Grenton

User Manual

Grenton 2

Document version: 1.0.19

Date: 2025-05-12

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. Creating new project
4. CLU Discovery function
    4.1. Adding modules to the project
    4.2. Replacing / Reassigning modules during Discovery process
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. Removing Z-Wave modules
    6.2. Removing 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
    2.4. Find / Replace function
    2.5. Copying scripts
3. Copying virtual objects
4. Date and time

VIII. Visual Builder - Smartphone control

Important information - end of support for functionality of Visual Builder
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. Thermostat
5. Push
6. Presence Sensor
7. Sunrise and Sunset Calendar
8. Event Scheduler
9. MultiFanACThermostat

X. Media measurement

Important information - end of support for the Virtual Media measurement functionality
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
   2.1. Time setting via NTP server
3. Virtual objects
   3.1. Modbus RTU protocol
   3.2. Modbus TCP protocol
4. Register parameters
   4.1. ModbusRTU and ModbusClient object
   4.2. ModbusSlaveRTU and ModbusServer object
5. Restoring factory settings - *Hard Reset*
6. 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)
   2.1.3. Setting Request Headers / Reading Response Headers
   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
   2.2.4. Setting Response Headers / Reading Request Headers
   2.3.1. Timer
   2.4.1. Sonos
   2.5.1. MusicCast
   2.6.1. CoolMasterNet
   2.7.1. CoolMaster
   2.8.1. HEOS
   2.9.1. DenonMarantzAVR
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. FAKRO ZWMR 24
    14.1. General information
    14.2. Objects
15. FAKRO ZWS 230
    15.1. General information
    15.2. Objects
16. Fibaro RGBW
    16.1. General information

16.2. Objects
17. Remotec ZXT-120
17.1. General information
17.2. Description of device configuration
17.3. Objects
18. Remotec ZXT-310
18.1. General information
18.2. Device configuration
18.3. Objects
19. Aeotec Nano Switch
19.1. General information
19.2. Objects
20. Aeotec Dual Nano Switch
20.1. General information
20.2. Objects
21. Aeotec Nano Dimmer
21.1. General information
21.2. Objects
22. Aeotec Nano Shutter
22.1. General information
22.2. Objects
23. Aeotec Nano Shutter (V2)
23.1. General information
23.2. Objects
24. Aeotec Smart Switch 7
24.1. General information
24.2. Objects
25. Aeotec Multisensor 6
25.1. General information
25.2. Objects

XVIII. myGrenton mobile application

1. Installation and first launch of the myGrenton application
1.1. Installation
1.2. First start-up, demonstration interface
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 (HEADER)
3.2. Value (VALUE)
3.3. Value v2 (VALUE_V2)
3.4. Value Double (VALUE_DOUBLE)
3.5. On/Off (ON_OFF)
3.6. On/Off Double (ON_OFF_DOUBLE)
3.7. Scene (SCENE)
3.8. Scene Double (SCENE_DOUBLE)
3.9. Dimmer (DIMMER)
3.10. Dimmer v2 (DIMMER_V2)
3.11. LED Lighting (LED)
3.12. Thermostat (THERMOSTAT)
3.13. Thermostat v2 (THERMOSTAT_V2)
3.14. Roller Shutter (ROLLER_SHUTTER)
3.15. Roller Shutter v2 (ROLLER_SHUTTER_V2)
3.16. Roller Shutter v3 (ROLLER_SHUTTER_V3)
3.17. Camera (CAMERA)
3.18. Text (TEXT)

3.19. Scheduler (SCHEDULER)
3.20. Event Scheduler (EVENT_SCHEDULER)
3.21. Multisensor (MULTISENSOR)
3.22. TV Remote Control (TV_REMOTE_CONTROL)
3.23. Audio Remote Control (AUDIO_REMOTE_CONTROL)
3.24. Contact Sensor (CONTACT_SENSOR)
3.25. Contact Sensor Double (CONTACT_SENSOR_DOUBLE)
3.26. Slider (SLIDER)
3.27. Air conditioning (COOL_MASTER)
3.28. Personalization of the widget
3.29. Widget removal
3.30. Copying widgets
3.31. Run the SCENE widget using Shortcuts
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
6.3. Interface and widget lock settings
6.4. Intercom 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

XX. RS232 Controller

1. General information
2. Example of usage in scripts
2.1. Sending a command to the device without waiting for a response
2.2. Detecting the received command
2.3. Detecting the received command with value analysis
3. Object description

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

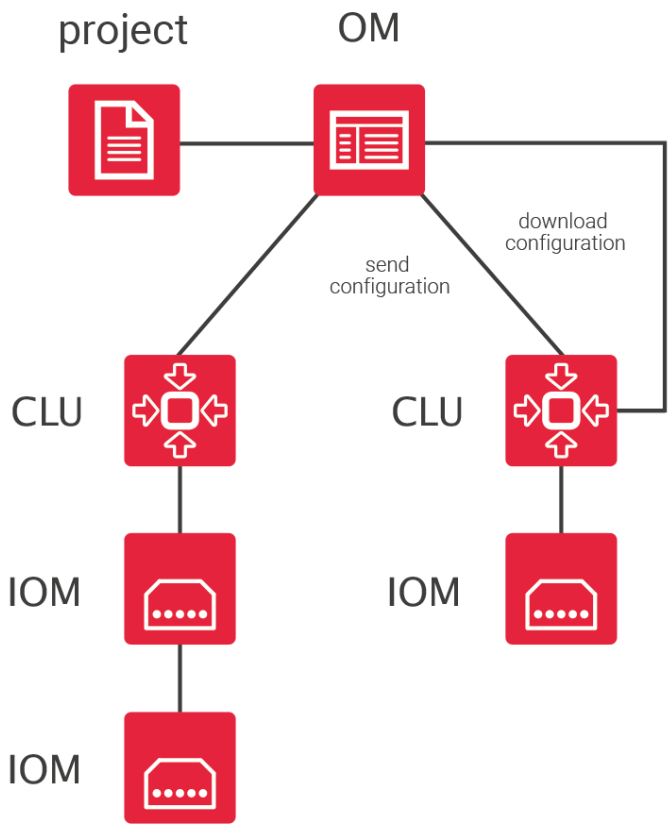
GRENTON Smart Building System was designed to operate small, medium, and large objects. System created on its basis can be easily modified, expanded, and integrated with other systems.

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).


project
OM
download
configuration
send
configuration
CLU
CLU
IOM
IOM
IOM

II. Foundation - Grenton Logical Interface

1. Introduction

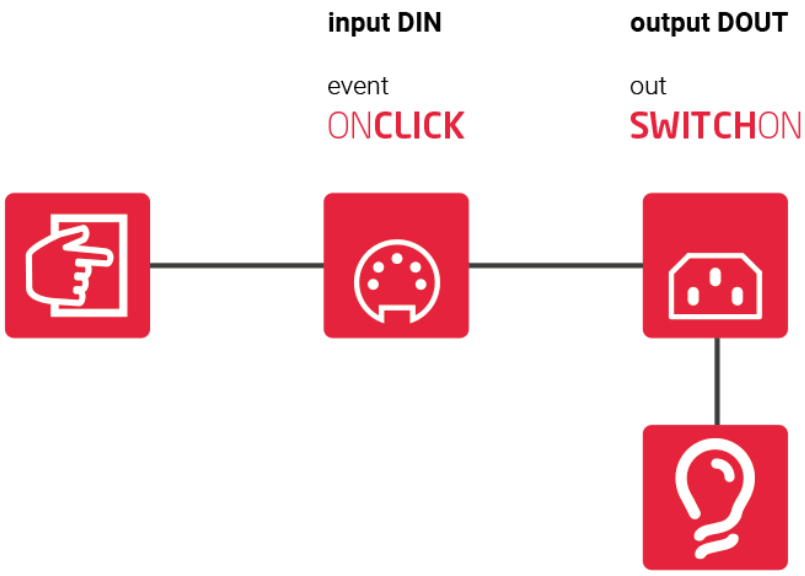
GRENTON system works on the basis of event driven model. Household members and their environment cause generation of events in the system, to which system reactions are connected, e. g. turning on the lamp in response to pressing a switch.

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

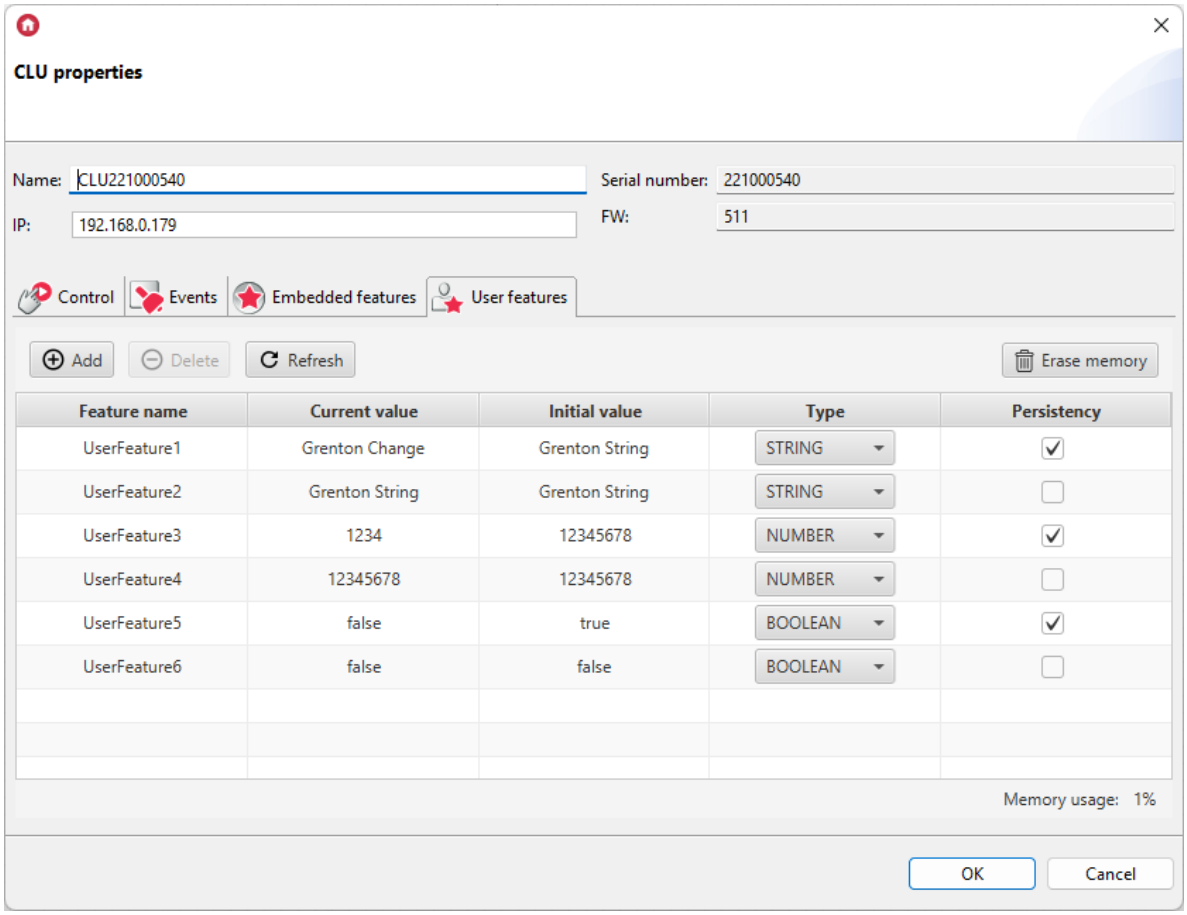
In CLU you can define features that will be used as variables in storing parameters during system operations, e.g. counters, markers (flags). User features can be used exactly as built-in features, except all user features can be saved and read.

The user feature name cannot contain spaces, special characters and cannot start with a number. The maximum name length is 100 characters.

Persistence

> **Note!**
> The described functionality is available for Object Manager version 1.8.0 or higher and CLU version 5.11.1 or higher.

Features for which persistence (restore) is checked save the value in memory after each change to them in the system. In case of a CLU restart, the features do not lose the value they had before the restart.



Registering a feature in the system as persistent is done by selecting the checkbox in the Persistence column and then sending the configuration to the CLU. In order to deregister a feature, you must deselect it and send the configuration to the CLU.

> **Note!**
> The allowed length of a STRING type user feature with selected persistence is 1000 characters.

In the lower right part of the user features window, the amount of used memory is shown in %. The amount of memory used depends on the type, number of characters stored in the feature name and the value itself.

After exceeding the memory limit, further added features are no longer persistent. The same applies to the situation when the memory overrun occurs for features that are already registered in the system as persistent, but their value (e.g. after a change) exceeds the limit.

When the memory is exceeded, a system log is generated in the console informing about exceeding the limit. To use logging, refer to chapter (VI.11.).

The Erase memory button is used to clear the memory of features that were not unregistered before deleting from the configuration. The memory is also cleared after the Hard Reset CLU procedure.

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.

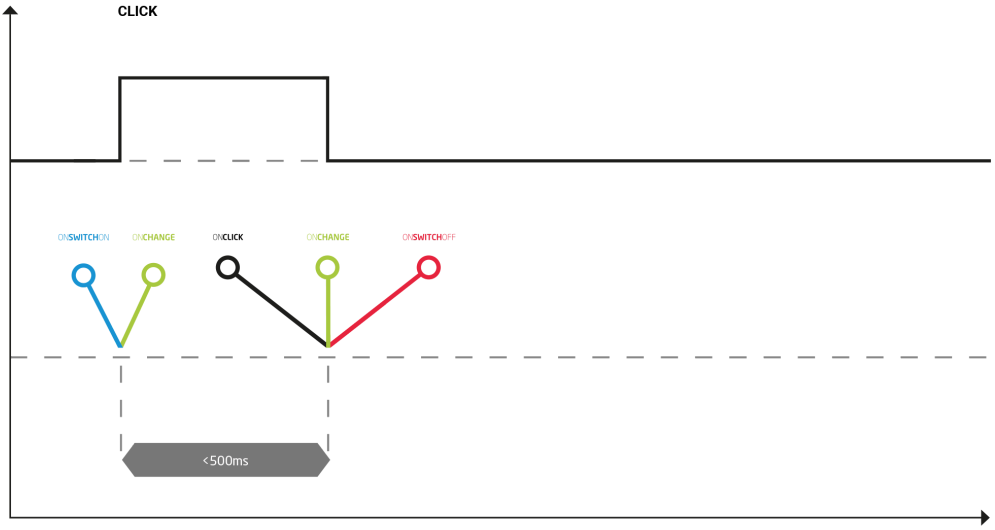
4. Events

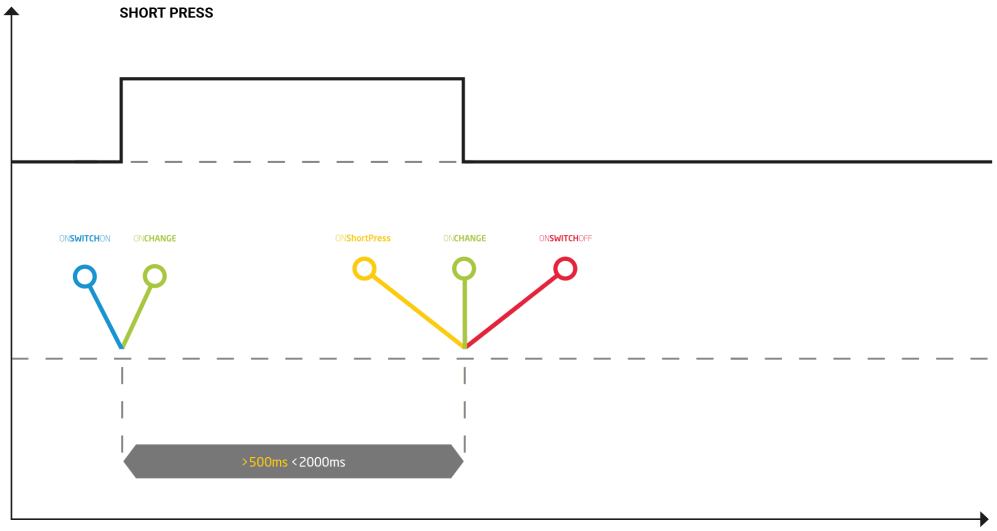
Events are elements of logical interface, invoked in reaction to the changes occurring in relation to an object (e.g. pressing a button, temperature change, etc.). We can connect one or more methods to each event, which will be invoked when an event occurs, e.g. when the button is pressed, the lights 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.

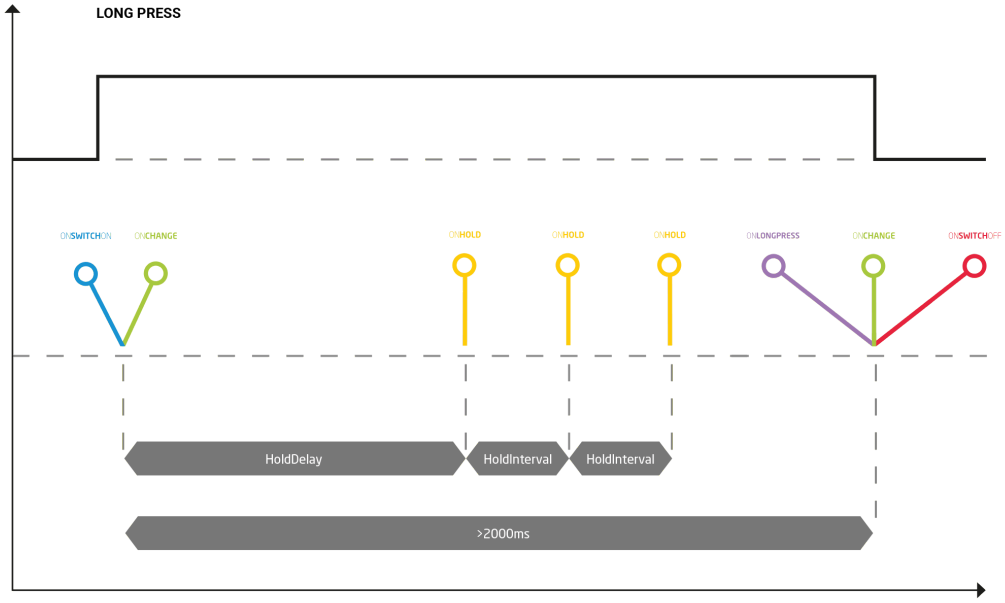
Each type of object (type of input/output) has its own list of events, which are invoked in a precisely specified way, depending on the actions performed by the user. For instance, binary input has the following list of events:

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

which are invoked according to the following scheme:


CLICK
ONSWITCHON
ONCHANGE
ONCLICK
ONCHANGE
ONSWITCHOFF
<500ms


SHORT PRESS
ONSWITCHON
ONCHANGE
ONShortPress
ONCHANGE
ONSWITCHOFF
>500ms <2000ms



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^2 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.

