value of the W component (0-255) - white
RampTime	Rise / fall time of the dimmer value change in milliseconds. The value of this feature affects the actions triggered by CLU - it does not affect the rise / fall time after pressing the buttons connected directly to the module

### METHODS

Name	Description
SetRed	Sets the value of the R component (0-255) - red
SetGreen	Sets the value of the G component (0-255) - green
SetBlue	Sets the value of the B component (0-255) - blue
SetWhite	Sets the value of the W component (0-255) - white
SetRampTime	Sets the rise / fall time of the dimmer value change

### EVENTS

Name	Description
OnChange	An event dispatched when the dimmer value is changed
OnSwitchOn	An event dispatched when the dimmer is switched on
OnSwitchOff	An event dispatched when the dimmer is switched off

## B. ZWAVE\_CONFIG

The object displays information regarding parameters and communication with the module in the Z-Wave network. It allows you to set advanced configuration parameters for a given module (specified individually in the manual).

**NOTE!** For Fibaro RGBW modules already added to the project - the ZWAVE\_CONFIG object will be added only when the module is completely removed from the project and after CLU Discovery.

### FEATURES

Name	Description
NodeID	Module (node) number in the Z-Wave network (assigned for each Z-Wave module after adding it to the controller)
Banned	Information on blocking Z-Wave communication with the module: <code>0</code> - communication with the module is not blocked, <code>1</code> - blocked communication with the module (banned module). Blocking occurs when 3 consecutive attempts to communicate with the module fail (incrementing the <code>FailCount</code> feature by 3). A request is sent to the banned module every 1.5 minutes - if the CLU receives a response, then the blocking will be removed and it is possible to retry sending the order to the module
FailCount	Number of failed communication attempts with the Z-Wave module. In the event of communication failure with the module (no response, confirmation, etc.), the feature is incremented by 1, then the retry is attempted twice (in 15s intervals). In case of failure, communication with the module is blocked ( <code>Banned = 1</code> )
Register	Number of the configuration register (parameter) that has been recently read / set using the available methods
Value	Value of the configuration register (parameter)

### METHODS

Name	Description
RemoveBan	Removes the blocking of communication with the Z-Wave module (in case the <code>Banned</code> = 1 feature). Calling the method enables the command to be sent again to the module. <b>NOTE!</b> The <code>RemoveBan</code> feature is not synonymous with correct communication with the module again - it allows you to resend the command / query to the module! In case of failure the whole blocking process is restarted!
ClearFailCount	Clears the number of failed communication attempts
Set	Sets the value of a given configuration register (parameter): 1 - Register (register or parameter number), 2 - Value (register or parameter value), 3 - Size (size of sent register value or parameter - in bytes)
Get	Gets the value of the given configuration register (parameter)
SetDefault	Sets the default value for a given configuration register (parameter)

## EVENTS

Name	Description
OnBanned	An event dispatched when the device is banned

# 15. Remotec ZXT-120

Module version: ZXT-120EU V1.0

## 15.1. General information

The handling of the Remotec ZXT-120 module includes the possibility of learning and sending IR code, defining transmission parameters and reading the learning status of a given code by the device. It is also possible to define the module wake-up period.

**How to add / remove:** 1x click the *PROG* button in the module during inclusion / exclusion - the red LED will flash 1x and then it will turn on continuously.

**How to restore the device to factory settings:** hold down the *PROG* button on the device for 10 seconds. After about 5 seconds, the red LED will light up and then start blinking twice at the end of the process (about 10 seconds).

## 15.2. Description of device configuration

The device can be configured in two ways:

1. Teaching your own IR codes
2. Use from the list of pre-defined codes available in the internal IR code library

## A. The way of teaching IR codes

1. Learning codes is done using the main object ZWAVE\_IR1
2. Call the method `SetAcDeviceNumber` with the parameter `AcDeviceNumber` equal to '0000' - sets the device in the mode of teaching new codes (outside the pre-defined list). After calling the method, the LED diode will blink 2x on the module.
3. Call the `LearnCode` method giving the IR code number from the range 0-22 under which we want the code to be saved. After calling the method, the LED on the device should go out and light up again.
4. Within 15 seconds, press and hold the remote control button that you want to learn by pointing the remote control towards the top of the device at a distance of 1-3 cm.
  - If the IR code is programmed correctly, the LED on the device should blink 2x.
  - In case of failure, the LED on the device should blink 6x.

The learning status can also be read from the `LearningStatus` parameter. In addition, appropriate events are generated depending on the learning status (`OnLearning`, `OnLearningOK`, `OnLearningFail`)

**NOTE!** The position of the remote control relative to the device during learning is crucial. It is recommended that the remote control is stationary relative to the device when the button is pressed. Incorrect position can cause the stored code to be incorrect despite the correct learning status.

**NOTE!** Memory of learned codes is saved after disconnecting the device's power supply. This memory is cleared after changing the AC device number and after removing the device from the Z-Wave network.

## B. The way of sending IR codes

1. Call the `SendCode` method specifying the number of the learned IR code from the range 0-22.
2. After calling the method, the LED on the device should go out and light up again and the assigned code is sent to the target device.

**NOTE!** The external transmitter has very low power and a small angle of light, so they should be placed near the IR receiver of the controlled device and properly directed. The light direction of the IR transmitters is consistent with the axis of the cable entering the IR transmitter housing.

**NOTE!** It is recommended not to change the AC device number (`AcDeviceNumber` feature) if you do not use the internal IR code of the device.

## 15.3. Objects

## A. ZWAVE\_IR

The object allows reading and writing of configuration parameters and sending IR codes.

### FEATURES

Name	Description
<code>AcDeviceNumber</code>	Returns the number of the AC device from the internal library of IR codes (number from the ZXT-120 Code List)
<code>EmitterPower</code>	Returns the power of the external (connected) infrared transmitter: <code>0</code> - normal power <code>255</code> - high power
<code>LearningStatus</code>	Returns the status of learning the IR codes: <code>0</code> - IR channel idle, <code>1</code> - learning successful, <code>2</code> - the learning procedure is in progress, <code>4</code> - learning failed
<code>SurroundIrControl</code>	Multidirectional IR signal transmission: <code>0</code> - Disabled, <code>255</code> - Enabled

**NOTE!** The value of the set configuration parameters is refreshed at the time of `WakeUp` of the given device (values are taken from the Z-Wave device). For the time of configuring the device parameters ( `SetAcDeviceNumber` , `SetEmitterPower` , `SetSurroundIrControl` ) and correct reading of the set features, it is possible to set the `WakeUpInterval` time for less than 60s. After making changes and completing the configuration of the above parameters, change the waking time to at least 60s.

### METHODS

Name	Description
<code>SendCode</code>	Sends an IR code with a specific number (code number in the range 0-22, learned or available in the internal IR code library for a given AC device)
<code>LearnCode</code>	Invokes the learning mode of the IR code with a specific number (code number in the range 0-22)
<code>SetAcDeviceNumber</code>	Sets the AC device number from the internal IR code library (number from the ZXT-120 Code List)
<code>SetEmitterPower</code>	Sets the power of the external infrared transmitter
<code>SetSurroundIrControl</code>	Sets the multidirection of the IR signal

### EVENTS

Name	Description
OnIrSend	An event triggered when the IR code is sent
OnLearningStatusChange	An event triggered when the status of the IR code learning mode changes
OnLearningOK	An event triggered when the status of learning the IR code changes to "OK"
OnLearning	An event triggered when the IR learning mode status changes to "Learning"
OnLearningFail	An event triggered when the IR learning mode status changes to "Learning Fail"

## B. ZWAVE\_BATTERY

The object allows reading the battery status. The status read is done cyclically every set time for the Interval feature of the ZWAVE\_WAKEUP object

### FEATURES

Name	Description
BatteryLevel	Battery level of the Z-Wave module in percent
WarningLevel	Battery level below which warning events are generated

### METHODS

Name	Description
SetWarningLevel	Sets the warning level of the Z-Wave module battery

### EVENTS

Name	Description
OnChange	An event triggered when the device is banned
OnLowBattery	An event triggered when a battery drop is detected below the warning level
OnBatteryGood	An event triggered when the battery level returns to a value above the warning level

## C. ZWAVE\_WAKEUP

An object that allows setting and reading the reading time of the Z-Wave module parameters. The default setting value for the CLU is 3600s (60 minutes). The minimum value is 10s, maximum 16777200s (about 194 days). It is possible to set the value in step 5s.



**NOTE!** It is not recommended to set the value of the `WakeUp` feature less than 60s during normal device operation. Decreasing the value may be useful only in the case of 'teaching' codes by the device (generating changes in the status of learning mode, as well as reading the `LearningStatus` feature), as well as in setting configuration parameters

## FEATURES

Name	Description
<code>Interval</code>	The period of self-awakening of the Z-Wave module from the sleep mode in seconds
<code>LastWakeUp</code>	Time of the last awakening of the Z-Wave module from sleep mode

## METHODS

Name	Description
<code>SetInterval</code>	Sets the period of automatic awakening of the Z-Wave module from the sleep mode

## EVENTS

Name	Description
<code>OnWakeUp</code>	An event triggered when the Z-Wave module wakes up from sleep mode

## D. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network.

## FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information of blocking Z-Wave communication with the module:</p> <p>0 - communication with the module is not blocked,  1 - blocked communication with the module (module banned).</p> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the FailCount feature by 3). A query is sent to the banned module every 1.5 minutes - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked (Banned = 1)

## METHODS

Name	Description
RemoveBan	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature Banned = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> RemoveBan it is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</p>
ClearFailCount	Clears the number of unsuccessful communication attempts

## EVENTS

Name	Description
OnBanned	An event triggered when the device is banned

# 16. Remotec ZXT-310

Module version: ZXT-310EU HW: 00 FW: 1.10

## 16.1. General information

The support of the ZXT-310 Remotec module includes handling for learning and sending IR code, defining transmission parameters and reading the learning status of a given code by the device. It is also possible to define the module's wake-up period.

**How to add / remove:** 1x click the *PROG* button in the module during inclusion / exclusion - the red LED will flash 1x and then will turn on continuously. If the LED blinks 6x, it means that the adding process has failed.

**How to restore the device to the factory settings:** hold down the *PROG* button device for 10 seconds. After the procedure, the red LED should turn off and turn on again.

Port 1 is the internal IR LEDs of the device. Ports 2-6 are the external IR ports of the device, to which the cables connected to the set with IR transmitters are connected.

## 16.2. Device configuration

### A. The way of teaching IR codes

1. Learning codes is done using the main object ZWAVE\_IR1
2. Select the Endpoint to which codes will be assigned by calling the `SetEndpointNumber` method. Each Endpoint has a representation in the form of an object (ZWAVE\_IR\_EP1, ... ZWAVE\_IR\_EP6)
3. Call the `LearnCode` method giving the IR code number between 1-384 at which we want the code to be saved. After calling the method, the LED on the device should turn off and light up again.
4. Within 15 seconds, press and hold the button on the remote control that you want to learn by pointing the remote control towards the "L" mark on the unit's casing at a distance of 1-3 cm.
  - If the IR code is programmed correctly, the LED on the device should blink 2x.
  - In case of failure, the LED on the device should blink 6x.

The learning status can also be read from the `LearningStatus` parameter. In addition, appropriate events are generated depending on the learning status (`OnLearning`, `OnLearningOK`, `OnLearningFail`, `OnCommandFull`)

Learning codes must be done for each endpoint separately. The maximum number of codes you can remember is  $6 * 64$ .

**NOTE!** The position of the remote control relative to the device during learning is crucial. It is recommended that the remote control is stationary relative to the device when the button is pressed. Incorrect position can cause the stored code to be incorrect despite the correct learning status.

**NOTE!** Memory of learned codes is saved after disconnecting the device's power supply. This memory is cleared after changing the AV device number and after removing the device from the Z-Wave network.

### B. The method of sending IR codes

1. Call the `SendCode` method specifying the number of the learned IR code from 1-384.
2. After calling the method, the LED on the device should go out and light up again, and the assigned code is sent to the target device.

**NOTE!** Sending codes can be performed for each of the six endpoints directly by selecting one of the ZWAVE\_IR\_EP objects or indirectly by selecting the ZWAVE\_IR object and configuring the endpoint number.

## C. Endpoints configuration

Endpoints (ZWAVE\_IR\_EP1, ZWAVE\_IR\_EP2, itd.) can be configured in two ways:

- indirectly through a common ZWAVE\_IR object - in this case, first set the endpoint number, which will be configured using the `SetEndpointNumber` method.
- directly through individual ZWAVE\_IR\_EP objects coherent to individual endpoints. For a common ZWAVE\_IR object

You can assign a different IR port to each endpoint. There are 6 IR ports available. By default, port 1 is assigned to all endpoints. Port 1 is the device's internal IR LEDs. Ports 2-6 are the external IR ports of the device, to which the cables connected to the set with IR transmitters are connected.

After assigning an IR port to a given endpoint, you can set other parameters such as IR power (external transmitters only) and transmission mode.

**NOTE!** External transmitters have very low power and a small lighting angle, so they should be available near the IR receiver of the controlled device and properly directed. The light direction of the IR transmitters is coherent with the axis of the cable entering the IR transmitter housing.

**NOTE!** It is recommended not to change the AV device number (feature `AvDeviceNumber` ) if you do not use the internal IR code of the device.

## 16.3. Objects

### A. ZWAVE\_IR

The object enables reading and writing configuration parameters of the previously selected endpoint and sending IR codes via this defined endpoint.

#### FEATURES

Name	Description
PortRouting	Returns the IR port number assigned to the currently selected endpoint (1 - internal IR port, 2 ÷ 6 - external IR ports)
AvDeviceNumber	Returns the number of the AV device from the internal IR code library assigned to the currently selected endpoint (four-digit number from the ZXT-310 Code List)
EmitterPower	Returns the power of the external infrared transmitter to the set IR port: 0 - normal power 255 - high power  <b>NOTE!</b> Parameter <code>EmmitterPower</code> it is not configurable for port 1
TransmissionMode	Returns the IR code transmission mode: 0 - continuous transmission, 255 - single pulse
EndpointNumber	Returns the number of the controlled endpoint (1 ÷ 6)
FirmwareVersion	Returns the version number of the software
LibraryVersion	Returns the version number of the built-in IR code library
LearningStatus	Returns the status of learning IR codes: 0 - IR channel idle, 1 - learning successful, 2 - learning procedure on progress, 3 - the maximum number of codes for a given Endpoint has been reached, 4 - learning failed

**NOTE!** The value of the set configuration parameters is refreshed at the time of `WakeUp` of the given device (values are taken from the Z-Wave device). For the time of configuration of the device parameters (`SetAvDeviceNumber`, `SetEmitterPower`, `SetTransmissionMode`, `SetPortRouting`) and correct reading of set features, it is possible to set the `WakeUpInterval` time for less than 60s. After making changes and completing the configuration of the above parameters, change the waking time to at least 60s.

## METHODS

Name	Description
<code>SendCode</code>	Sends an IR code with a specific number (code number in the 1-384 range, learned or available in the internal IR code library for the given AV device)
<code>LearnCode</code>	Invokes the learning mode of the IR code with a specific number (code number in the 1-384 range)
<code>SetPortRouting</code>	Sets the IR port number to be assigned to the currently selected endpoint
<code>SetAvDeviceNumber</code>	Sets the AV device number from the internal IR code library assigned to the currently selected endpoint (four-digit number from the ZXT-310 Code List)
<code>SetEmitterPower</code>	Sets the power of the external infrared transmitter  <b>NOTE!</b> Parameter <code>EmitterPower</code> is not configurable for port 1
<code>SetTransmissionMode</code>	Sets the IR code transmission mode
<code>SetEndpointNumber</code>	Sets the endpoint number to be controlled (1 ÷ 6)

## EVENTS

Name	Description
<code>OnIrSend</code>	An event triggered when the IR code is sent
<code>OnLearningStatusChange</code>	An event triggered when the status of the IR code learning mode changes
<code>OnLearningOK</code>	An event triggered when the status of learning IR code changes to "OK"
<code>OnLearning</code>	An event triggered when the IR learning mode status changes to "Learning"
<code>OnLearning</code>	An event triggered when the IR learning mode status changes to "Command Full"
<code>OnLearningFail</code>	An event triggered when the IR learning mode status changes to "Learning Fail"

## B. ZWAVE\_IR\_EP

The object enables direct reading and writing of endpoint configuration parameters to which it relates, as well as sending IR codes via this endpoint. By default, port 1 is assigned to all endpoints (the value of the `PortRouting` attribute).

**NOTE!** In order for each subsequent object (ZWAVE\_IR\_EP1, ZWAVE\_IR\_EP2, etc.) to refer to the next port of the device (1-6), the `PortRouting` feature should be set first, for example:  
ZWAVE\_IR\_EP1 - `PortRouting`: 1 ZWAVE\_IR\_EP2 - `PortRouting`: 2 ... ZWAVE\_IR\_EP6 - `PortRouting`: 6

then send the configuration.

## FEATURES

Name	Description
<code>PortRouting</code>	Returns the number of the IR port assigned to the endpoint (1 - internal IR port, 2 ÷ 6 - external IR ports)
<code>AvDeviceNumber</code>	Returns the number of the AV device from the internal IR code library assigned to the endpoint (four-digit number from the ZXT-310 Code List)
<code>EmitterPower</code>	Returns the power of the external infrared transmitter to the set IR port: <code>0</code> - normal power <code>255</code> - high power  <b>NOTE!</b> The <code>EmmitterPower</code> parameter is not configurable for port 1
<code>TransmissionMode</code>	Returns the IR code transmission mode: <code>0</code> - continous transmission, <code>255</code> - single pulse

## METHODS

Name	Description
<code>SendCode</code>	Sends an IR code with a specific number (code number 1-465, learned or available in the internal IR code library for the given AV device)
<code>SetPortRouting</code>	Sets the IR port number to be assigned to the currently selected endpoint
<code>SetAvDeviceNumber</code>	Sets the AV device number from the internal IR code library assigned to the currently selected endpoint (four-digit number from the ZXT-310 Code List)
<code>SetEmitterPower</code>	Sets the power of the external infrared transmitter  <b>NOTE!</b> Parameter <code>EmmitterPower</code> is not configurable for port 1
<code>SetTransmissionMode</code>	Sets the IR code transmission mode

## EVENTS

Name	Description
<code>OnIrSend</code>	An event triggered when the IR code is sent

## C. ZWAVE\_WAKEUP

An object that allows setting and reading the reading time of the Z-Wave module parameters. The default setting value for the CLU is 3600s (60 minutes). The minimum value is 10s, maximum 16777200s (about 194 days). It is possible to set the value in step 5s.

**NOTE!** It is not recommended to set the value of the `WakeUp` feature less than 60s during normal device operation. Decreasing the value can be useful in the case of 'teaching' codes by the device (generation of events changing the status of learning mode, as well as reading the `LearningStatus` feature), as well as setting configuration parameters.

### FEATURES

Name	Description
<code>Interval</code>	The period of self-awakening of the Z-Wave module from the sleep mode in seconds
<code>LastWakeUp</code>	Time of the last awakening of the Z-Wave module from sleep mode

### METHODY

Name	Description
<code>SetInterval</code>	Sets the period of automatic awakening of the Z-Wave module from the sleep mode

### EVENTS

Name	Description
<code>OnWakeUp</code>	An event triggered when the Z-Wave module wakes up from sleep mode

## D. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network.

### FEATURES



Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information of blocking Z-Wave communication with the module:  0 - communication with the module is not blocked,  1 - blocked communication with the module (module banned).</p> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the FailCount feature by 3). A query is sent to the banned module every 1.5 minutes - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In the case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 30s intervals). In case of failure, communication with the module is blocked ( Banned = 1)

## METHODS

Name	Description
RemoveBan	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature Banned = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> RemoveBan it is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</p>
ClearFailCount	Clears the number of unsuccessful communication attempts

## EVENTS

Name	Description
OnBanned	An event triggered when the device is banned

# 17. Aeotec Nano Switch

Module version: ZW116-C

## 17.1. General information

**NOTE!** The module service is available from CLU version 5.06.03 (build 2043).

Aeotec Nano Switch support includes the ability to turn on / off, permanently or for a specified period of time, the module output and read its status. The device can be controlled by the methods of the ZWAVE\_DOUT object or by means of external switches connected to the Nano Switch inputs. Their configuration (operation mode) is possible by changing the appropriate parameters specified individually in the module manual.

After Inclusion the module to the CLU Z-Wave , basic parameters configuring the operation of switches are set:

Parameter	Default Value
120 - external switch mode for S1	1 - 2-state switch mode
121 - external switch mode for S2	1 - 2-state switch mode

**Operation of S1, S2 inputs:** S1, S2 inputs directly control the module output:

- High state of input -> output switched on,
- Low state of input -> output switched off.

**How to add / remove:** Adding / removing the device is done by pressing button on the Nano Switch during Inclusion / Exclusion (called on the CLU).

**NOTE!** After CLU reboot (sending configuration), wait 10 seconds before the first attempt to turn on the Nano Switch module.

## 17.2. Objects

### A. ZWAVE\_DOUT

Object that enables the device to be turned on / off permanently or for a specified period of time and its current status to be read.

#### FEATURES

Name	Description
Value	Returns 1 for output set at On and 0 for output set at Off state

#### METHODS

Name	Description
SetValue	Sets output state to 1 or 0
Switch	Changes the output value from 0 to 1 or from 1 to 0. The first parameter is the time of change: 0 - switches output to continuous modespec number - switches output for a time specified by a parameter (in milliseconds)
SwitchOn	Sets output value to 1
SwitchOff	Sets output value to 0

#### EVENTS

Name	Description
OnChange	Occurs when a change in the state takes place (regardless of the value)
OnSwitchOn	Occurs when On (1) is set at output
OnSwitchOff	Occurs when Off (0) is set at output

## B. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network. It enables to set advanced configuration parameters of a given module.

**NOTE!** The change of configuration parameters is possible only after setting parameter 252 to 0 (Unlock), by default set to 1 (Lock).

### FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:</p> <ul style="list-style-type: none"> <li>0 - communication with the module is not blocked,</li> <li>1 - blocked communication with the module (module banned).</li> </ul> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the <code>FailCount</code> attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked ( <code>Banned</code> = 1)
Register	The register number (parameter) of the configuration that has been read / set recently using the available methods
Value	The value of the configuration register (parameter)

### METHODS

Name	Description
<code>RemoveBan</code>	It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module. <b>NOTE!</b> <code>RemoveBan</code> it is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!
<code>ClearFailCount</code>	Clears the number of unsuccessful communication attempts
<code>Set</code>	Sets the value of a given configuration register (parameter): <code>Register</code> (register or parameter number), <code>Value</code> (the value of the register or parameter), <code>Size</code> (size of the sent register or parameter value - in bytes)
<code>Get</code>	Gets the value of a given configuration (parameter) register
<code>SetDefault</code>	Sets the default value for a given configuration (parameter) register

## EVENTS

Name	Description
<code>OnBanned</code>	An event triggered when the device is banned

# 18. Aeotec Dual Nano Switch

Module version: ZW140-C

## 18.1. General information

**NOTE!** The module service is available from CLU version 5.06.03 (build 2043).

Aeotec Dual Nano Switch support includes the ability to turn on / off, permanently or for a specified period of time, the outputs of the module as well as read their status. The device can be controlled by the methods of the ZWAVE\_DOUT object or by means of external switches connected to the Dual Nano Switch inputs. Their configuration (operation mode) is possible by changing the appropriate parameters specified individually in the module manual.

After Inclusion the module to the CLU Z-Wave, basic parameters configuring the operation of switches are set:

Parameter	Default Value
<code>120</code> - external switch mode for S1	<code>1</code> - 2-state switch mode
<code>121</code> - external switch mode for S2	<code>1</code> - 2-state switch mode

**Operation of S1, S2 inputs:** S1, S2 inputs directly control the module outputs (OUT1, OUT2), respectively S1 -> OUT1, S2 -> OUT2:

- High state of input -> output switched on,
- Low state of input -> output switched off.

**How to add / remove:** Adding / removing the device is done by pressing button on the Dual Nano Switch during Inclusion / Exclusion (called on the CLU).

**NOTE!** After CLU reboot (sending configuration), wait 10 seconds before the first attempt to turn on the Dual Nano Switch module.

## 18.2. Objects

### A. ZWAVE\_DOUT

Object that enables the device to be turned on / off permanently or for a specified period of time and its current status to be read.

#### FEATURES

Name	Description
Value	Returns 1 for output set at On and 0 for output set at Off state

#### METHODS

Name	Description
SetValue	Sets output state to 1 or 0
Switch	Changes the output value from 0 to 1 or from 1 to 0. The first parameter is the time of change: 0 - switches output to continuous modespec number - switches output for a time specified by a parameter (in milliseconds)
SwitchOn	Sets output value to 1
SwitchOff	Sets output value to 0

#### EVENTS

Name	Description
OnChange	Occurs when a change in the state takes place (regardless of the value)
OnSwitchOn	Occurs when On (1) is set at output
OnSwitchOff	Occurs when Off (0) is set at output

### B. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network. It enables to set advanced configuration parameters of a given module.

**NOTE!** The change of configuration parameters is possible only after setting parameter 252 to 0 (Unlock), by default set to 1 (Lock).

## FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:            0 - communication with the module is not blocked,            1 - blocked communication with the module (module banned).</p> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the FailCount attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked ( Banned = 1)
Register	The register number (parameter) of the configuration that has been read / set recently using the available methods
Value	The value of the configuration register (parameter)

## METHODS

Name	Description
RemoveBan	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature Banned = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> RemoveBan it is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</p>
ClearFailCount	Clears the number of unsuccessful communication attempts
Set	Sets the value of a given configuration register (parameter): Register (register or parameter number), Value (the value of the register or parameter), Size (size of the sent register or parameter value - in bytes)
Get	Gets the value of a given configuration (parameter) register
SetDefault	Sets the default value for a given configuration (parameter) register

## EVENTS

Name	Description
OnBanned	An event triggered when the device is banned

## 19. Aeotec Nano Dimmer

Module version: ZW111-C

### 19.1. General information

**NOTE!** The module service is available from CLU version 5.06.03 (*build 2043*).

Aeotec Nano Dimmer support includes the ability to smoothly control the level of light intensity and read this value. Lighting can be controlled by the methods of the ZWAVE\_DIMMER object or by means of external switches connected to the Nano Dimmer inputs. Their configuration (operation mode) is possible by changing the appropriate parameters specified individually in the module manual.

After Inclusion the module to the CLU Z-Wave , basic parameters configuring the operation of switches are set:

Parameter	Default Value
120 - external switch mode for S1	3 - momentary push button mode
121 - external switch mode for S2	1 - 2-state switch mode.
125 - dimming rate in seconds	3

**Operation of S1 input:** S1 input directly controls the module output:

- Single click -> sets last memorized value of the dimmer / turns the dimmer off,
- Press and hold -> to increase or decrease the dimmer value.

**Operation of S2 input:** S2 input is associated with the ZWAVE\_DIN object, it does not control the module output. The exact operation can be found in the description of the ZWAVE\_DIN object.

**How to add / remove:** Adding / removing the device is done by pressing button on the Nano Dimmer during Inclusion / Exclusion (called on the CLU).

**NOTE!** After CLU reboot (sending configuration), wait 10 seconds before the first attempt to turn on the Nano Dimmer module.

### 19.2. Objects

#### A. ZWAVE\_DIMMER

Object that allows you to control the level of light intensity and read the current state of the device.

#### FEATURES

Name	Description
Value	Specifies the current output value
MinValue	Minimum value which Value can adopt
MaxValue	Maximum value which Value can adopt
RampTime	Delay value when changing illumination (in ms)

## METHODS

Name	Description
SetValue	Sets output value
SetMinValue	Setting the minimum value which can be adopted by an output. Attempting to set a lower value will generate an error
SetMaxValue	Setting the maximum value which can be adopted by an output. Attempting to set a higher value will generate an error
SetRampTime	Determinates the time of output value increment (in ms)
Switch	Changes the output value from 0 to 1 or from 1 to 0. The first parameter is the time of change: 0 - switches output to continous mode number - switches output for a specified time by a parameter( in milliseconds) The second parameter is the ramp (time of value increments which is optional. If this parameter is not specified, the default ramp is used
SwitchOn	Sets output value to 1. The first parameter is the time of switching (how long it is to be switched for). The second parameter is the ramp (time of value increments) which is optional
SwitchOff	Sets output value to 0. The first parameter is the time of switching (how long it is to be switched for). The second parameter is the ramp (time of value increments) which is optional
Hold	Executes the function of illuminating/ dimming
HoldUp	Executes the function of illuminating
HoldDown	Executes the function of dimming

## EVENTS



Name	Description
OnChange	Event resulting from changing the output state
OnSwitchOn	Event occurring when the output value is changed from 0 to a higher value than 0
OnSwitchOff	Event occurring when 0 is set at the output
OnValueRise	Event occurring when the set value is higher than the current value
OnValueDrop	Event occurring when the set value is lower than the current value

## B. ZWAVE\_DIN

The operation of the object is determined by the setting of the configuration parameter `121` of the Nano Dimmer module, which defines the operating mode for the S2 input:

- for the **2 - state switch mode**:
  - High state of input -> the embedded feature `Value` takes the value 1,
  - Low state of input -> the embedded feature `Value` takes the value 0.
- for the **Momentary push button mode**:
  - Singel click -> the embedded feature `Value` takes the value 1,
  - Single click again -> the embedded feature `Value` takes the value 0.

**NOTE!** Based on the descriptions of changes (presented above) to an embedded `Value` feature of ZWAVE\_DIN object, the configuration in the ZWAVE\_DIN -> ZWAVE\_DIMMER binding should be adjusted accordingly, in order to achieve the desired functionality (on / off, dimming). If you want to perform a standard configuration in the Grenton system to control the DIMMER object, an adequate input of DIN or TouchPanel module should be used.

## FEATURES

Name	Description
HoldDelay	Time in milliseconds after which, when pressing and holding a button, the OnHold event occurs
HoldInterval	Cyclical interval in milliseconds after which, when pressing and holding a button, the OnHold event occurs
Value	Returns input state as 0 or 1

## METHODS

Name	Description
SetHoldDelay	Sets <code>HoldDelay</code> value
SetHoldInterval	Sets <code>HoldInterval</code> value

## EVENTS

Name	Description
<code>OnChange</code>	Occurs when a change in the input state takes place (regardless of the value)
<code>OnSwitchOn</code>	Occurs when the high state is set at input
<code>OnSwitchOff</code>	Occurs when the low state is set at input
<code>OnShortPress</code>	Occurs after pressing the button for 500 - 2000ms
<code>OnLongPress</code>	Occurs after pressing the button for at least 2000ms
<code>OnHold</code>	Occurs for the first time after HoldDelay time and then cyclically every HoldInterval value
<code>OnClick</code>	Occurs after pressing the button for less than 500 ms

## C. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network. It enables to set advanced configuration parameters of a given module.

**NOTE!** The change of configuration parameters is possible only after setting parameter 252 to 0 (Unlock), by default set to 1 (Lock).