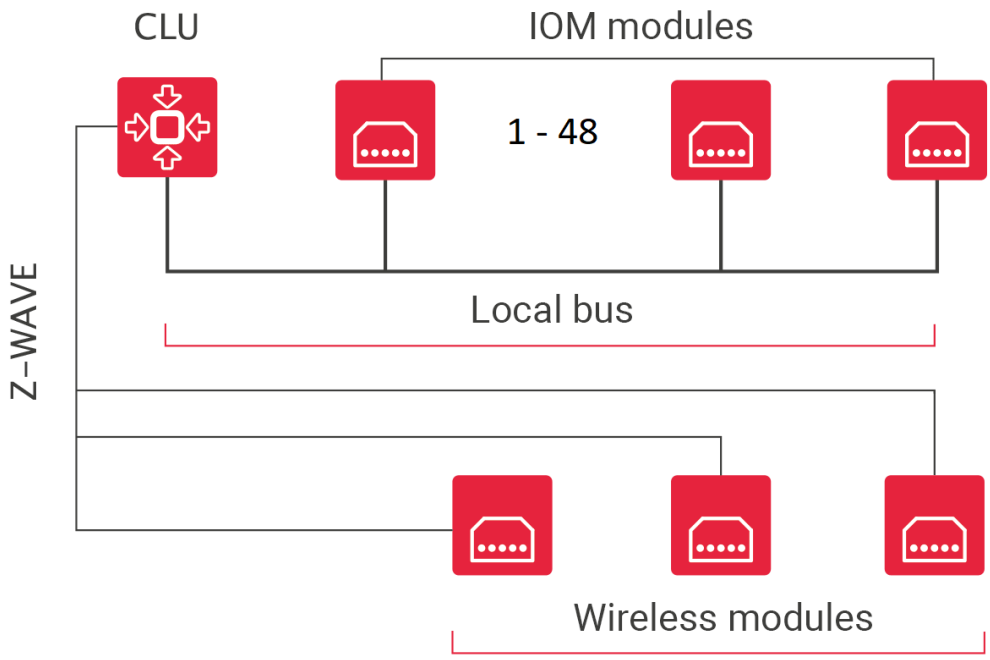
- If you don't know yet, which devices will be controlled by the system, make sure the wiring reaches all possible places.
- For light switches, any thin wire may be used, e.g. YTDY – it will allow saving on wire.
- Remember to prepare the system for temperature sensors and weather station.
- Place power 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:

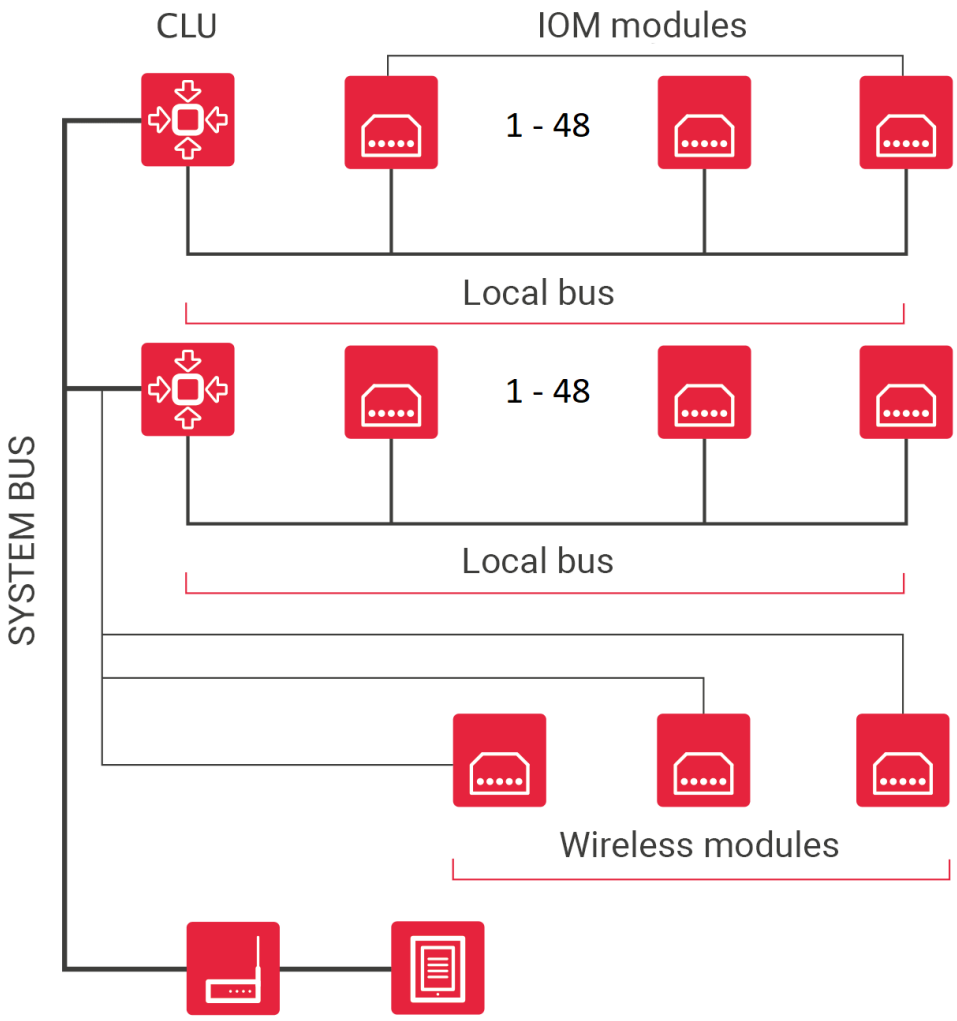
A. Basic configuration - centralized system with one CLU

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.



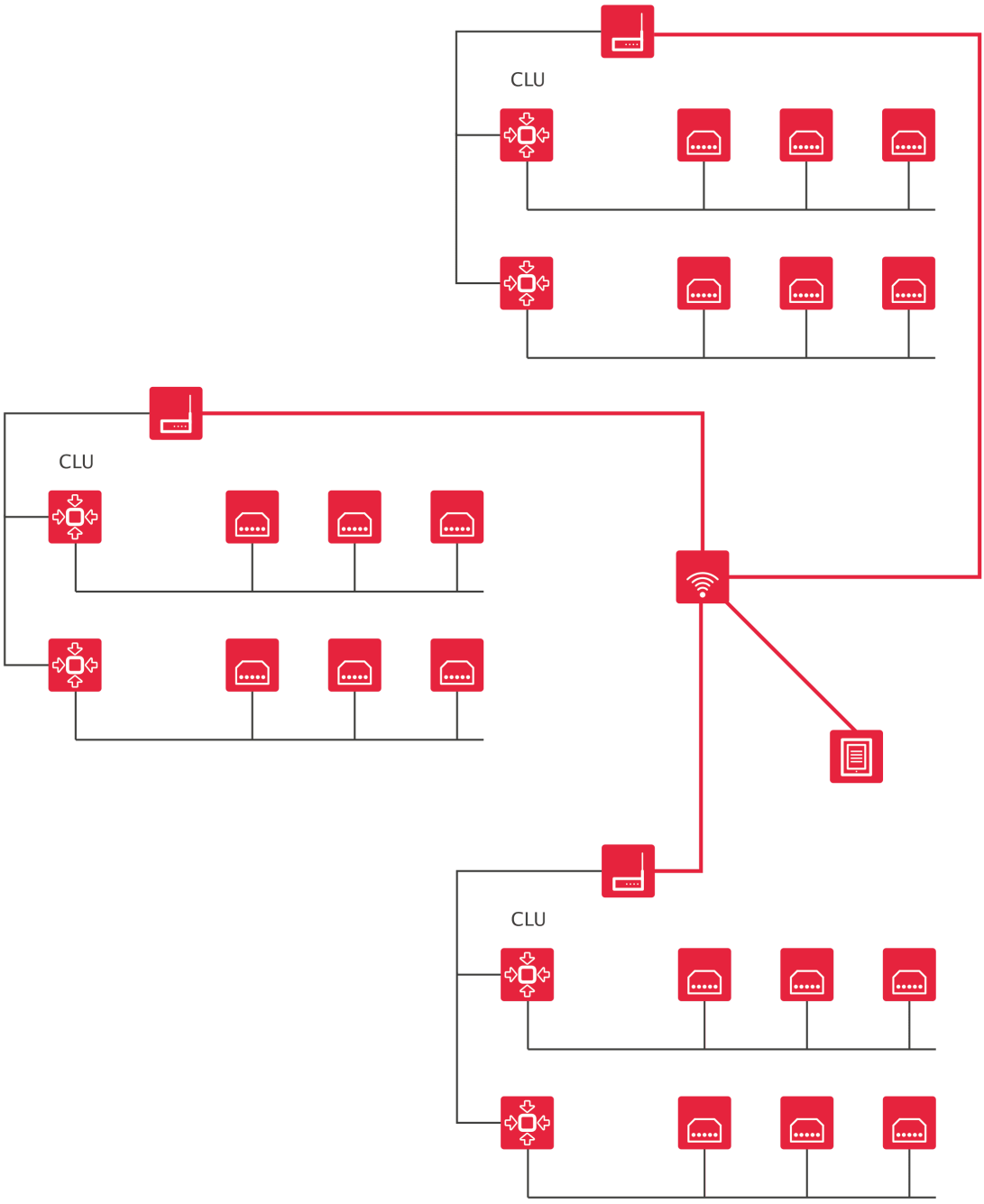
B. Advanced configuration - tablet-controlled distributed system with many CLU

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

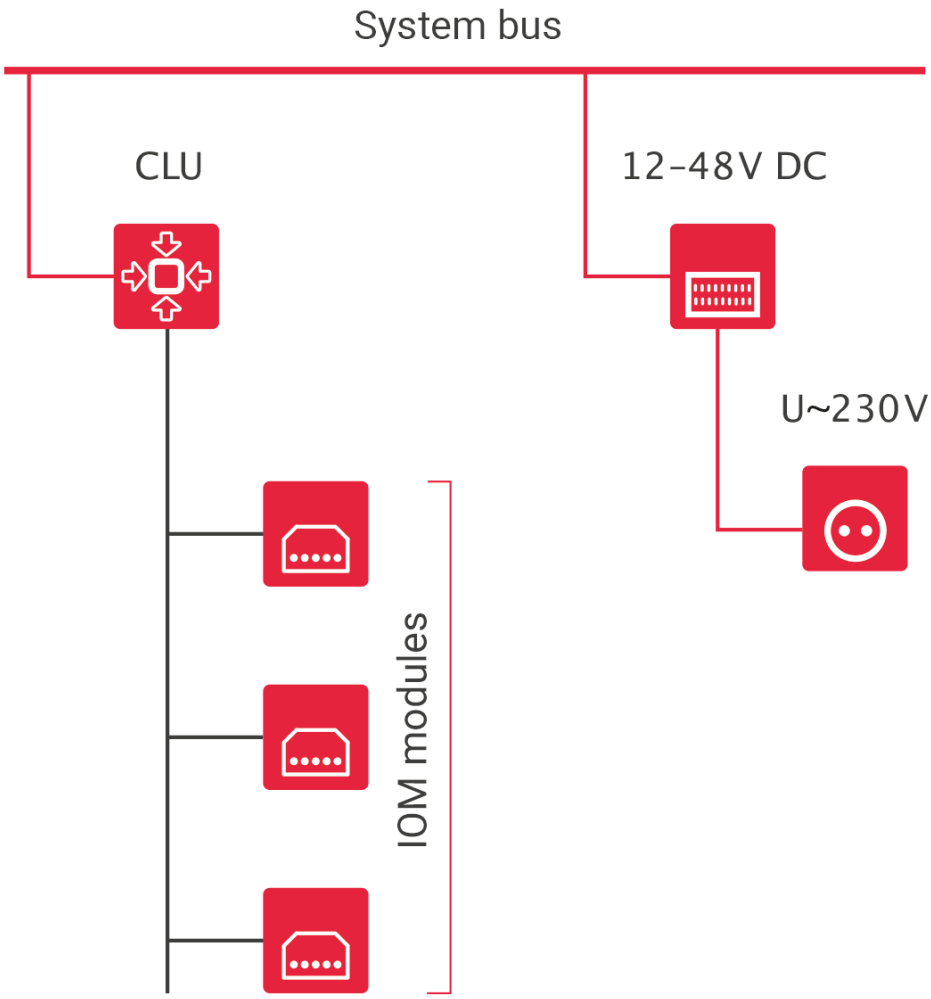
System expansion is practically unlimited. Several objects can be connected to one system. Thanks to that, you can have central control using only one system.



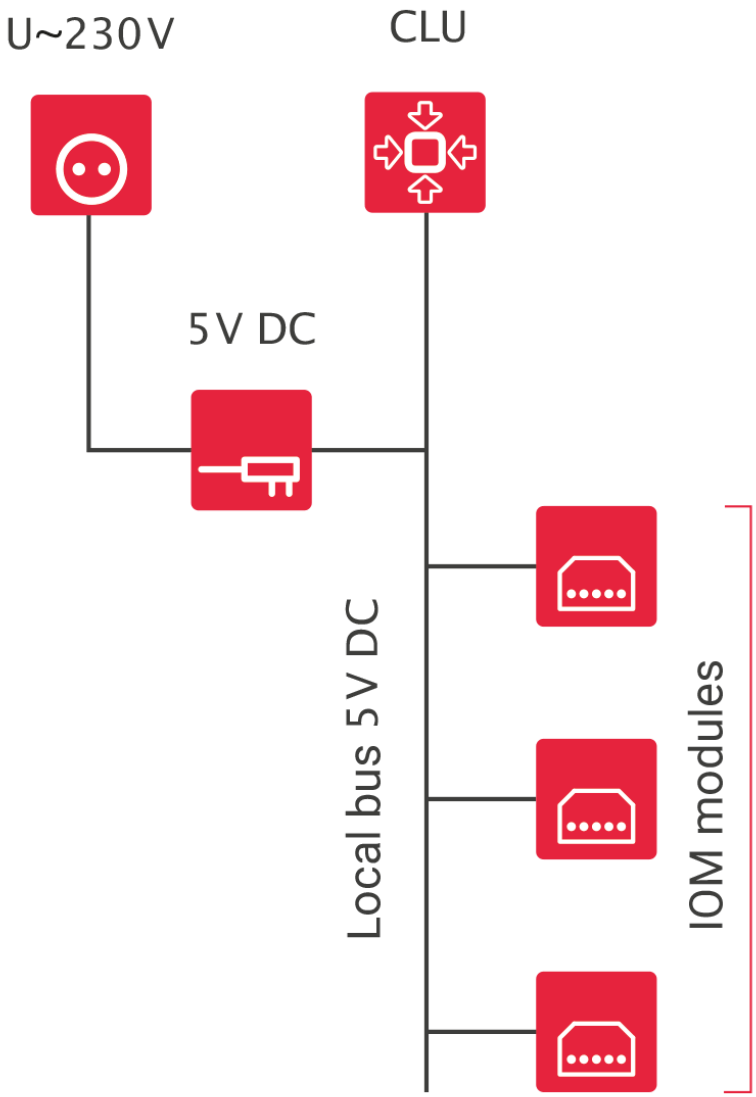
3. Modules power supply

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

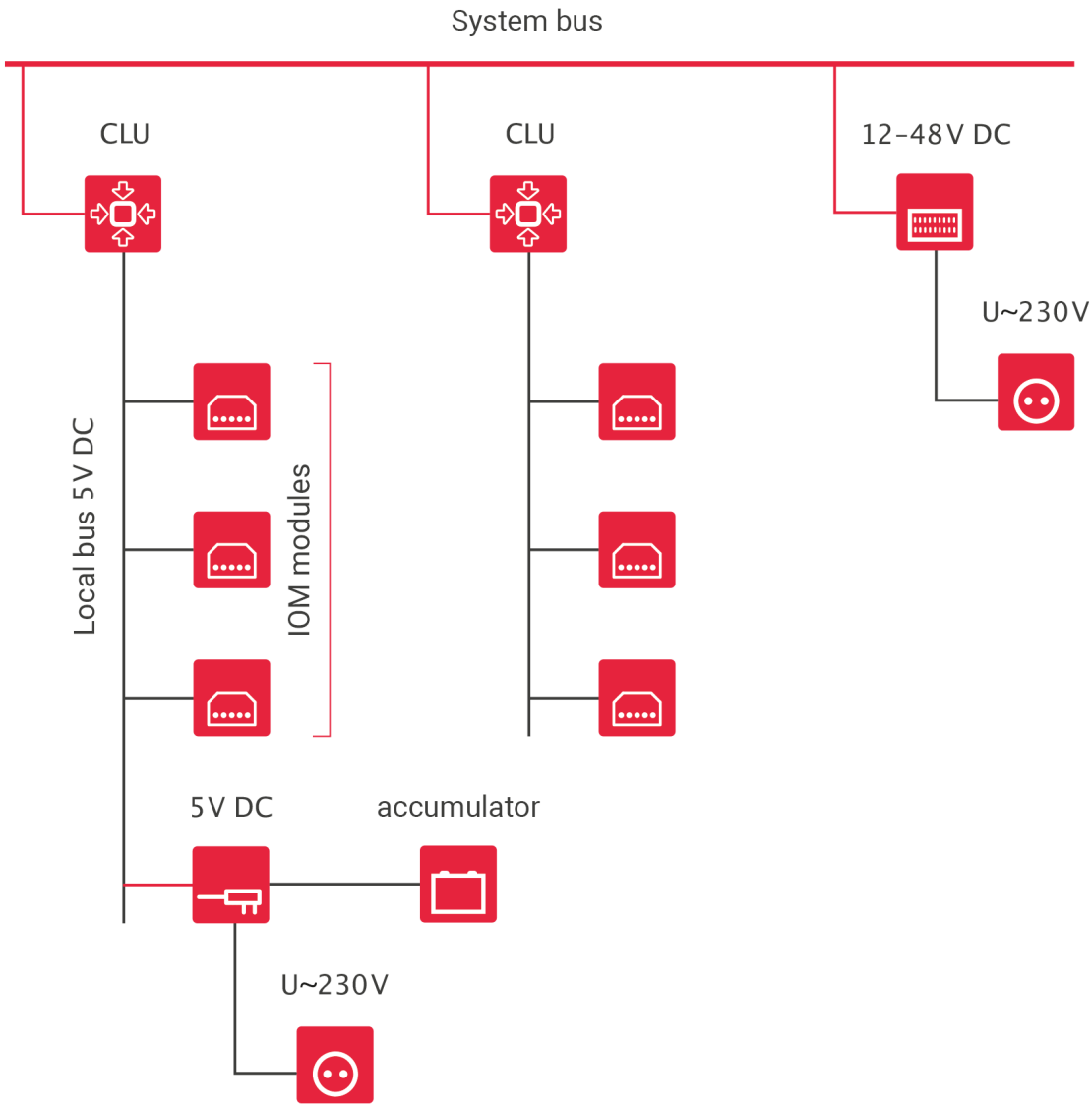
1. By connecting the power supply to the 24V DC system bus - in this case the CLU module will power the IOM modules connected to it via the local bus. A maximum current of 1000 mA can flow through the local bus (TFbus).



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.

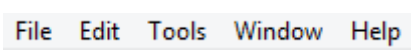
C. Linux

- 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**

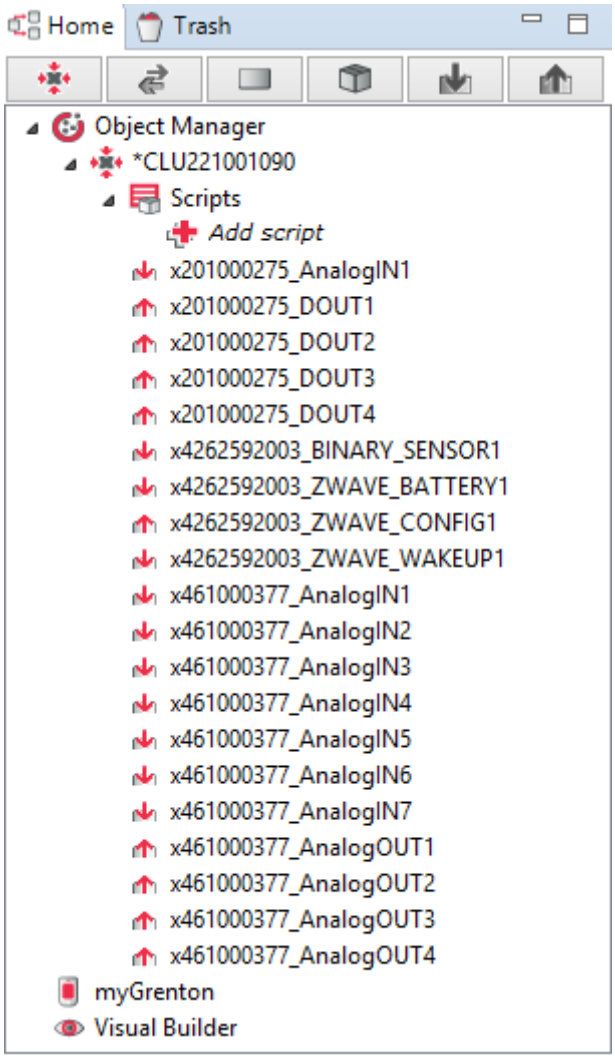


Contains basic commands for handling the project.

- **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**



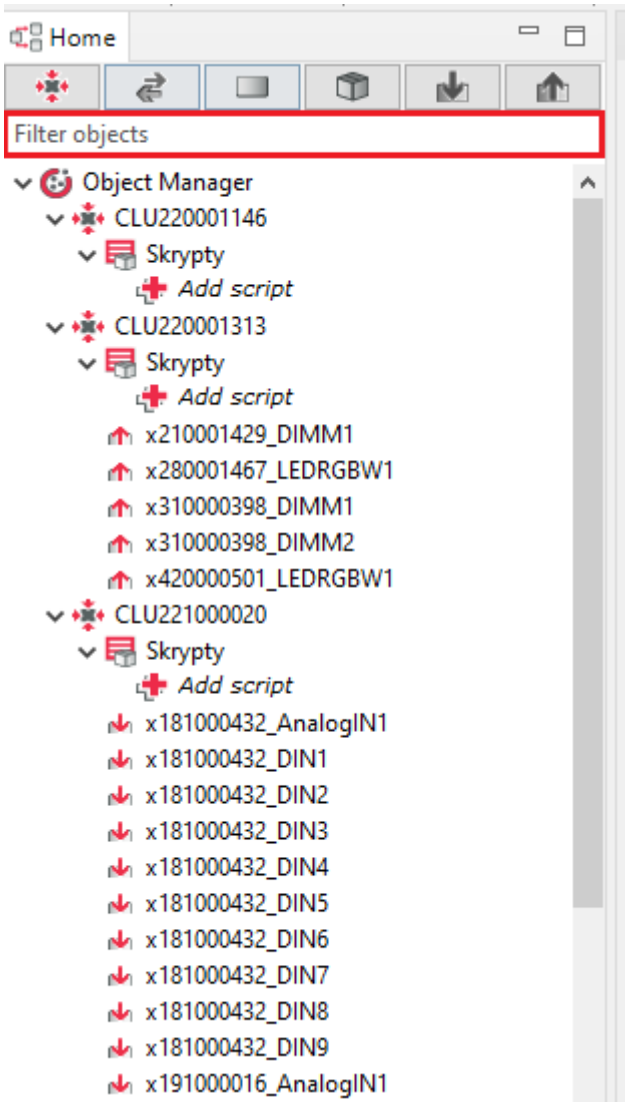
Consists of three parts: Object (CLU, inputs, outputs) list, the myGrenton tab (creating the myGrenton application interface) and Visual Builder (creating the Home Manager application interface).

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.

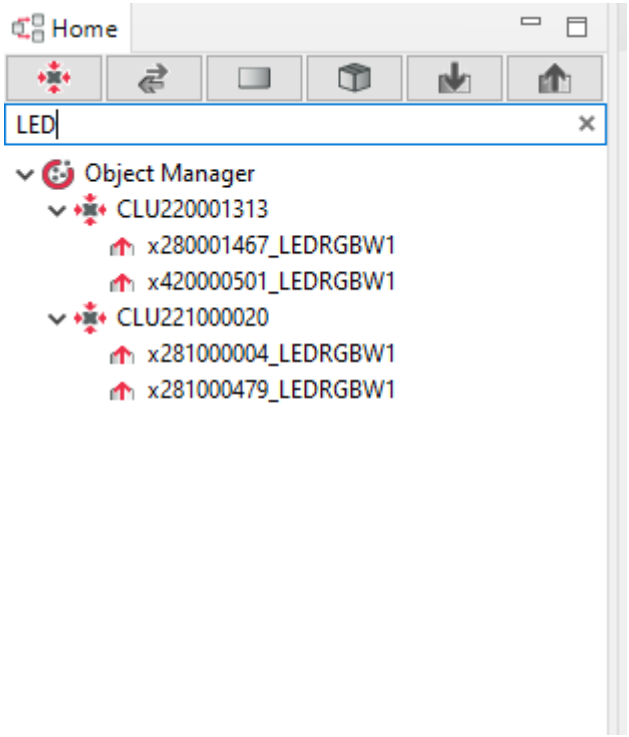
2.1. Object filtering

> **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.



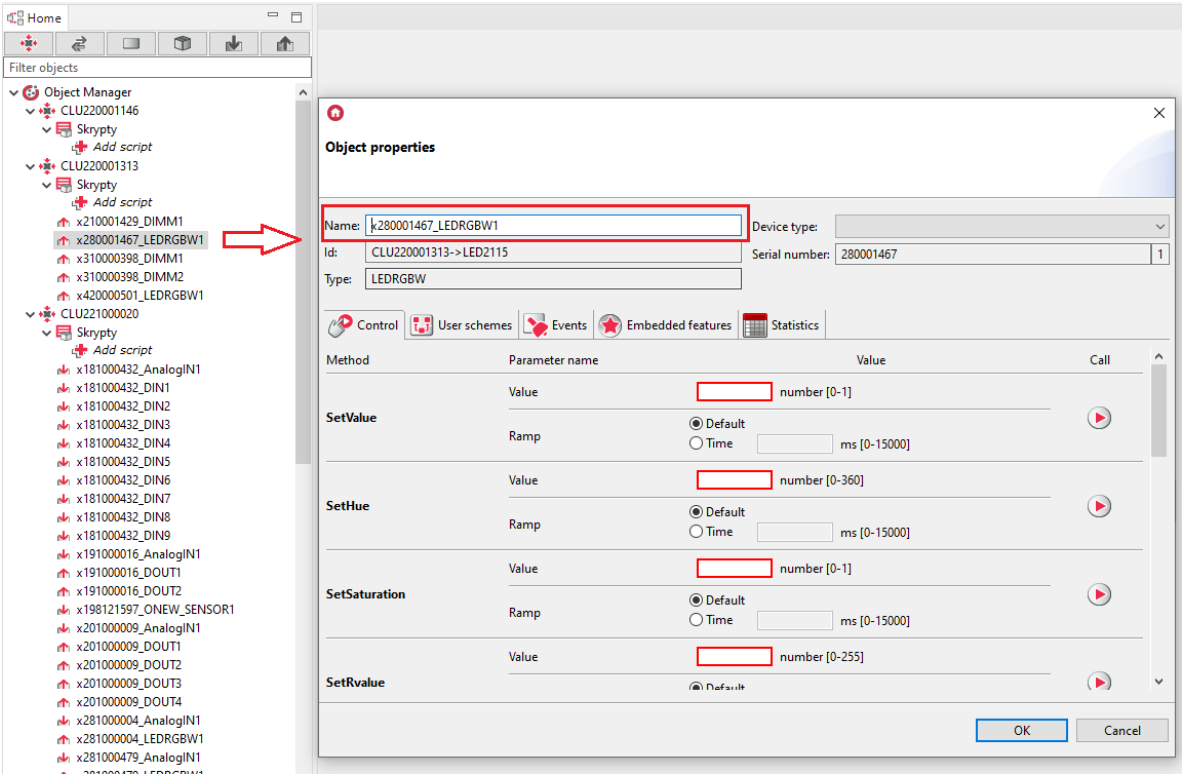
The given phrase is included in the filtering after switching to another group view (tab).

2.2. Renaming an object

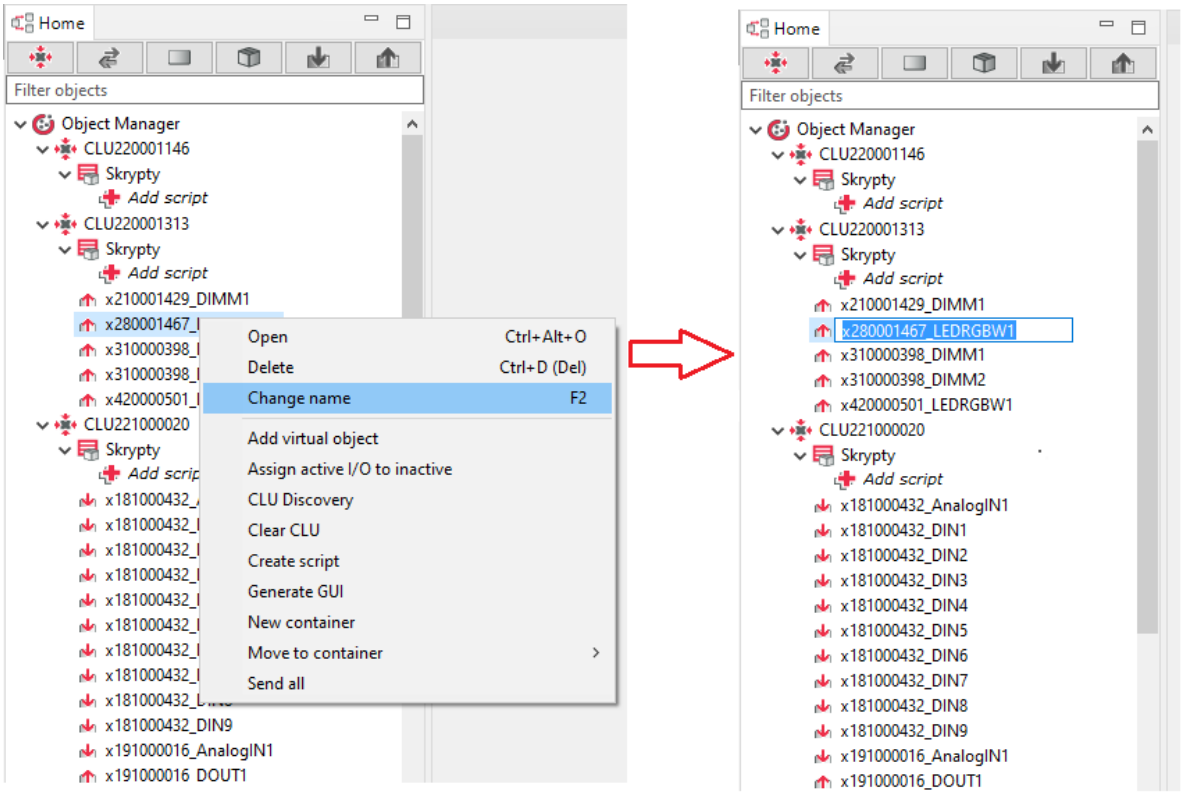
Note!
The possibility of renaming an object using the context menu / `F2` is available for Object Manager version 1.5.0 or higher.

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`

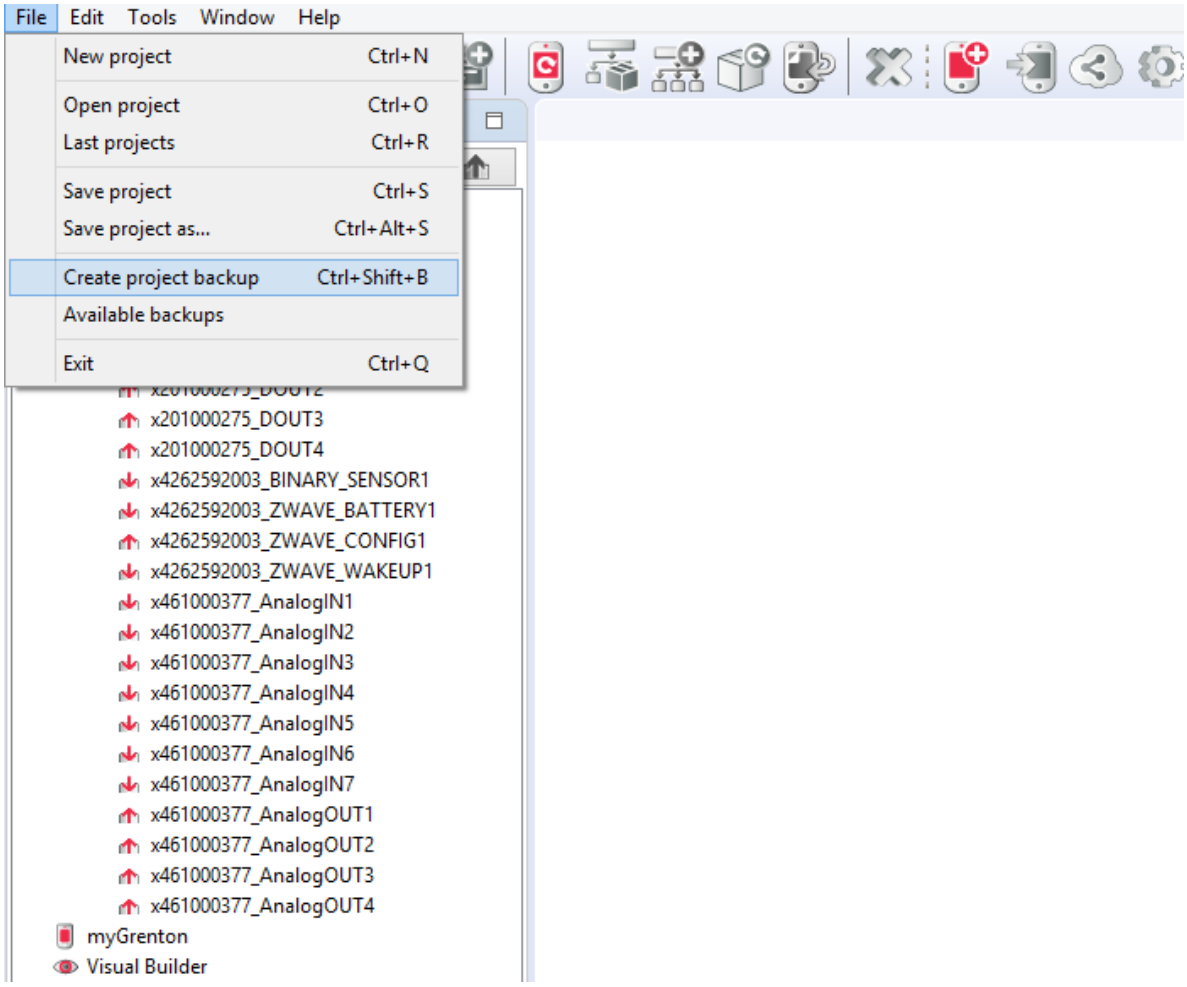
All files of created and saved projects are saved in the catalogue with `*.omp` extension.
(e.g. `projekt.omp`).

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`).



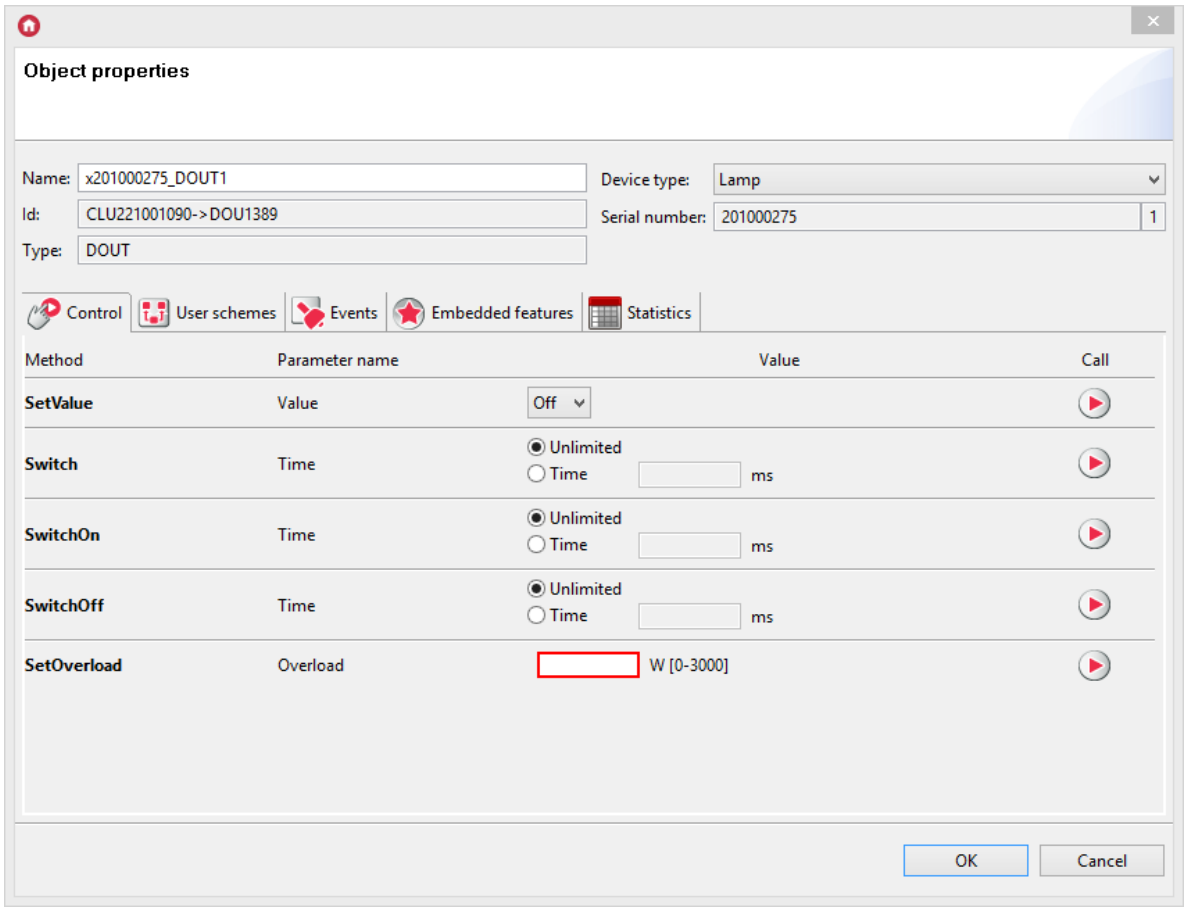
Saved backups are available in the list opened by clicking `Available backups`, or in the project opening window in a tab `Backups`.

> **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

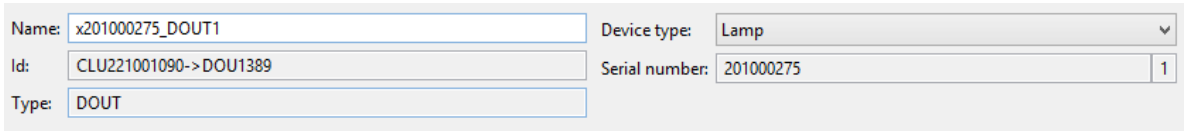
4.1. Objects configurator

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.



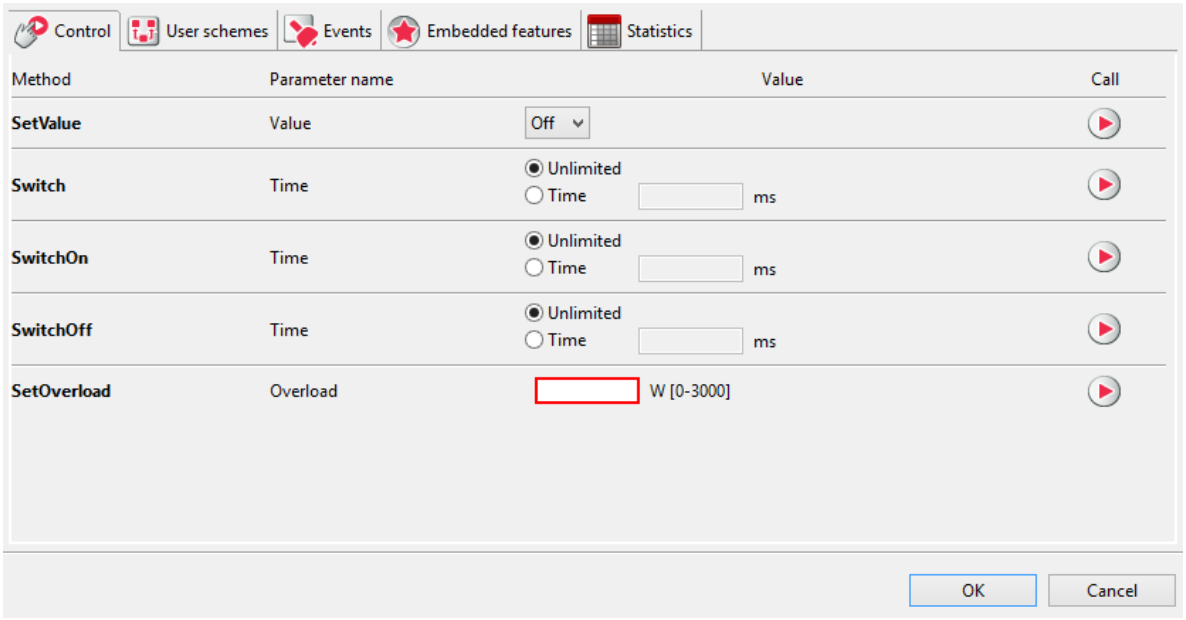
The form above consists of the following sections:

1. Basic information



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.

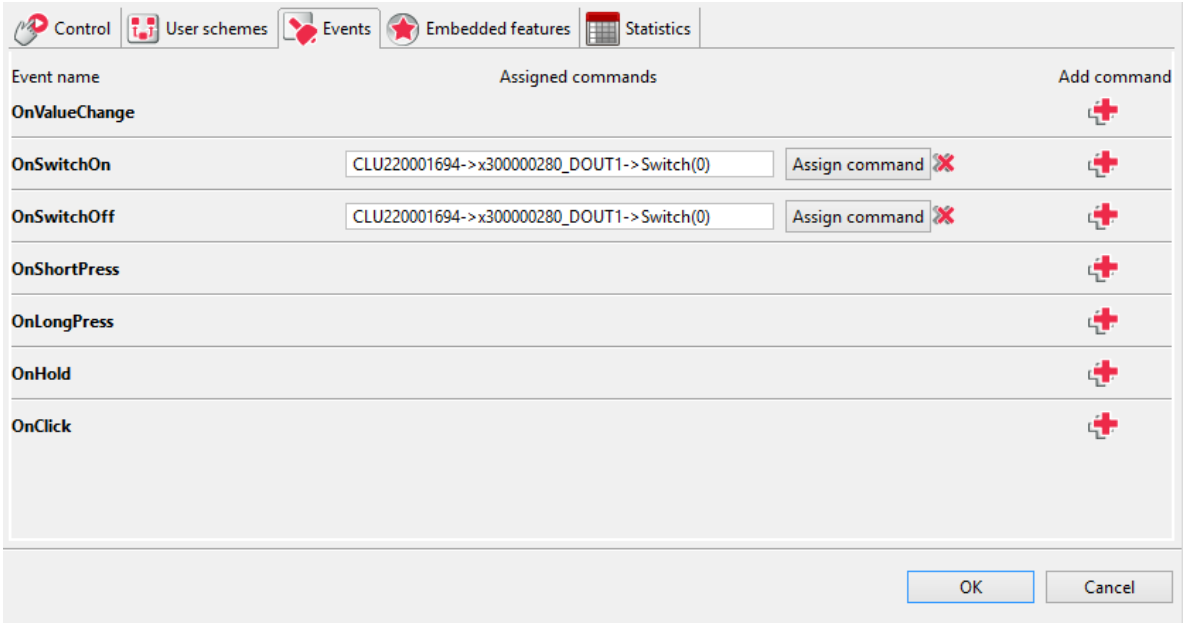
2. Control Tab



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. Events - tab description

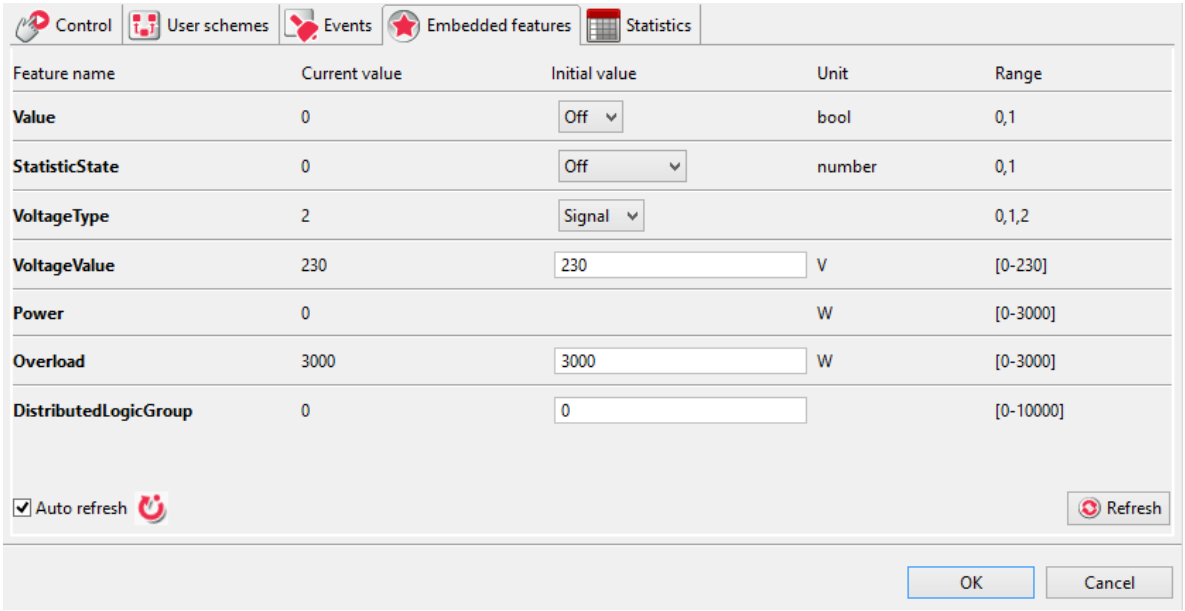
The tab contains list of events applicable for the specific object type and methods connected to them, which are invoked after event occurrence (if the user defined them).