### FEATURES

Name	Description
<code>NodeID</code>	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
<code>Banned</code>	<p>Information on blocking Z-Wave communication with the module:</p> <ul style="list-style-type: none"> <li><code>0</code> - communication with the module is not blocked,</li> <li><code>1</code> - blocked communication with the module (module banned).</li> </ul> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the <code>FailCount</code> attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
<code>FailCount</code>	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked ( <code>Banned</code> = 1)
<code>Register</code>	The register number (parameter) of the configuration that has been read / set recently using the available methods
<code>Value</code>	The value of the configuration register (parameter)

### METHODS

Name	Description
<code>RemoveBan</code>	It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module. <b>NOTE!</b> <code>RemoveBan</code> <i>it is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</i>
<code>ClearFailCount</code>	Clears the number of unsuccessful communication attempts
<code>Set</code>	Sets the value of a given configuration register (parameter): <code>Register</code> (register or parameter number), <code>Value</code> (the value of the register or parameter), <code>Size</code> (size of the sent register or parameter value - in bytes)
<code>Get</code>	Gets the value of a given configuration (parameter) register
<code>SetDefault</code>	Sets the default value for a given configuration (parameter) register

## EVENTS

Name	Description
<code>OnBanned</code>	An event triggered when the device is banned

# 20. Aeotec Nano Shutter

Module version: ZW141-C

## 20.1. General information

**NOTE!** The module service is available from CLU version 5.06.03 (*build 2043*).

Aeotec Nano Shutter support includes the ability to control the module outputs (up / down / stop). The device can be controlled by the methods of the ZWAVE\_ROLLER\_SHUTTER object or by means of external switches connected to the Nano Shutter inputs. Their configuration (operation mode) is possible by changing the appropriate parameters specified individually in the module manual.

After Inclusion the module to the CLU Z-Wave, basic parameters configuring the operation of switches are set:

Parameter	Default Value
<code>35</code> - moving time from up to down in seconds	<code>11</code>
<code>120</code> - external switch mode for S1	<code>1</code> - 2-state switch mode
<code>121</code> - external switch mode for S2	<code>3</code> - momentary push button mode

**Operation of S1 input:** S1 input directly controls the module outputs:

- High state of input -> shutter up (OUT1 on) if previously down / shutter down (OUT2 on) if previously up,
- Low state of input -> shutter stopped, outputs off.

**Operation of S2 input:** S2 input directly controls the module outputs:

- Single click -> shutter up (OUT1 on) if previously down / shutter stopped, outputs off / shutter down (OUT2 on) if previously up.

**How to add / remove:** Adding the device is done by pressing button on the Nano Shutter during Inclusion, removing is done by pressing button 6 times during Exclusion (called on the CLU).

**NOTE!** After CLU reboot (sending configuration), wait 10 seconds before the first attempt to turn on the Nano Shutter module.

## 20.2. Objects

### A. ZWAVE\_ROLLER\_SHUTTER

Object enabling the roller shutter control (up / down / stop). The condition of the roller shutter is determined on the basis of the called methods.

**NOTE!** The object does not take over the information about the real state of the device controlled by external switches connected to inputs S1, S2.

**NOTE!** For the correct operation of the object, the `MaxTime` feature and the `35` configuration parameter of the Nano Shutter module should be set to the same value.

#### FEATURES

Name	Description
<code>OUT1</code>	State of OUT1 relay (moving upwards)
<code>OUT2</code>	State of OUT2 relay (moving downwards)
<code>State</code>	Output state: <code>0</code> - no movement, <code>1</code> - moving upwards, <code>2</code> - moving downwards
<code>MaxTime</code>	Default Time parameter value. 0 if not specified

#### METHODS

Name	Description
Up	Roller shutter up or STOP if moving. Parameter Time: <input type="text" value="number"/> - output is active for specified timer <input type="text" value="0"/> - output is active for the time specified in MaxTime
Down	Roller shutter down or STOP if moving. Parameter Time: <input type="text" value="number"/> - output is active for specified timer <input type="text" value="0"/> - output is active for the time specified in MaxTime
Start	Roller shutter up if the preceding motion was down or roller shutter down if the preceding motion was up. Parameter Time: <input type="text" value="number"/> - output is active for specified timer <input type="text" value="0"/> - output is active for the time specified in MaxTime
Stop	STOP if moving
Hold	Hold with direction change
HoldUp	Hold always up
HoldDown	Hold always down

## EVENTS

Name	Description
OnChange	Result from a change in the state of any of the outputs
OnUp	Occurs when changing the Stop state to the Up state
OnDown	Occurs when changing the Stop state to the Down state
OnStart	Occurs when Start is requested
OnStop	Occurs when Stop is requested

## B. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network. It enables to set advanced configuration parameters of a given module.

**NOTE!** The change of configuration parameters is possible only after setting parameter 252 to 0 (Unlock), by default set to 1 (Lock).

## FEATURES

Name	Description
<code>NodeID</code>	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
<code>Banned</code>	<p>Information on blocking Z-Wave communication with the module:  <code>0</code> - communication with the module is not blocked,  <code>1</code> - blocked communication with the module (module banned).</p> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the <code>FailCount</code> attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
<code>FailCount</code>	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked ( <code>Banned</code> = 1)
<code>Register</code>	The register number (parameter) of the configuration that has been read / set recently using the available methods
<code>Value</code>	The value of the configuration register (parameter)

## METHODS

Name	Description
<code>RemoveBan</code>	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> <code>RemoveBan</code> it is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</p>
<code>ClearFailCount</code>	Clears the number of unsuccessful communication attempts
<code>Set</code>	<p>Sets the value of a given configuration register (parameter):</p> <p><code>Register</code> (register or parameter number),  <code>Value</code> (the value of the register or parameter),  <code>Size</code> (size of the sent register or parameter value - in bytes)</p>
<code>Get</code>	Gets the value of a given configuration (parameter) register
<code>SetDefault</code>	Sets the default value for a given configuration (parameter) register

## EVENTS

Name	Description
<code>OnBanned</code>	An event triggered when the device is banned

# 21. Aeotec Nano Shutter (V2)

Module version: ZW141-C

## 21.1. General information

**NOTE!** The module service is available from CLU version 5.08.01.

Aeotec Nano Shutter support includes the ability to control the module outputs (up / down / stop). The device can be controlled by the methods of the ZWAVE\_ROLLER\_SHUTTER object or by means of external switches connected to the Nano Shutter inputs. Their configuration (operation mode) is possible by changing the appropriate parameters specified individually in the module manual.

**NOTE!** There may be numerous delays and anomalies in the execution of commands by the modules, especially when they are in the range of other Z-WAVE devices or devices using the radio frequency of 868.42 MHz such as gate remote controls.

After Inclusion the module to the CLU Z-Wave, basic parameters configuring the operation of switches are set:

Parameter	Default Value
35 - moving time from up to down in hundredths of a seconds	15000
120 - external switch mode for S1	1 - 2-state switch mode
121 - external switch mode for S2	3 - momentary push button mode

**Operation of S1 input:** S1 input directly controls the module outputs:

- High state of input -> shutter up (OUT1 on) if previously down / shutter down (OUT2 on) if previously up,
- Low state of input -> shutter stopped, outputs off.

**Operation of S2 input:** S2 input directly controls the module outputs:

- Single click -> shutter up (OUT1 on) if previously down / shutter stopped, outputs off / shutter down (OUT2 on) if previously up.

**How to add / remove:** Adding the device is done by pressing button on the Nano Shutter during Inclusion, removing is done by pressing button 6 times during Exclusion (called on the CLU).

**NOTE!** After CLU reboot (sending configuration), wait 10 seconds before the first attempt to turn on the Nano Shutter module.

## 21.2. Objects

## A. ZWAVE\_ROLLER\_SHUTTER

Object enabling the roller shutter control (up / down / stop). The condition of the roller shutter is determined on the basis of the called methods.

**NOTE!** The object does not take over the information about the real state of the device controlled by external switches connected to inputs S1, S2 or button on the module housing.

### FEATURES

Name	Description
OUT1	State of OUT1 relay (moving upwards)
OUT2	State of OUT2 relay (moving downwards)
State	Output state: 0 - no movement, 1 - moving upwards, 2 - moving downwards
Position	Percentage value of the shutter opening: 0% - fully closed, 100% - fully open.
MoveTime	The time in milliseconds it takes to fully open / close the blind

### METHODS

Name	Description
Up	Roller shutter up or STOP if moving.
Down	Roller shutter down or STOP if moving.
Start	Roller shutter up if the preceding motion was down or roller shutter down if the preceding motion was up.
Stop	STOP if moving
Hold	Hold with direction change
HoldUp	Hold always up
HoldDown	Hold always down

### EVENTS



Name	Description
OnChange	Result from a change in the state of any of the outputs
OnUp	Occurs when changing the Stop state to the Up state
OnDown	Occurs when changing the Stop state to the Down state
OnStart	Occurs when Start is requested
OnStop	Occurs when Stop is requested

## B. ZWAVE\_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network. It enables to set advanced configuration parameters of a given module.

**NOTE!** The change of configuration parameters is possible only after setting parameter 252 to 0 (Unlock), by default set to 1 (Lock).

### FEATURES

Name	Description
NodeID	The number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
Banned	<p>Information on blocking Z-Wave communication with the module:</p> <ul style="list-style-type: none"> <li>0 - communication with the module is not blocked,</li> <li>1 - blocked communication with the module (module banned).</li> </ul> <p>The blocking occurs when 3 consecutive attempts to communicate with the module fail (increment of the <code>FailCount</code> attribute by 3). A query is sent to the banned module every 1 minute - if the CLU receives a response, then the blocking will be removed and it is possible to try again to send the order to the module</p>
FailCount	The number of unsuccessful attempts to communicate with the Z-Wave module. In case of failure of communication with the module (no response, confirmation, etc.), the feature is incremented by 1, then the attempt to repeat is twice (in 10s intervals). In case of failure, communication with the module is blocked ( <code>Banned</code> = 1)
Register	The register number (parameter) of the configuration that has been read / set recently using the available methods
Value	The value of the configuration register (parameter)

### METHODS

Name	Description
<code>RemoveBan</code>	<p>It removes the blocking of communication with the Z-Wave module (in the case when the feature <code>Banned</code> = 1). Calling the method enables re-sending the command to the module.</p> <p><b>NOTE!</b> <code>RemoveBan</code> <i>it is not synonymous with the correct communication with the module again - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!</i></p>
<code>ClearFailCount</code>	Clears the number of unsuccessful communication attempts
<code>Set</code>	<p>Sets the value of a given configuration register (parameter):</p> <p><code>Register</code> (register or parameter number),</p> <p><code>Value</code> (the value of the register or parameter),</p> <p><code>Size</code> (size of the sent register or parameter value - in bytes)</p>
<code>Get</code>	Gets the value of a given configuration (parameter) register
<code>SetDefault</code>	Sets the default value for a given configuration (parameter) register

## EVENTS

Name	Description
<code>OnBanned</code>	An event triggered when the device is banned

# XVIII. MyGrenton mobile application

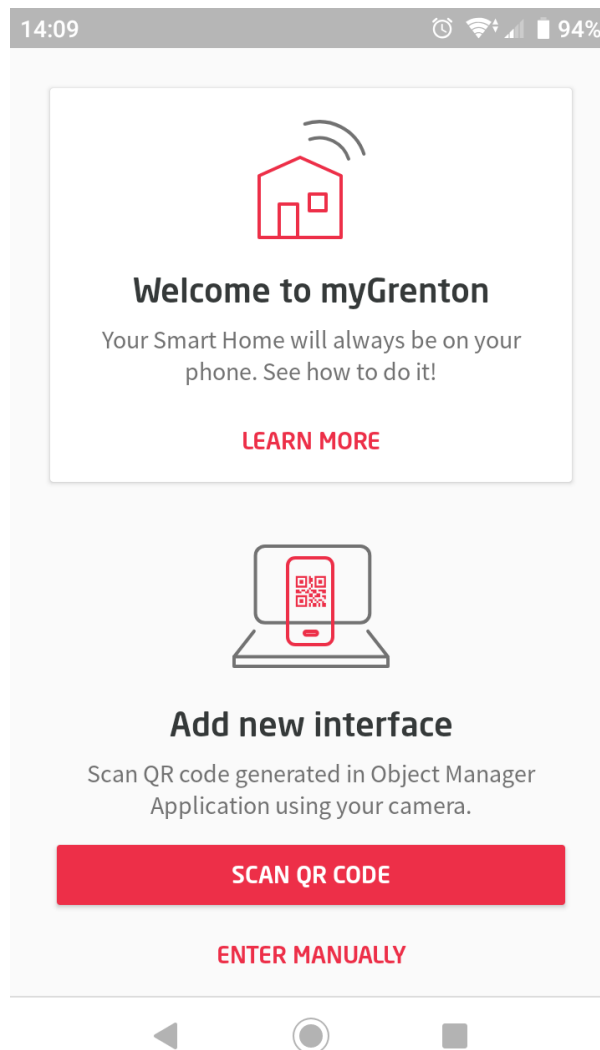
---

## 1. MyGrenton installation

---

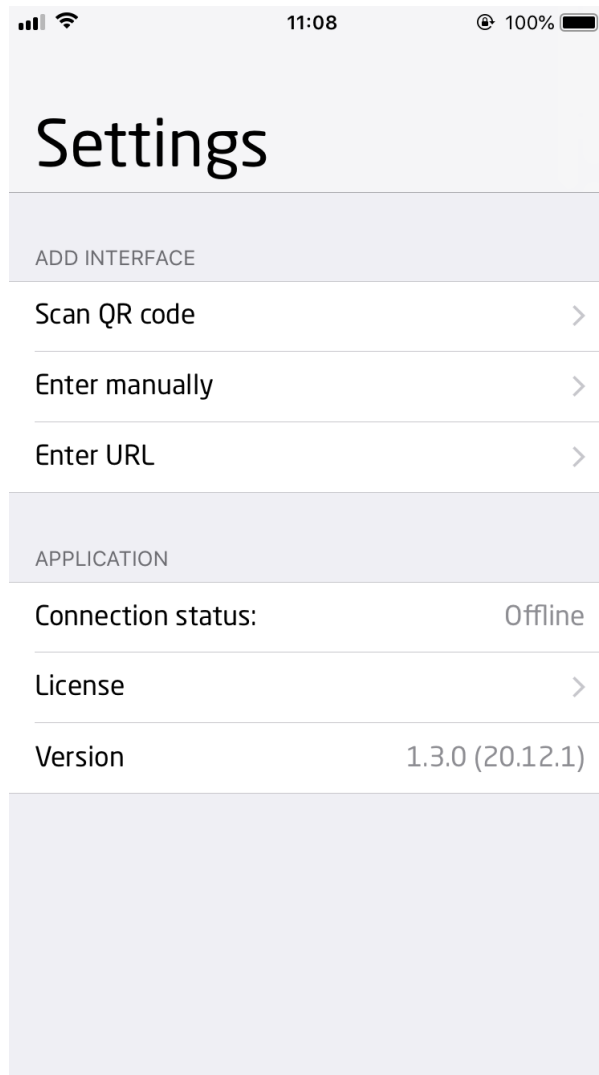
### A. Android

The current version of the application is available in the Play Store for phones or tablets with the **Android system (version 5.0 or higher)**. After opening the application, a welcome window with information about the application and the possibility of adding a new interface will appear.



### B. iOS

The current version of the application is available in the App Store for phones or tablets with the **iOS system (12 or higher)**. After opening the application, a welcome window with the possibility of adding a new interface will appear.



**NOTE!** Minimum version CLU required **05.03.06**

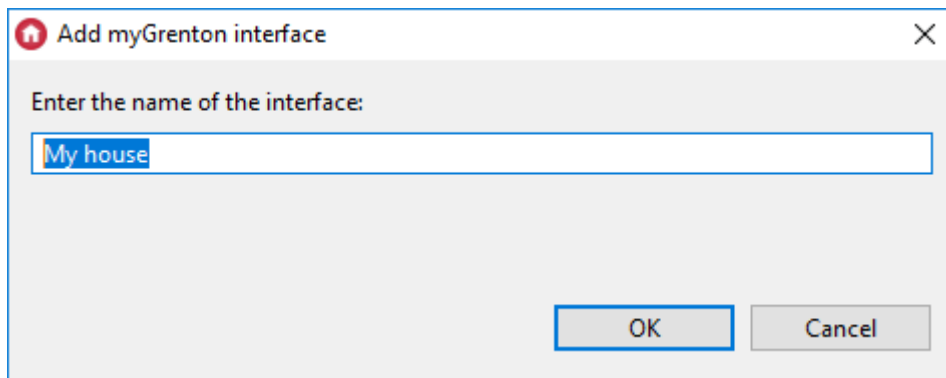
**NOTE!** The application may ask, among others for permission to take photos and videos, full access to the network, display network connections, prevent the phone from going to sleep, receive data from the Internet. For correct working of the application, you must agree to the above conditions.

## 2. Creating the interface

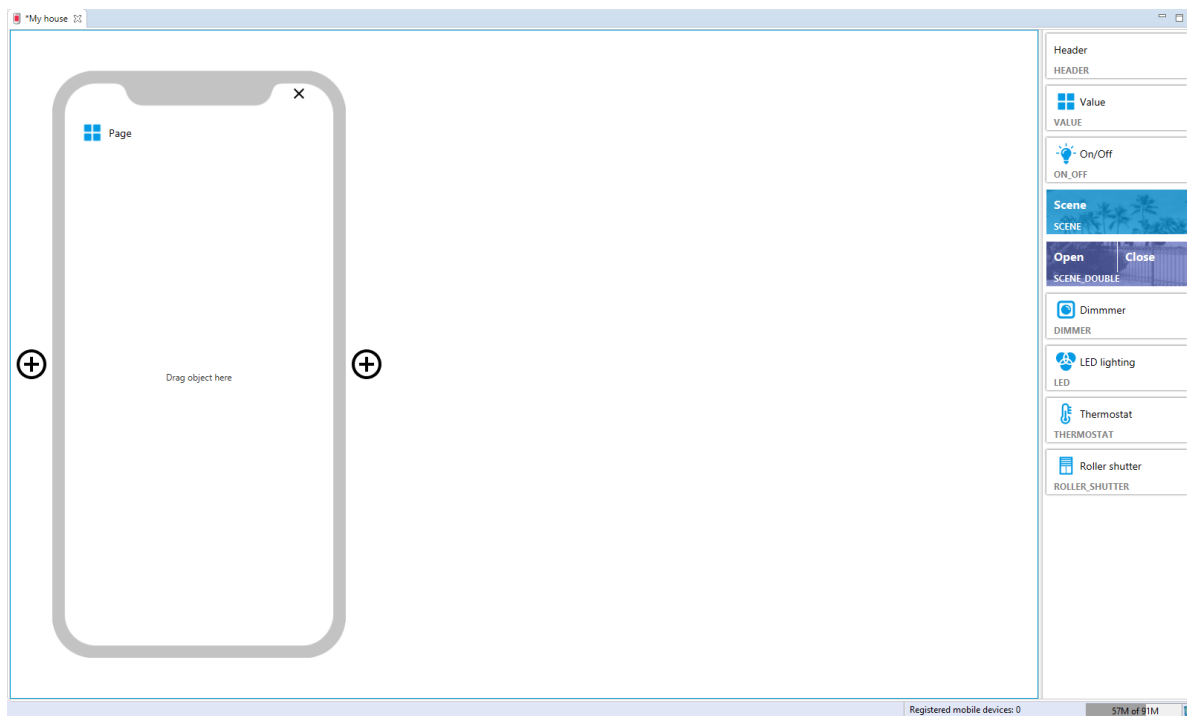
To create an interface to the myGrenton application, click on the 'Add myGrenton interface' icon in the Main Menu of the Object Manager:




Then a window will appear with the option to change the interface name:




After accepting the name, an empty interface will be created:



## 2.1. Adding a page to the interface

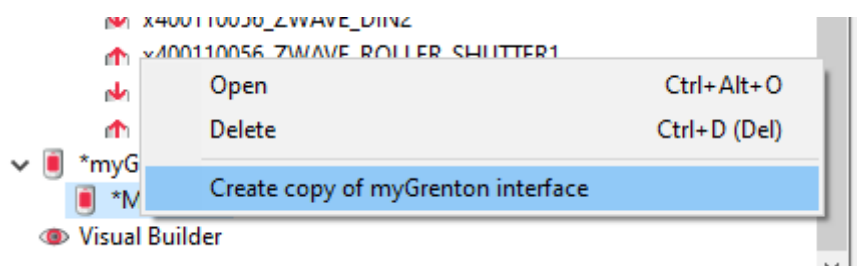
To add another interface page, click on the  icon next to the phone graphic. Up to 30 pages can be created.

## 2.2. Deleting a page from the interface

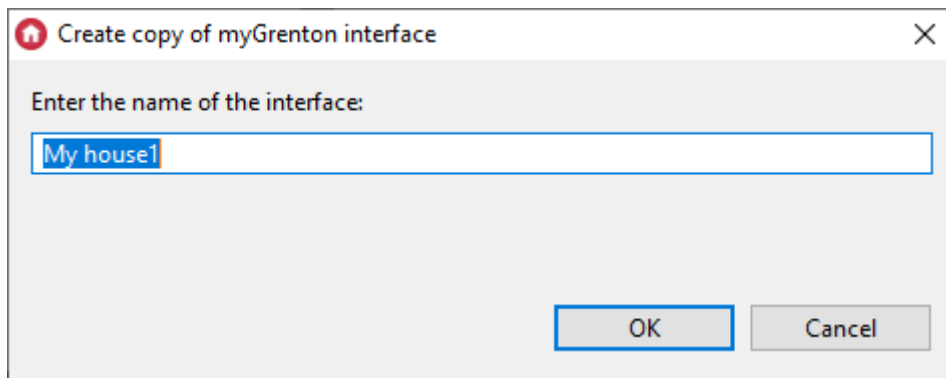
To remove a page from the interface, click on the  icon, which is located in the upper right corner of the phone and confirm the removal of the page.

## 2.3. Copying the interface

To copy an interface, right-click on the interface and select "Create copy of myGrenton interface" from the menu:



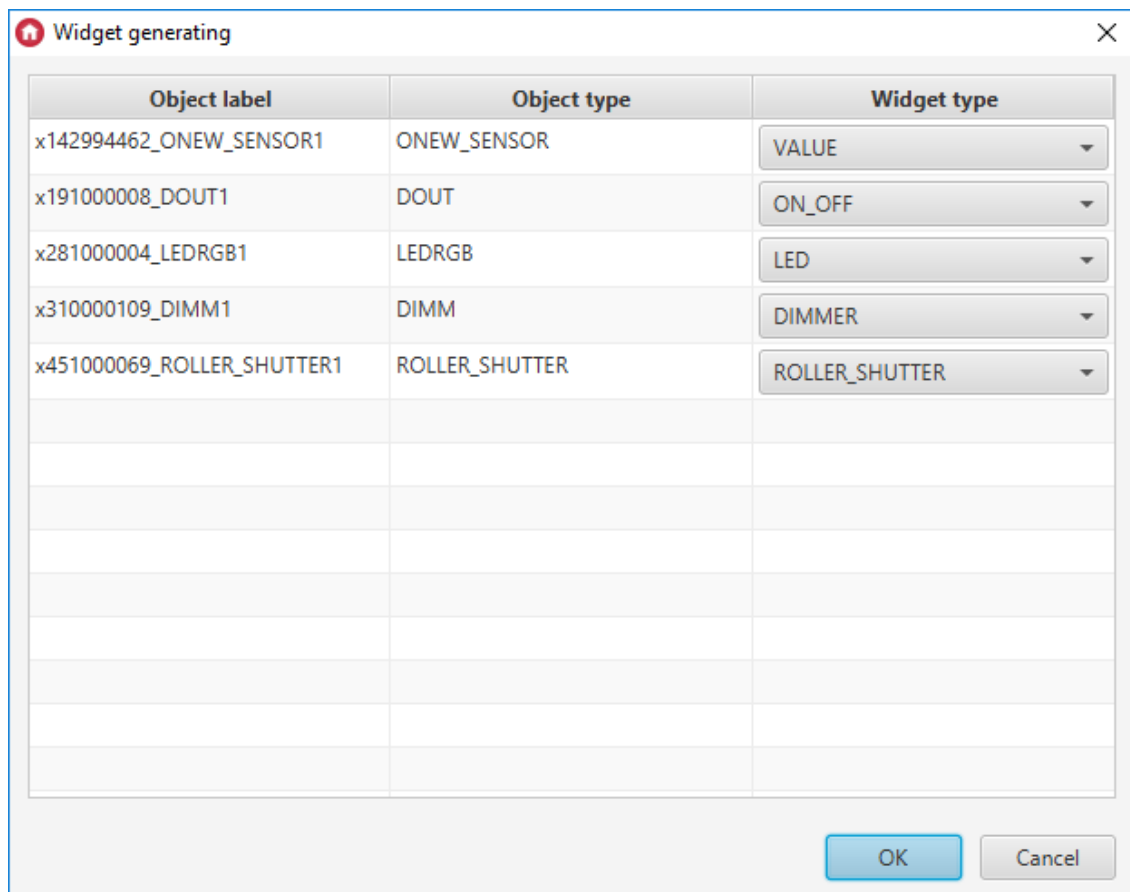
Then choose a name for the new interface:



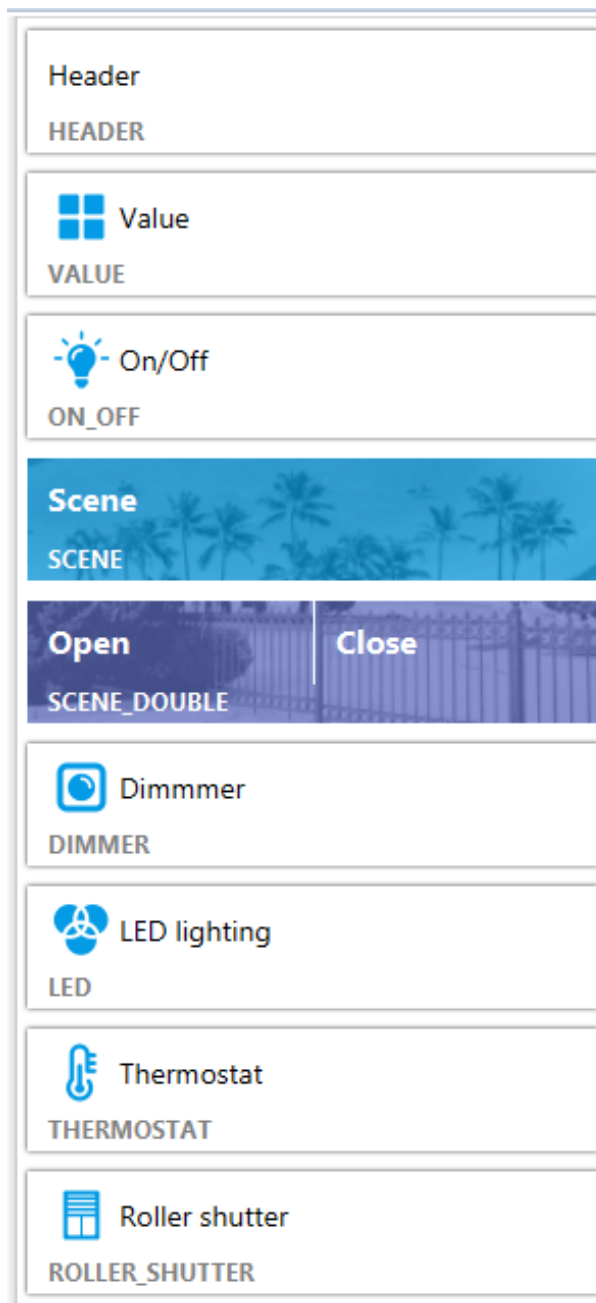
### 3. Widgets

To control the system using myGrenton application from the level of the phone or tablet, widgets are used. Each of them offers a different functionality. Widgets can be added to the interface in 2 ways:

1. Dragging a specific object from the list of objects in the Object Manager (then the created widget will have a predefined template)



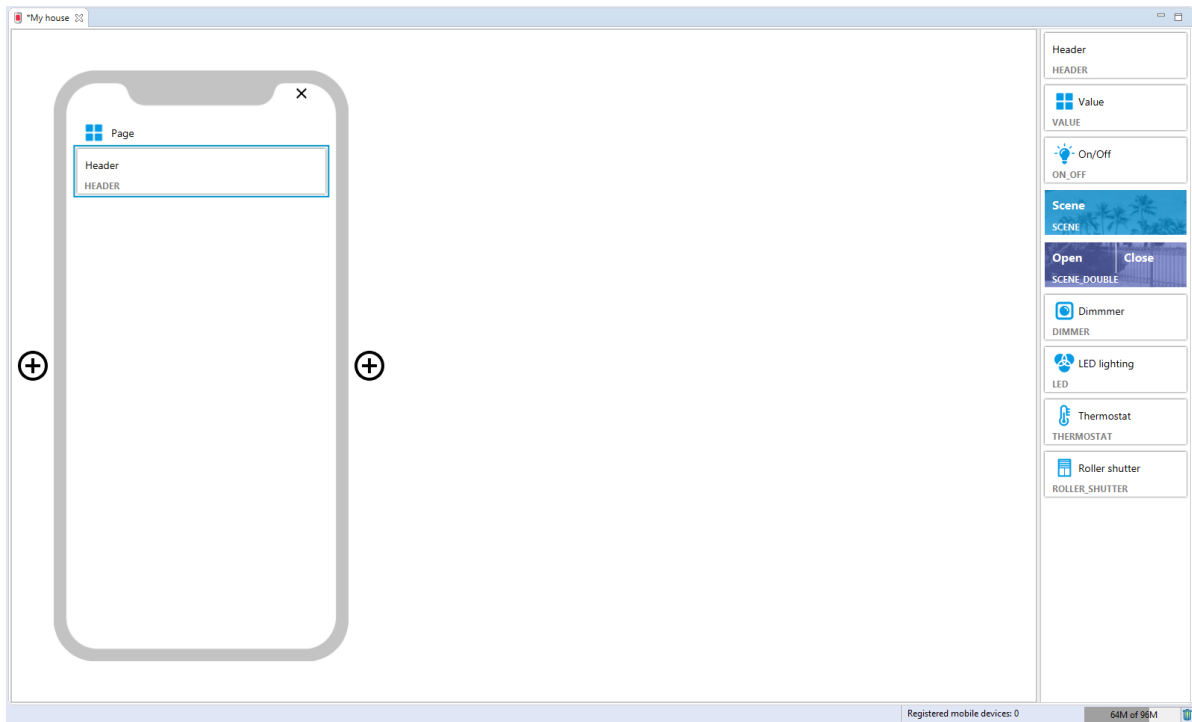
2. Dragging a specific widget from the tab on the right side and then completing its desired parameters.