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

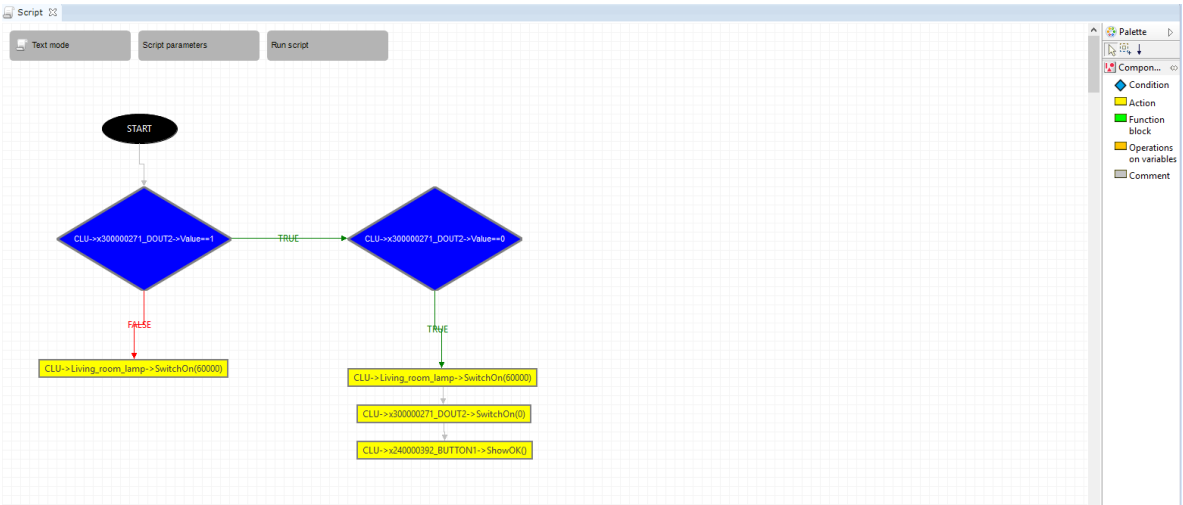
This part presents values which the selected objects currently possesses, and initial values which was saved in it (initial vaues set in the case of system restart, e.g. after power supply break). Entering value in the "Initial values" field will result in setting it during CLU start.



4.2. Script builder

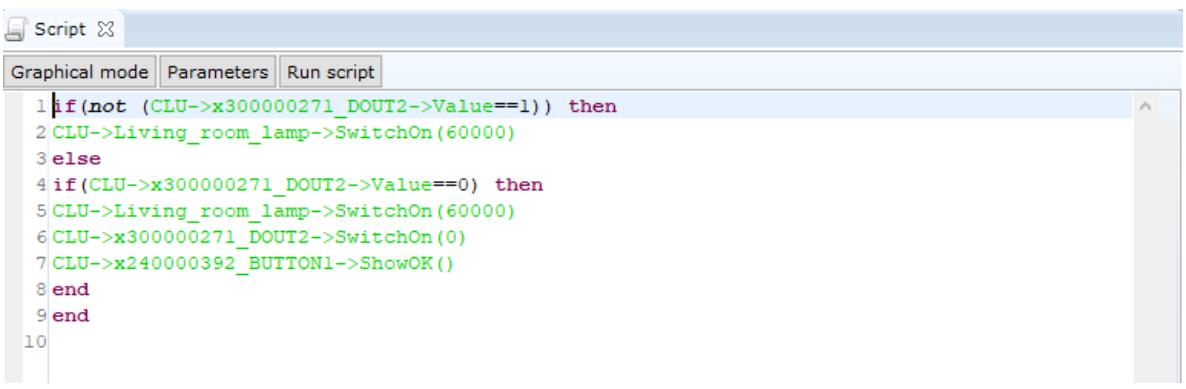
It's a tool for script creation, which can work in two modes:

1. **Graphic (simplified) mode**, in which the chart can be created in an easy wat 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



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

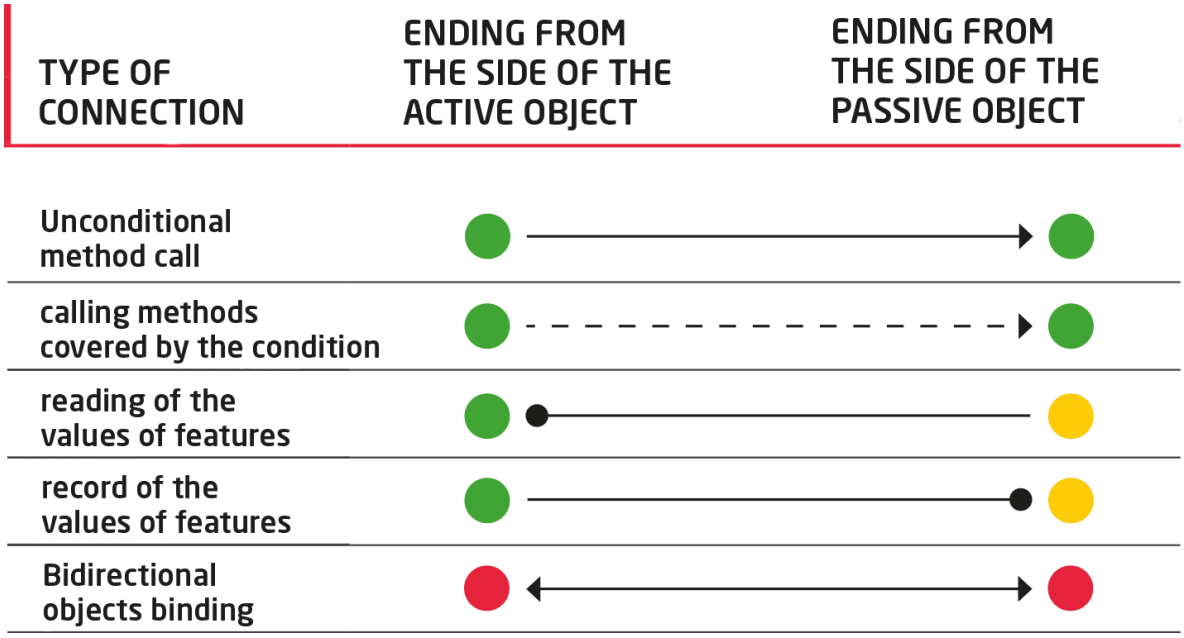
A tool showing dependencies and connections between all objects in the system. Thanks to that tool, you can easily and quickly find a dependency that interests you, or check dependencies of a specific module without going through configurations.

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.



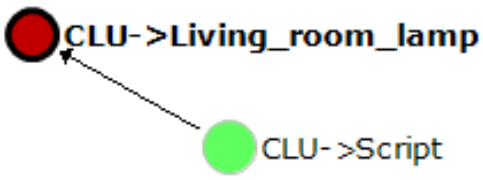
Connections are displayed on three different levels:

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.

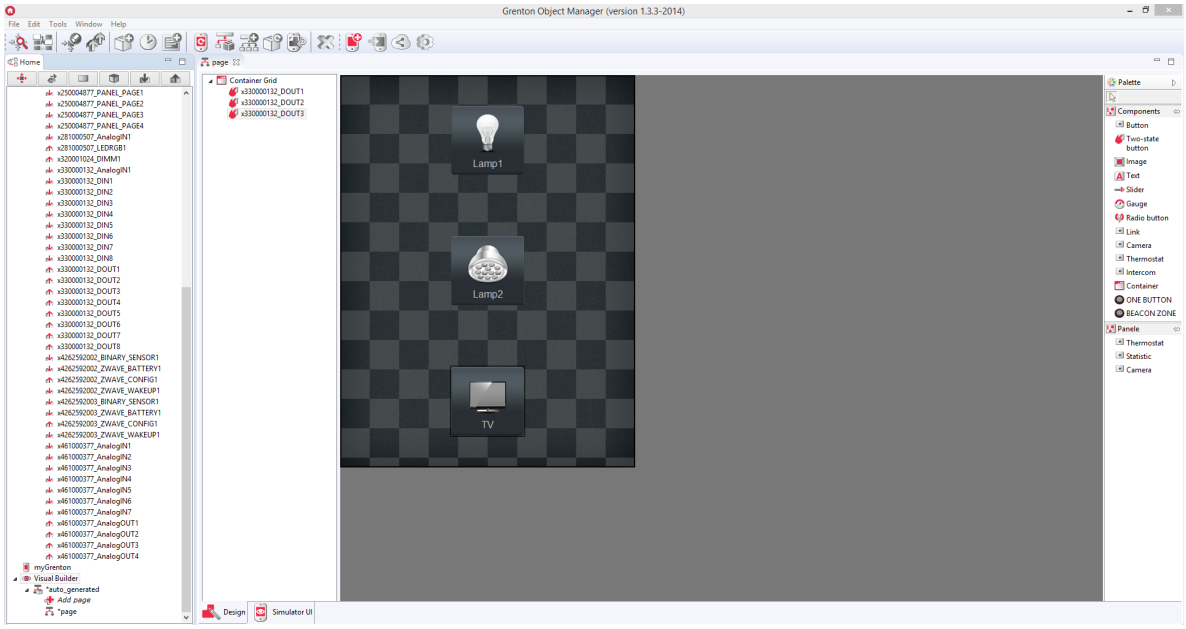
2. In the diagonal plane - allows going up and down between levels by clicking on the central object and selecting an object from the appearing list (for going down) or by pressing "up" button in the upper part of the chart (for going up).



4.4. Visual Builder

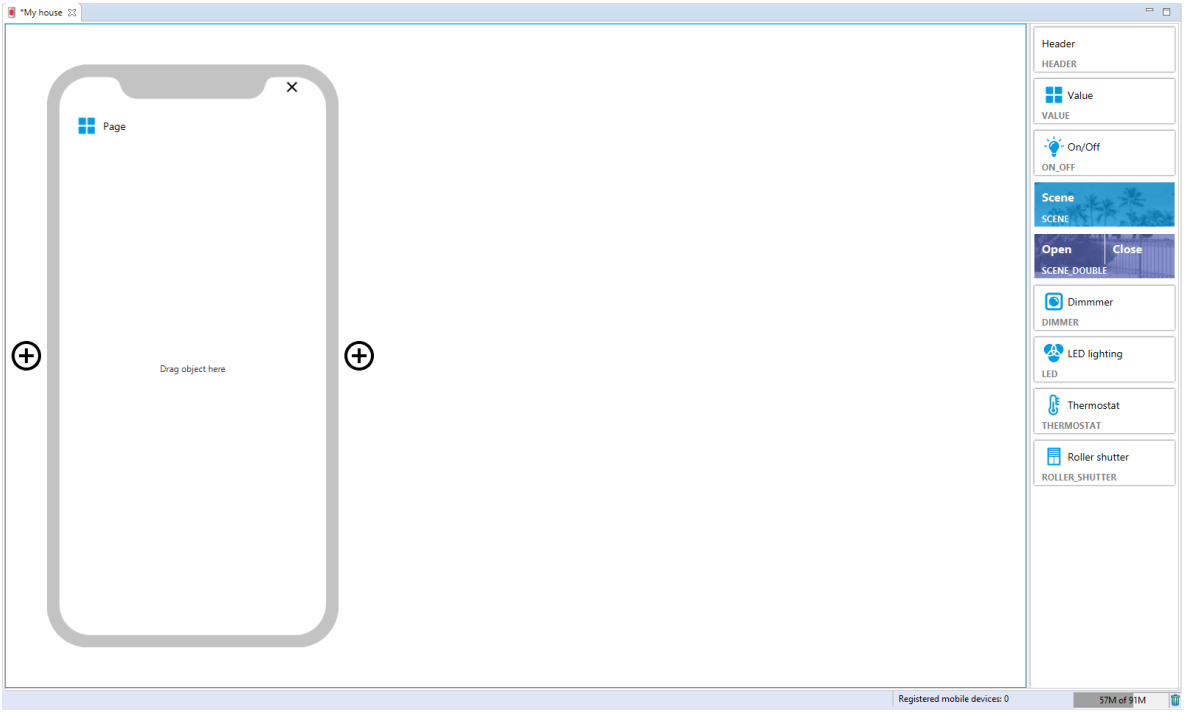
> **Note!**
> **Support for the virtual media measurement functionality has been ended in Object Manager version 1.9.0 and higher. Assumes Statistics (in the features of built-in objects) and options related to virtual measurement have been removed.**

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.

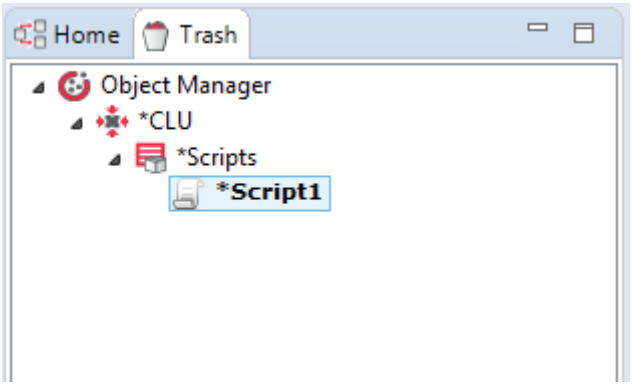


4.6. Bin

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.

An object can be permanently removed from the recycle bin by selecting *Delete* from the context menu. Restoring a removed module to the project (removing a module means removing all objects of the removed module) is only possible by performing CLU Discovery. Restoring a single module object is only possible by clearing the configuration on the CLU.



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.

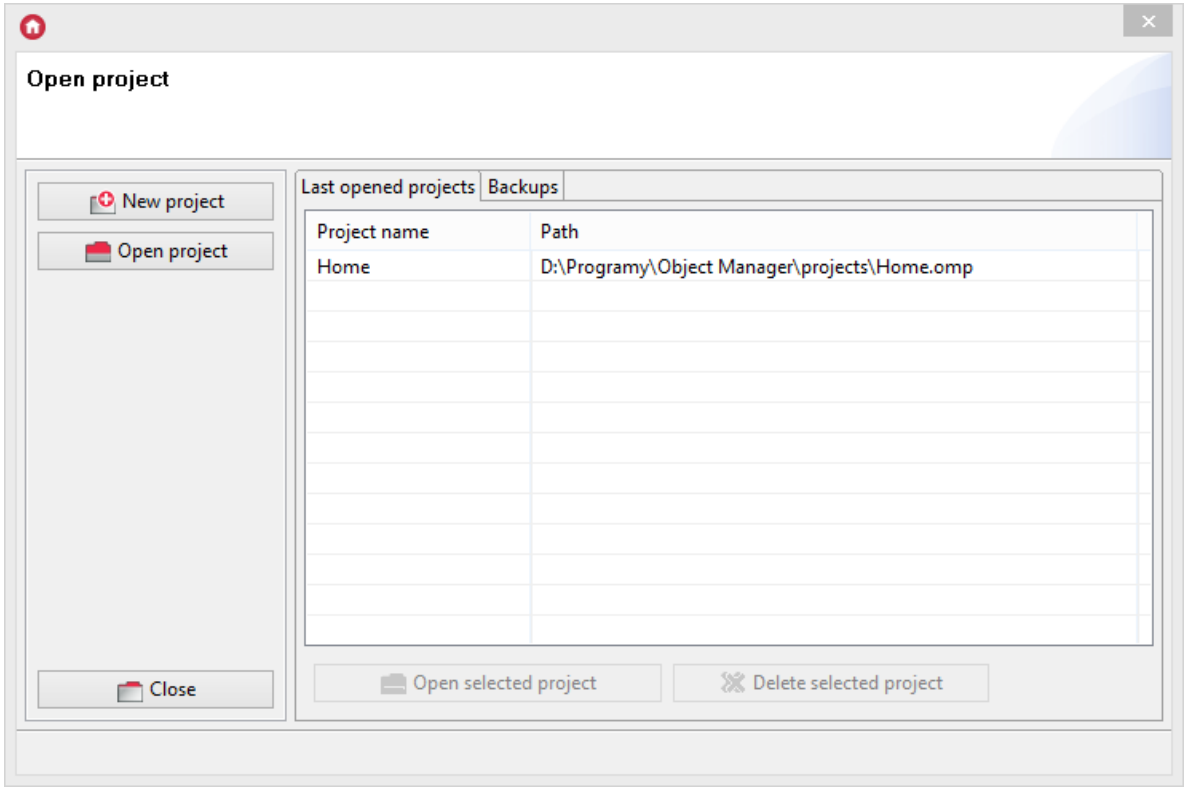
2. IP addresses

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.

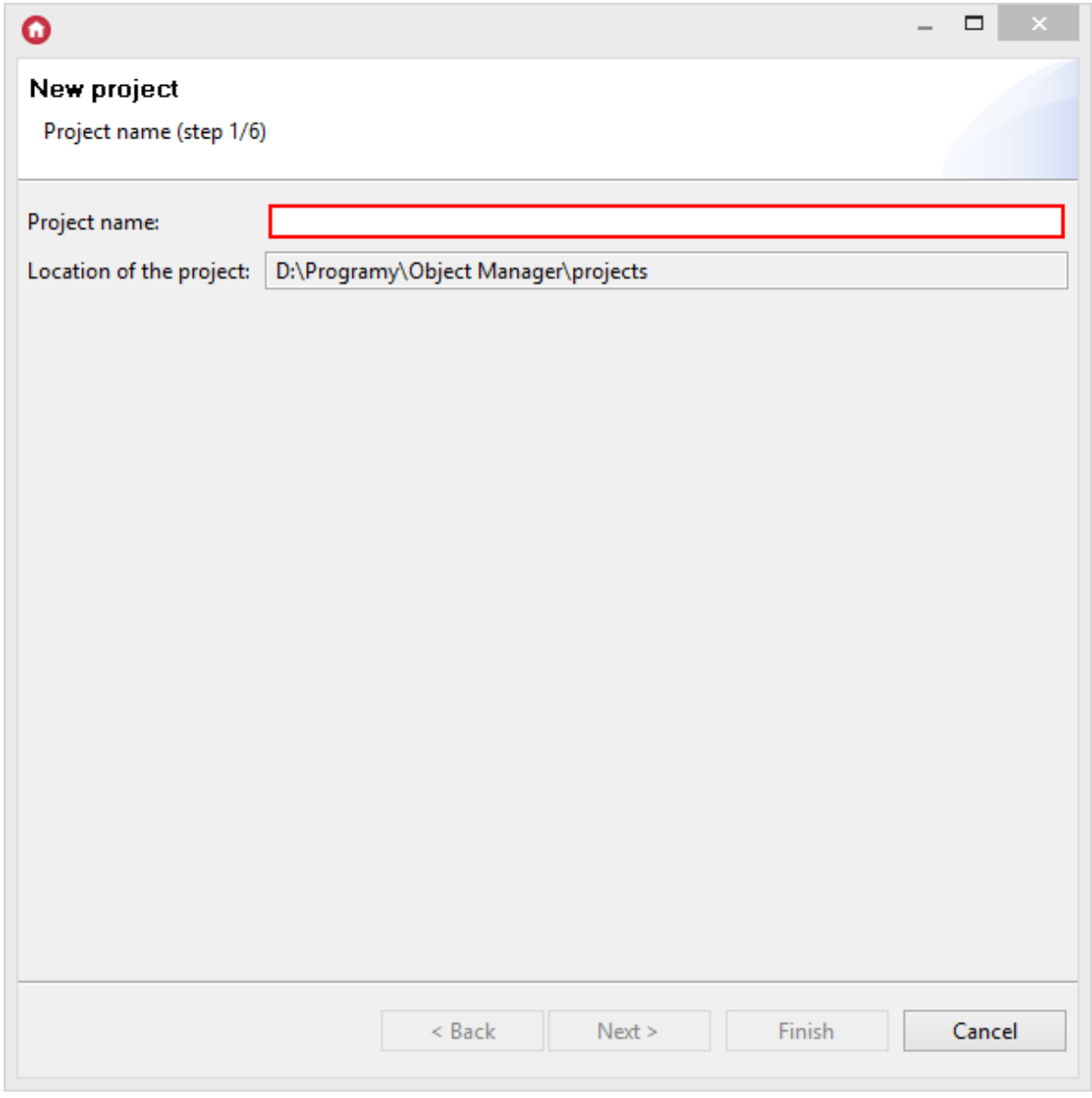
> **Note!**
> After connecting CLU (or several CLU) to computer's network card, it will receive a new IP address consistent with pool of addresses in which computer's network card is.