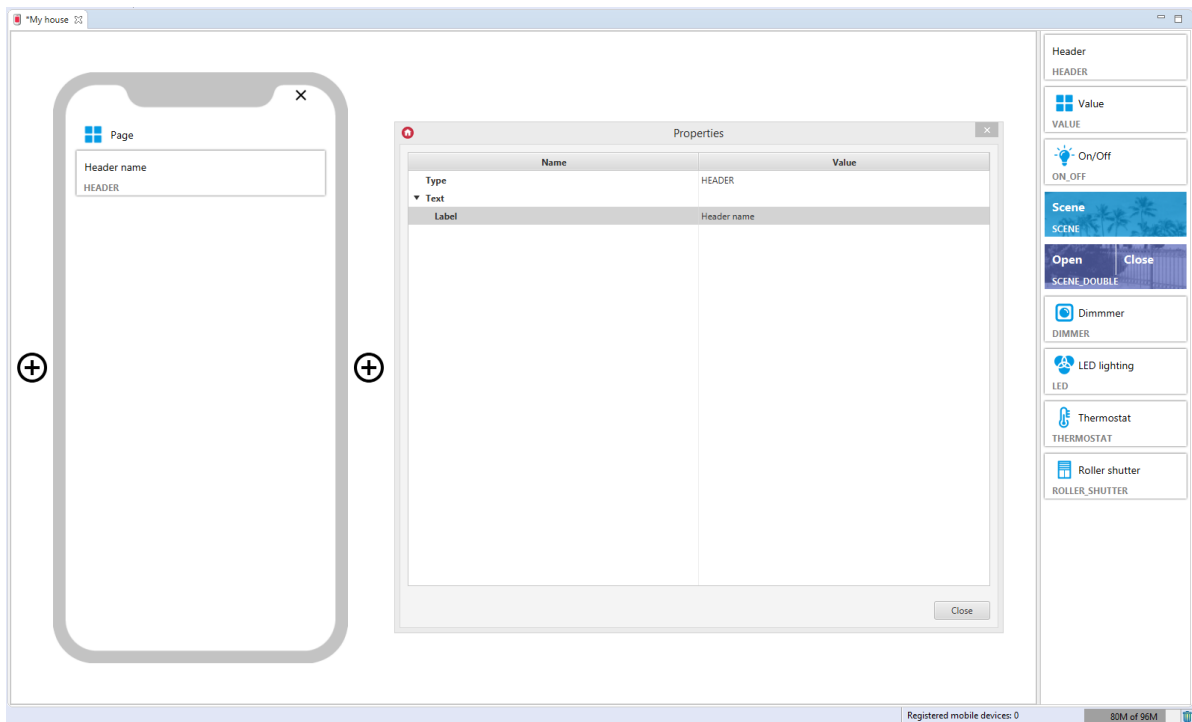
**NOTE!** The maximum number of widgets per page is 30.

### 3.1. Header

Used to display a string. All alphabetic, numeric and special characters are supported.



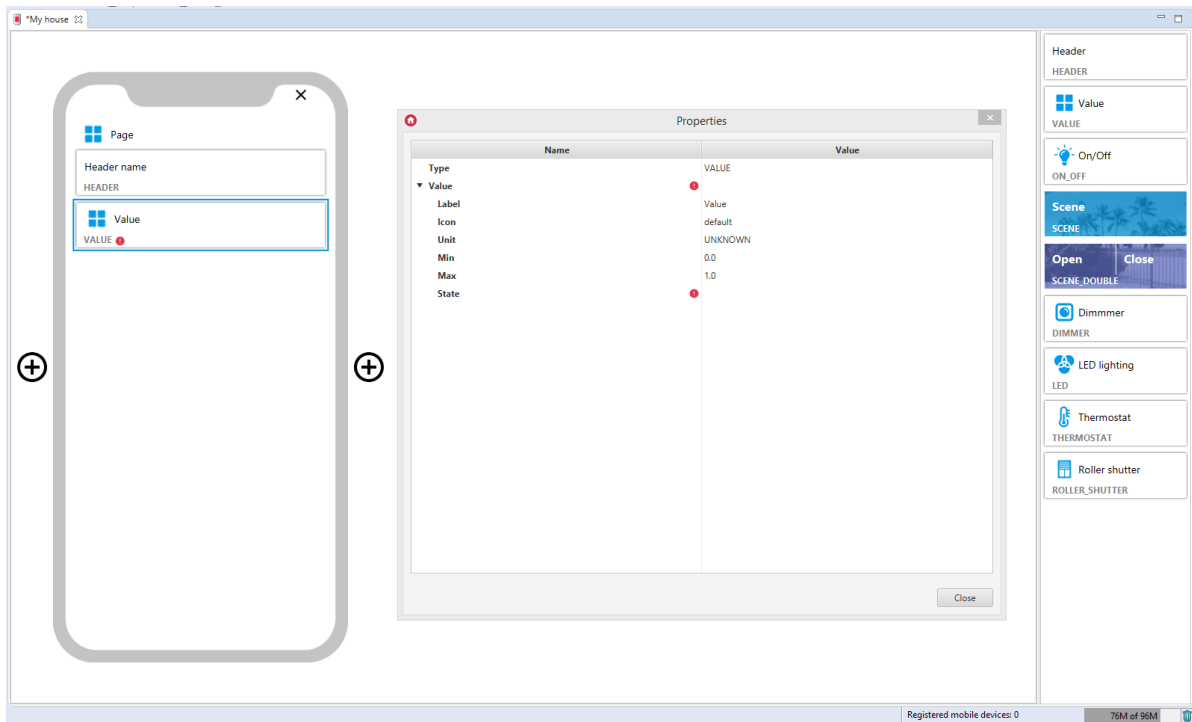
To change the header, click the word `Header` in the phone field. It is also possible to change the `Label` field in the properties window, which appears after double-clicking on the widget with the left mouse button.



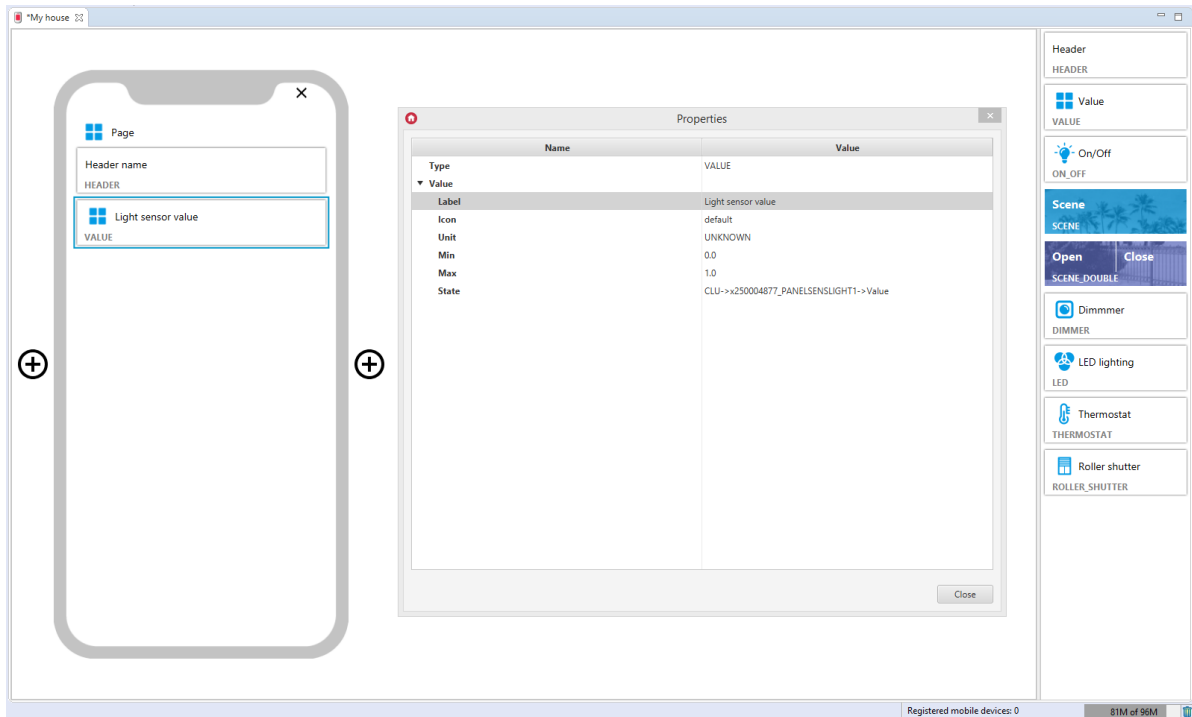
## 3.2. Value

The widget returns the property values of the object. This widget has three units to choose from: unknown, percent and degree. In addition, it is possible to describe the widget in the `Label` field and also to change the icon in the `Icon` field in the properties window, which appears after double-clicking on the widget with the left mouse button.





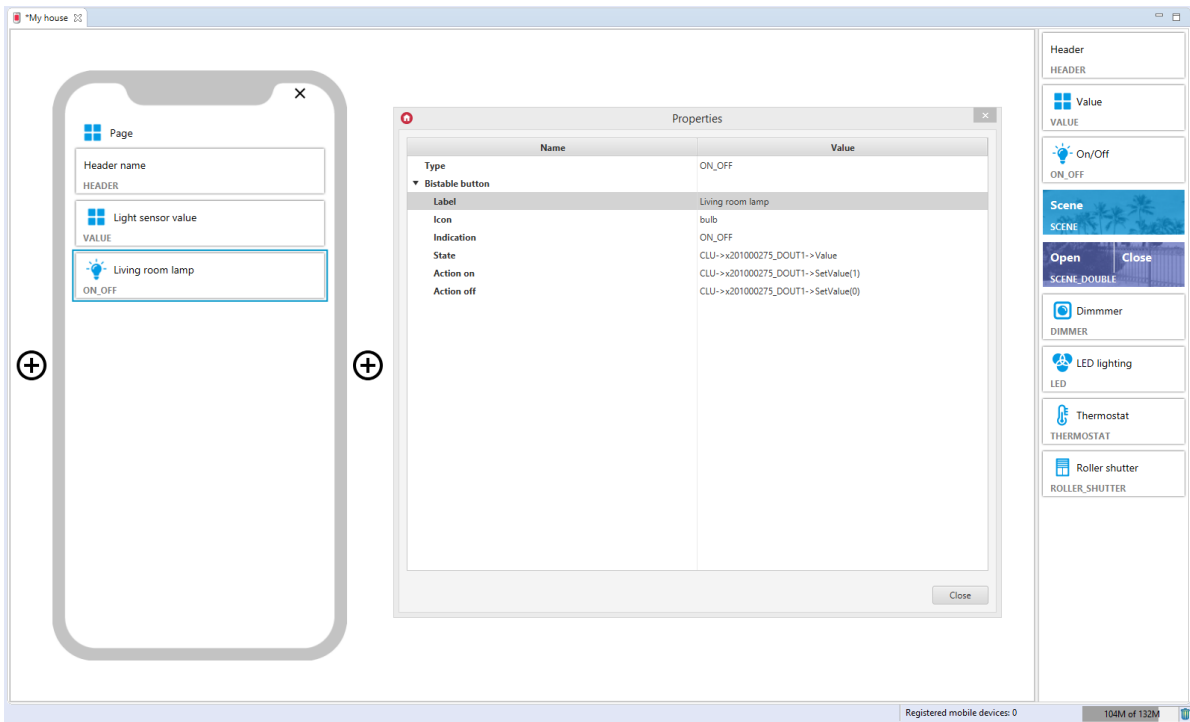
Filled widget:



### 3.3. On/Off

The widget is used to control the executive elements. When dragging the widget from the tab on the right side of the screen, it must be completed with the values in the `State`, `Action on`, `Action off` fields. Widget dedicated mainly to control relay outputs, however it can be used, among others, for switching LED lighting, starting virtual objects.

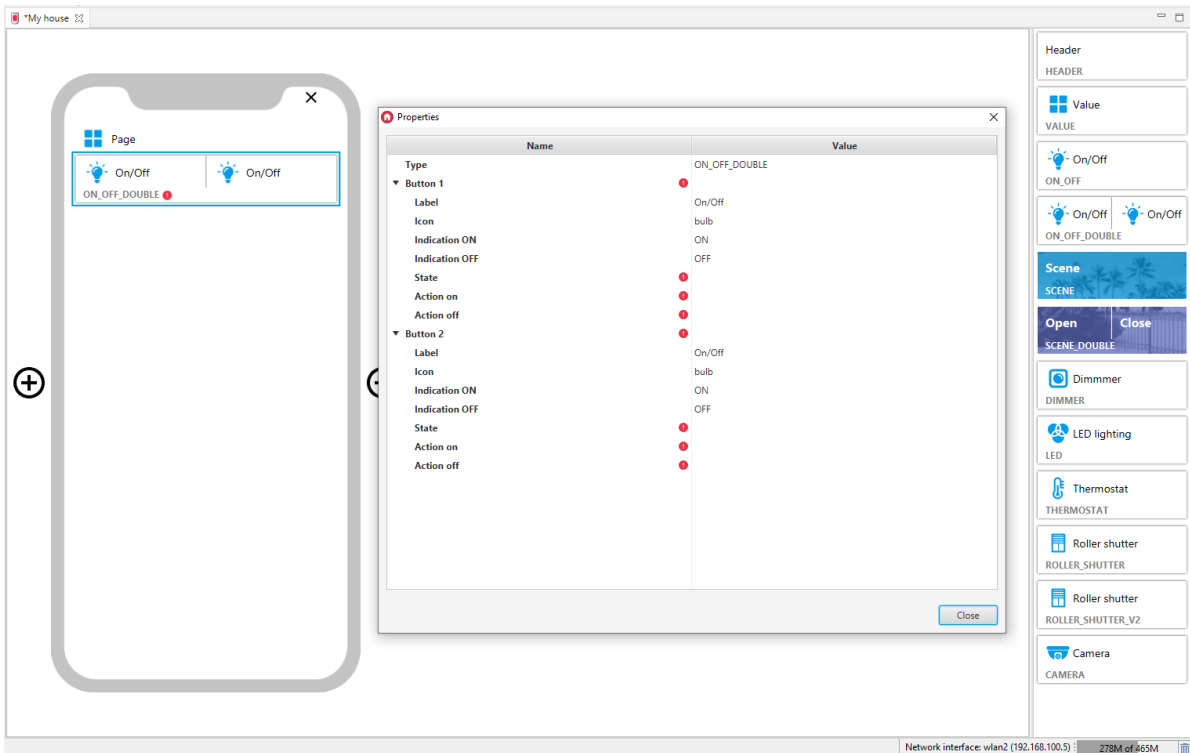




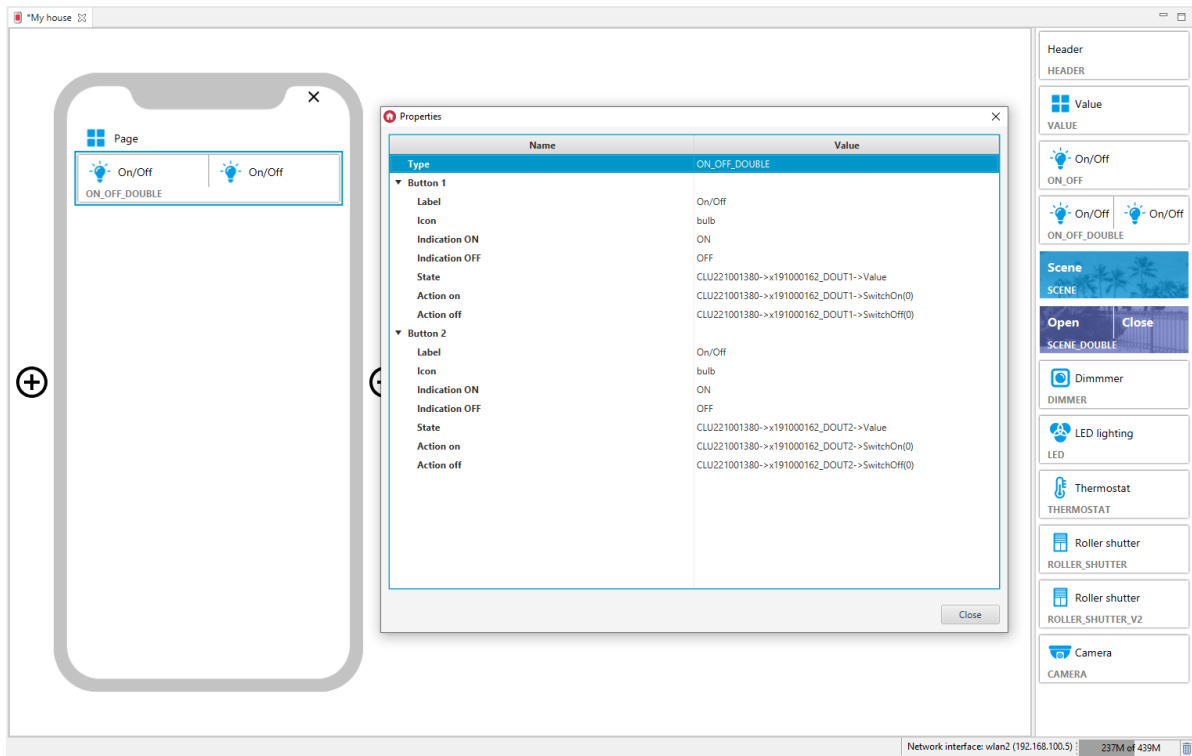
### 3.4. On/Off Double

**NOTE!** The On/Off Double widget is available for Object Manager version 1.4.0 or higher, and for myGrenton application version 1.2.3 or higher (Android) and version 1.6.0 or higher (iOS).

This is the On/Off double-version widget. By dragging the widget from the tab to the right of the screen, it must be supplemented with values in the fields `State`, `Action on`, `Action off` for both buttons.

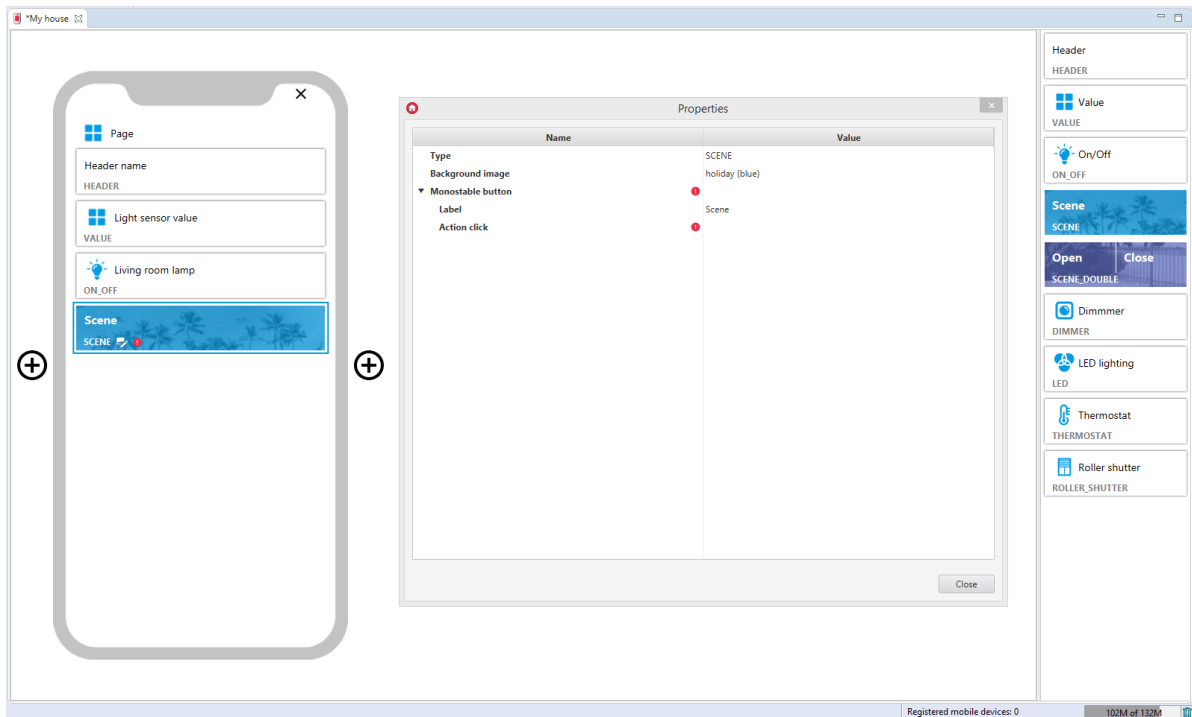


The widget you create will look like this:

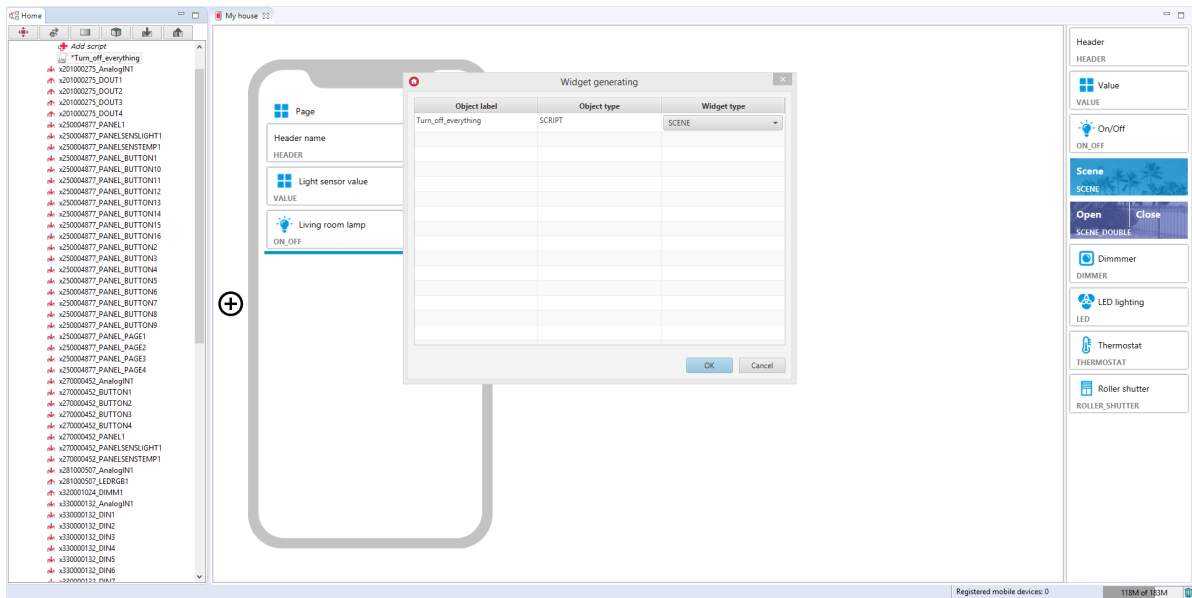


### 3.5. Scene

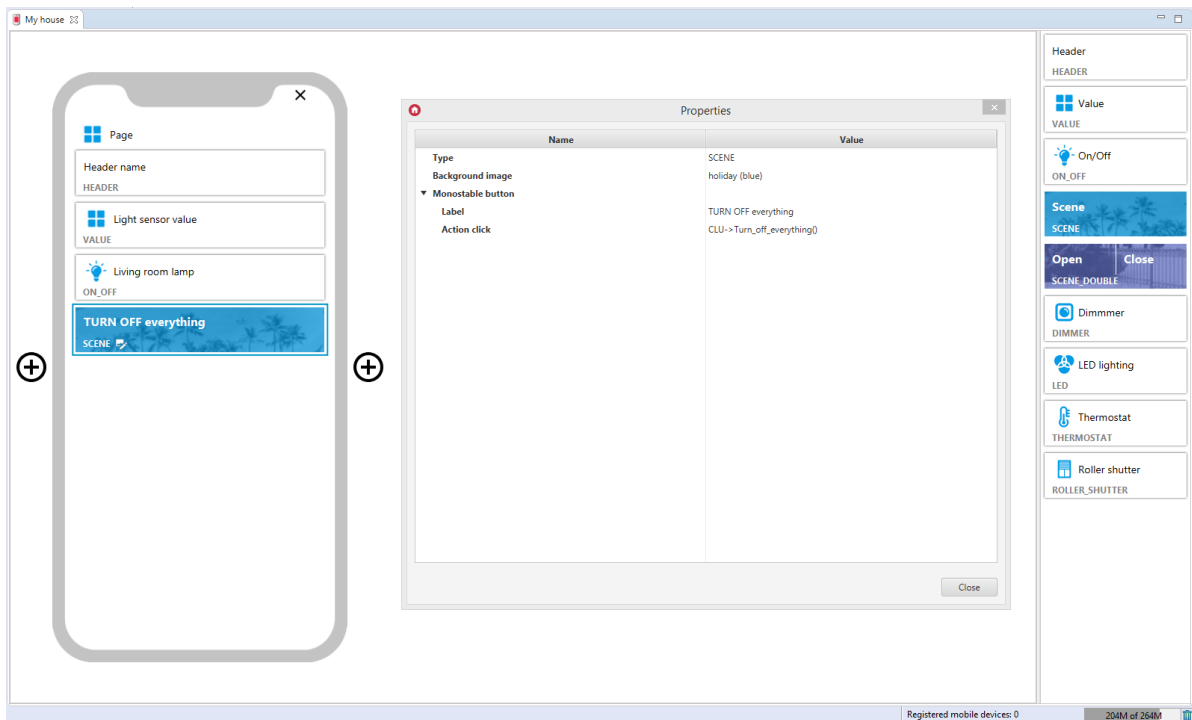
The widget is dedicated to calling created scripts.



For scripts, ready-made templates for the SCENE widget are defined. To add a SCENE widget with the desired digital output object, drag the script from the list of objects to the interface page:

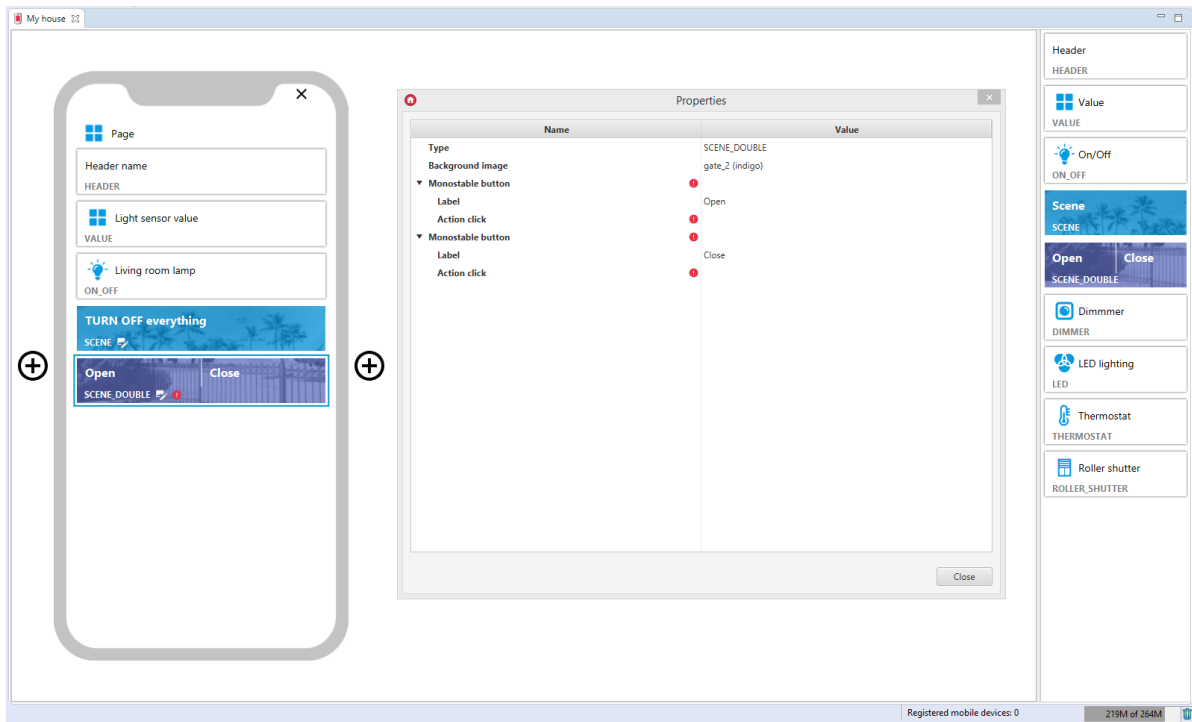


The widget you create will look like this:

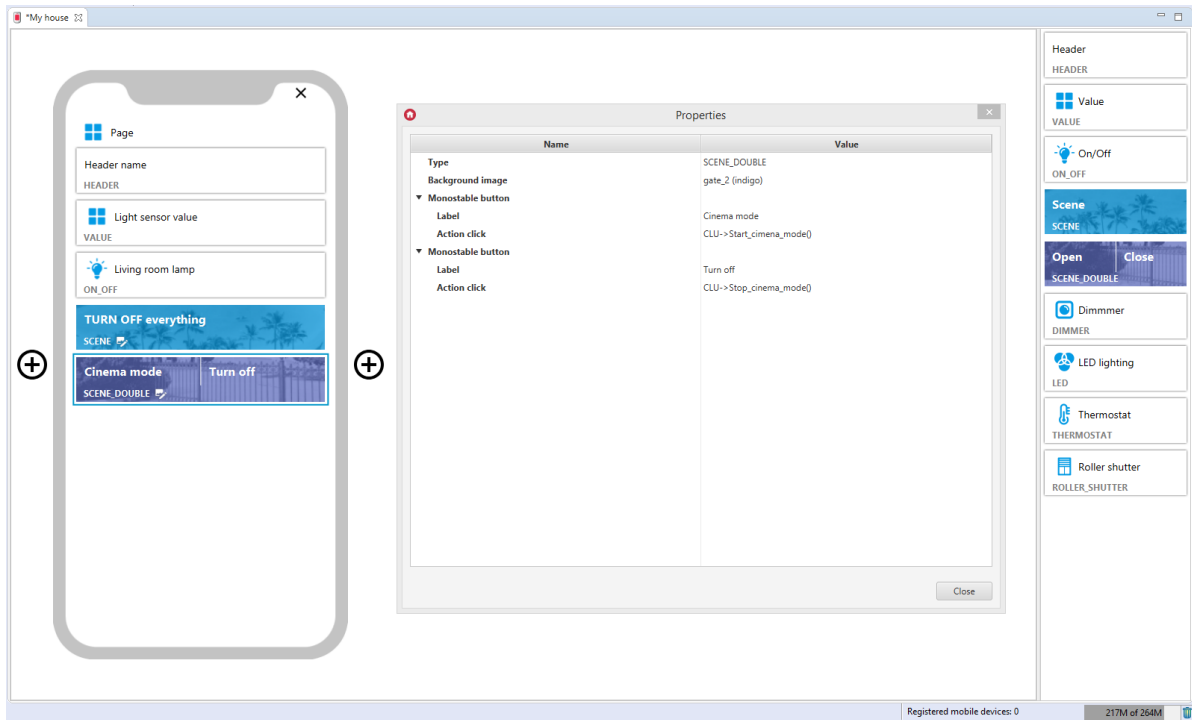


### 3.6. Open/Close (SCENE\_DOUBLE)

The widget is used to call scripts (object methods). In contrast to the widget SCENE has 2 buttons for actions.



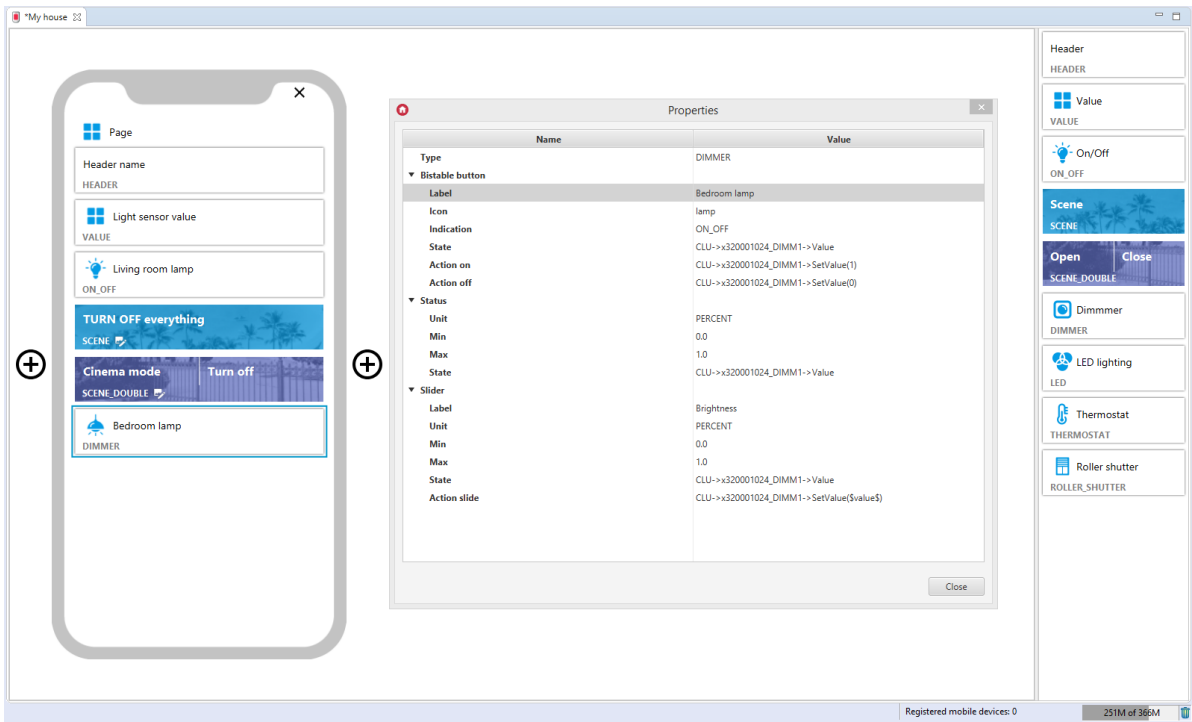
For example, the `Label` and `Action click` fields have been completed:



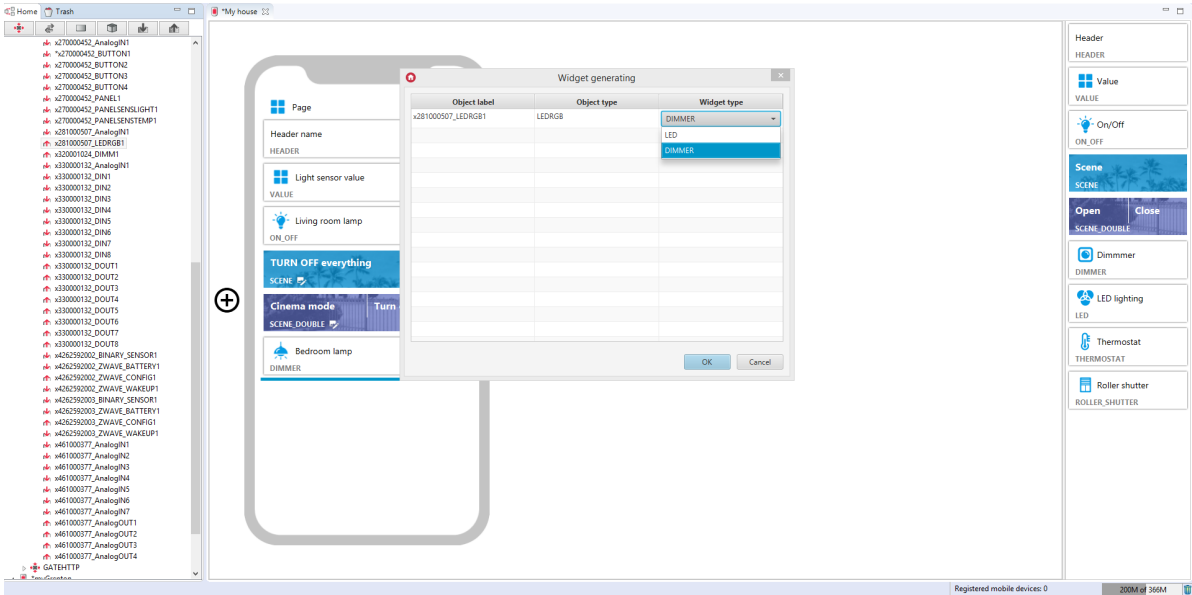
### 3.7. Dimmer

Widget dedicated for DIMMER and LEDRGB modules. When dragging the widget from the tab on the right side of the screen, it should be filled in with the values in the `State`, `Action on`, `Action off`, `Unit`, `Min`, `Max`, `Action slide` fields. This widget has a slider, which allows it to be controlled in a given range.



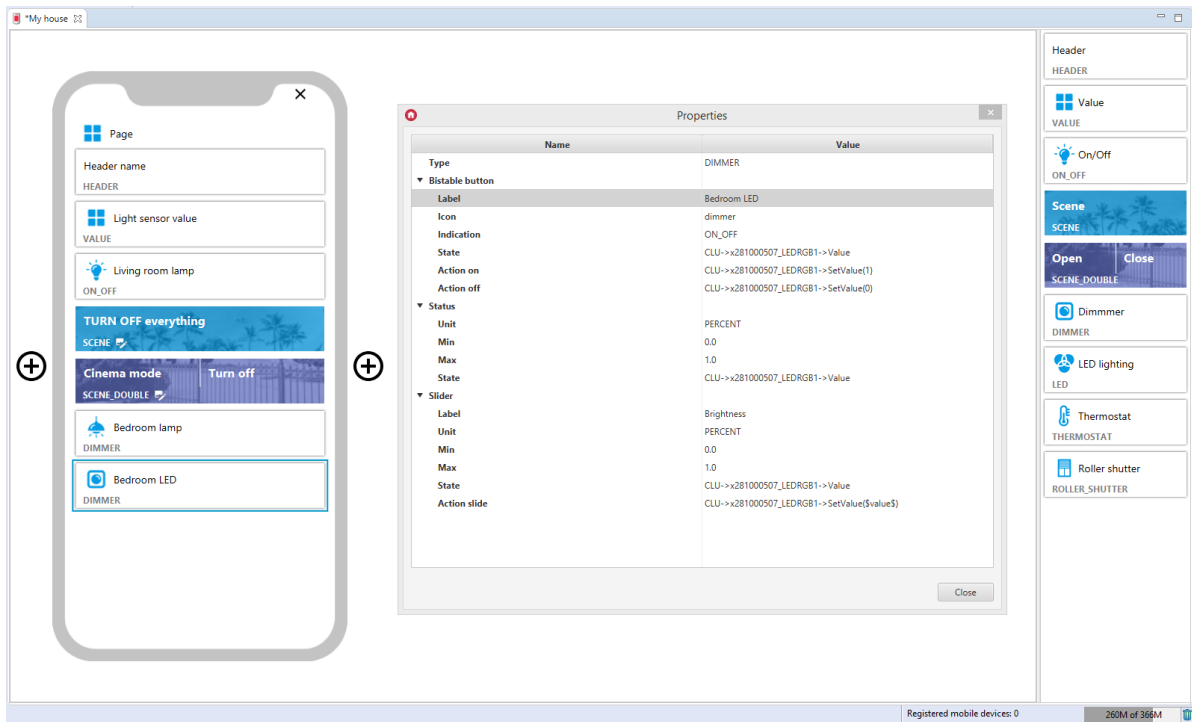


The DIMMER widget has a ready template also for the LEDRGB object:



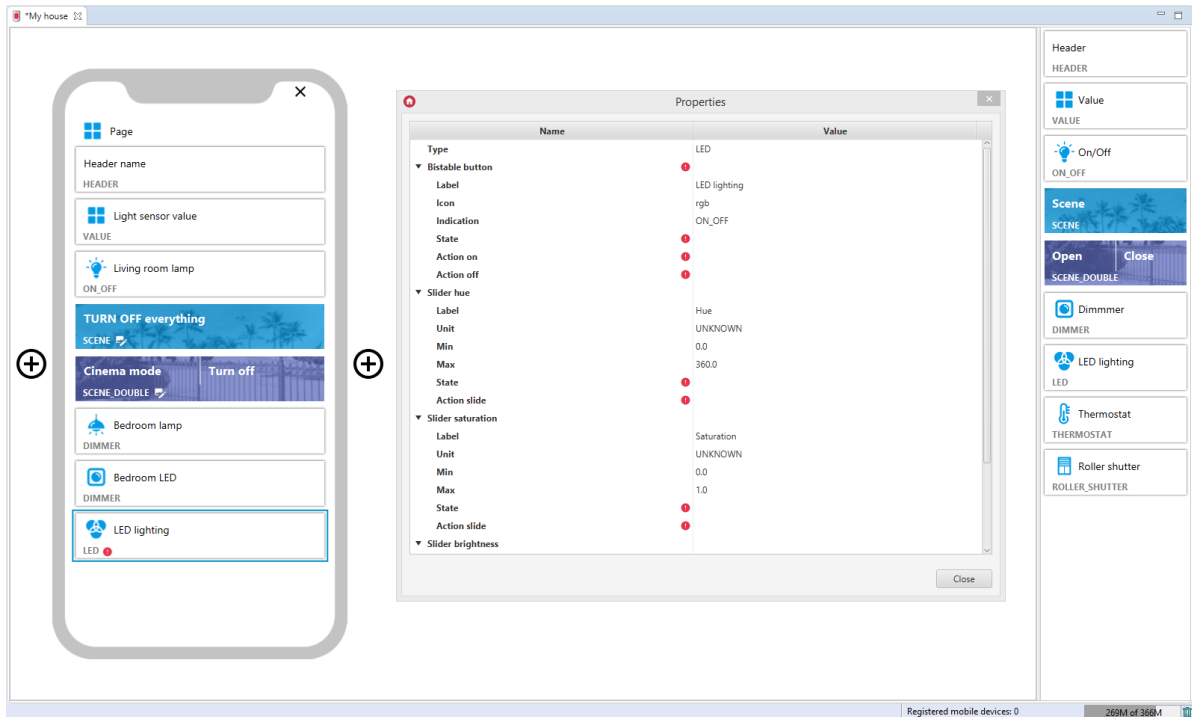
Widget created:



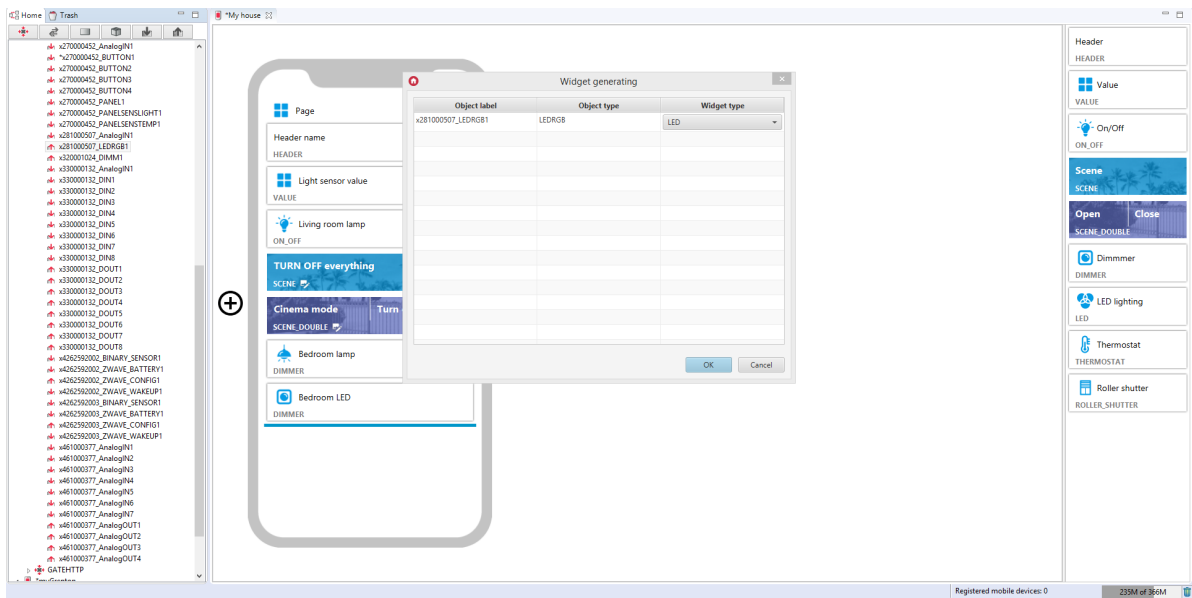


### 3.8. LED lighting

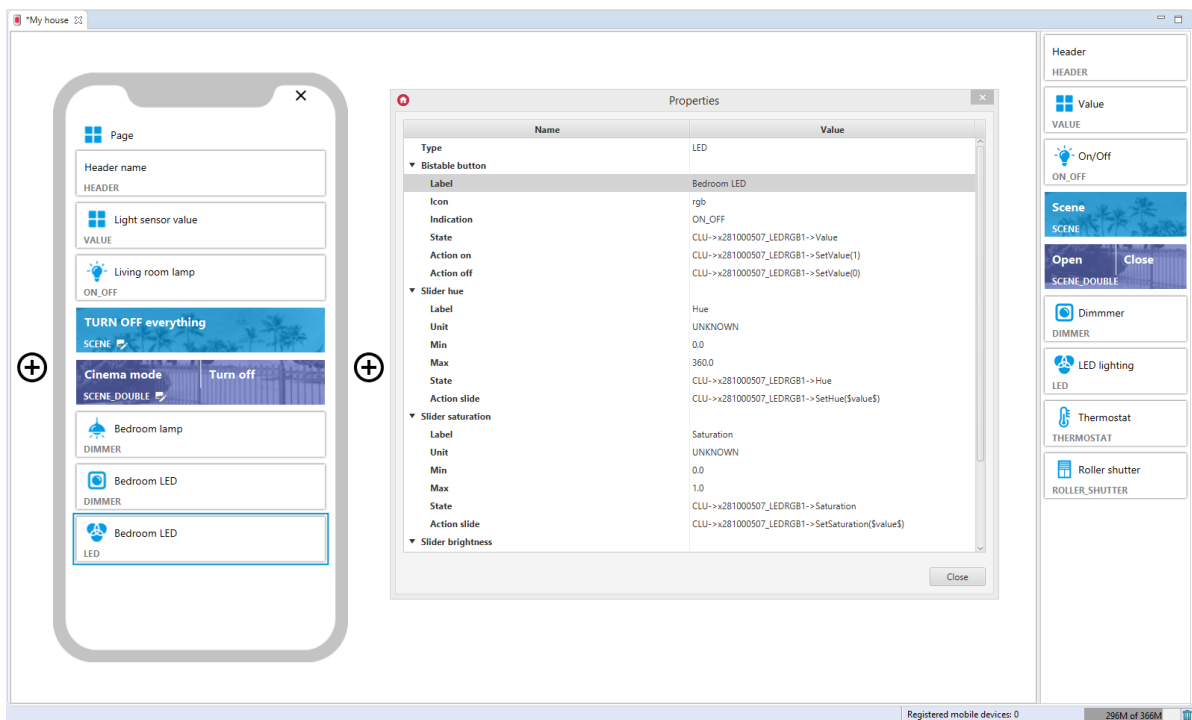
Widget dedicated to LED lighting. It has one bistable on/off switch and 3 sliders: Color control slider, Saturation control slider, Brightness control slider.



For LED lighting, ready-made templates for the LED widget are defined. To add a LED widget with a ready template, drag the LEDRGB object from the list of objects to the interface page:



Widget created:

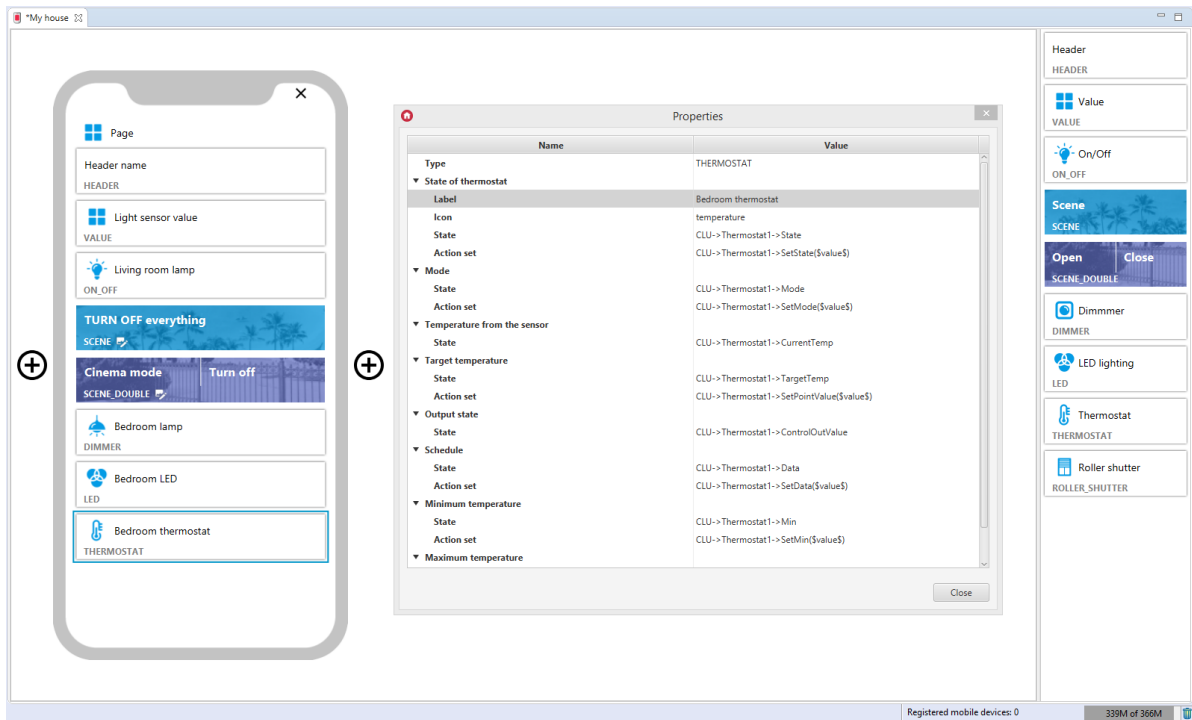


### 3.9. Thermostat

**NOTE!** Widget is supported for thermostats created in **CLU 2.0!**

Widget dedicated for virtual objects of the thermostat type. In the case when we draw from the list of objects already defined thermostat to the interface, the created widget is completed on the basis of given thermostat input and output features.

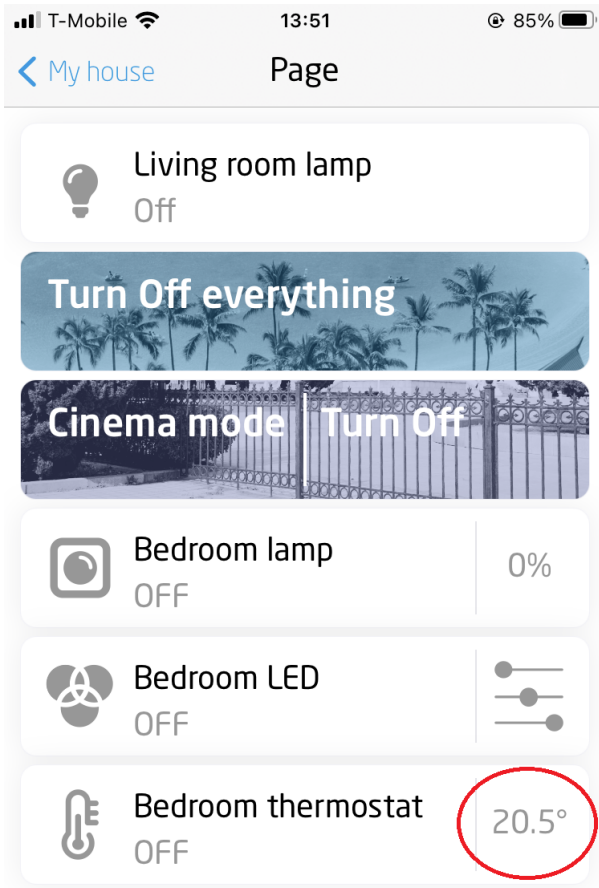




## A. Schedule configuration in the application

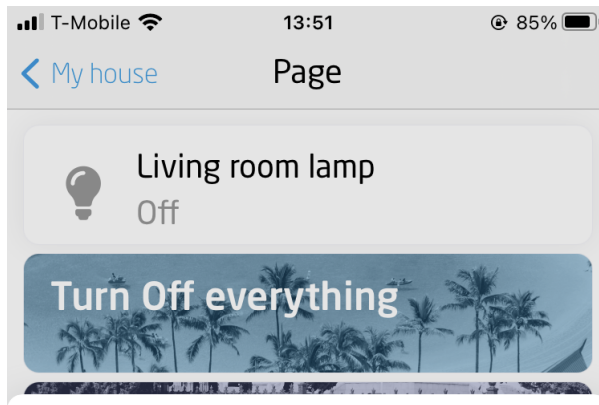
**NOTE!** The new schedule configuration is available for myGrenton application version 1.2.3 or higher (Android) and version 1.6.0 or higher (iOS).

In the myGrenton application, you can edit the thermostat schedule. To do this, click on the widget temperature field:



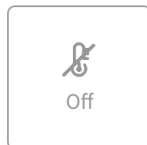
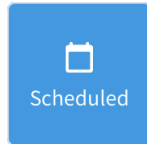
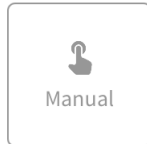
•

Then select the schedule mode and the `EDIT SCHEDULE` option:



## Details

20.5°

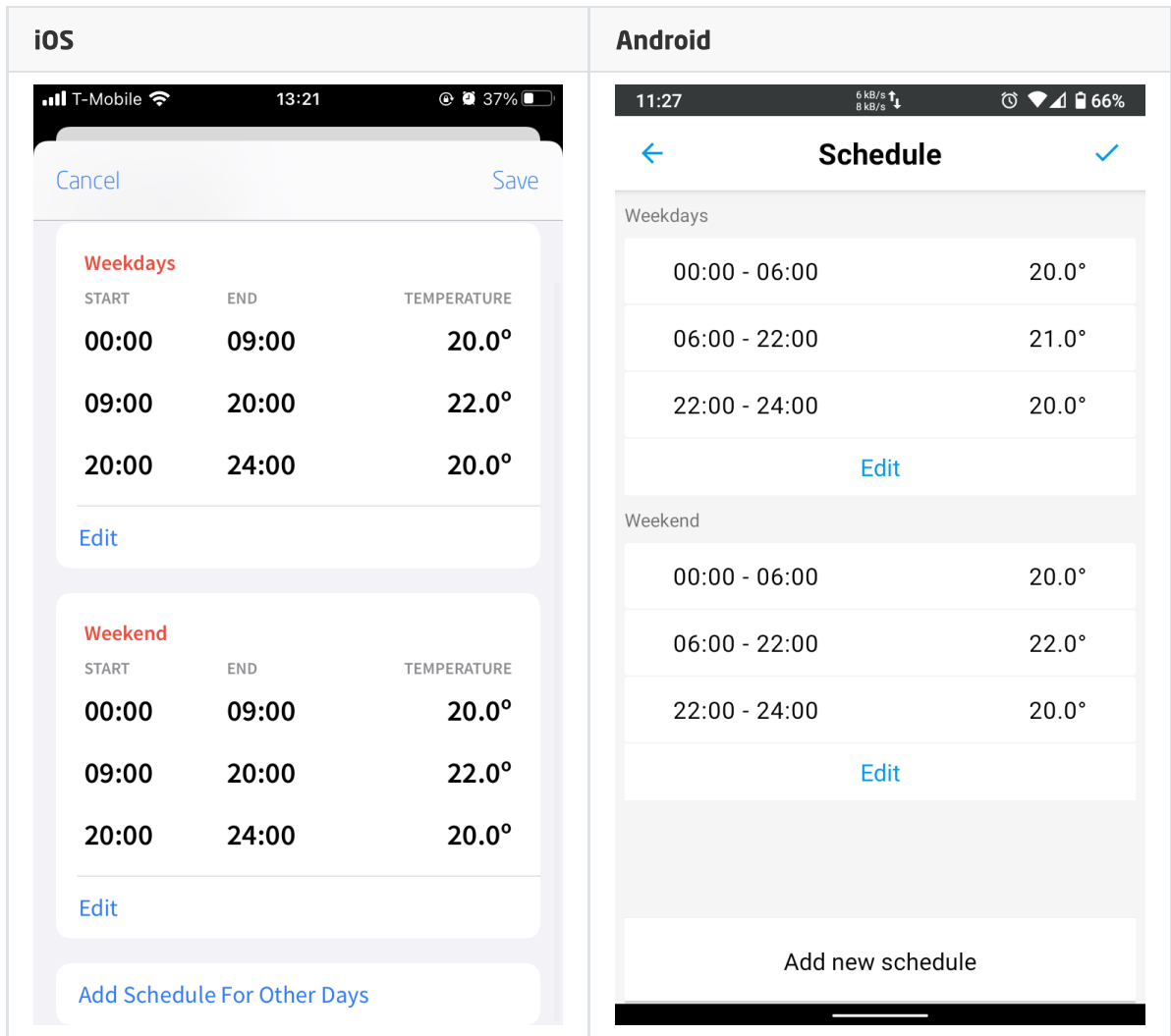


20.0°

Thermostat is running according to predefined schedule

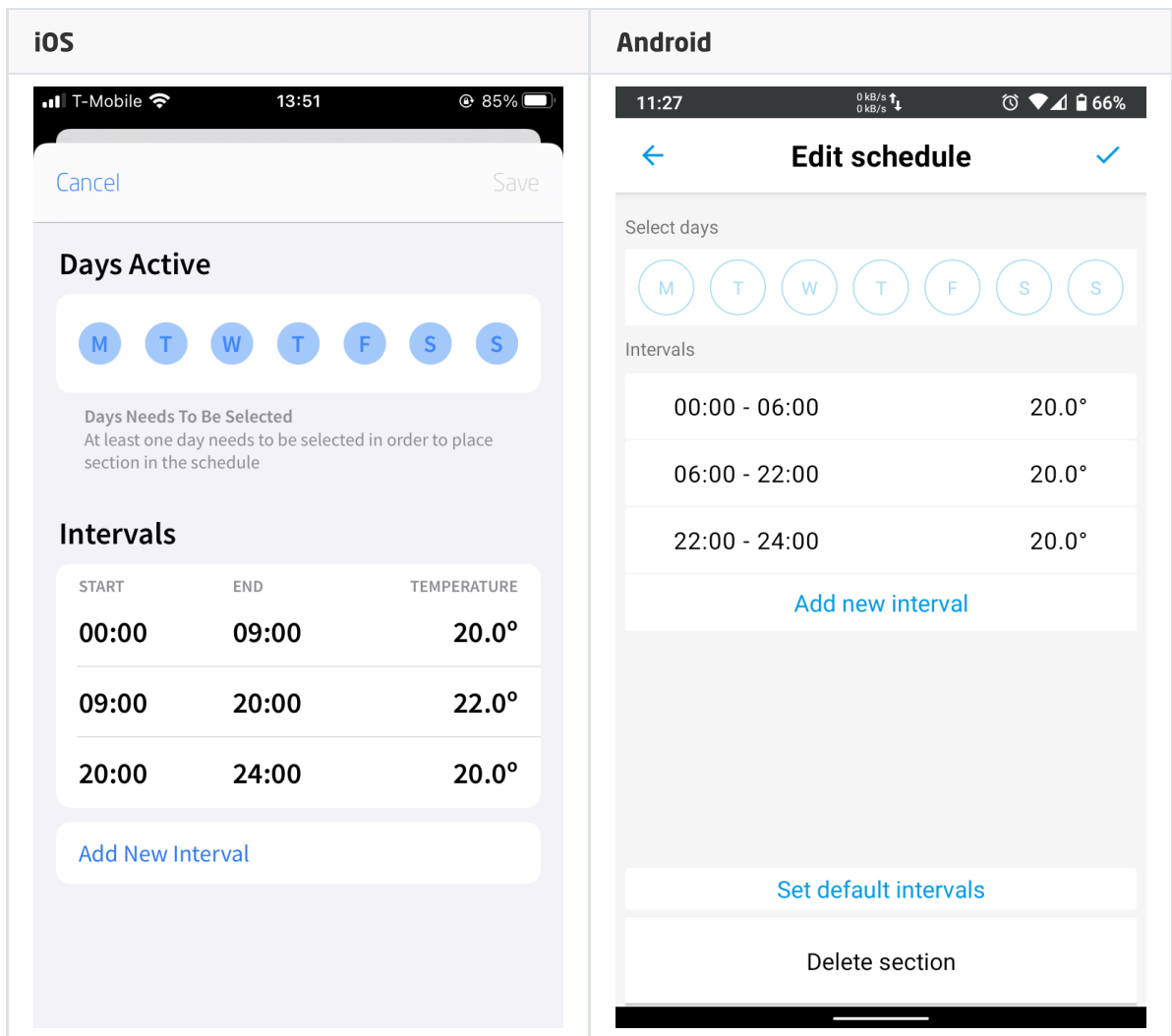
[EDIT SCHEDULE](#)

The window will display the schedule downloaded from the CLU. You can edit this schedule or add new schedules for each day of the week:



### Adding a new schedule

After selecting `Add Schedule For Other Days` (iOS) or `Add new schedule` (Android), the adding schedule window will open:



Then:

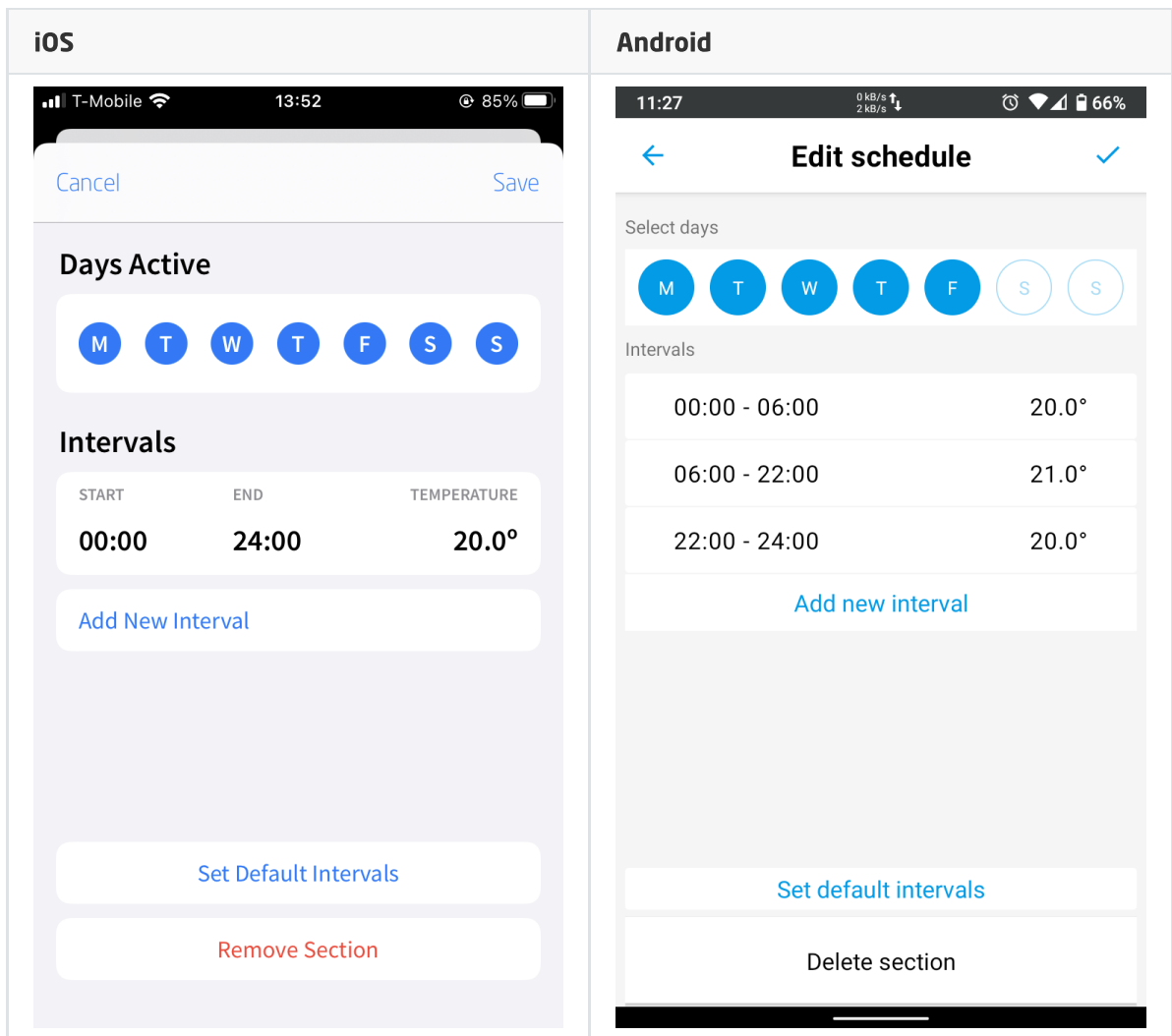
- Mark the desired days of the week (at least one day must be selected),
- Set the temperature for specific time intervals (after opening the window, default time intervals are displayed),
- Accept the changes by clicking on `Save`.