3. Creating new project

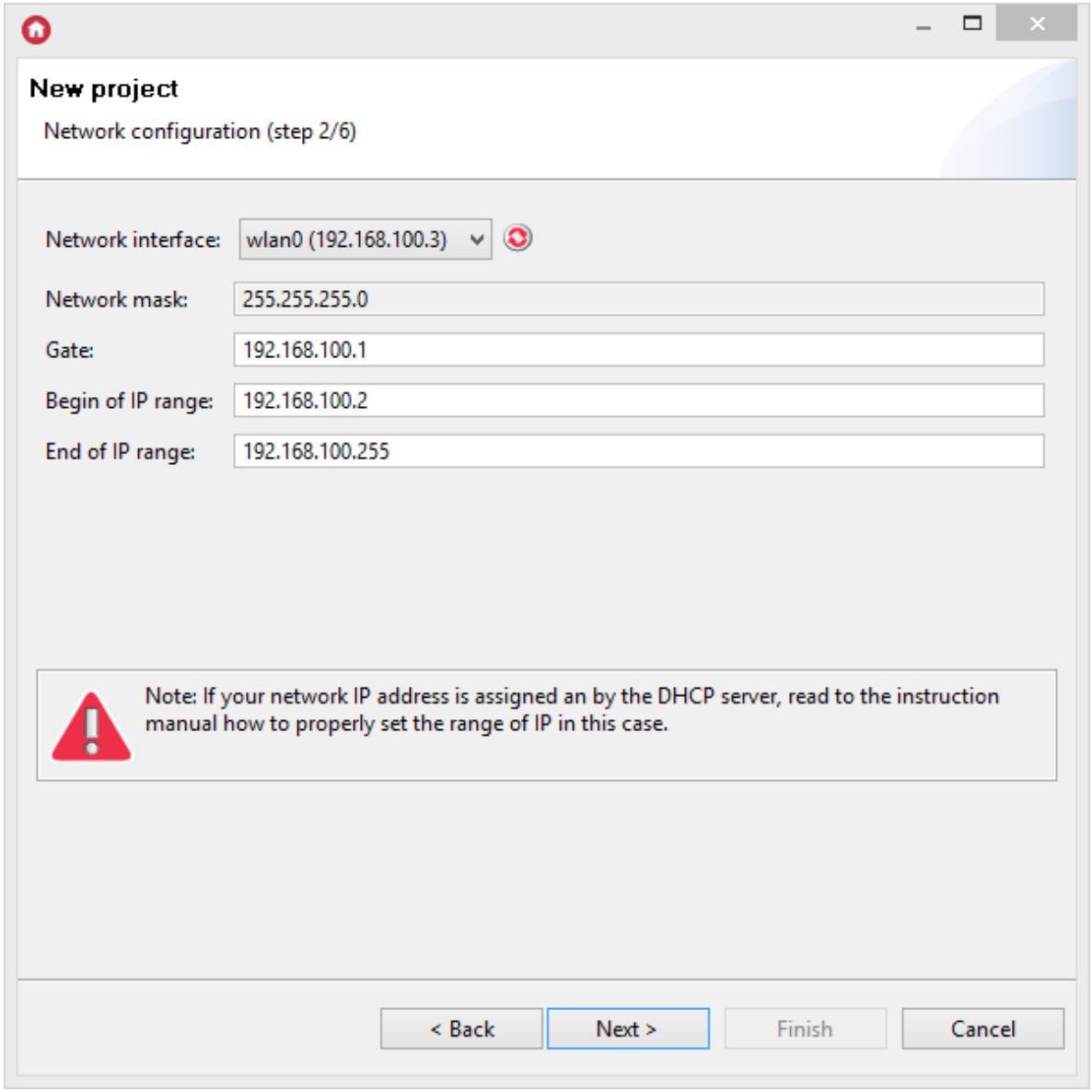
After opening Object Manager, a new window with two options appears: opening a saved project and creating a new project.



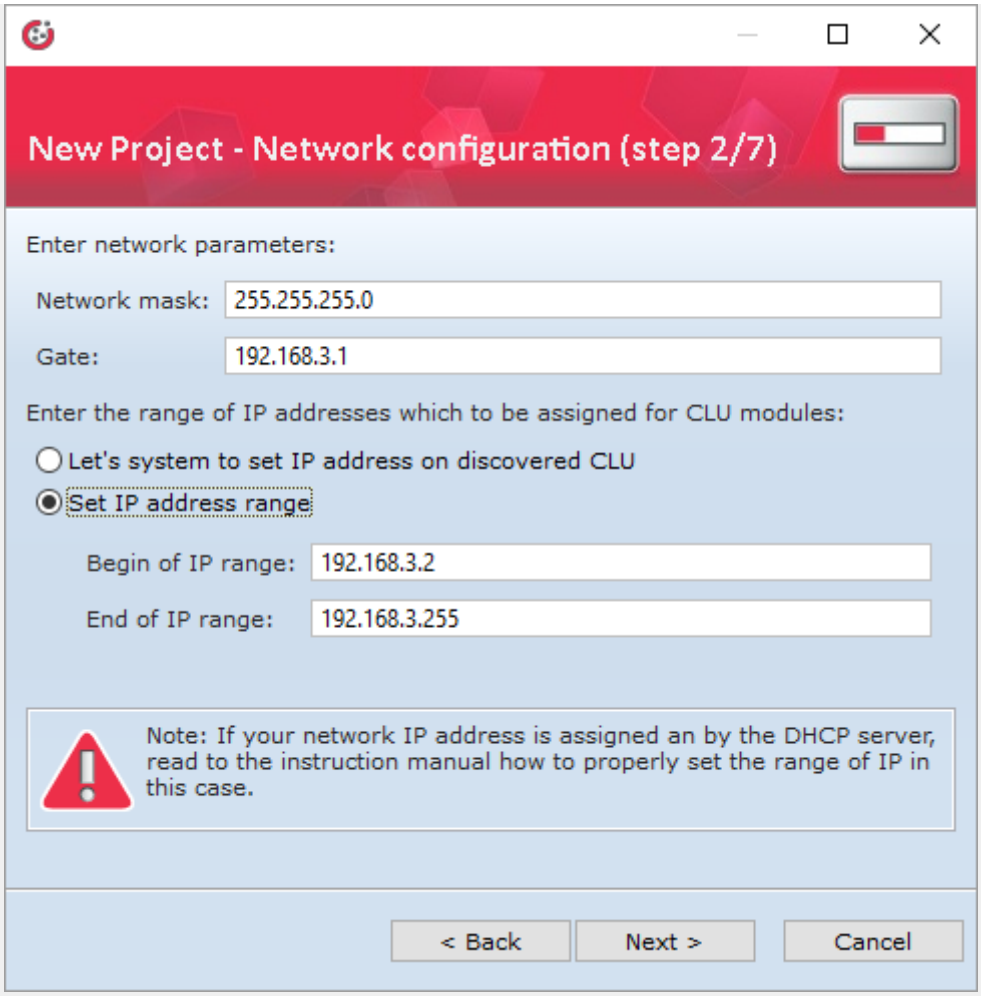
1. Select new project creation, then name the created project.



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.

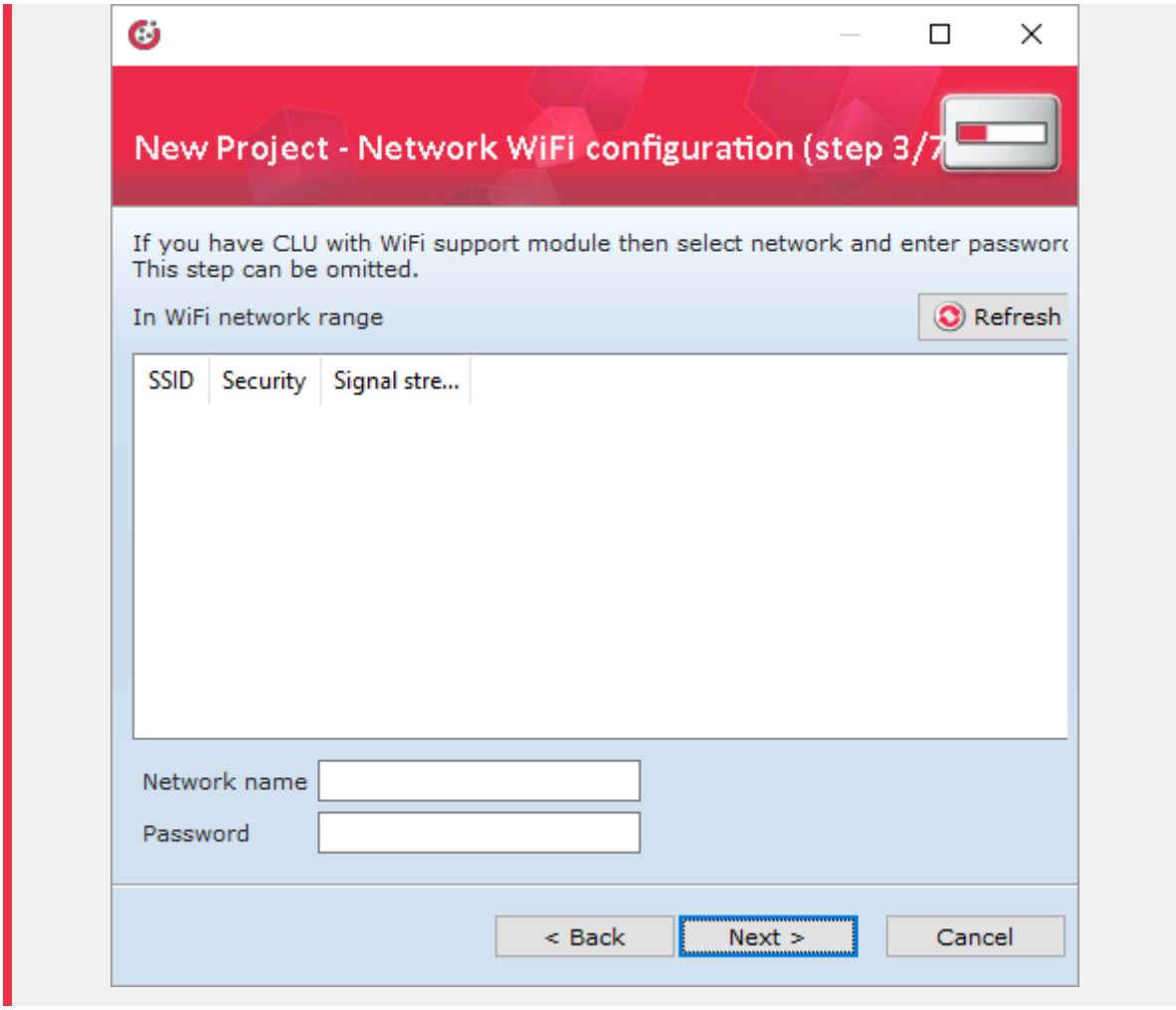


Note!
For Object Manager version below 1.2.1, the settings for the available network interface are not loaded automatically.

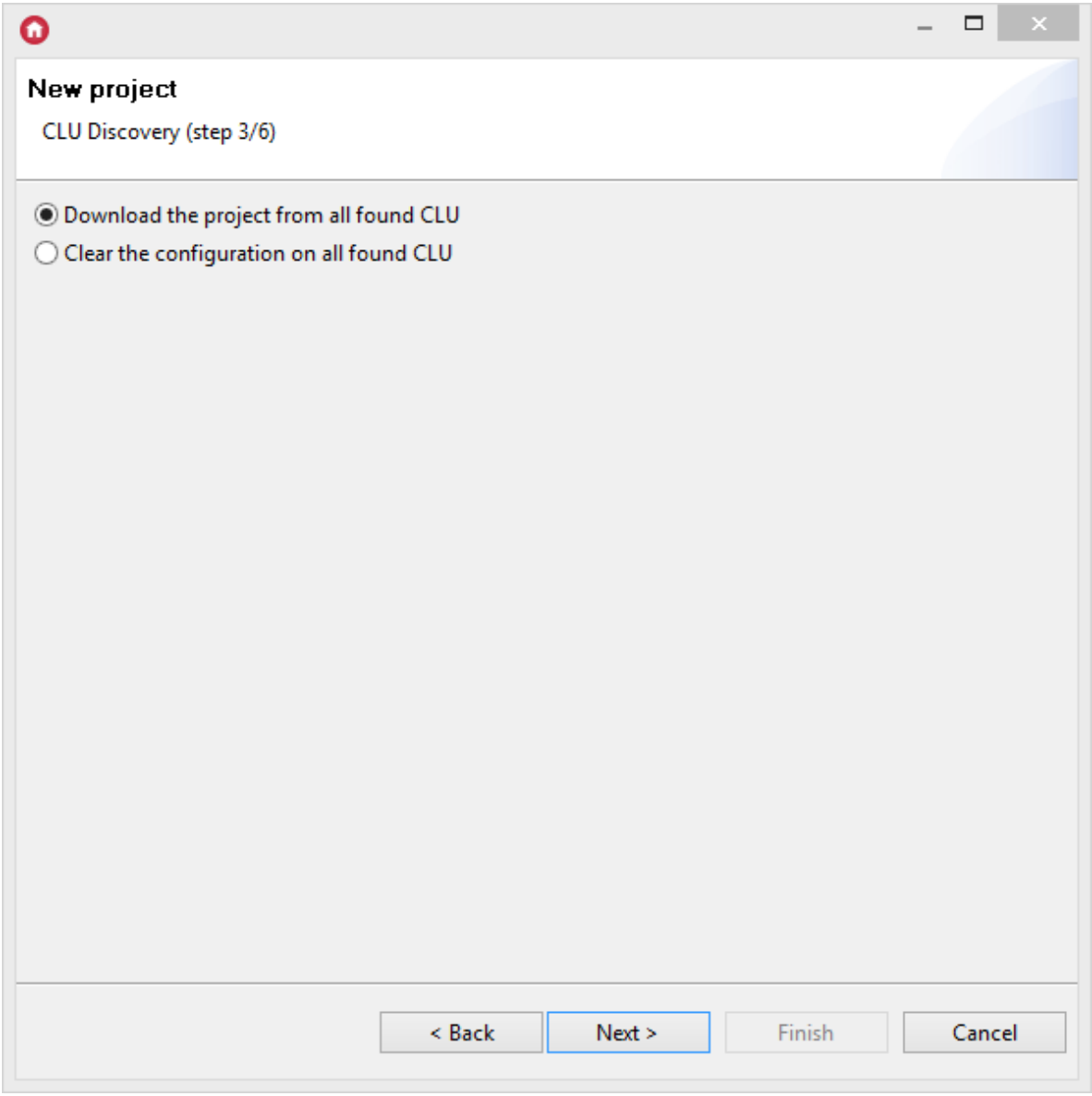


Note!

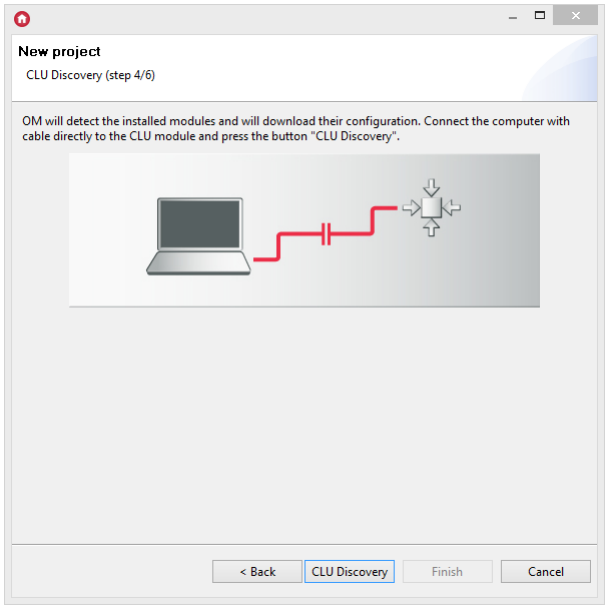
In the older version of Object Manager, there is the "Network WiFi configuration" step that should be skipped.



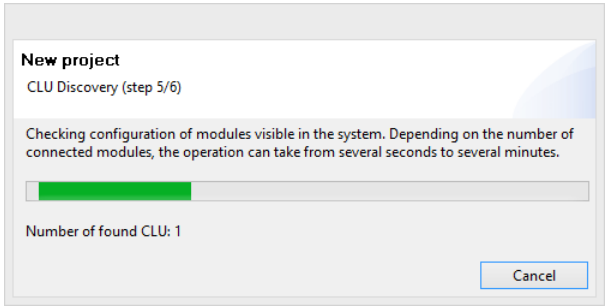
3. In the next step, you may choose between downloading existing system configuration to the newly created project, and complete configuration reset and starting a project from scratch. The first option is useful when necessity of recreating configuration after loss of project file occurs.



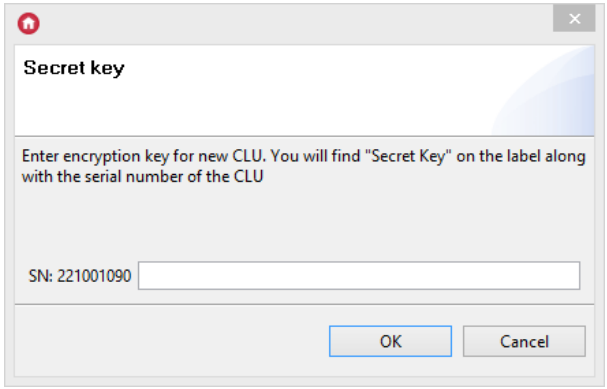
4. In the fourth step, available modules search procedure named CLU Discovery should be launched.



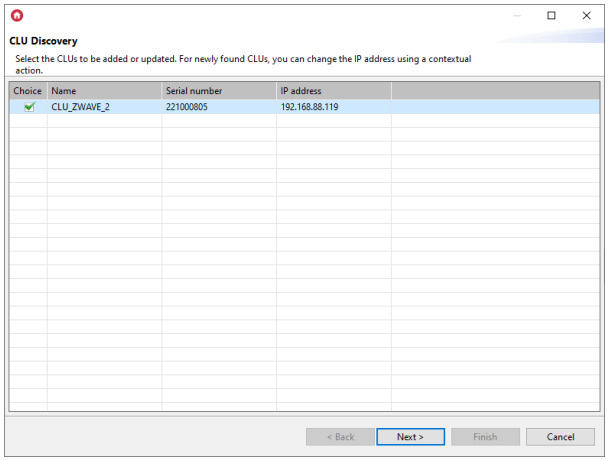
5. In the next step, OM starts searching available CLU modules.



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.

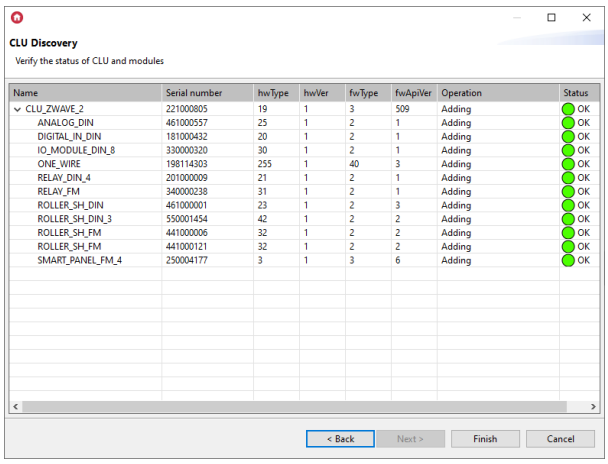


6. After entering the key for the found CLU, a window containing all CLUs found during the CLU Discovery procedure is displayed. For a given CLU module to be added to the project the check box in the *Choice* column must be selected.



In this window, it is also possible to change the IP - option is called from the context menu (PPM) for a given CLU (selected in the line).