The application for unselected days of the week will automatically create a new schedule or add them to the existing one to correctly complete the values for the whole week.

### Delete / edit a section of the schedule

After selecting `Edit` for the selected section of the schedule, the editing window will open, where you can edit the selected days of the week, time intervals or delete the section:



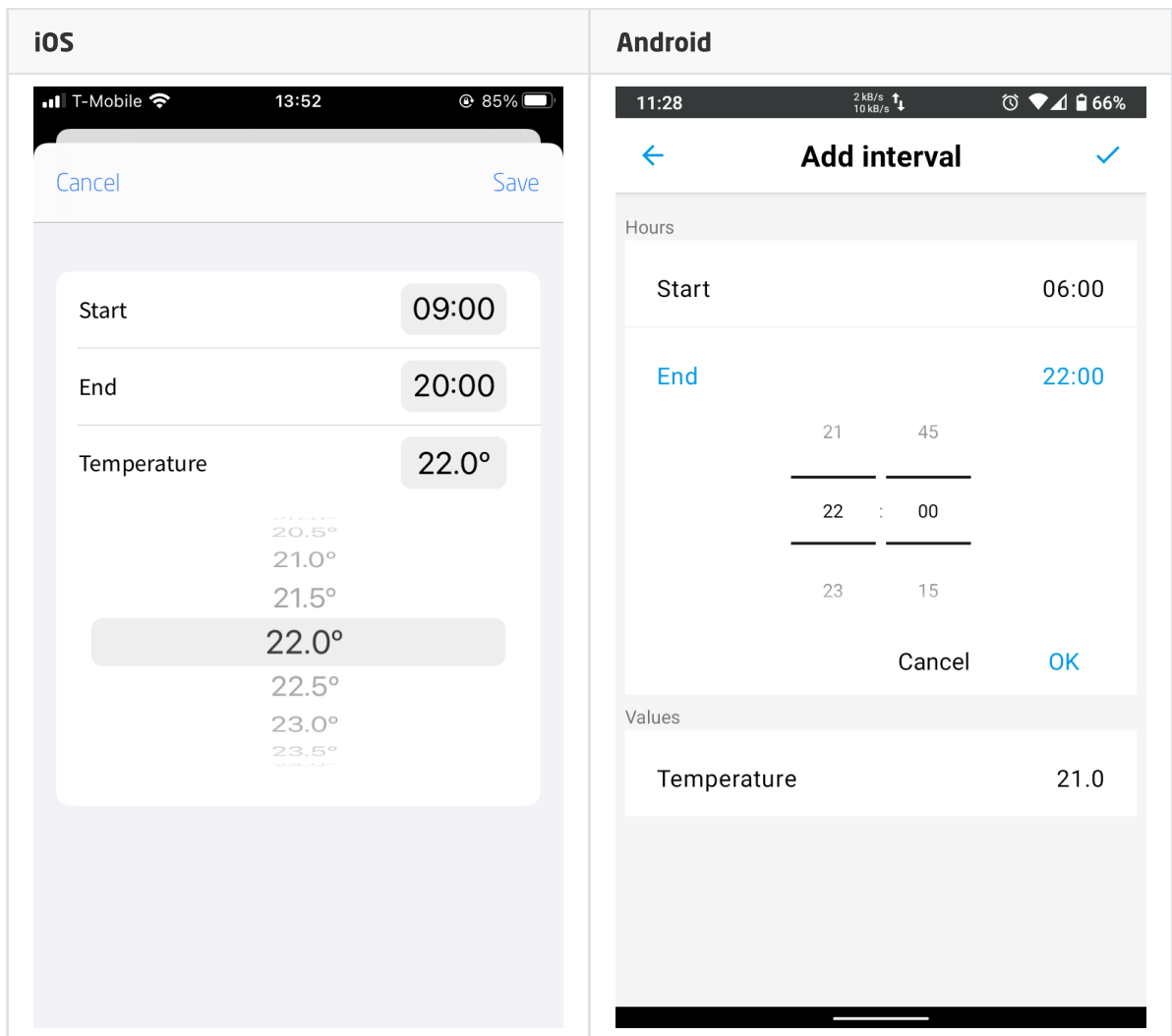


To delete a schedule section, click on `Remove Section`. After deleting the schedule, the application will automatically create a new schedule for the missing days of the week or add them to the existing one to correctly complete the values for the whole week.

With the option `Set Default Intervals`, you can replace the current time intervals with the default ones.

### Adding new time periods

After selecting `Add New Interval`, the window for adding a time interval will open:



Then:

- Enter the start time of the interval,
- Enter the end time of the interval,
- Set the desired temperature,
- Accept the changes by clicking on `Save`.

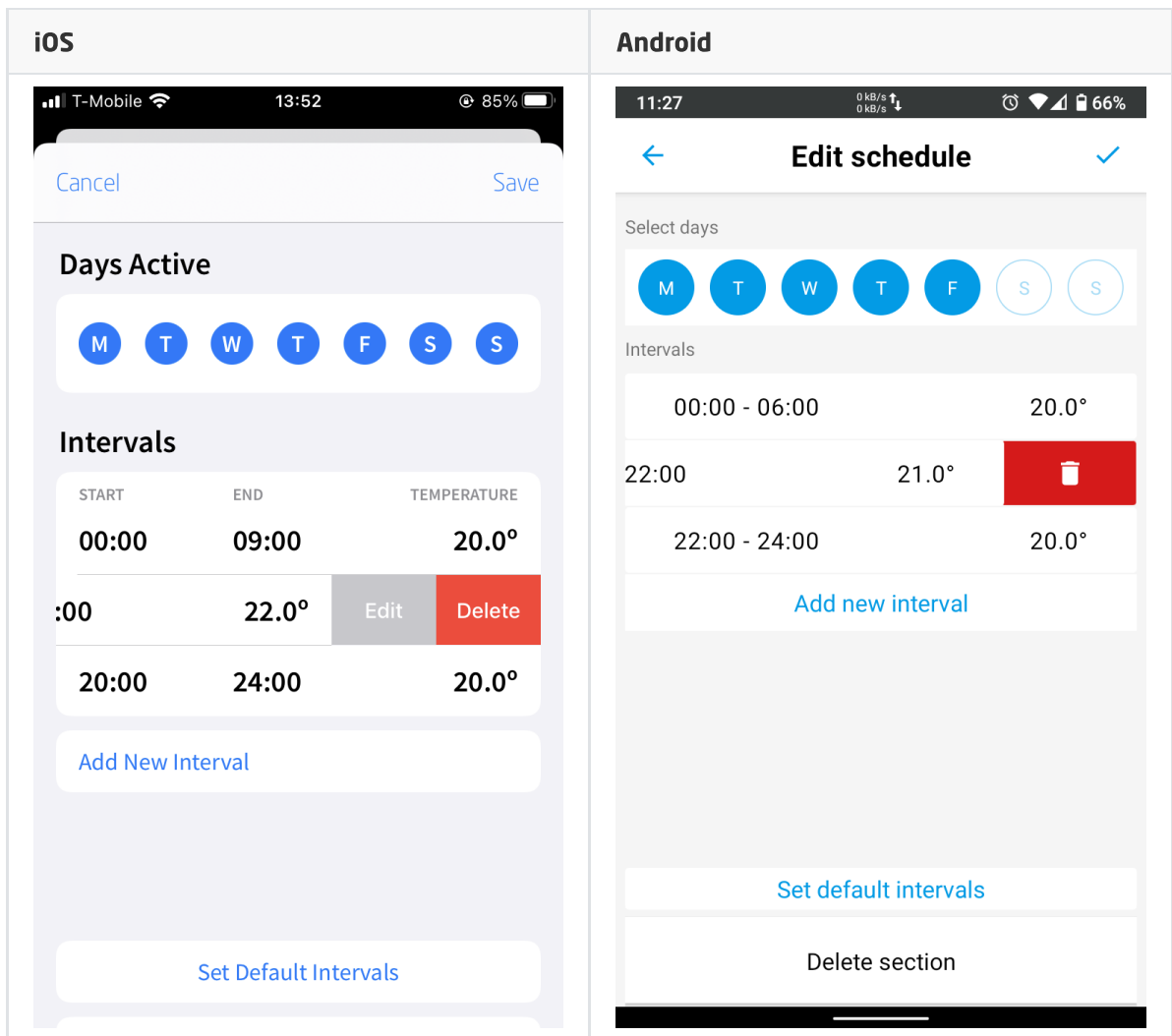
The application will automatically add intervals for unaccounted hours to correctly fill in the values for the whole day.

**NOTE!** The `Add New Interval` option allows you to add up to 6 time periods.

### Delete / edit time period

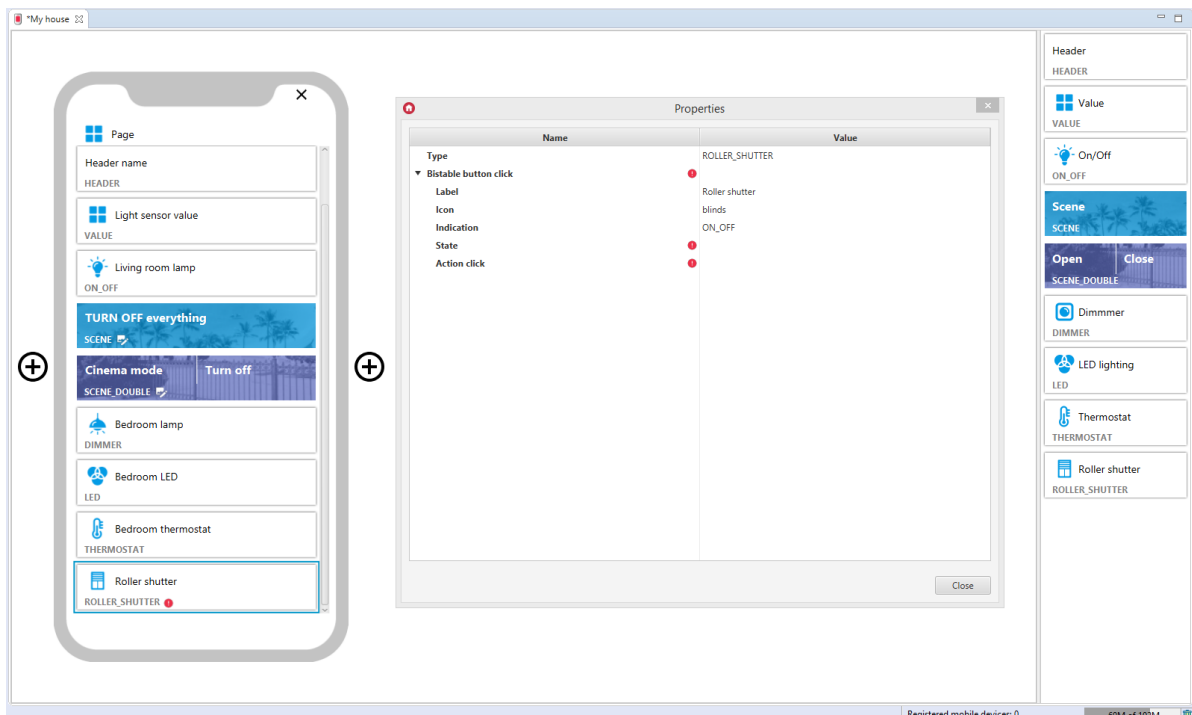
To edit an existing period, click on the time period (iOS / Android) or make a left-swipe gesture on the time period, and then click on the `Edit` option (available only for iOS).

To delete a time period, perform a left-swipe gesture on the time period, and then click `Delete`.

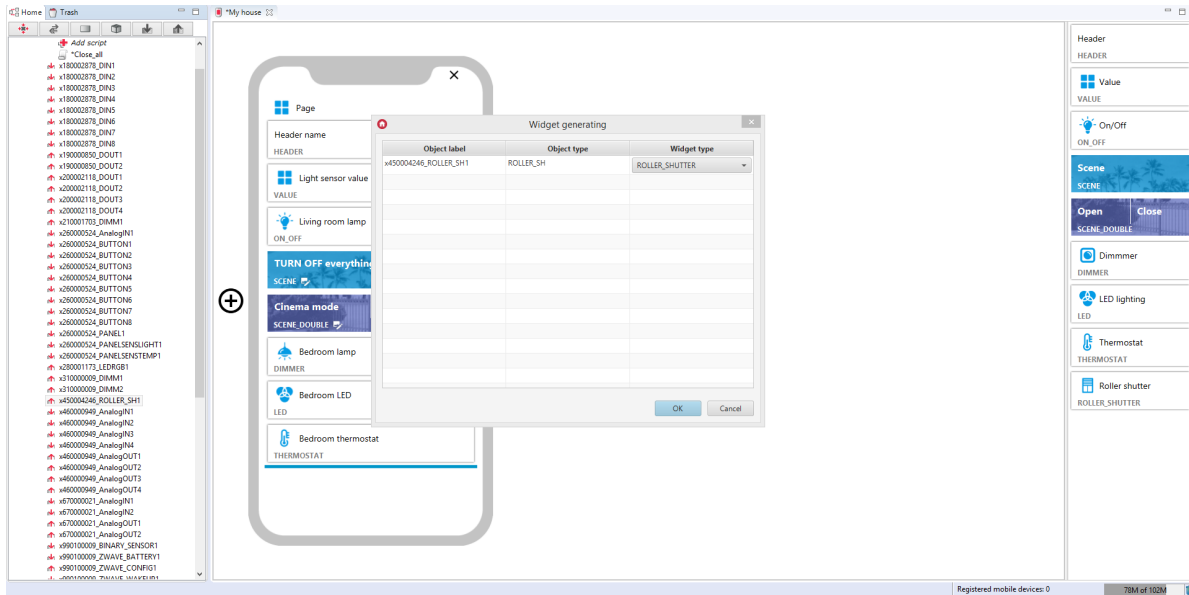


### 3.10. Roller shutter

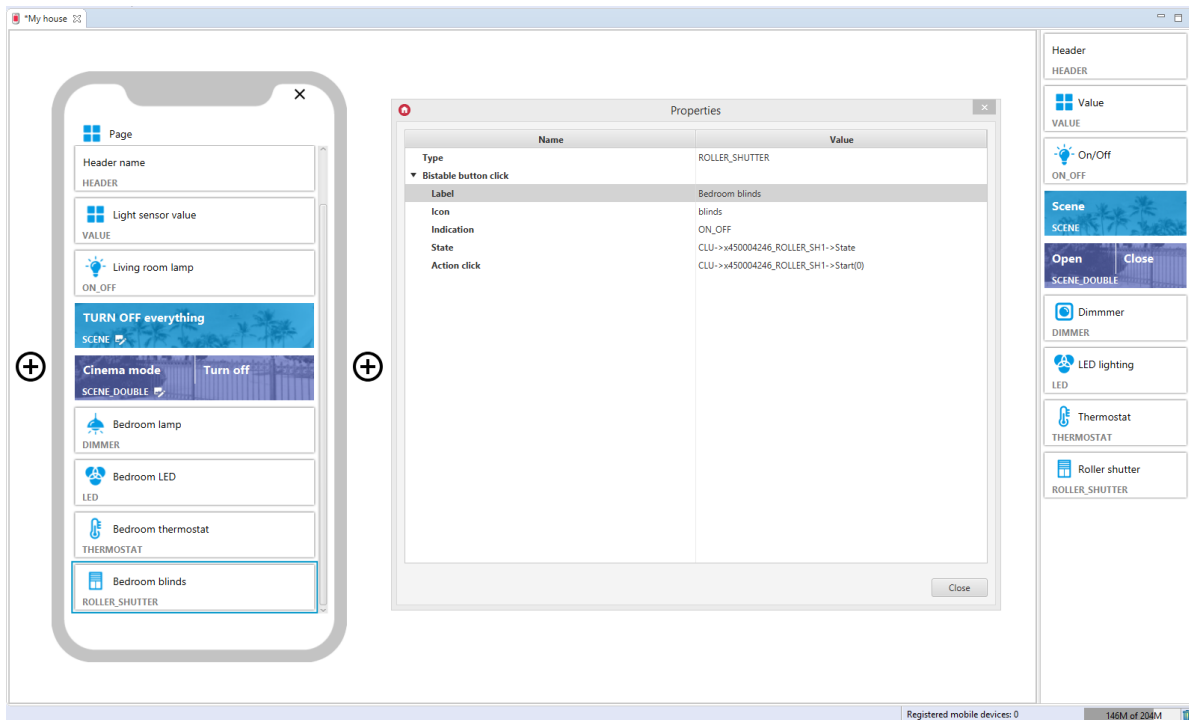
The widget is used to control the blinds. The application displays the current status of the blinds (STOPPED, CLOSED, OPEN).



For blind controllers, pre-defined templates for the ROLLER\_SHUTTER widget are defined. To add the ROLLER\_SHUTTER widget with a ready template, drag the ROLLER\_SHUTTER object from the list of objects to the phone:



Ready widget:



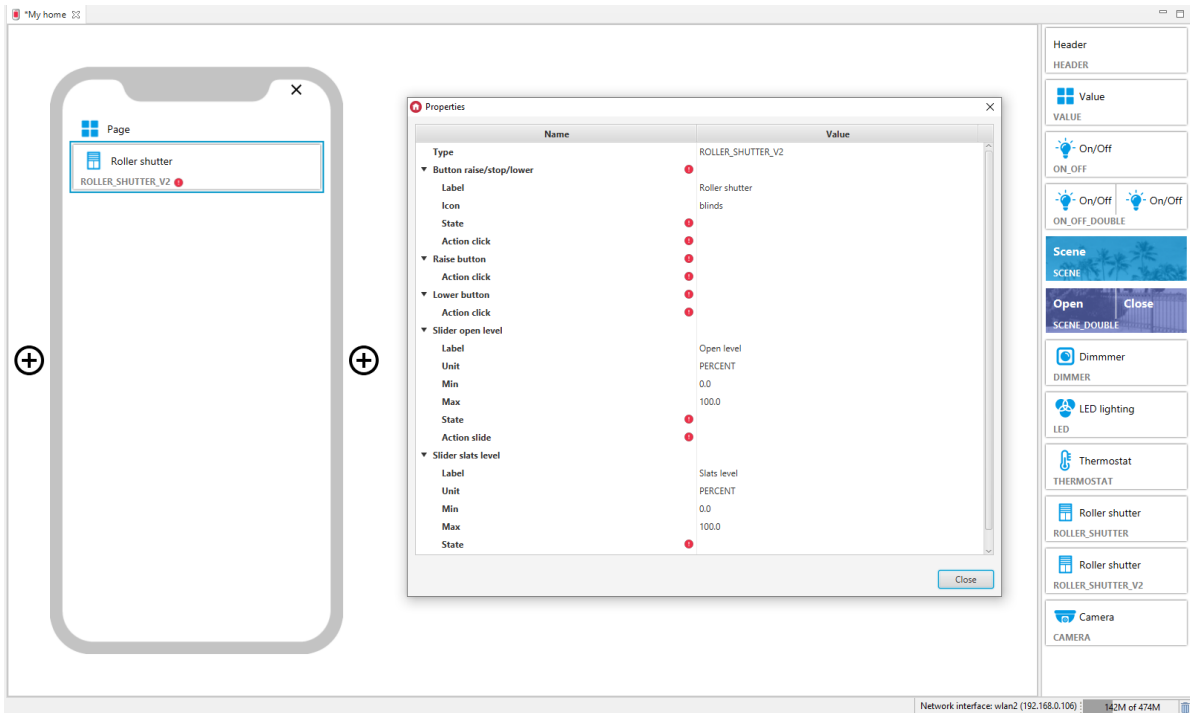
### 3.11. Roller shutter v2

**NOTE!** The ROLLER\_SHUTTER\_V2 widget is available for Object Manager version 1.4.0 or higher, CLU version 5.7.1 or higher, Roller Shutter x1 DIN / Roller Shutter x3 DIN / Roller Shutter FM module in version 2.1.1 or higher and for myGrenton application version 1.2.3 (Android) / 1.6.0 (iOS) or higher.

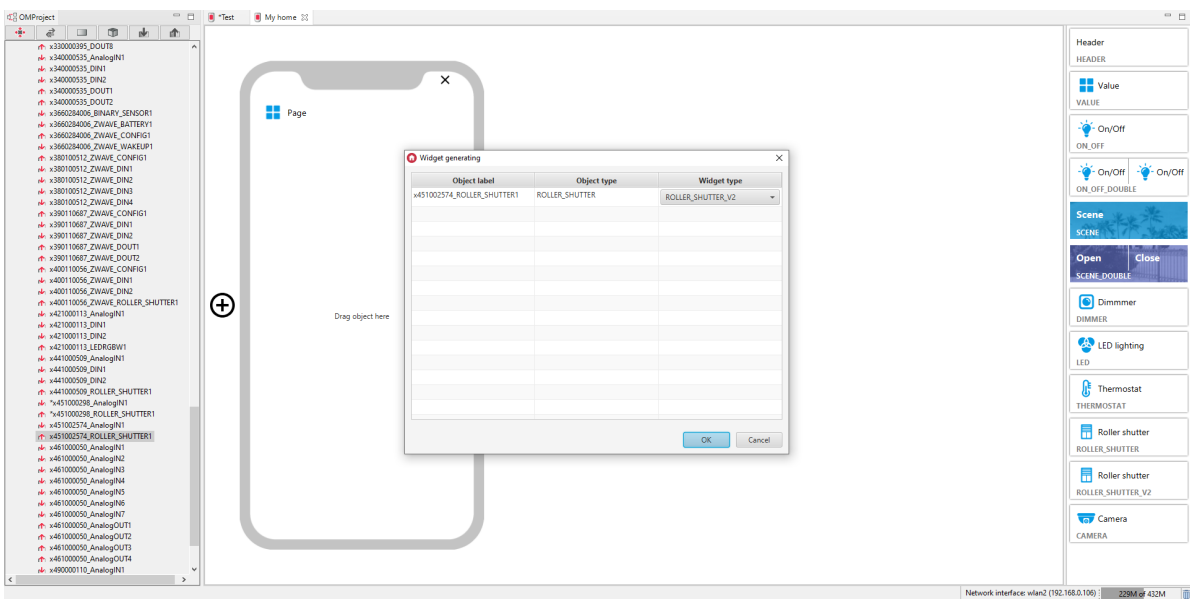
Widget dedicated to advanced control of blinds and slats. The ROLLER\_SHUTTER\_V2 widget supports:

- Buttons enabling the actions of OPENING / CLOSING / STOP the roller shutter,
- Percentage of shutter opening level,

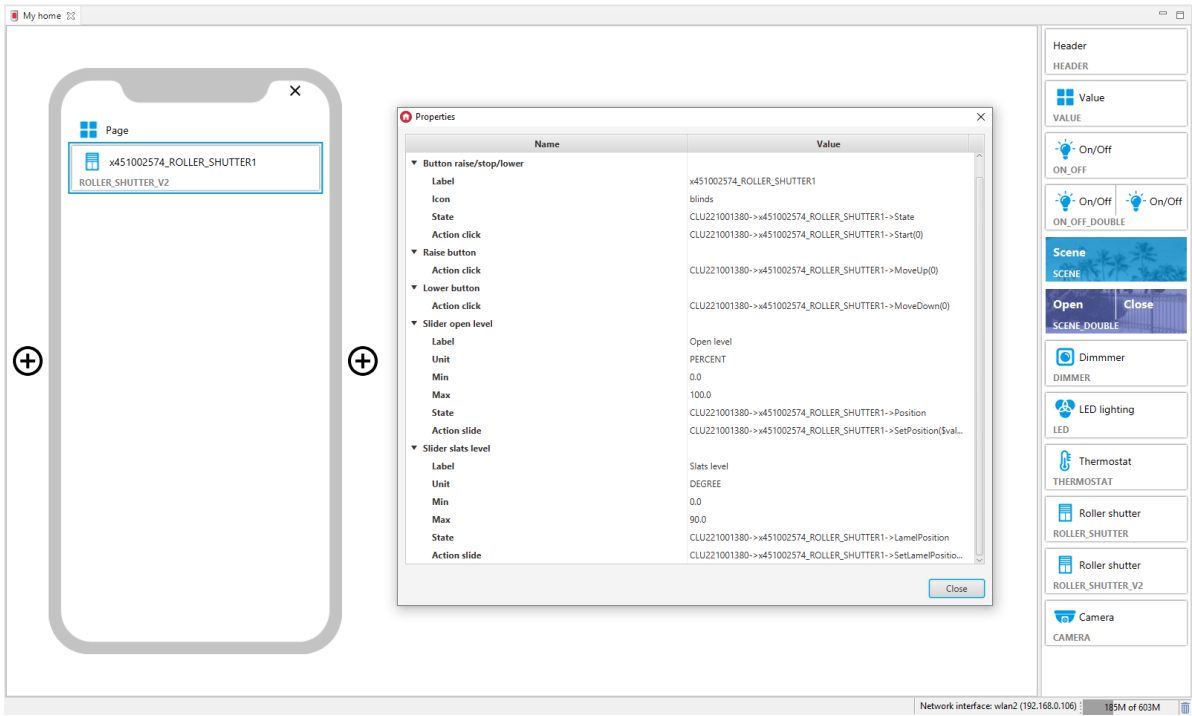
- Sliders for controlling and displaying the shutter and slat opening level.



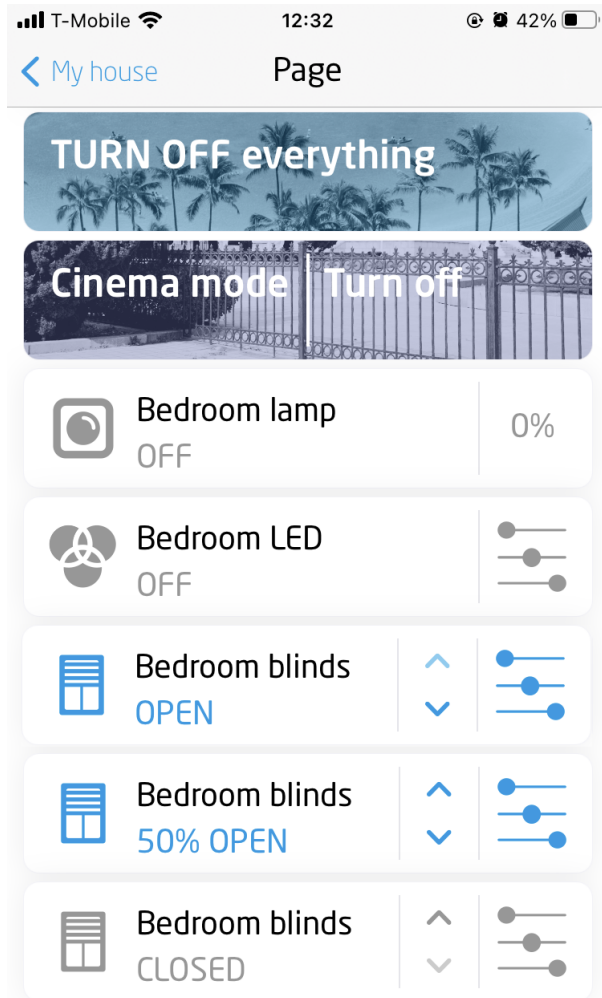
For blind controllers, pre-defined templates for the ROLLER\_SHUTTER\_V2 widget are defined. To add the ROLLER\_SHUTTER\_V2 widget with a ready template, drag the ROLLER\_SHUTTER object from the list of objects to the phone:

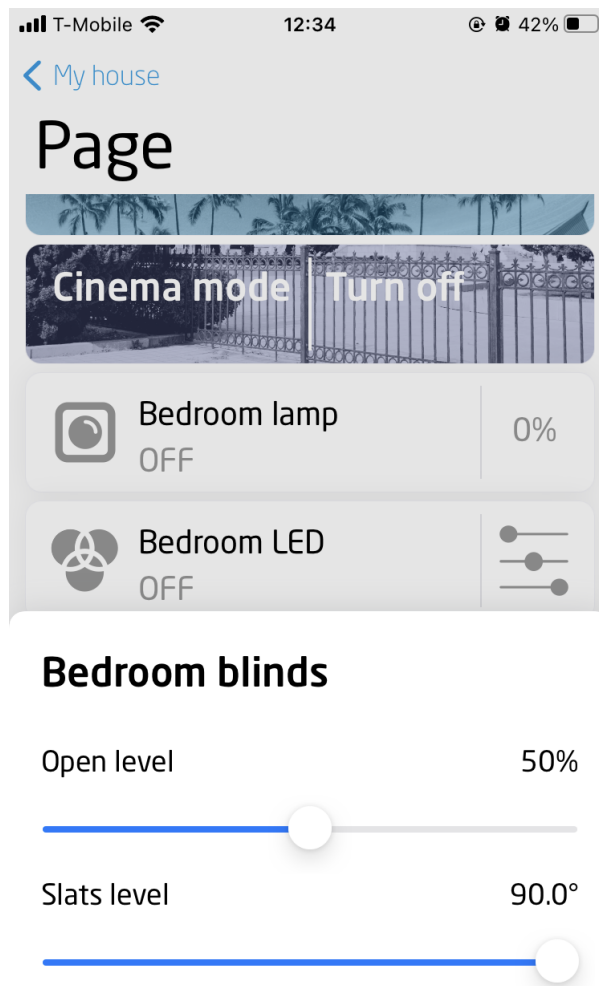


Filled ROLLER\_SHUTTER\_V2 widget:



Widget appearance in myGrenton application:

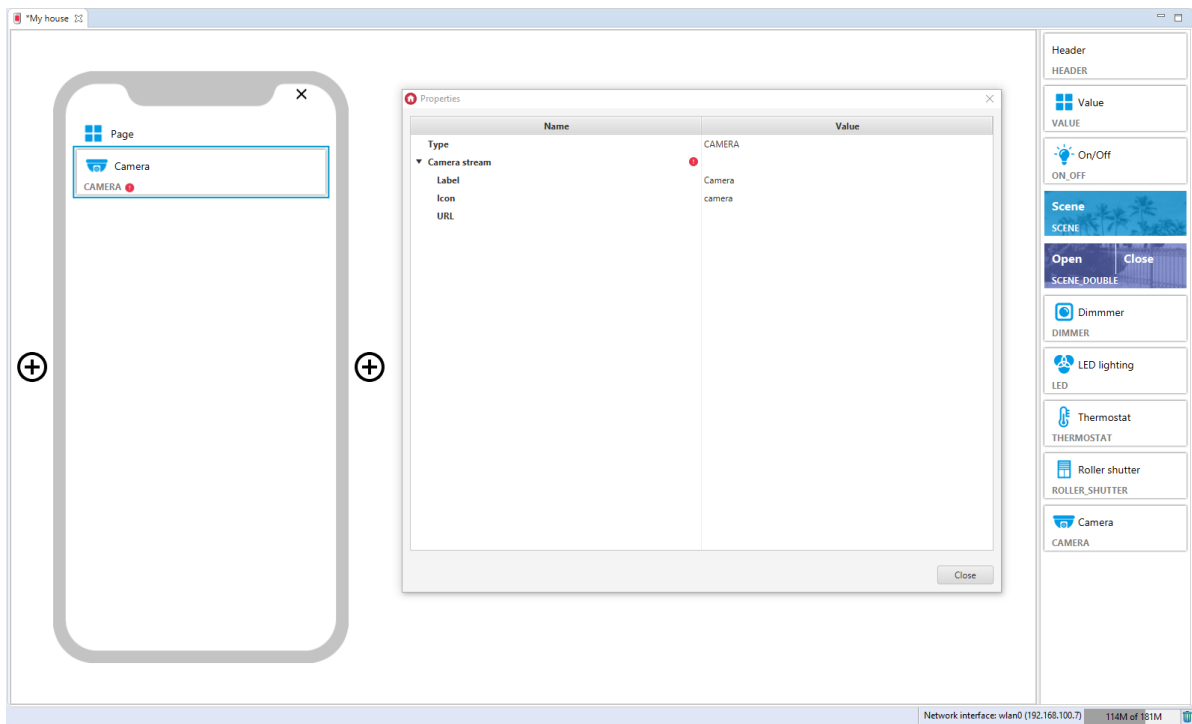




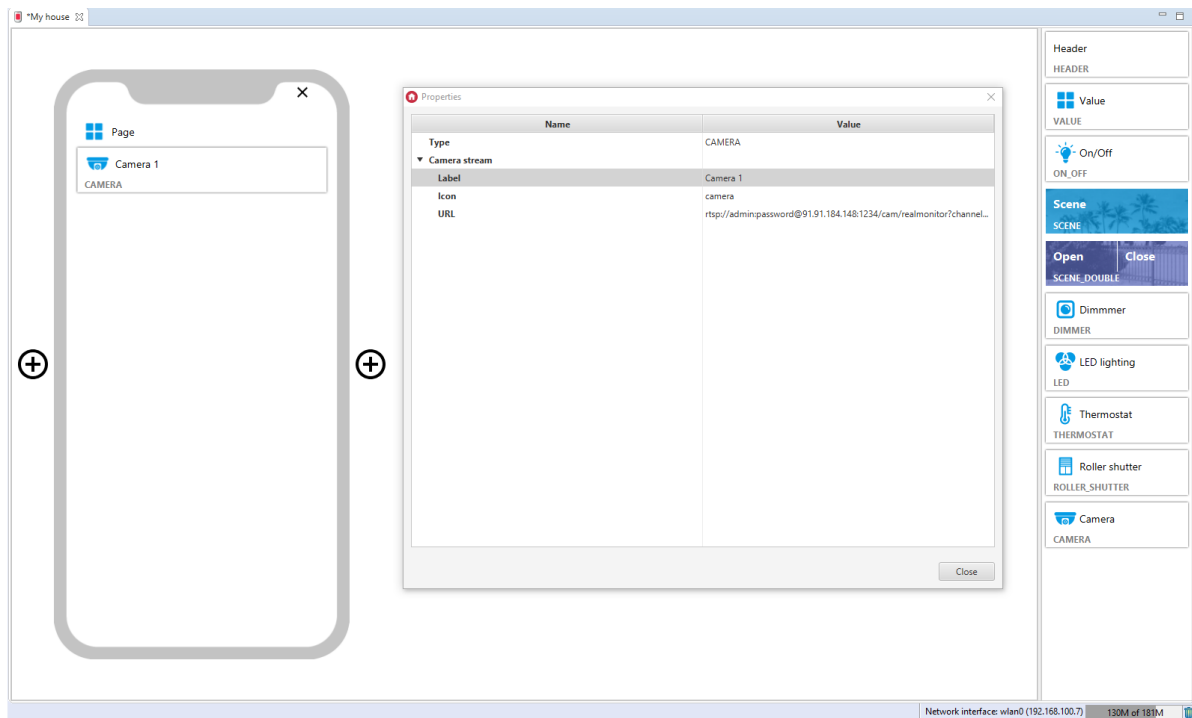
### 3.12. Camera

**NOTE!** The Camera widget is available for Object Manager version 1.3.5-204201 or higher, and for myGrenton application version 1.2.2 or higher (Android) and version 1.5.1 or higher (iOS).

Widget is used to display streaming video from IP camera. After dragging the widget from the tab on the right side of the screen, it should be completed with the value in the `URL` field (RTSP stream address for IP camera).



Ready widget:



## A. Camera configuration

The condition for the proper operation of the widget is to use the video stream with the RTSP protocol and the MJPG codec. Detailed information on camera configuration can be found in the documentation.

**NOTE!** It is recommended to use Dahua or Hikvision cameras. A configuration example is shown based on the settings for the models:

- Dahua IP CAMERA Model: IPC-HFW2531S-S-0280B-S2
- Hikvision NETWORK CAMERA Model: DS-2CD1043G0-I



## Example of how to get RTSP stream for Hikvision camera:

### 1. RTSP without authentication

```
rtsp: // < device IP address > : < RTSP port > / Streaming / channels / < channel number > < stream number > / ?transportmode=unicast
```

NOTE: < stream number > represents main stream (01) or sub stream (02)

Example:

```
rtsp://173.200.91.70:10554/Streaming/channels/101/?transportmode=unicast - get main stream for channel 1
```

```
rtsp://173.200.91.70:10554/Streaming/channels/102/?transportmode=unicast - get substream
```

### 2. RTSP with authentication

```
rtsp: // < username > : < password > @ < device IP address > : < RTSP port > / Streaming / channels / < channel number > < stream number > / ? transportmode=unicast
```

Example:

```
rtsp://admin:password@173.200.91.70:10554/Streaming/channels/101/?transportmode=unicast - get main stream for channel 1
```

```
rtsp://admin:password@173.200.91.70:10554/Streaming/channels/102/?transportmode=unicast - get substream
```

## Example of how to get RTSP stream for Dahua camera:

### 1. RTSP without authentication

```
rtsp: // < device IP address > : < RTSP port > / cam / realmonitor ? channel = < channel number > & subtype = < stream number >
```

Example:

```
rtsp://173.200.91.70:10554/cam/realmonitor?channel=1&subtype=1 - get main stream
```

### 2. RTSP with authentication

```
rtsp: // < username > : < password > @ < device IP address > : < RTSP port > / cam / realmonitor ? channel = < channel number > & subtype = < stream number >
```

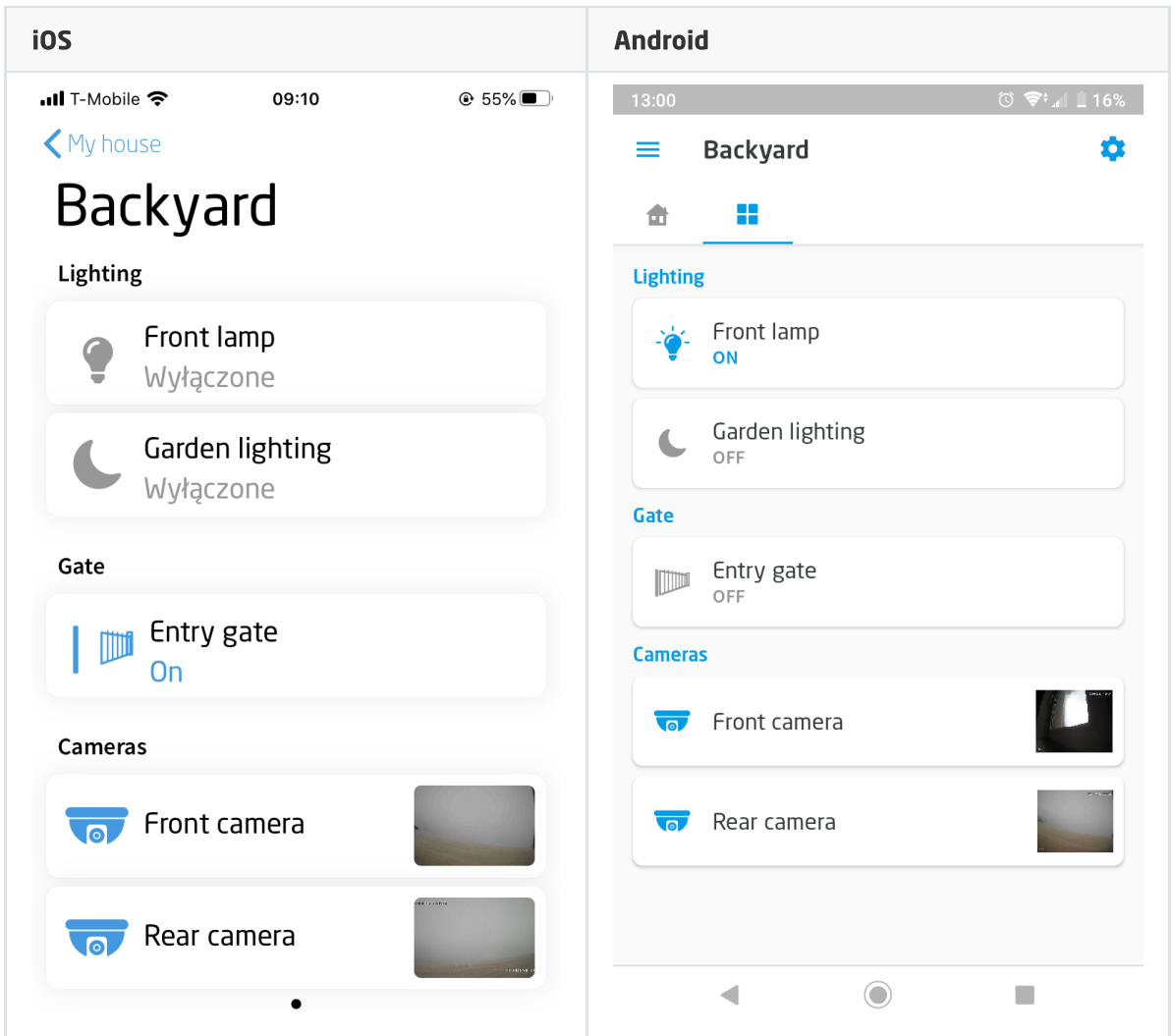
Example:

```
rtsp://admin:password@173.200.91.70:10554/cam/realmonitor?channel=1&subtype=1 - get main stream
```

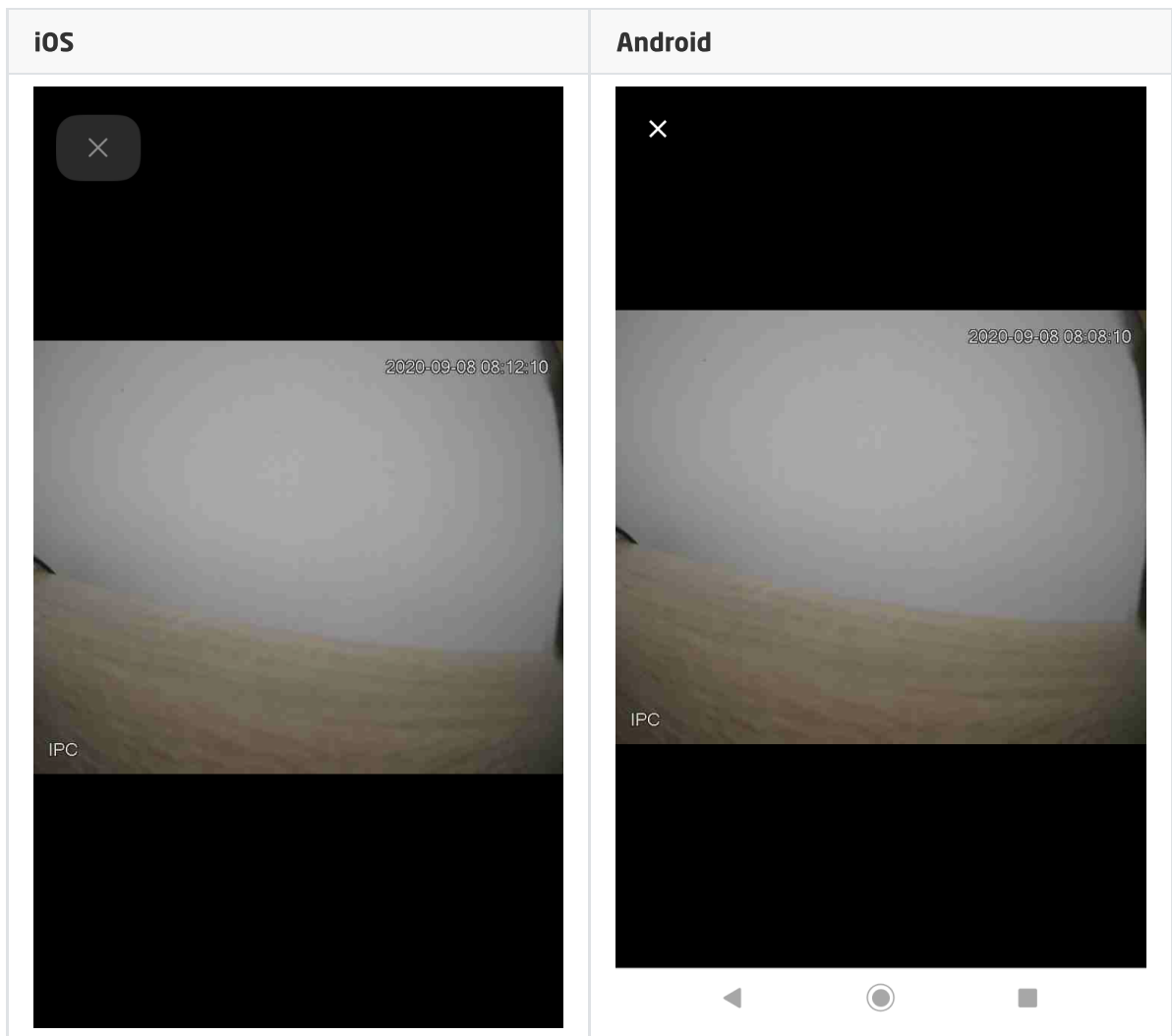
## B. Camera widget in the app

The stream from the camera is displayed after pressing the widget in the application. To close the preview, press the cross in the upper left corner of the screen or use the system back button. The thumbnail image from the camera is updated each time the stream from the camera is displayed.

Camera widget appearance:



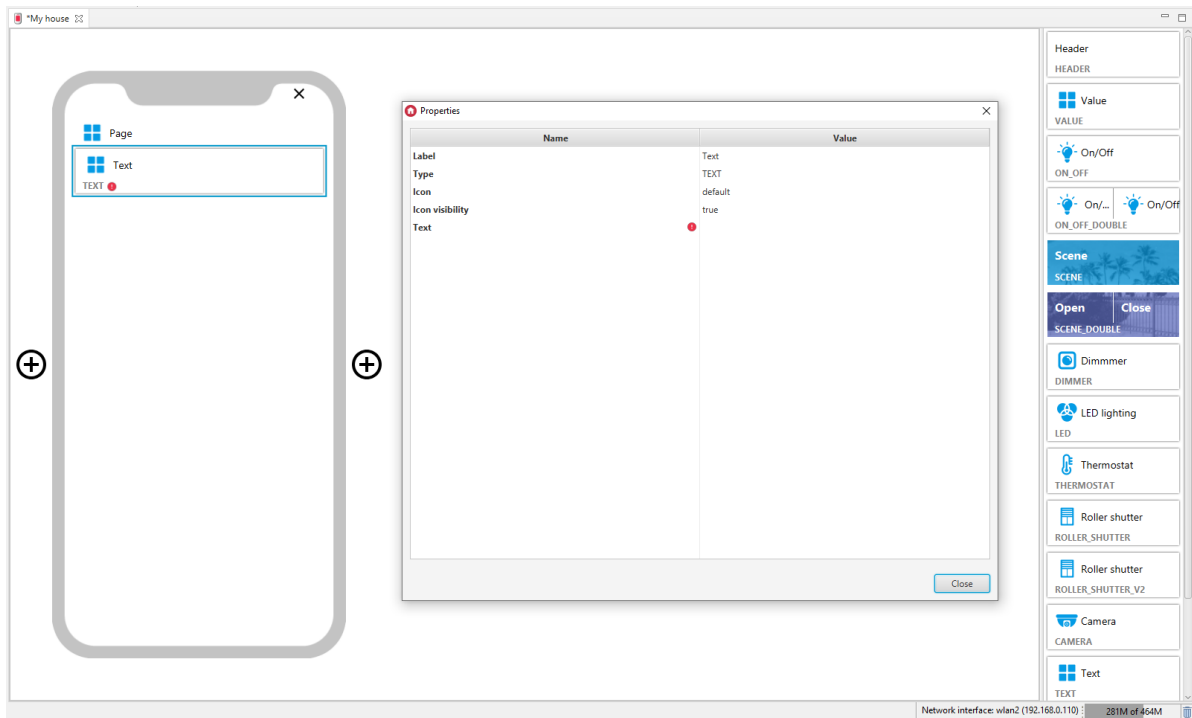
Camera stream preview view:



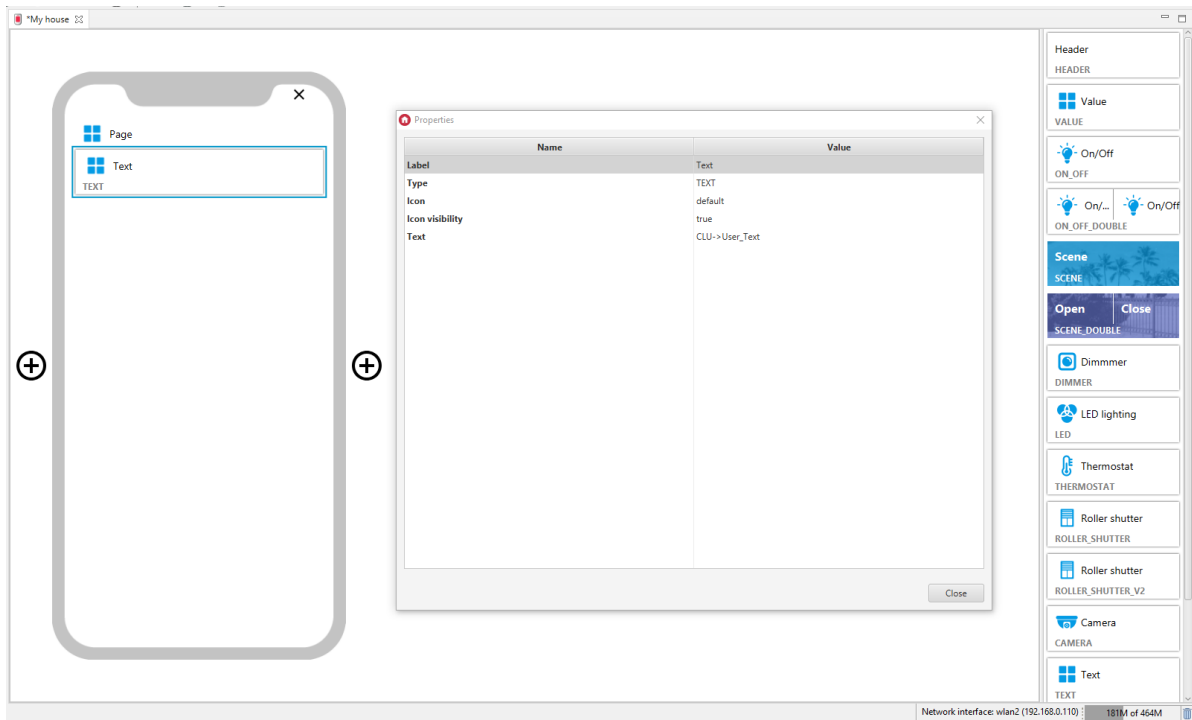
### 3.13. Text

**NOTE!** The TEXT widget is available for Object Manager version 1.5.0 or higher and for myGrenton application version 1.3.0 (Android) / 1.7.0 (iOS) or higher.

The widget is dedicated to display the value of a user feature or an embedded feature of a given object. The widget has a `Icon visibility` property that allows you to display or hide the widget icon in the application.



Filled TEXT widget:



The widget supports basic HTML language tags:

- `<br>` - Inserts a single line break,
- `<b>` - Bold font,
- `<i>` - Italic font,
- `<em>` - Highlight in the color of the interface theme.

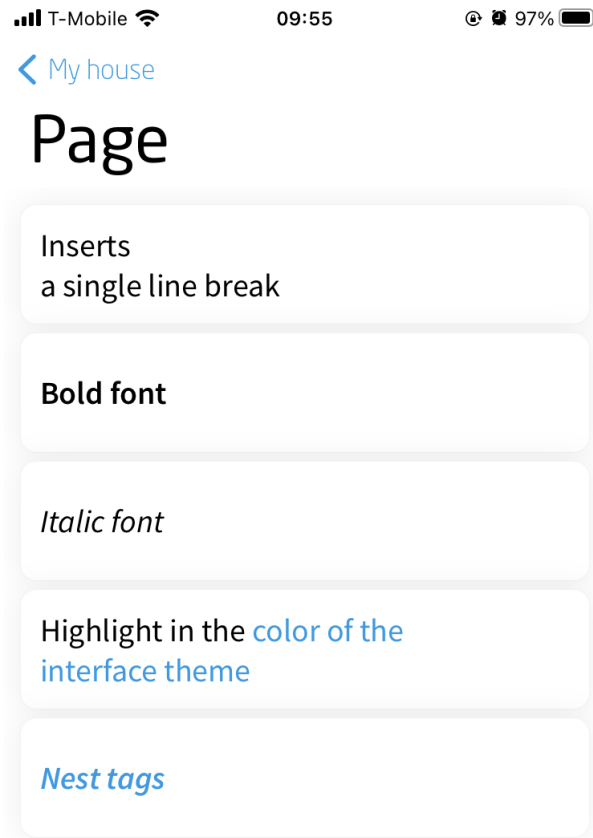
It is possible to nest tags.

Examples of the use of tags:

- Create user features of the String type, e.g:
  - `string_br` Initial value: `Inserts <br>a single line break` ,
  - `string_b` Initial value: `<b>Bold font</b>` ,
  - `string_i` Initial value: `<i>Italic font</i>` ,

- `string_em` Initial value: `Highlight in the <em>color of the interface theme</em>`.
- `string_nested` Initial value: `<b><i><em>Nest tags</em></i></b>`.
- Add the myGrenton interface containing configured TEXT widgets for the created user features.
- Send the interface to your mobile device.

Appearance in the application:



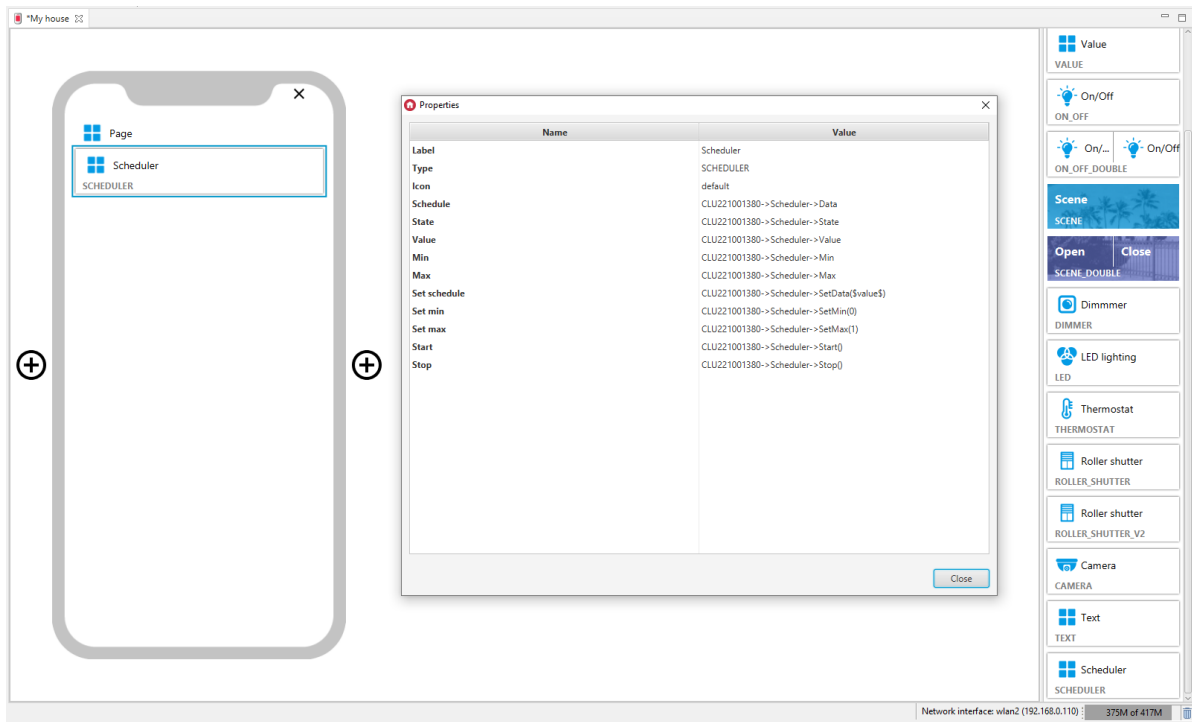
•

### 3.14. Scheduler

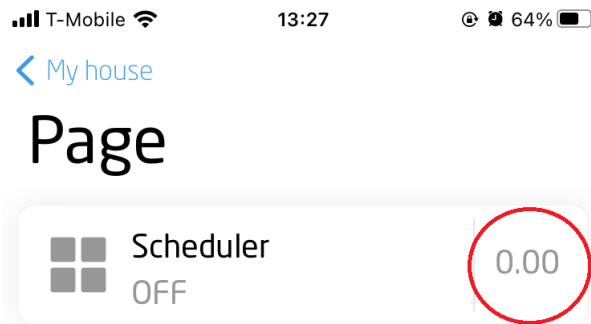
**NOTE!** The SCHEDULER widget is available for Object Manager version 1.5.0 or higher and for myGrenton application version 1.3.0 (Android) / 1.7.0 (iOS) or higher.

Widget dedicated for virtual objects of the scheduler type.





In the myGrenton application, you can edit the schedule. To do this, click on the widget value field:

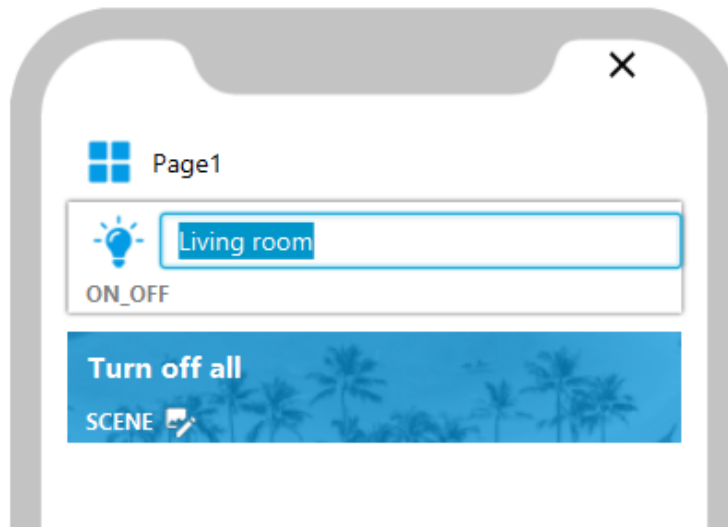


Schedule configuration has been described for the THERMOSTAT widget [look up XVIII.3:9.A.](#). Slight differences are due to the different characteristics of the virtual scheduler object.