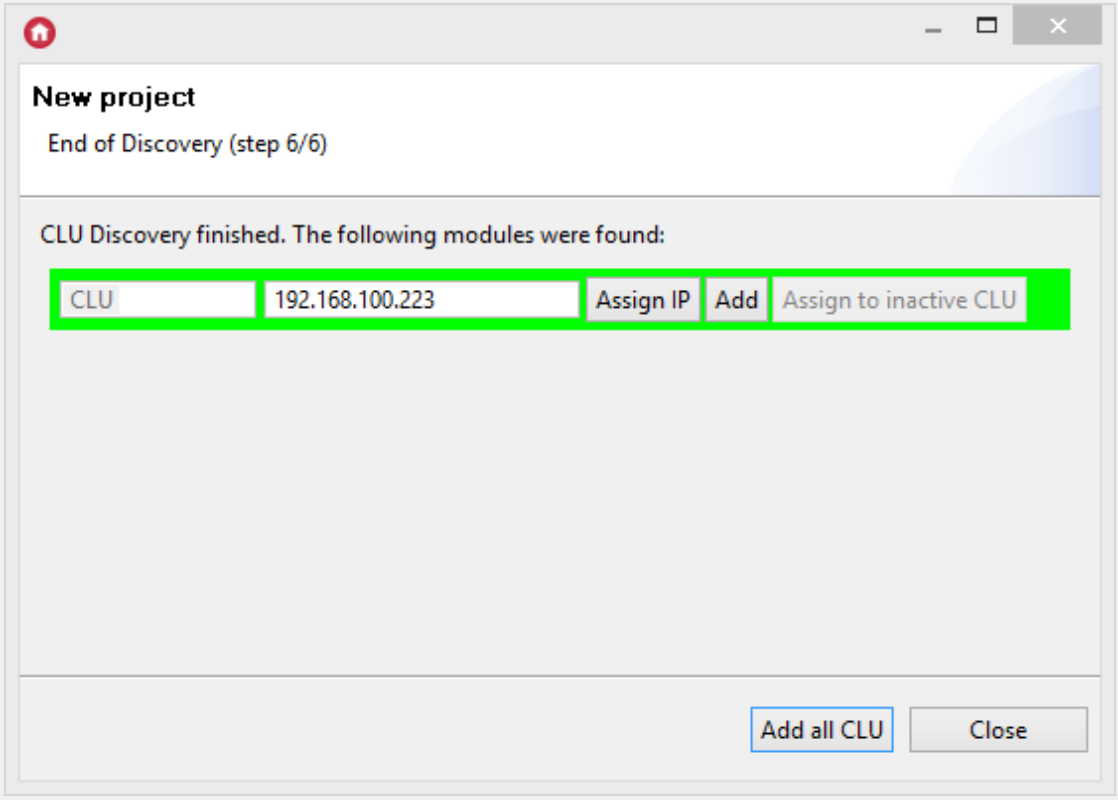
Going to the next window (*Next* button), a list of all selected (in the previous step) CLUs along with the modules that are connected to it (TF-Bus) / are paired (Z-Wave) is displayed.



The description of the information displayed in the above window can be found in chapter VI.4. CLU Discovery function.

Note!

For Object Manager version below 1.6.0, the windows referred to in point 6 are not displayed - 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.

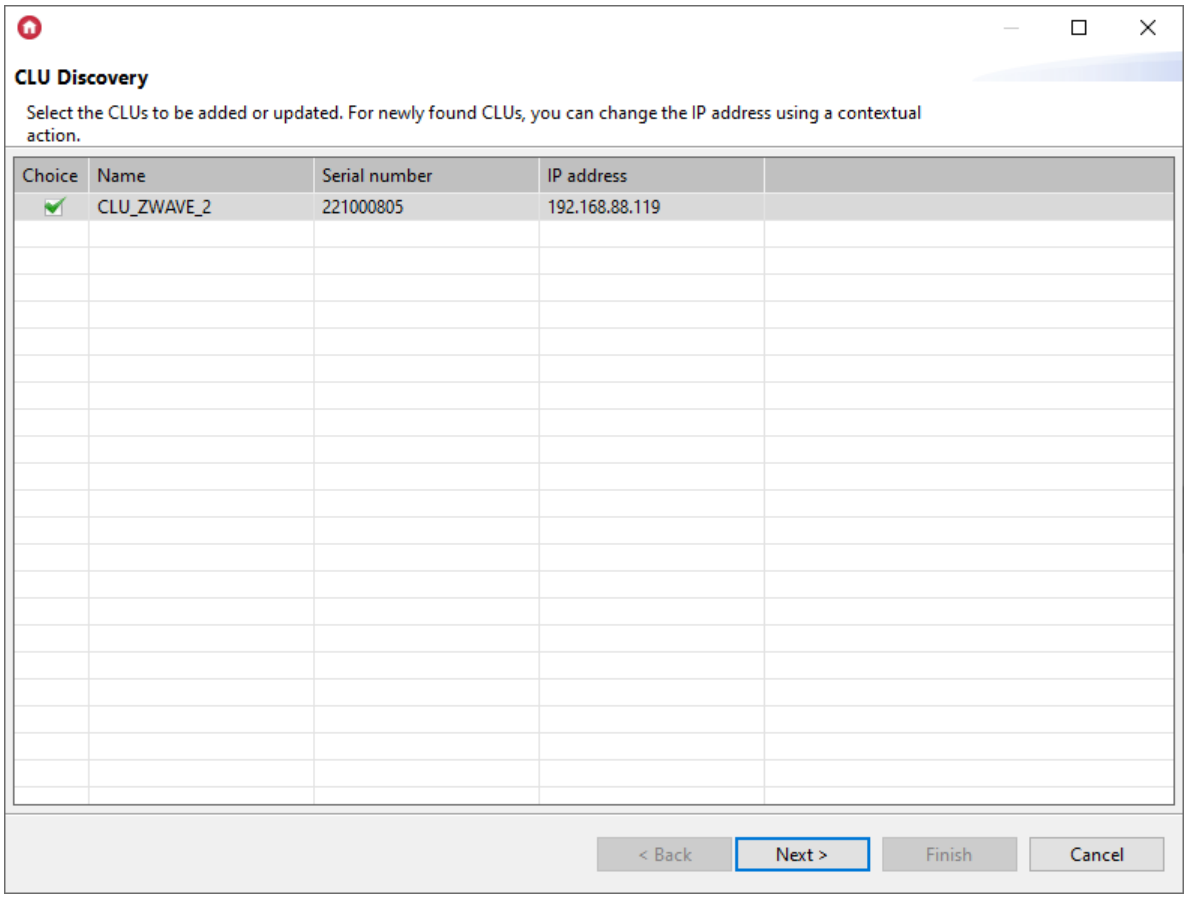
Before running CLU Discovery, make sure that:

- 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.

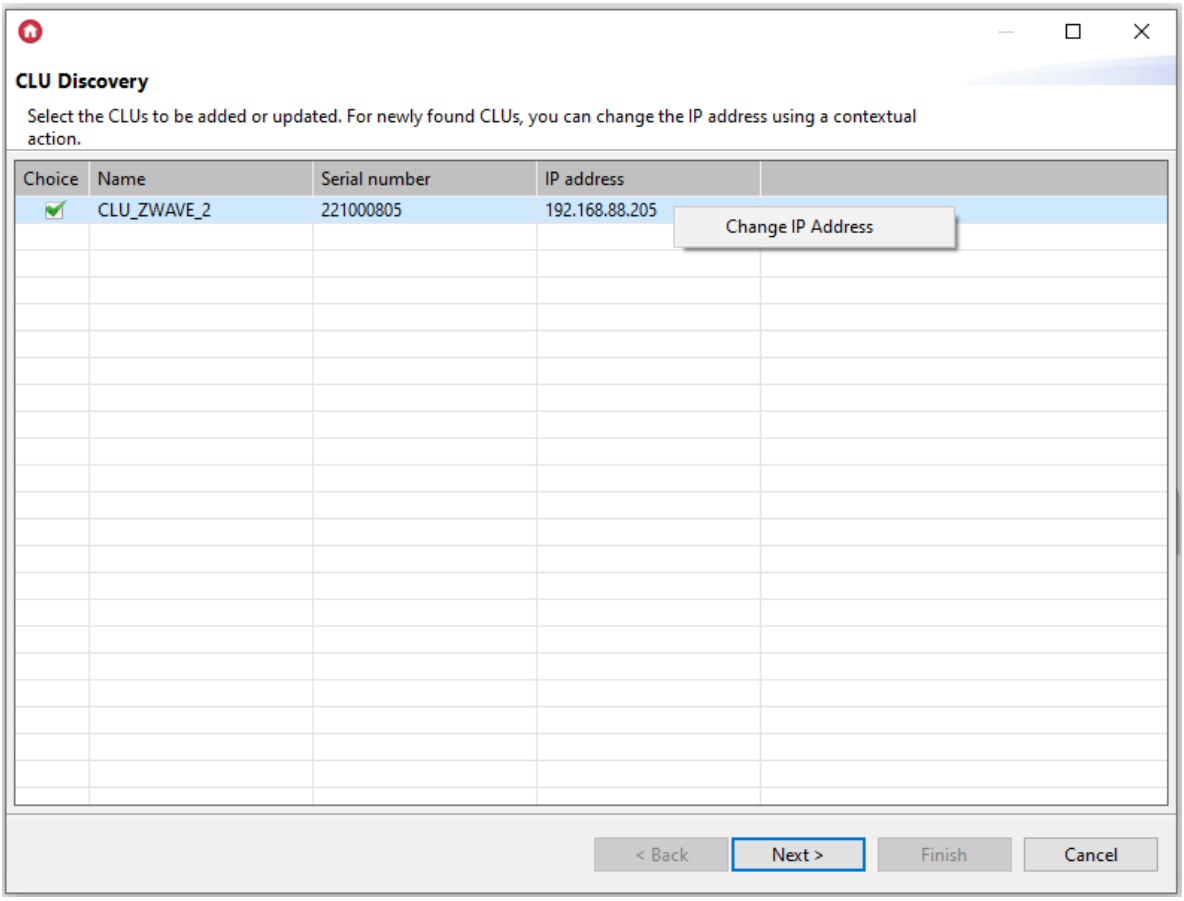
> **Note!**
> If the network consists of router, it is recommended to connect the computer directly to the CLU with a network cable when running CLU Discovery. In the majority of cases, CLU Discovery will run successfully also while connected through the router, however, in the case of a specific router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

4.1. Adding modules to the project

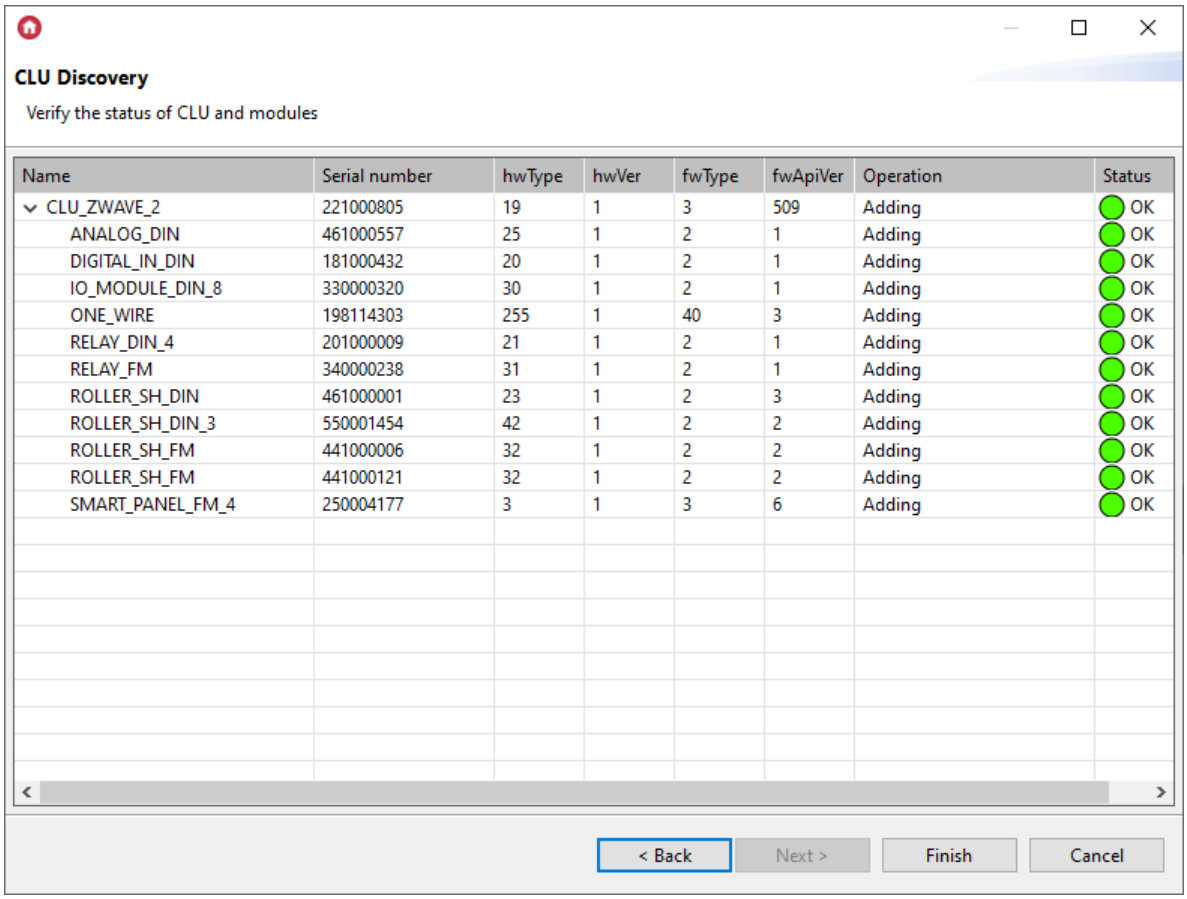
The displayed window contains all CLUs found during the CLU Discovery procedure. For a given CLU module to be added to the project, the check box in the *Choice* column must be selected.



In this window, it is also possible to change the IP address - the option is called from the context menu (PPM) for a given CLU (selected in the line). The option of changing the IP address / rewriting the configuration is available only for newly found CLU modules.



Going to the next window (using *Next* button) a list of all selected (marked in the previous step) CLU is displayed along with the modules that are connected to it (TF-Bus modules) or are paired (Z-Wave modules).



The window displays the following information:

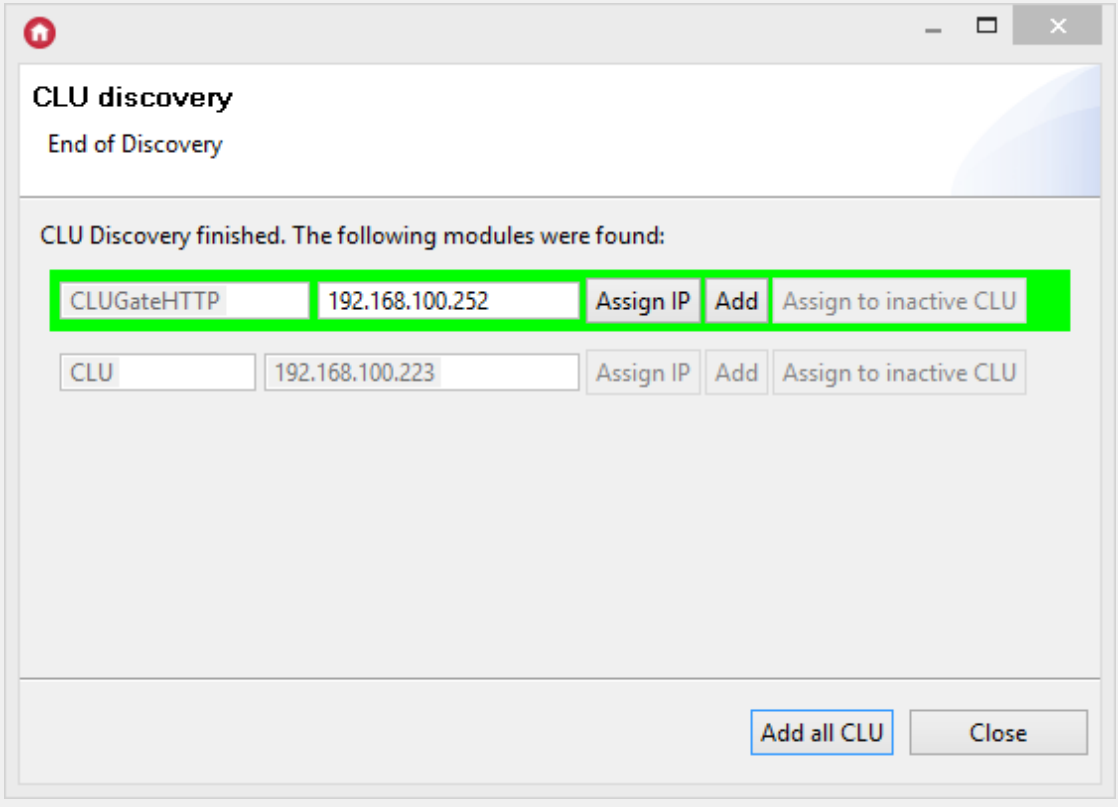
- **Name** - name of the device (CLU / module);

- **Serial number** - the number of a given device assigned by the manufacturer (TF-Bus modules) or during the pairing process (Z-Wave modules);
- **hwType, hwVer, fwType, fwApiVer** - configuration parameters of a given device;
- **Operation** - information on what action will be performed for a given device (CLU / module):
  - **Adding** - a new module is added to the project;
  - **Adding (Reassigning)** - the configuration is rewritten between the inactive CLU and the active CLU (applies to the situation in which CLU Discovery was run on an existing project);
  - **Updating** - objects of a given device will be updated - in accordance with the update of a given module (when fwApiVer is changed). In the case of an update two values are displayed in the *fwApiVer* column: current value and value before the module update process (applies to the situation where CLU Discovery was run on an existing project);
  - **Removing** - if a given module (TF-Bus / Z-Wave) is not found during CLU Discovery the objects of a given module in the project are grayed out (applies to a situation where CLU Discovery has been run on an existing project);
  - **None** - no changes to the configuration of a given module (applies to a situation where CLU Discovery was run on an existing project).
- **Status** - informing about the possibility of carrying out the actions listed in the *Operation* column:
  - **OK** - correct execution of the given operation to the module;
  - **Missing XML interfaces** - the given operation cannot be performed because the XML interface is missing for the given module.

If all modules have **Status: OK** - it is possible to add / apply changes for a given CLU - using the *Finish* button.

In a situation when a given device has the Status **Missing XML interfaces**, it is not possible to add / apply changes for the given CLU to which the given module is connected. In this case, update the interface database and then run CLU Discovery again.

Note!
For Object Manager version below 1.6.0, the above-mentioned windows are not displayed (CLU selection and device status) - all found modules will be displayed as a List.



Color 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 color** – CLU previously added to the project (only if CLU Discovery was used on pre-existing project)

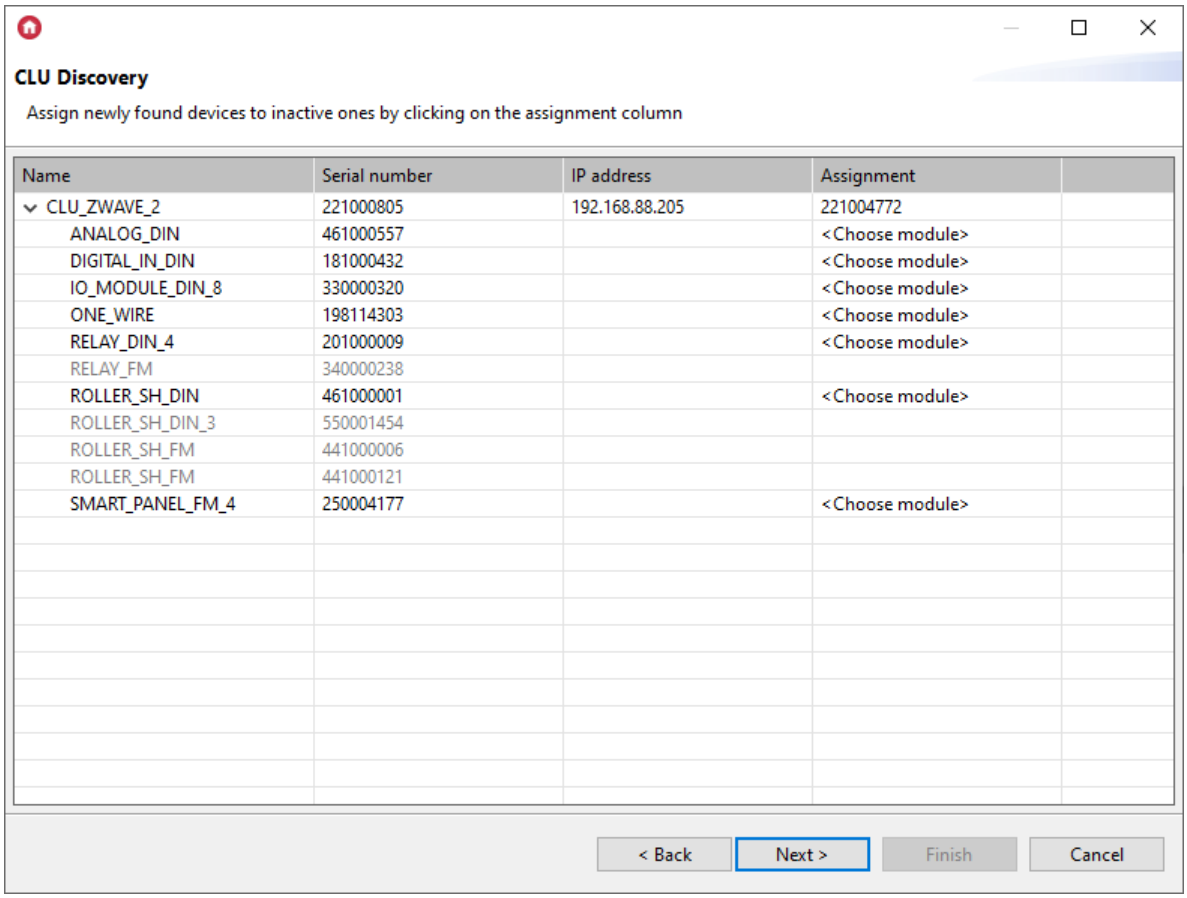
Modules may be added one by one by clicking *Add* button, or all at once by clicking *Add all CLU* button.