## 3.15. Personalization of the widget

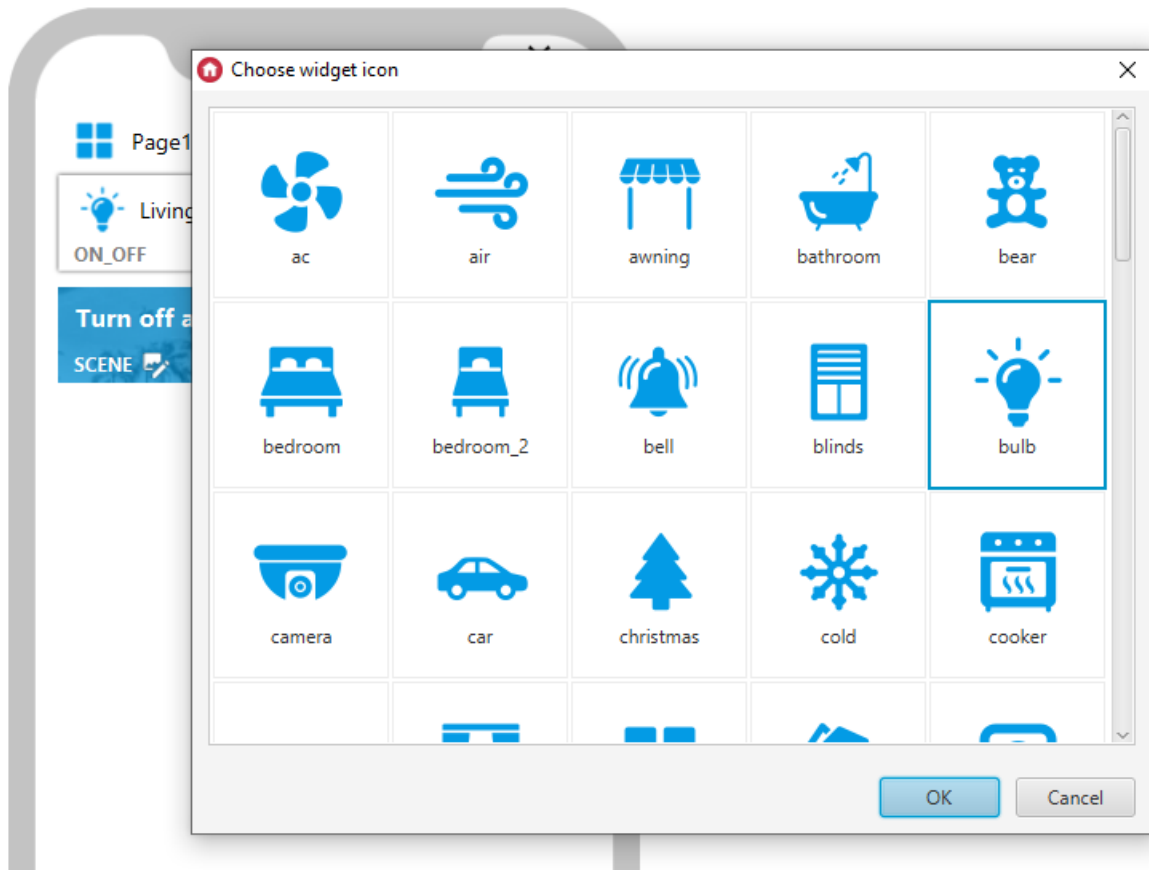
### A. Change the name

The widget name can be quickly changed by clicking on the current widget name. To confirm the change, press *Enter* on the keyboard. To cancel, press *Esc* on the keyboard.



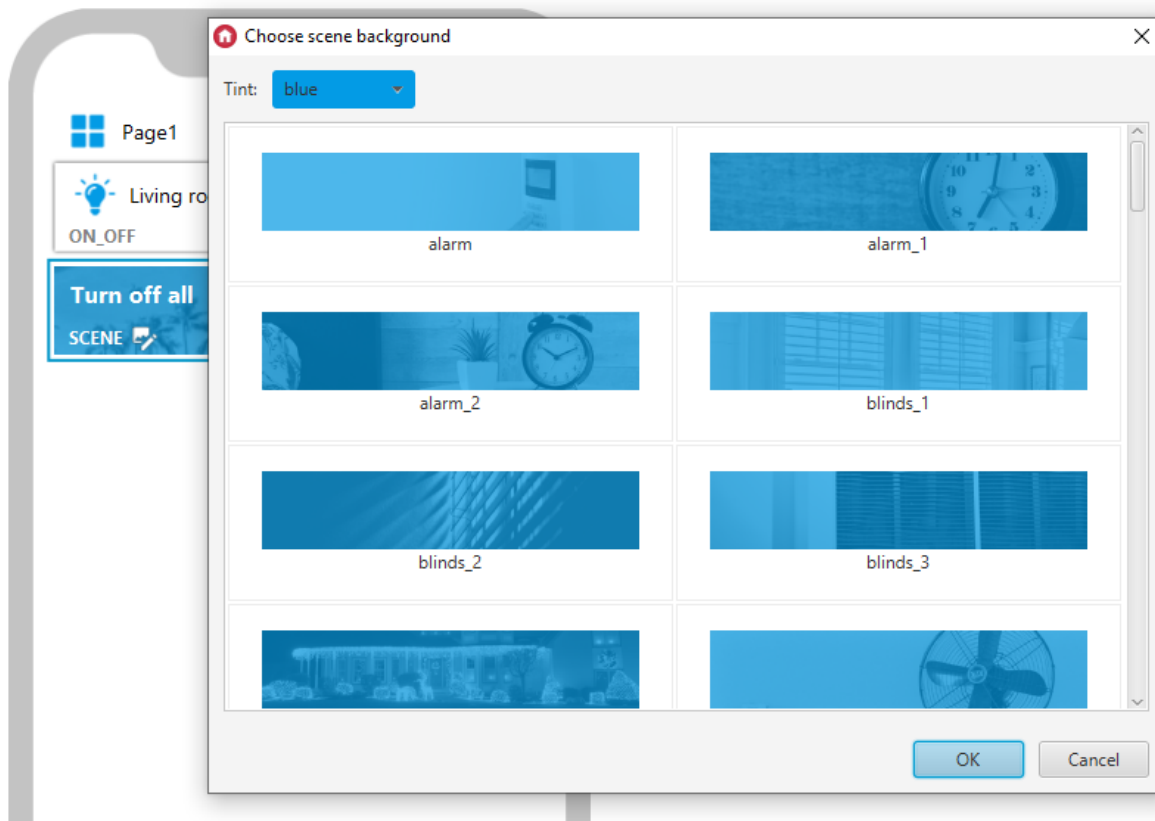
### B. Change icon / background picture

The widget icon can be quickly changed by clicking on the icon next to the widget name. Then a window with available icons will appear.





The background image of the Scene or Open / Close widget can be quickly changed by clicking on the edit icon under the widget name. Additionally, it is possible to choose the shade of the picture from the available 15 colors.



**NOTE!** If an icon or background image is not supported in a given version of myGrenton, it will be replaced with the default icon / background image.

### 3.16. Widget removal

To delete a widget, select it and press the `Delete` key or right-click and select the `Delete` option from the context menu. On the device with the macOS operating system installed, use the combination of `Fn` + `Backspace` characters.

### 3.17. Copying widgets

To copy a widget, select it and press the key combination `Ctrl` + `C` (then `Ctrl` + `V` in the destination) or choose the `Copy` option from the context menu (then RMB and `Paste` in the destination).

It is possible to select more widgets to copy. This can be done by selecting while holding down the `Ctrl` key, or by selecting a range while holding down the `Shift` key.

Widgets can be copied to all pages of the interface, also to pages of other interfaces within one project.

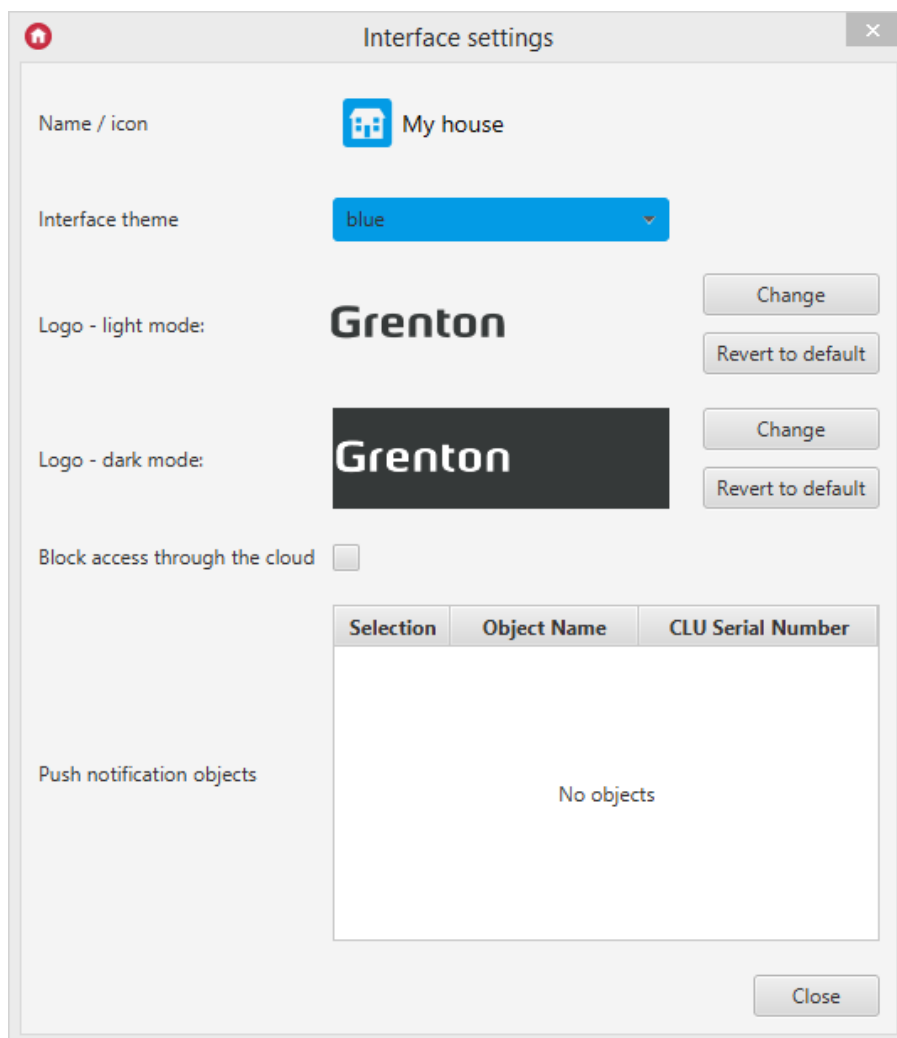
**NOTE!** If the widgets are copied to another project, they will have to be reconfigured.

## 4. Personalization of the interface

You can customize the appearance of the interface to your preferences. To personalize the interface, click the `MyGrenton interface settings` icon in the toolbar:

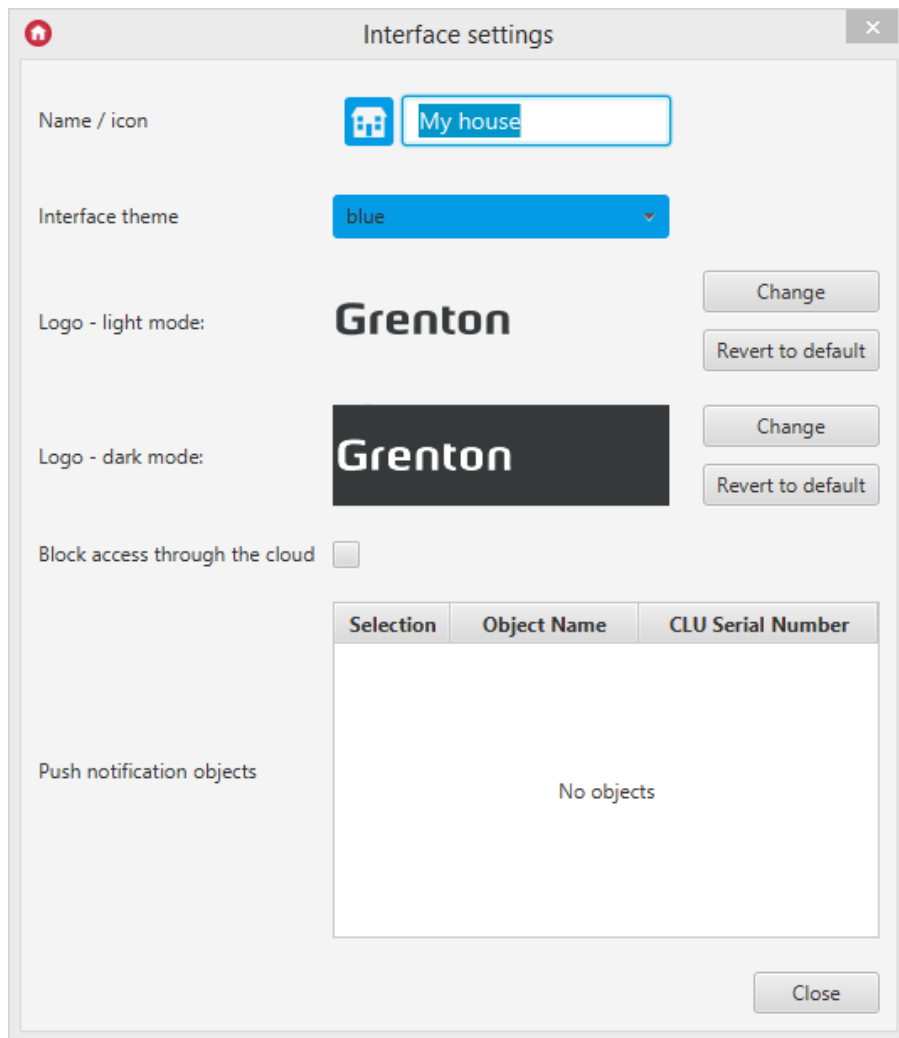


After clicking the icon, a window with the interface settings will appear:



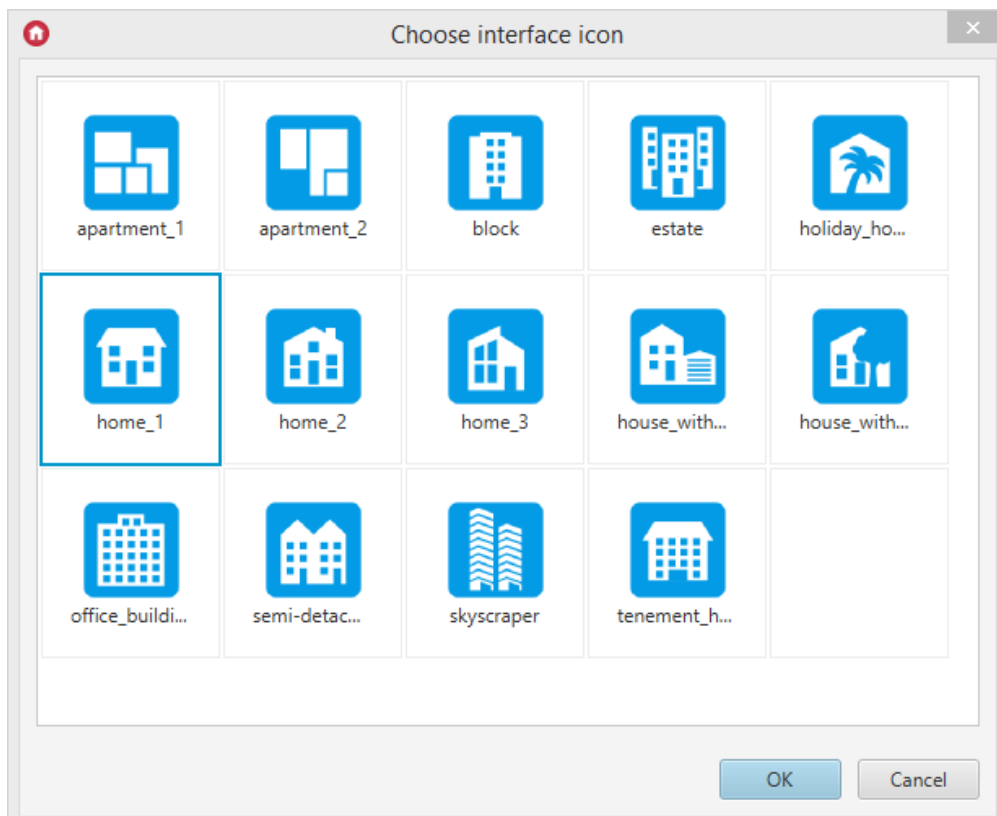
### 4.1. Change the name of the interface

To change the name of the interface, click on the current name displayed at the top of the window. To confirm the change, press *Enter* on the keyboard. To cancel, press *Esc* on the keyboard.



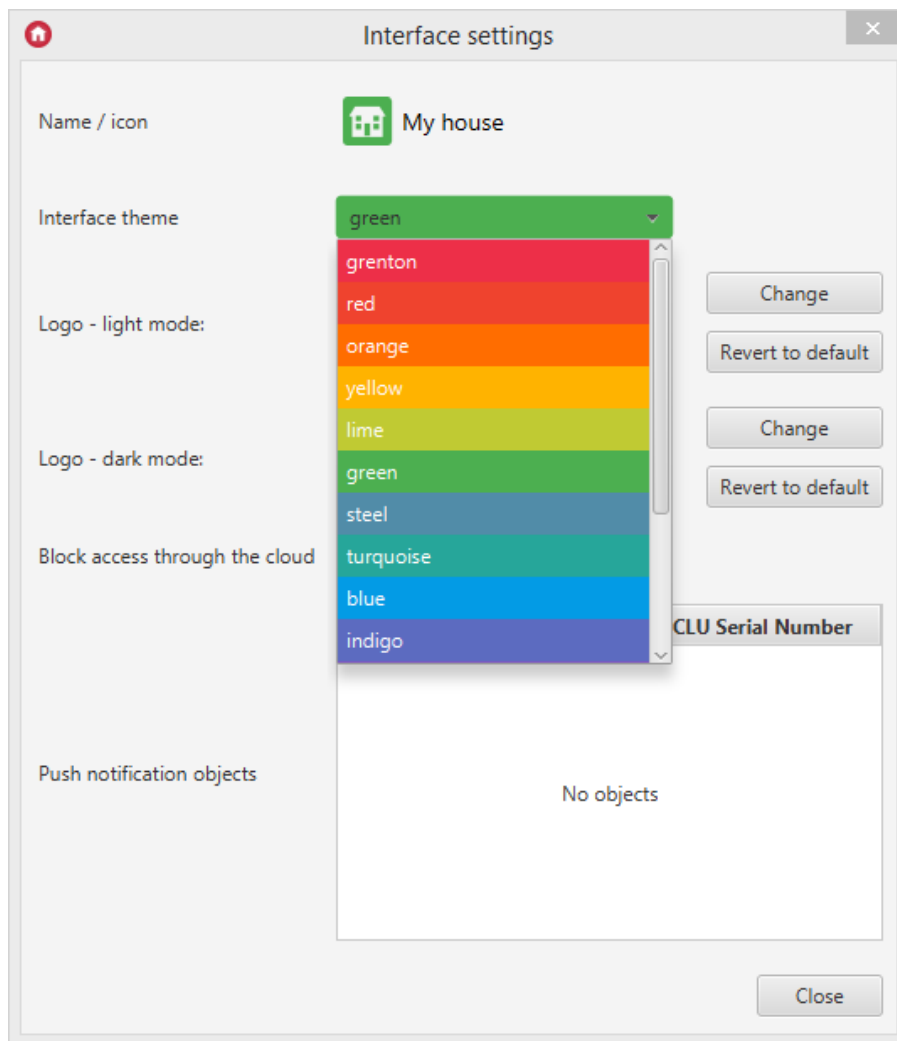
## 4.2. Change the interface icon

To change the interface icon, click on the icon image next to the interface name. A window with available icons will appear.



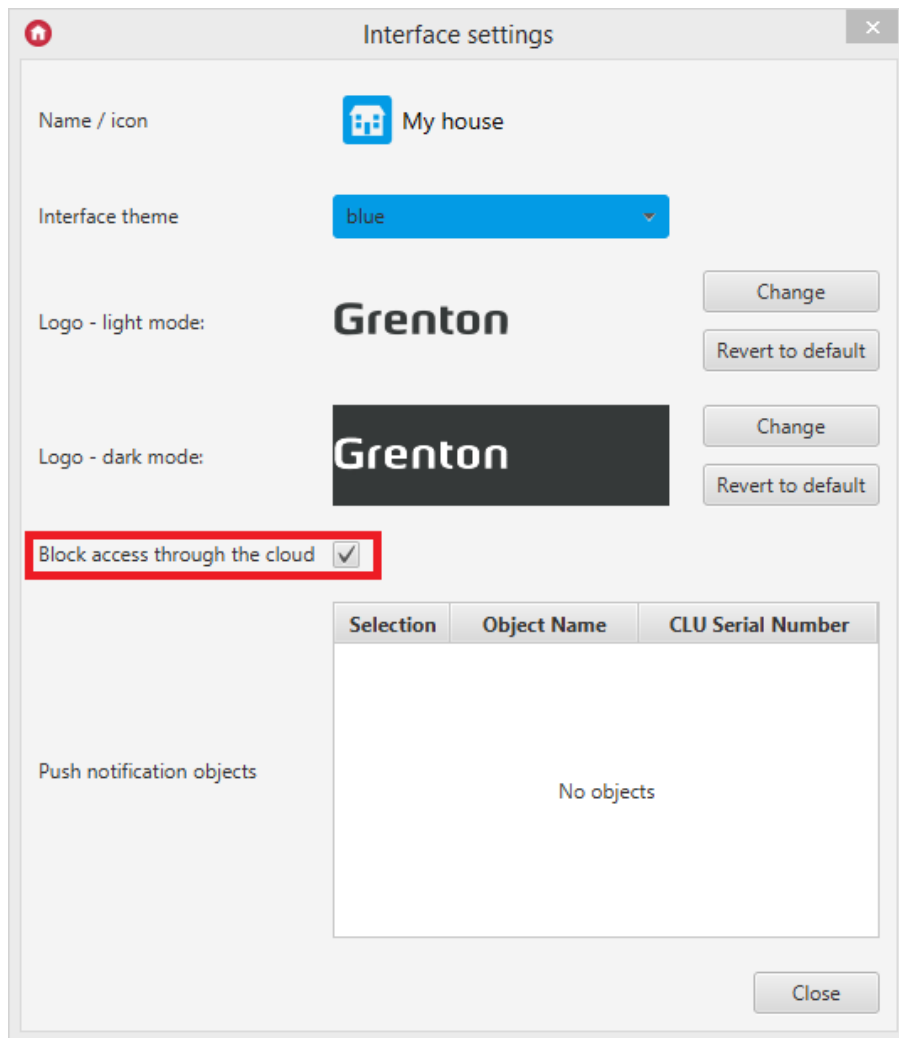
### 4.3. Change the color of the interface

It is also possible to change the color of the interface theme. There are 15 different theme colors available from the drop-down list.



### 4.4. Blocking access through the cloud

The option `Block access through the cloud` blocks the possibility of connecting the application to the system via the cloud. Then it will be possible to use the application only in the local connection mode.



## A. Android application behavior

After blocking access by the cloud and sending the configuration to the device, the connection mode with CLU for this interface will be set to local connection (otherwise, the connection mode via the cloud is set by default).

## Connection type

If this interface wasn't configured to communicate using Grenton Cloud, only local connection is supported.

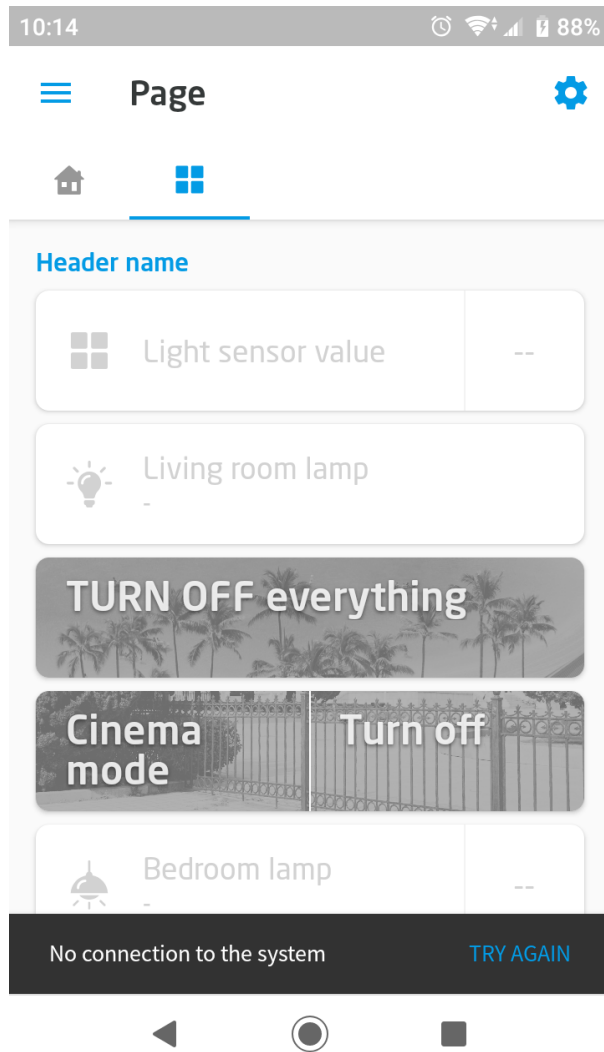
Local connection



Cloud connection

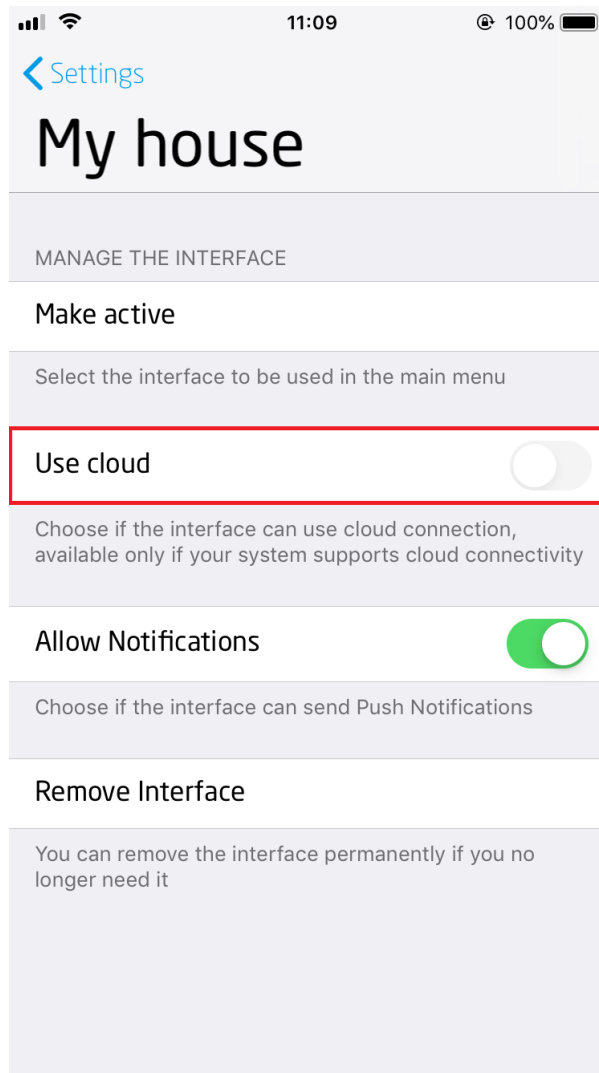


When changing the connection mode to connection via the cloud, connection with CLU will be impossible. A message will be displayed that there is no connection to the system.



## B. IOS application behavior

After blocking access by the cloud and sending the configuration to the device, it will not be possible to select the `Use cloud` option in this interface in the application.

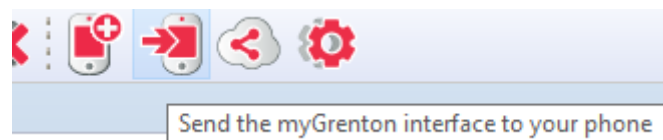


## 5. Sending the interface to the device

It is possible to send it to a mobile device in 2 ways:

### 5.1. Sending myGrenton interface to your phone using a QR code or manually


To send the interface to the phone, click the `Send the myGrenton interface to your phone` icon in the toolbar:



The window displayed allows you to send the interface by scanning the QR code or by providing interface parameters:



Send to your mobile device✕



### Now you can send the interface to your mobile device

Launch the myGrenton application on your mobile device, select the 'Add new interface' option from the menu and scan the QR code displayed on the left side of this text.

If you can not scan the QR code, select the manual interface loading mode in the myGrenton application and enter the following data:

**IP address:** 192.168.100.3  
**Port:** 9998  
**Token:** 566 710

**Note!**

- The computer with the Object Manager application and the mobile device with the myGrenton application must be connected to the same LAN.
- The dialog box must remain open until the transmission of the communication is complete.

Close

**NOTE!** The computer with the Object Manager application and the mobile device with the myGrenton application must be connected to the same LAN. The dialog window must remain open until the interface transfer is completed.

On the phone, select the option to scan the QR code or manual entry. If data is provided manually, the data displayed in the Object Manager window must be completed.

In the application for Android devices, after providing the correct data, select the option Load interface:



## Enter data to load interface

These data are necessary to load interface created in Object Manager Application.

IP Address

192.168.1.2

e.g. 192.168.1.2

Port

9998

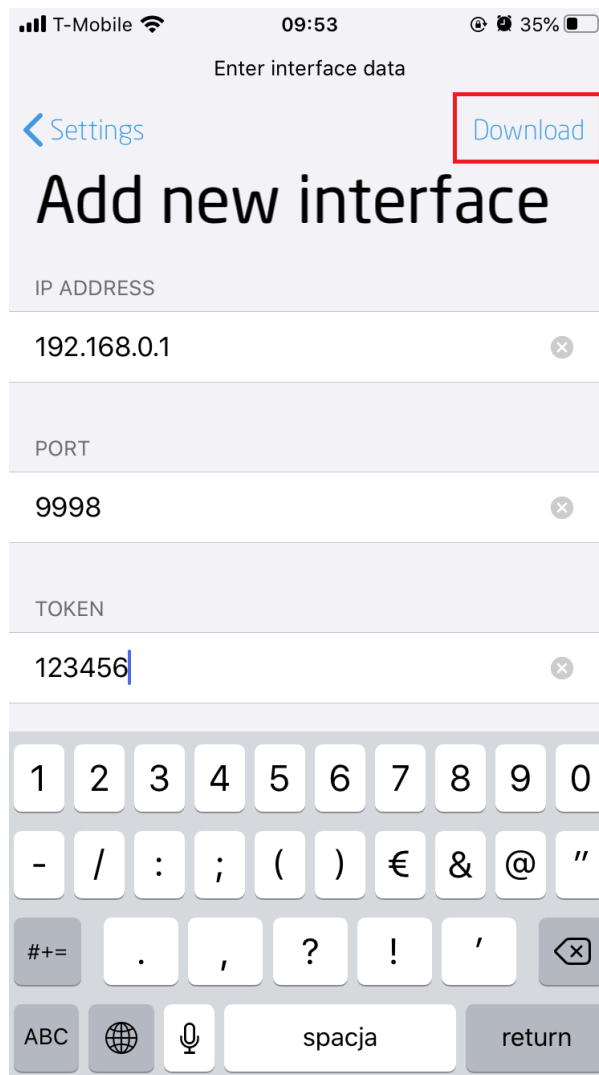
Token

123456

6 / 6

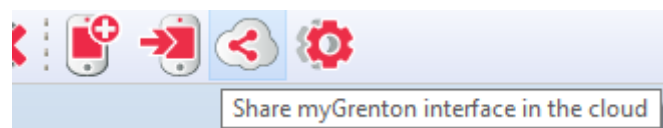
**LOAD INTERFACE**

In the application for iOS devices, select the `Download` option:

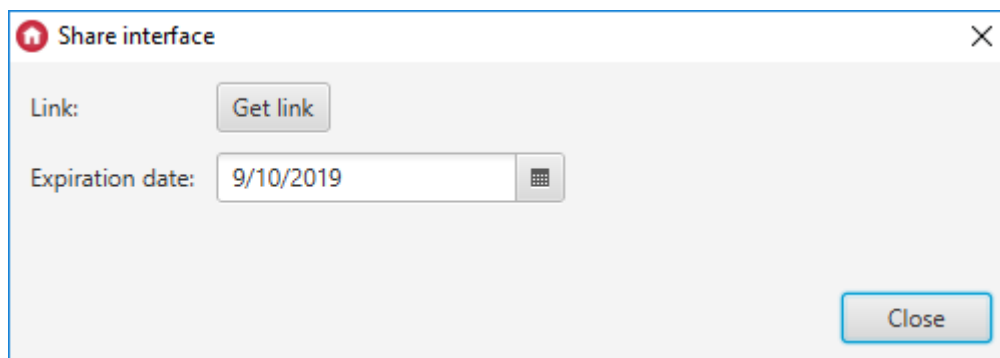


After correctly uploading the application will automatically launch the loaded interface.