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

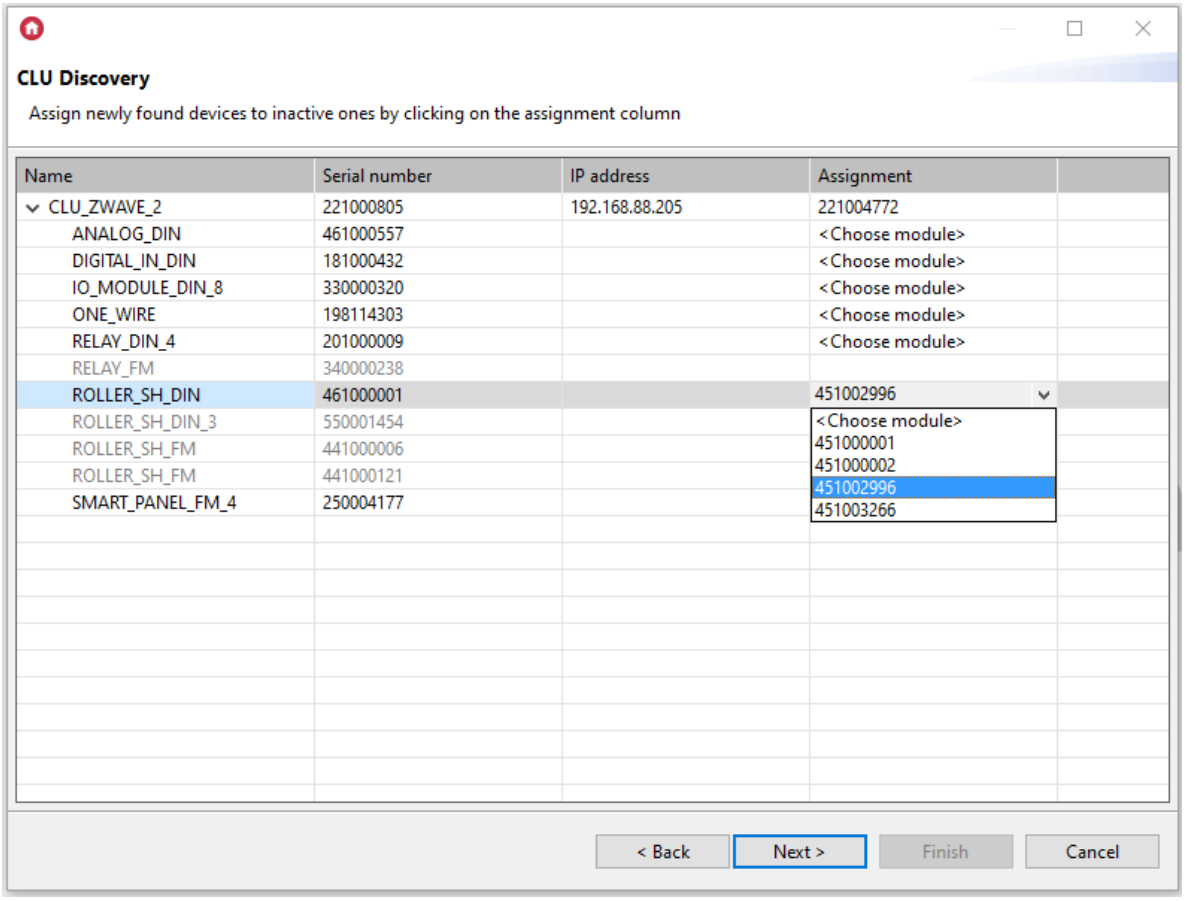
4.2. Replacing / Reassigning modules during Discovery process

Note!
The ability to assign modules during CLU Discovery is available for Object Manager version 1.7.0 or higher.

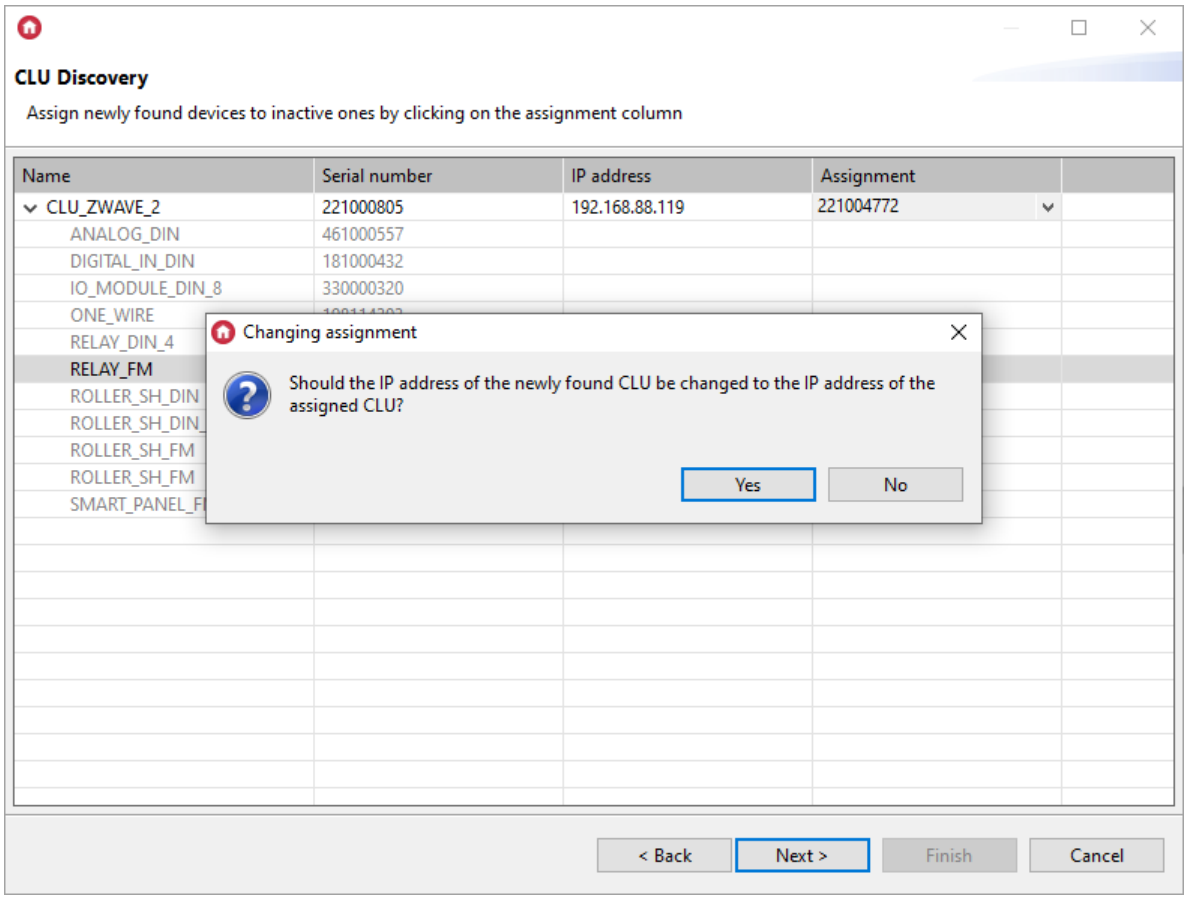
In the case of physical module replacement, during the Discovery process it is possible to assign the configuration of an inactive (disconnected) module to a new one added to the installation. Assignment takes place within the entire module (its objects), without the need to assign individual objects of a given module.



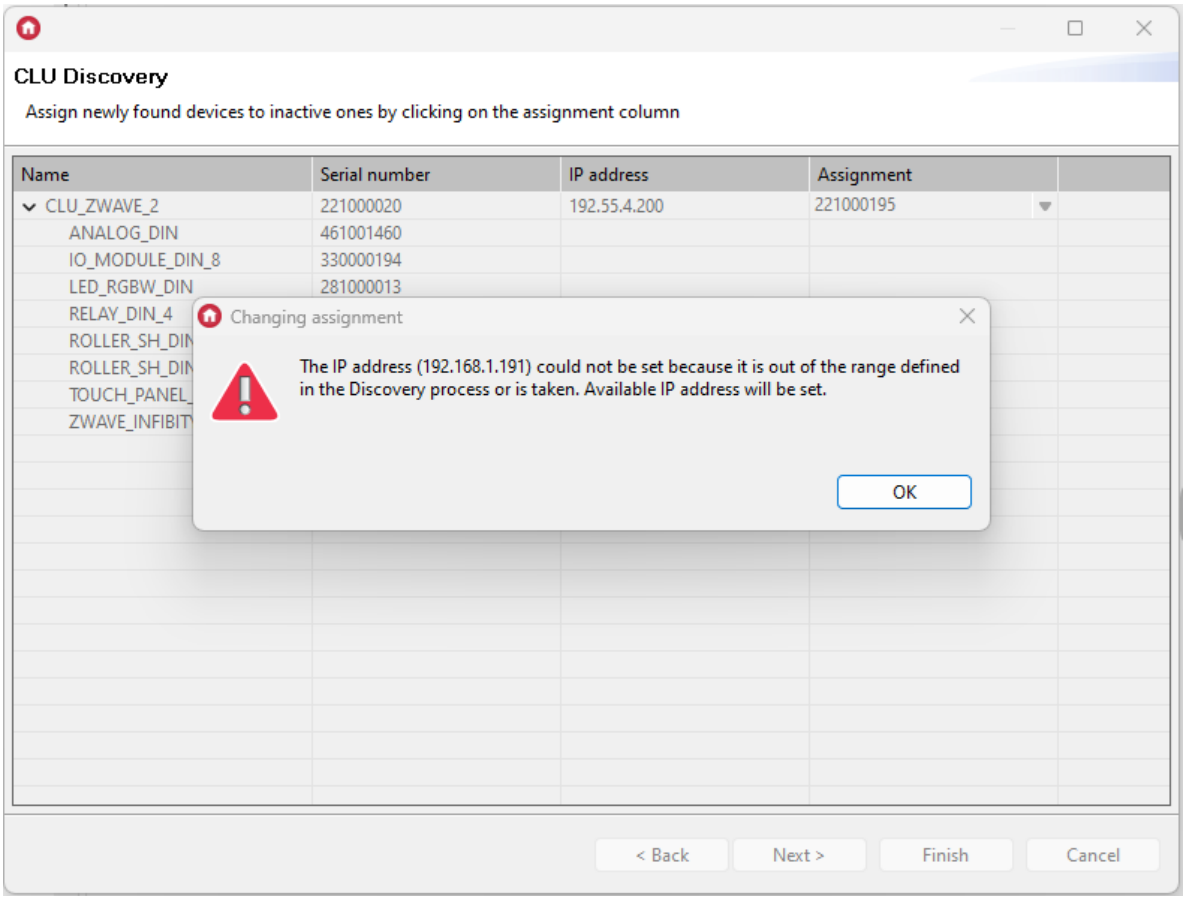
The window appears only when there are possible assignments between modules of the same type. The view presents the newly found CLU and / or modules and their possible assignments. Devices are presented in the hierarchical form `CLU-> Modules`. Modules that do not match the assignment they are greyed out. For matching modules the *Assignment* column for a given module displays a list of inactive modules whose configuration can be assigned.



In the case of assigning the CLU configuration, after selecting the inactive module, a message is displayed that the IP address has been changed to the inactive CLU address - the change will be performed if the given address is available.

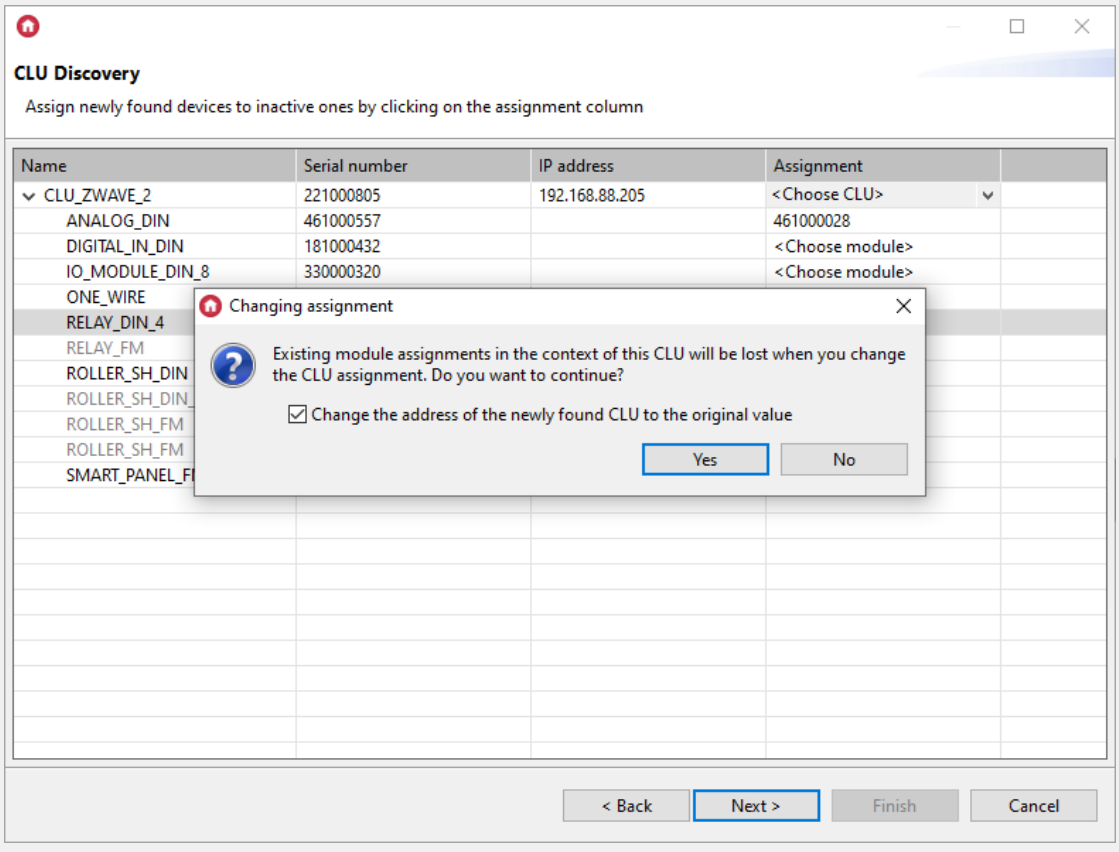


In case of busy / no access to the address an adequate message is displayed and the IP address remains the same as given during the Discovery process.

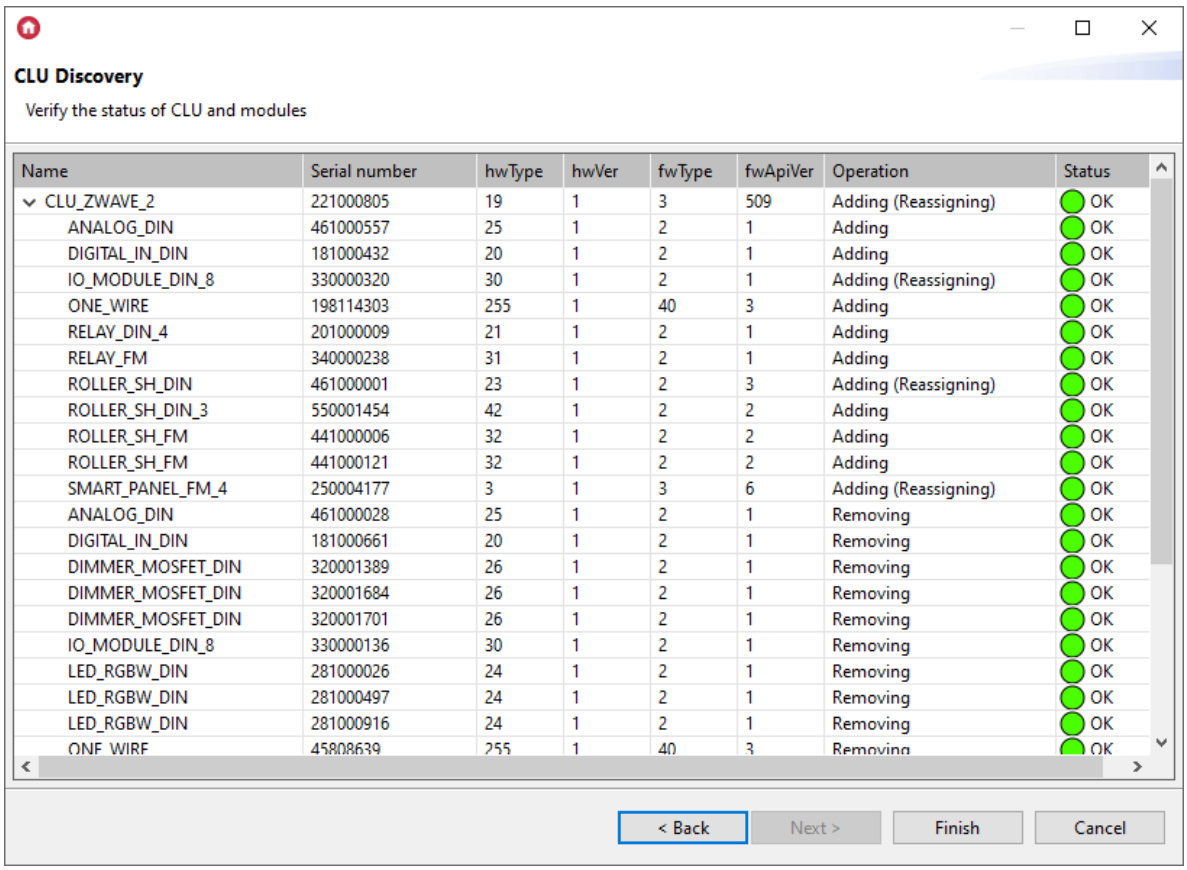


Note!

If the assignment for a CLU is removed, the assignment made between the modules of a given CLU is removed, and it is possible to restore the previous IP address given during Discovery (if it was changed during the assignment) - the following message is displayed:



After assigning in the next step (Discovery summary), the assigned modules have the status *Adding (Reassigning)*.



By clicking the *Finish* button, the configuration is sent to the CLU.

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
	Both LEDs blink every 700ms - CLU in logging mode

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.

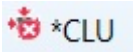
Disconnected

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:

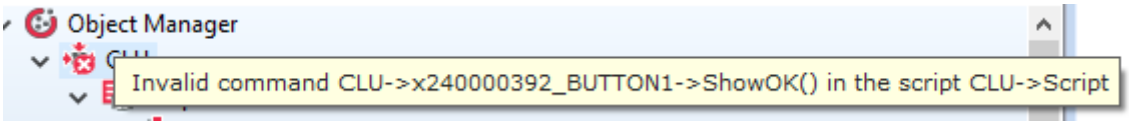
If the CLU is in disconnected mode, the user has an option of making and saving changes in the project, but the new configuration won't be sent to CLU – that is only possible in normal mode.

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 switch 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 edition), 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 color 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 recognizable (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

For IOM Z-Wave modules to be present in the system, it is necessary to add them to the CLU. This can be done in two ways:

By clicking *LINK* button located on the CLU module

For this purpose, it is necessary to press the `LINK` button located on the CLU module with the Z-Wave controller.

After pressing it, CLU goes into the module adding mode - the ON diode blinks constantly at 200ms intervals.