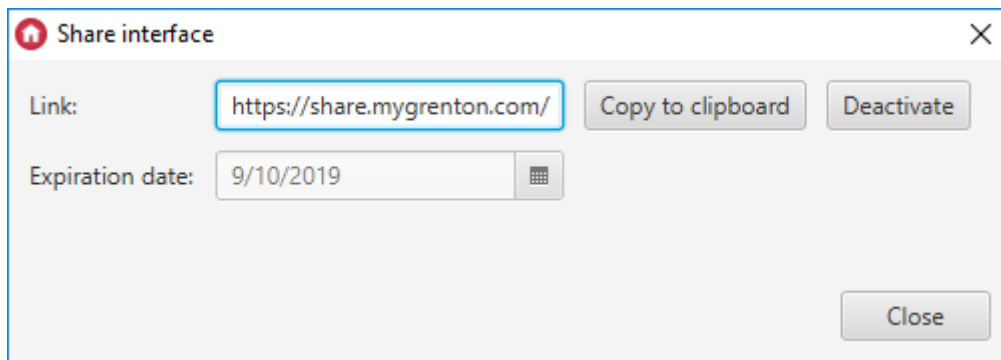
## 5.2. Sharing myGrenton interface via the cloud



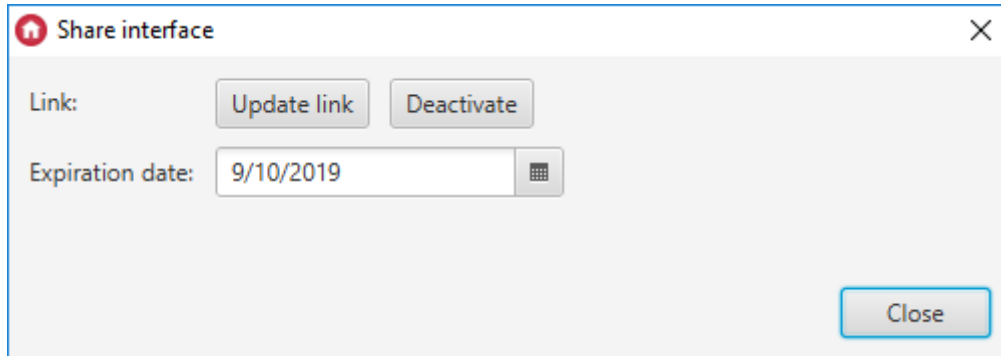
After clicking the icon [Share myGrenton interface in the cloud](#) the window for sharing the interface to the Grenton cloud will appear:



It is possible to set the expiration date of the link with the interface. After clicking [Get link](#) a link to the page with the generated interface appears



If the user has made changes to the interface, it is possible to update the shared interface:



After entering the shared link, a page will appear with two options for adding a new interface:

1. By opening the generated link in the browser of the Android phone (the myGrenton application will automatically open and add a new interface to it) or by entering the link in the myGrenton application on the iOS phone.
2. By scanning the created QR code in the myGrenton application.

# myGrenton Interface Sharing Link

## Option 1.

Open this link on your mobile, myGrenton compatible device.

<https://share.mygrenton.com/hihVTpRnfCsEZXXK7mbvAT9QIViwb5LJZ31AiEuW7r2V!>

Copy



## Option 2.

Scan the below QR Code using myGrenton application.



Don't have the app? Download it below.



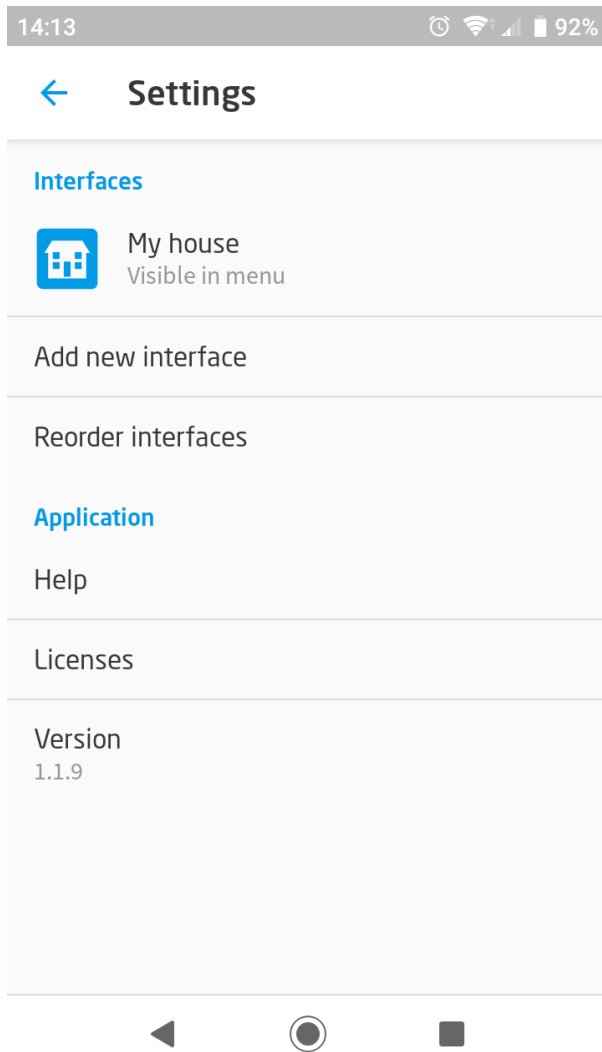
App Store and the Apple logo are trademarks of Apple Inc. Google Play and the Google Play logo are trademarks of Google LLC.

## 6. Application and interface settings

### 6.1. Application settings

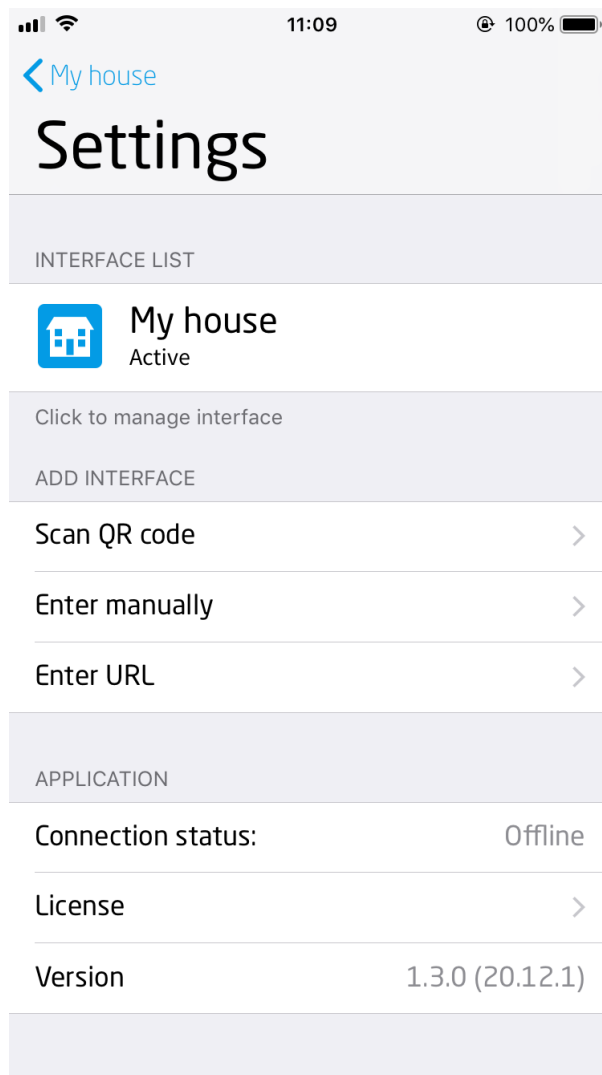
#### A. Android

In the settings, we have the option of adding a new interface and changing the order of interfaces. The `Help` button redirects you to the technical support page where you can find useful information about the Grenton system. `Licenses` takes the user to a new page where all the licenses used are listed. The `Version` field displays the version of the application being used.



## B. iOS

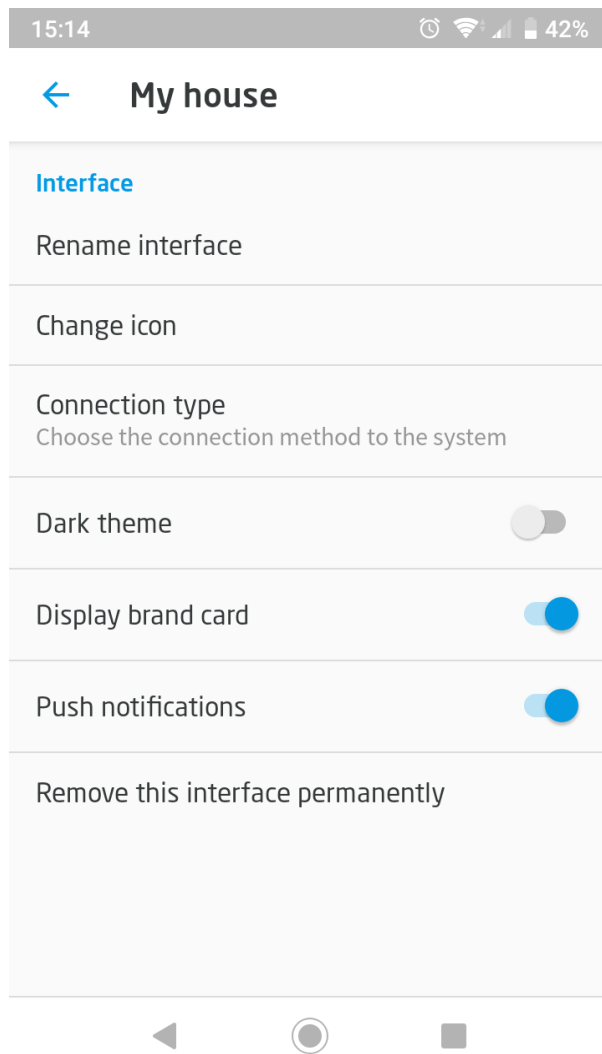
In the settings, we can add an interface by scanning the QR code, manually entering data or entering the URL address of the interface shared via the cloud. Here we can also see the current status of the connection to the system and the version of the current myGrenton application. The `License` button opens the page where the licenses are shown.



## 6.2. Interface Settings

### A. Android

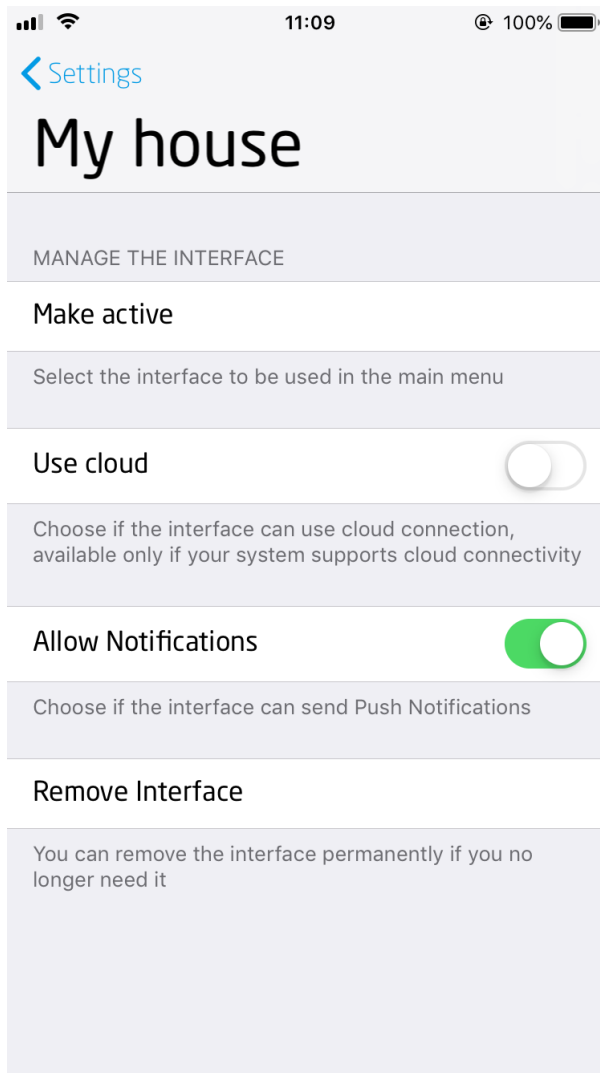
By clicking the interface you go to the interface settings. After entering the settings, you can customize its appearance. It is possible to change the interface name, change the icon, choose the type of connection, select the light or dark mode, display a card with a logo on the home page, set permissions to display push notifications and also remove the interface.



## B. iOS

By clicking the interface you go to the interface settings. There is an option to activate the interface. In the settings we can also choose the type of connection, enable push notifications or remove the interface.





**NOTE!** Required minimum CLU version for cloud support: **05.03.06**

**NOTE!** To connect correctly via the cloud, a CLU connection is required. To do this, set the parameter `UseCloud == true` and then send the configuration to CLU. Correct connection to the cloud will be signaled by the `cloudConnection == true` parameter.



**CLU properties**

Name:  Serial number:

IP:  FW:

Control  Events  **Embedded features**  User features

Feature name	Current value	Initial value	Unit	Range
Uptime	152		s	
Log	nil			
State	1			0,1,2,3,4,5,6,7,8
IsLocalPower	true		bool	
Date	2019-09-03			
Time	12:46:47			
Day	3			[1-31]
Month	9			[1-12]
Year	2019			
DayOfWeek	2			[0-6]
Hour	12		h	[0-23]
Minute	46		m	[0-59]
UnixTime	1567514807		s	
FirmwareVersion	05.03.09			
<b>UseCloud</b>	true	True ▾	bool	
<b>cloudConnection</b>	true		bool	

Auto refresh   Refresh

**NOTE!** If the CLU has connected to the cloud, the interface sent to the mobile application on the Android device will use the remote connection by default. To switch to local communication, select the local system connection type. For devices running on iOS, the UseCloud parameter is turned off by default.

14:29

🕒 📶 📶 📶 🔋 89%

## ✕ Connection type

If this interface wasn't configured to communicate using Grenton Cloud, only local connection is supported.

Local connection

Cloud connection



# XIX. Grenton 2.0 Logic Distribution

**NOTE!** The functionality is available only for modules from the Grenton 2.0 series!

The Grenton version 2.0 system has the functionality of distributed logic and connections. Thanks to this, it is possible for the modules to communicate directly with each other in order to trigger actions between the inputs and the outputs without using the CLU central module. In the absence of communication of the executive modules with the CLU unit or in the event of a CLU failure, the system can still function in the set configuration. The functionality is available for modules that have the Distributed Logic mode enabled. Ultimately, all Grenton 2.0 devices will support this functionality.

Distributed Logic setting is available for the following modules:

- GRENTON DIGITAL IN 6+3, DIN, TF-Bus (INP-209-D-01)
- GRENTON RELAY 4HP, DIN, TF-Bus (REL-204-D-01)
- GRENTON RELAY 2HP, DIN, TF-Bus (REL-202-D-01)
- GRENTON DIMMER MOSFET, DIN, TF-Bus (DIM-211-D-01)
- GRENTON I/O MODULE 8/8, DIN, TF-Bus (INO-288-D-01)
- GRENTON I/O MODULE 2/2, Flush, TF-Bus, 1-wire (INO-222-T-01)
- GRENTON ROLLER SHUTTER, DIN, TF-Bus (RSH-201-D-01)
- GRENTON ROLLER SHUTTER x3, DIN, TF-Bus (RSH-203-D-01)
- GRENTON ROLLER SHUTTER, Flush, TF-Bus (RSH-201-T-01)
- GRENTON LED RGBW, Flush, TF-Bus (RGB-042-T-16)
- GRENTON TOUCH PANEL 8B, TF-Bus (TPA-208-T-0X)
- GRENTON TOUCH PANEL 4B, TF-Bus (TPA-204-T-0X)
- GRENTON SMART PANEL 4B, OLED, TF-Bus (SPS-204-T-01)

## 1. Configuration of the Distributed Logic mode

**NOTE!** The modules on the first connection to the bus have Logic Distributed Mode enabled - value 1 of the `DistributedLogicGroup` (Default Mode) feature - the inputs can control the outputs, as described below. After performing CLU Discovery and sending the configuration, the mode turns off.

**NOTE!** The Smart Panel module on the first connection to the bus have Logic Distributed Mode enabled - value 1 of the `DistributedLogicGroup_1 - DistributedLogicGroup_4` (Default Mode) feature - assigned to four physical buttons - the inputs can control the outputs, as described below. After performing CLU Discovery and sending the configuration, the mode turns off.

### A. Mode configuration for IN / OUT modules

Object properties
✕

Name:

Id:

Type:

Device type:

Serial number:  1

Control
 User schemes
 Events
 Embedded features
 Statistics

Feature name	Current value	Initial value	Unit	Range
<b>Value</b>	0	Off <input type="button" value="v"/>	bool	0,1
<b>StatisticState</b>	0	Off <input type="button" value="v"/>	number	0,1
<b>VoltageType</b>	0	AC <input type="button" value="v"/>		0,1,2
<b>VoltageValue</b>	230	<input type="text" value="230"/>	V	[0-230]
<b>Power</b>	0		W	[0-3000]
<b>Overload</b>	3000	<input type="text" value="3000"/>	W	[0-3000]
<b>DistributedLogicGroup</b>	0	<input type="text" value="0"/>		[0-10000]

Auto refresh 
 Refresh

The configuration of Distributed Logic is the same as for any other system functionality through the OM application, and is defined for each IN / OUT object of a given module. The `DistributedLogicGroup` feature is used for this. The default value of `DistributedLogicGroup` is 0, which means that the mode is off.

If the connection of modules with CLU is lost and an event is detected for an input object (IN object of a given module), a message is sent to each output object (OUT object of a given module) that has the same value of the `DistributedLogicGroup` feature. As a result of receiving a message, the appropriate action assigned to the received event is triggered on the output object.

## B. Mode configuration for Smart Panel module

**Object properties**

Name:  Device type:

Id:  Serial number:

Type:

Control  User schemes  Events  Embedded features  Statistics

Feature name	Current value	Initial value	Unit	Range
PageType	1	<input type="text" value="Buttons"/>		0,1,2,3
PageName	-	<input type="text" value=""/>	-	[0-15]
Object_1_Id	1	<input type="text" value="1"/>	-	[0-23]
Object_1_Name	-	<input type="text" value=""/>	-	[0-15]
Object_1_CustomIcon	-	<input type="text" value=""/>		[0-9]
DistributedLogicGroup_1	1	<input type="text" value="1"/>		[0-10000]
Object_2_Id	2	<input type="text" value="2"/>	-	[0-23]
Object_2_Name	-	<input type="text" value=""/>	-	[0-15]
Object_2_CustomIcon	-	<input type="text" value=""/>		[0-9]
DistributedLogicGroup_2	2	<input type="text" value="2"/>		[0-10000]

Auto refresh

For Smart Panel module, the binding on the input takes place in the `PANEL_PAGEx` object. Before that, `PANEL_PAGEx` objects must be linked with the appropriate `PANEL_BUTTONx` objects by setting the appropriate values in the `Object_x_Id` field. If there is no connection to the CLU, in Distributed Logic mode all 16 buttons can work with the change of pages using gestures.

**NOTE!** Distributed Logic is only possible in the `Buttons` mode of the `PANEL_PAGEx` object.

## 1.1. Operation of Distributed Logic between DIN and output objects

**NOTE!** Events and triggered actions are set statically and cannot be changed.

Available actions while running in the Distributed Logic Mode:

### A. DIN and DOUT objects

- Switching on the DIN input object (SwitchOn) -> Switching on the given DOUT output.
- Switching off the DIN input object (SwitchOff) -> Switching off the given DOUT output.
- Short change of state of an input object DIN (Click) -> Change of state to the opposite of the given DOUT output.

### B. DIN and DIMM objects

- Switching on the DIN input object (SwitchOn) -> Switching on the given DIMM output.
- Switching off the DIN input object (SwitchOff) -> Switching off the given DIMM output.
- Short change of state of an input object DIN (Click) -> Change of state to the opposite of the given DIMM output.

## C. DIN and LEDRGBW objects

- Switching on the DIN input object (SwitchOn) -> Switching on the given LEDRGBW channel.
- Switching off the DIN input object (SwitchOff) -> Switching off the given LEDRGBW channel.
- Short change of state of an input object DIN (Click) -> Change of state to the opposite of the given RGBW LED channel.

## D. DIN and ROLLER\_SHUTTER objects

- Switching on the DIN input object (SwitchOn) -> Switching on the UP or DOWN ROLLER\_SHUTTER output, depending on the previous direction.
- Switching off the DIN input object (SwitchOff) -> Switching off the given connected output (UP or DOWN).
- Short change of state of an input object DIN (Click) -> Change of state to opposite ROLLER\_SHUTTER:
  - if the relays (UP / DOWN) are turned off - switching on the UP or DOWN relay, depending on the previous direction,
  - if the UP or DOWN relay is on - the relay is turned off.

**NOTE!** Switching the UP or DOWN relay means switching the relay on without switching off after the MaxTime time has elapsed. The relays should be turned off with a given Roller Shutter control input in the Distributed Logic mode.

## 1.2. Operation of Distributed Logic between BUTTON and output objects

**NOTE!** Events and triggered actions are set statically and cannot be changed.

Available actions while running in the Distributed Logic mode:

### A. BUTTON and DOUT objects

- Press the button (SwitchOn) -> Switching on the given DOUT output.
- Release the button (SwitchOff) -> Switching off the given DOUT output.
- Short button press (Click) -> Change of state to the opposite of the given DOUT output.

### B. BUTTON and DIMM objects

- Press the button (SwitchOn) -> Switching on the given DIMM output.
- Release the button (SwitchOff) -> Switching off the given DIMM output.
- Short button press (Click) -> Change of state to the opposite of the given DIMM output.

### C. BUTTON and LEDRGBW objects

- Press the button (SwitchOn)-> Switching on the given LEDRGBW channel.
- Release the button (SwitchOff)-> Switching off the given LEDRGBW channel.
- Short button press (Click) -> Change of state to the opposite of the given RGBW LED channel.

## D. BUTTON and ROLLER\_SHUTTER objects

- Press the button (SwitchOn) -> Switching on the UP or DOWN ROLLER\_SHUTTER output, depending on the previous direction.
- Release the button (SwitchOff) -> Switching off the given connected output (UP or DOWN).
- Short button press (Click) -> Change of state to opposite ROLLER\_SHUTTER:
  - if the relays (UP / DOWN) are turned off - switching on the UP or DOWN relay, depending on the previous direction,
  - if the UP or DOWN relay is on - the relay is turned off.

**NOTE!** Switching the relay UP or DOWN means switching on the relay without switching off after the MaxTime time has elapsed. The relays should be turned off with a given control input of the ROLLER\_SHUTTER object in the Distributed Logic mode.

## 1.3. Operation of Distributed Logic between PANEL\_PAGE with PANEL\_BUTTON and output objects

**NOTE!** Events and triggered actions are set statically and cannot be changed.

Available actions while running in the Distributed Logic mode:

### A. PANEL\_PAGE with PANEL\_BUTTON object set and DOUT objects

- Press the button (SwitchOn) -> Switching on the given DOUT output.
- Release the button (SwitchOff) -> Switching off the given DOUT output.
- Short button press (Click) -> Change of state to the opposite of the given DOUT output.

### B. PANEL\_PAGE with PANEL\_BUTTON object set and DIMM objects

- Press the button (SwitchOn) -> Switching on the given DIMM output.
- Release the button (SwitchOff) -> Switching off the given DIMM output.
- Short button press (Click) -> Change of state to the opposite of the given DIMM output.

### C. PANEL\_PAGE with PANEL\_BUTTON object set and LEDRGBW objects

- Press the button (SwitchOn)-> Switching on the given LEDRGBW channel.
- Release the button (SwitchOff)-> Switching off the given LEDRGBW channel.
- Short button press (Click) -> Change of state to the opposite of the given RGBW LED channel.

### D. PANEL\_PAGE with PANEL\_BUTTON object set and ROLLER\_SHUTTER objects

- Press the button (SwitchOn) -> Switching on the UP or DOWN ROLLER\_SHUTTER output, depending on the previous direction.
- Release the button (SwitchOff) -> Switching off the given connected output (UP or DOWN).
- Short button press (Click) -> Change of state to opposite ROLLER\_SHUTTER:
  - if the relays (UP / DOWN) are turned off - switching on the UP or DOWN relay, depending on the previous direction,
  - if the UP or DOWN relay is on - the relay is turned off.



**NOTE!** Switching the relay UP or DOWN means switching on the relay without switching off after the MaxTime time has elapsed. The relays should be turned off with a given control input of the ROLLER\_SHUTTER object in the Distributed Logic mode.

## 2. Default Mode

If the value of the `DistributedLogicGroup` feature is set to 1 for a given object, it works in the **Default Mode**. This is a special operating mode set by default for each object:

**NOTE!** The events and triggered actions are statically set and cannot be changed.

### 2.1. Default Mode for input modules and output modules

- input module (DIGITAL IN 6+3 DIN) - controls all output modules (RELAY 4HP DIN, RELAY 2HP DIN, DIMMER MOSFET DIN, ROLLER SHUTTER DIN, ROLLER SHUTTER DIN) in TFBUS network, which are also in Default Mode, for example:
  - DIGITAL IN1 --> RELAY 4HP OUT1 | RELAY 2HP OUT1 | DIMMER MOSFET DIMM1 | ROLLER\_SHUTTER1.
  - DIGITAL IN2 --> RELAY 4HP OUT2 | RELAY 2HP OUT2 | ROLLER\_SHUTTER2.
  - DIGITAL IN3 --> RELAY 4HP OUT3 | ROLLER\_SHUTTER3.
  - DIGITAL IN4 --> RELAY 4HP OUT4.
- Touch Panel module (TOUCH PANEL 4B, TOUCH PANEL 8B) - controls all output modules (RELAY 4HP DIN, RELAY 2HP DIN, DIMMER MOSFET DIN, ROLLER SHUTTER DIN) in TFBUS network, which are also in Default Mode, for example:
  - TOUCH PANEL BUTTON1 --> RELAY 4HP OUT1 | RELAY 2HP OUT1 | DIMMER MOSFET DIMM1 | ROLLER\_SHUTTER1.
  - TOUCH PANEL BUTTON2 --> RELAY 4HP OUT2 | RELAY 2HP OUT2 | ROLLER\_SHUTTER2.
  - TOUCH PANEL BUTTON3 --> RELAY 4HP OUT3 | ROLLER\_SHUTTER3.
  - TOUCH PANEL BUTTON4 --> RELAY 4HP OUT4.
- Smart Panel module - controls all output modules (RELAY 4HP DIN, RELAY 2HP DIN, DIMMER MOSFET, ROLLER SHUTTER DIN) in TFBUS network, which are also in Default Mode, for example:
  - SMART PANEL PANEL\_PAGE1 feature `DistributedLogicGroup_1` --> RELAY 4HP OUT1 | RELAY 2HP OUT1 | DIMMER MOSFET DIMM1 | ROLLER\_SHUTTER1.
  - SMART PANEL PANEL\_PAGE1 feature `DistributedLogicGroup_2` --> RELAY 4HP OUT2 | RELAY 2HP OUT2 | ROLLER\_SHUTTER2.
  - SMART PANEL PANEL\_PAGE1 feature `DistributedLogicGroup_3` --> RELAY 4HP OUT3 | ROLLER\_SHUTTER3.
  - SMART PANEL PANEL\_PAGE1 feature `DistributedLogicGroup_4` --> RELAY 4HP OUT4.

**NOTE!** Switching the relay UP or DOWN means switching on the relay without switching off after the MaxTime time has elapsed. The relays should be turned off with a given control input of the ROLLER\_SHUTTER object in the Distributed Logic mode.

#### **NOTE!**

For a Smart Panel module running in the Distributed Logic Default Mode, the behavior described above is identical for any PANEL\_PAGE object (PANEL\_PAGE1 - PANEL\_PAGE4).

## 2.2. Default Mode for modules with their own inputs / outputs

- module with inputs and outputs (I/O MODULE DIN 8, I/O MODULE FM, ROLLER SHUTTER FM) - controls its own channels with the appropriate number (IN1-> OUT1, IN2-> OUT2, etc.) for example:
  - I/O MODULE FM IN1 -> I/O MODULE FM OUT1 | ROLLER SHUTTER.
  - I/O MODULE FM IN2 -> I/O MODULE FM OUT2.
  - ROLLER SHUTTER FM IN1 --> ROLLER\_SHUTTER1.
- LED RGBW FM module - controls its own channels (Red, Green):
  - LED RGBW FM IN1 --> LED RGBW FM R-channel.
  - LED RGBW FM IN2 --> LED RGBW FM G-channel.

## 3. Restoring communication between the CLU and the module

---

When communication between the CLU and modules is restored, the value of the Value property of the given objects is updated to the actual value of the I / O (changed to the value changed during Logic Distributed Mode operation) and the modules perform actions in accordance with the CLU's programmed logic.

---

1. Depending on the type of router used, its interface may differ from the general port configuration instruction. [↪](#)
2. This is the default port for the camera stream *rtsp*. [↪](#)
3. Its IP address can be found in the list of currently connected devices in the router's interface. [↪](#)
4. Depending on what type of device is in use, its configuration may differ from the one provided in the manual. [↪](#)
5. In addition to the connection settings in the same section, you can check the box that determines the use of the hands-free mode after receiving a call [↪](#)
6. Where X and Y are the CLU names. [↪](#)