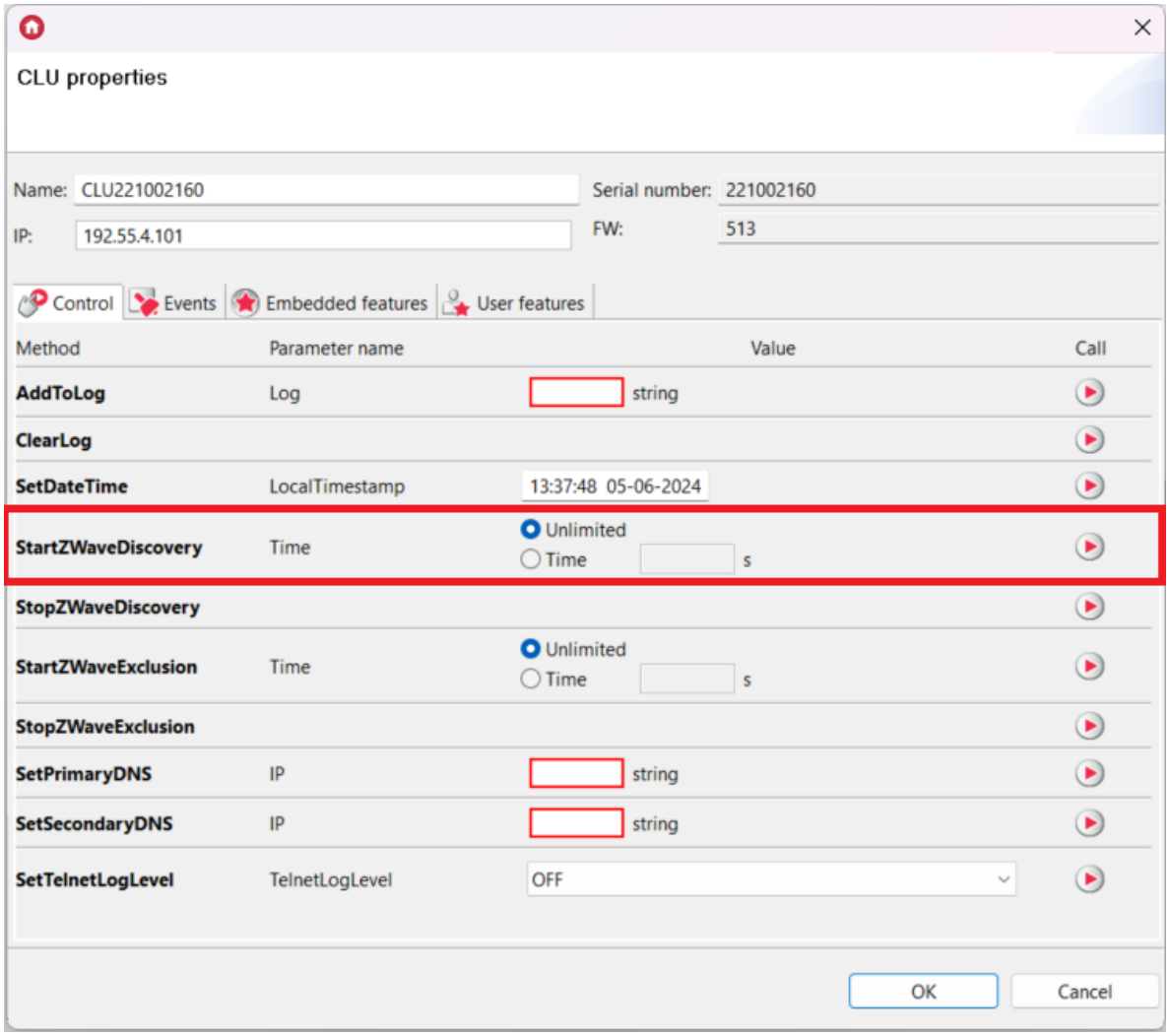
Then the Z-Wave module should be put into inclusion mode (according to the module instructions). Correct addition of the module will be signaled by the ON diode turning on for 1 second, and then by the ON and ERR diodes flashing three times at 200ms intervals. After adding the Z-Wave module, the ON diode will start blinking at a frequency of 500ms.

After adding Z-Wave modules, perform *CLU Discovery* - new Z-Wave modules will be added to the project.

Using the *StartZWaveInclusion* method from the Object Manager

This method of adding allows you to define the time during which CLU will wait for the wireless modules to "present themselves". This solution is very useful when the added modules are located at a greater distance from the CLU and more time is needed to press the button located on them.

In order to add wireless modules using OM, open the object configurator for the CLU Z-Wave module to which wireless modules will be added (double-click on the CLU icon in the object list).



There are two time options available for the `Time` parameter of the `StartZWaveInclusion` method:

- `Unlimited` - the adding process will run until the Z-Wave module is added to the network.
- `Time` - the given time will be the time that CLU waits for new Z-Wave modules to be registered. Once this time has elapsed, the search ends, even if no modules were found.

After calling the `StartZWaveInclusion` method, CLU goes into the module adding mode - the ON diode blinks constantly at 200ms intervals.

Then the Z-Wave module should be put into inclusion mode (according to the module instructions). Correct addition of the module will be signaled by the ON diode turning on for 1 second, and then by the ON and ERR diodes flashing three times at 200ms intervals. After adding the Z-Wave module, the ON diode will start blinking at a frequency of 500ms.

After adding Z-Wave modules, perform the *CLU Discovery* process - 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 given module has been added previously, 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 - first perform the procedure for removing such a module.

6.2. Removing Z-Wave modules

In order for the wireless module to stop appearing in the system configuration, it must be removed from the Z-Wave CLU.

By clicking *UNLINK* button located on the CLU module

For this purpose, it is necessary to press the `Unlink` button located on the CLU module with the controller.

After pressing it, CLU goes to the module removal mode - the ERR diode blinks constantly at 200ms intervals.

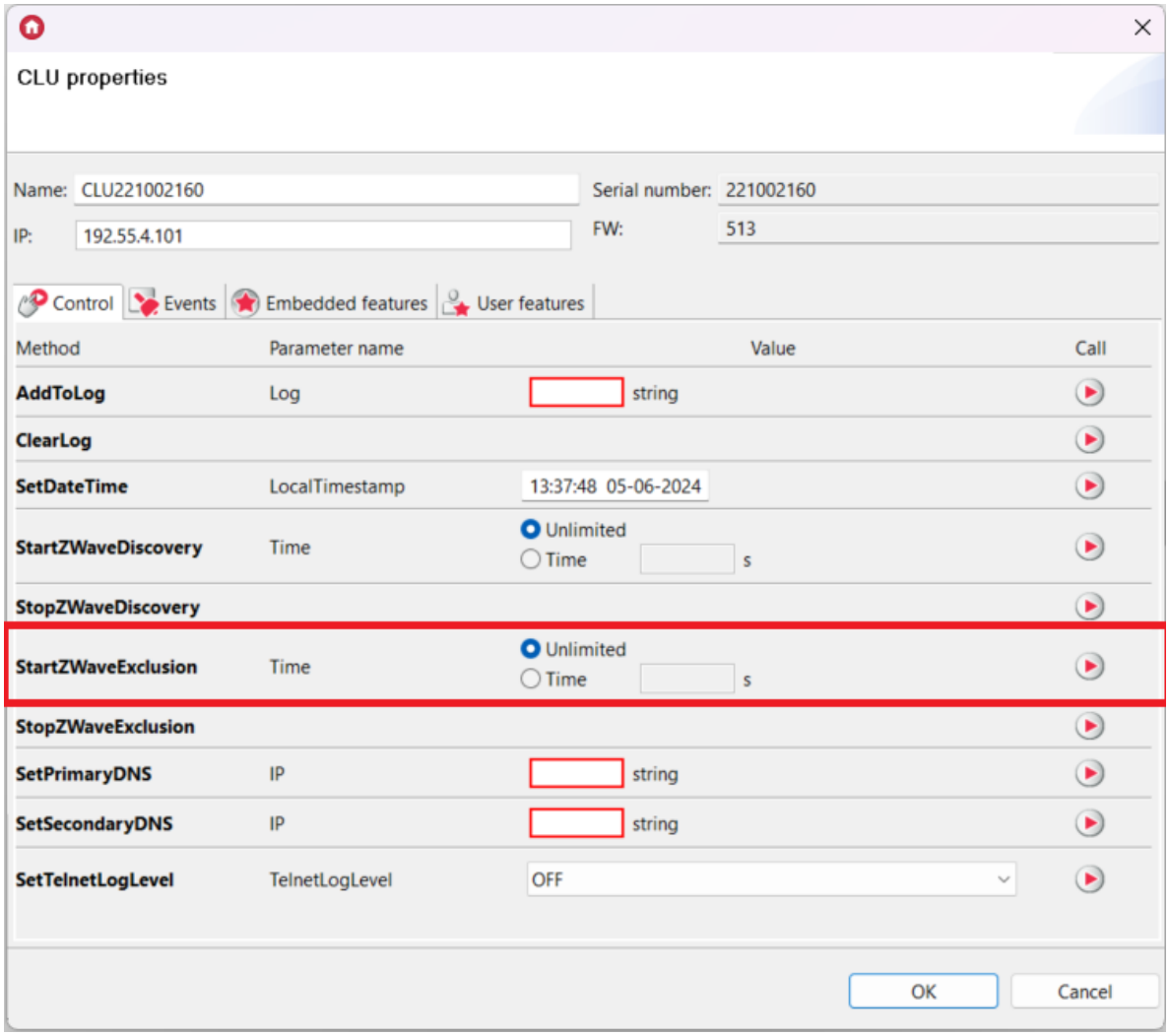
Then the Z-Wave module should be put into exclusion mode (according to the module instructions). Correct removal of the module will be signaled by the ON and ERR LEDs flashing three times at 200ms intervals. After the Z-Wave module removal is completed, the ERR LED will turn off and ON will start blinking at a frequency of 500ms.

The last step will be to perform *CLU Discovery* - the deleted modules will be grayed out.

Using the *StartZWaveExclusion* method from the Object Manager

This method of adding allows you to define the time during which CLU will wait for the wireless modules to "present themselves". This solution is very useful when the modules to be removed are located at a greater distance from the CLU and more time is needed to press the button located on them.

To remove wireless modules using OM, open the object configurator for the Z-Wave CLU module to which wireless modules will be added (double-click on the CLU icon in the object list).



There are two time options available for the `Time` parameter of the `StartZWaveExclusion` method:

- `Unlimited` - the adding process will run until the Z-Wave module is removed from the network.
- `Time` - the given time will be the time that CLU waits for Z-Wave modules to respond. Once this time has elapsed, the search ends, even if no modules have been removed.

After calling the `StartZWaveExclusion` method, CLU goes to the module removal mode - the ERR diode blinks constantly at 200ms intervals.

Then, the Z-Wave module to be removed should be put into exclusion mode (according to the module instructions). Correct removal of the module will be signaled by the ON and ERR LEDs flashing three times at 200ms intervals. After the Z-Wave module removal is completed, the ERR LED will turn off and ON will start blinking at a frequency of 500ms.

After adding Z-Wave modules, perform the *CLU Discovery* process - 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 given module has been added previously, 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 - first perform the procedure for removing such a module.

6.2. Removing Z-Wave modules

In order for the wireless module to stop appearing in the system configuration, it must be removed from the Z-Wave CLU.

By clicking the *UNLINK* button located on the CLU module

For this purpose, it is necessary to press the `UNLINK` button located on the CLU module with the controller.

After pressing it, CLU goes to the module removal mode - the ERR diode blinks constantly at 200ms intervals.

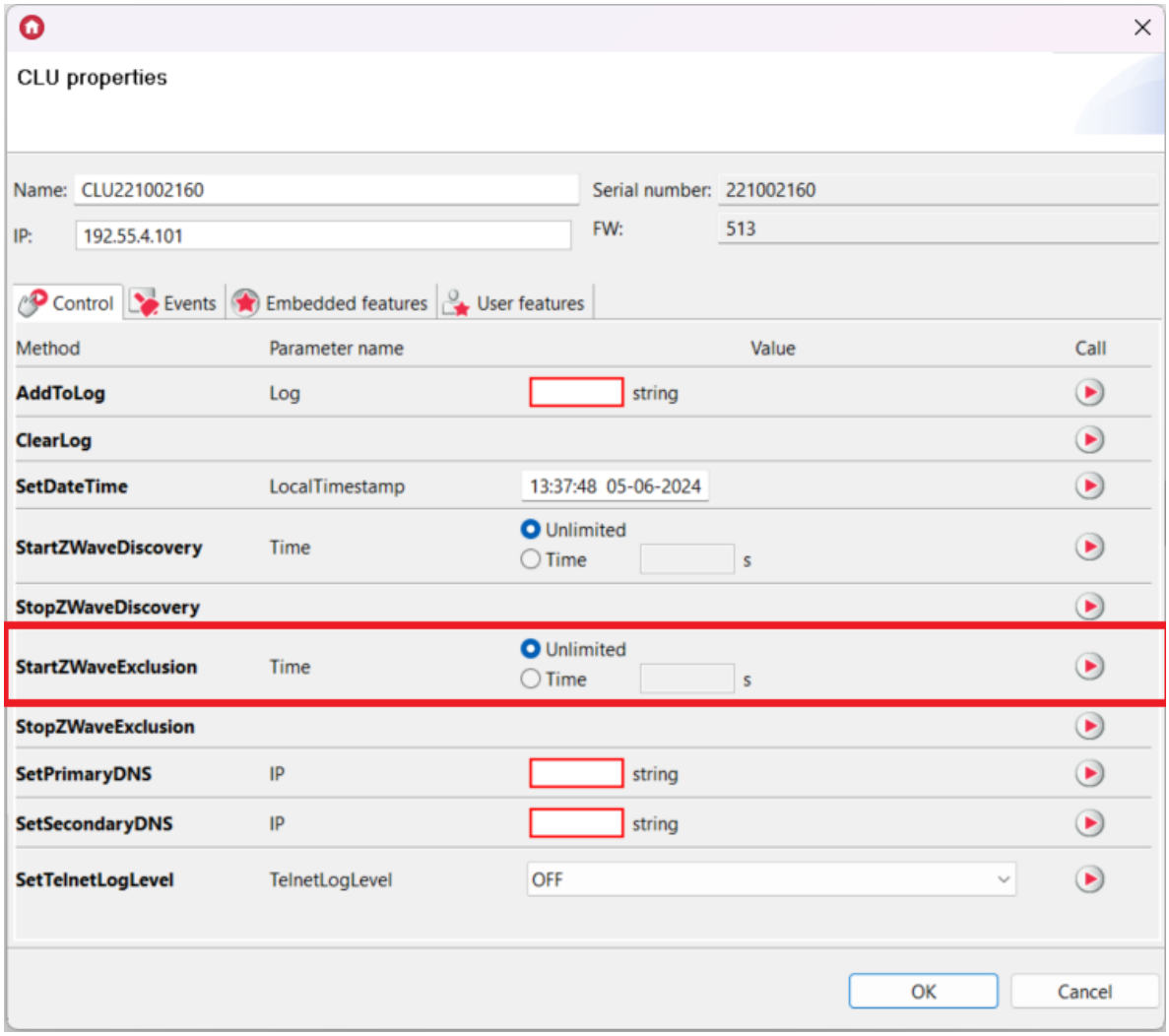
Then press the button on the wireless module (according to the module manual) that is to be removed. Correct removal of the module will be signaled by the ON and ERR LEDs flashing three times at 200ms intervals. After the Z-Wave module removal is completed, the ERR LED will turn off and ON will start blinking at a frequency of 500ms.

The last step will be to perform *CLU Discovery* - the deleted modules will be grayed out.

Using the *StartZWaveExclusion* method from the Object Manager

This method of adding allows you to define the time during which CLU will wait for the wireless modules to "present themselves". This solution is very useful when the modules to be removed are located at a greater distance from the CLU and more time is needed to press the button located on them.

To remove wireless modules using OM, open the object configurator for the Z-Wave CLU module to which wireless modules will be added (double-click on the CLU icon in the object list).



There are two time options available for the `Time` parameter of the `StartZWaveExclusion` method:

- `Unlimited` - the adding process will run until the Z-Wave module is removed from the network.
- `Time` - the given time will be the time that CLU waits for Z-Wave modules to respond. Once this time has elapsed, the search ends, even if no modules have been removed.

After calling the `StartZWaveExclusion` method, CLU goes into the module removal mode - the ERR LED blinks constantly at 200ms intervals.

Then press the button on the wireless module (according to the module manual) that is to be removed. Correct removal of the module will be signaled by the ON and ERR LEDs flashing three times at 200ms intervals. After the Z-Wave module removal is completed, the ERR LED will turn off and ON will start blinking at a frequency of 500ms.

The last step will be to perform *CLU Discovery* - the deleted modules will be grayed out.

> **Note!**
> Calling the `StopZWaveExclusion` method interrupts the search for 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

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

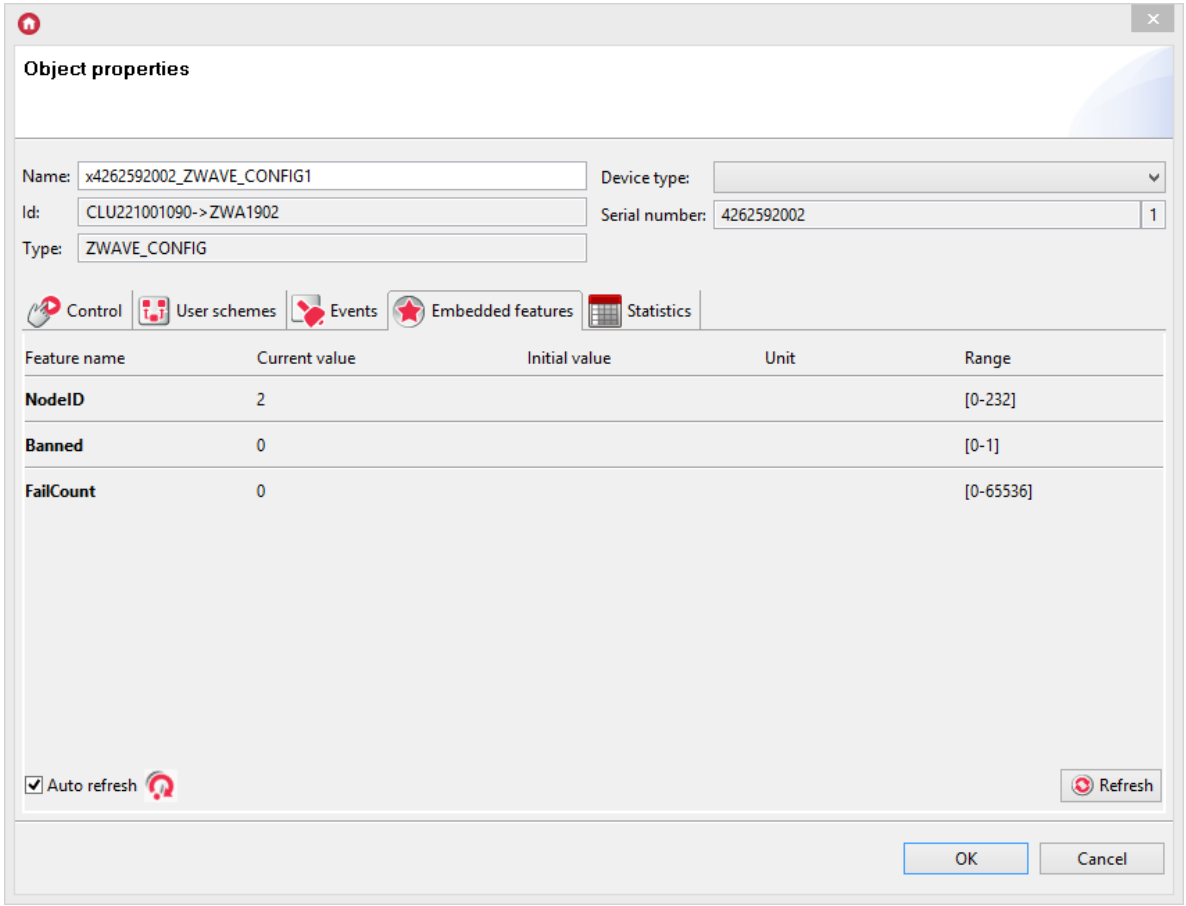
Failures in communication with a Z-Wave device may occur when:

- 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.

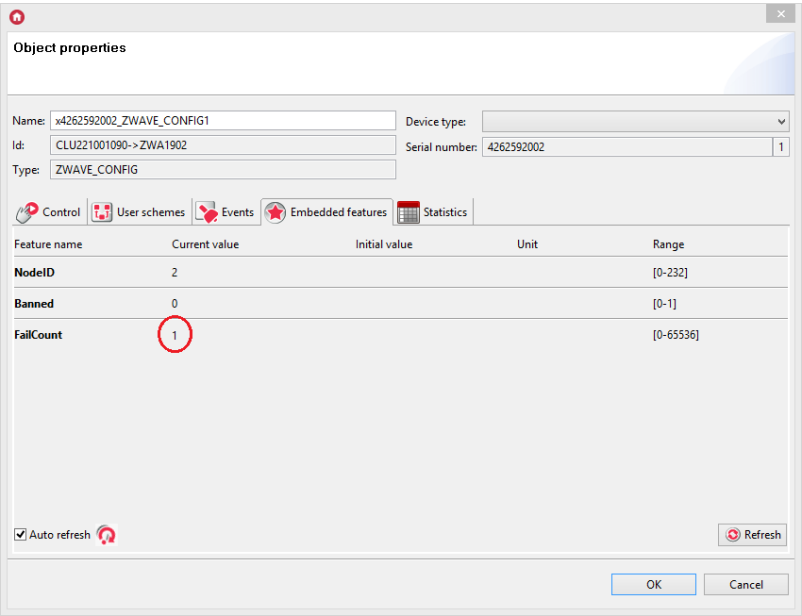
The following features are available for a given object:



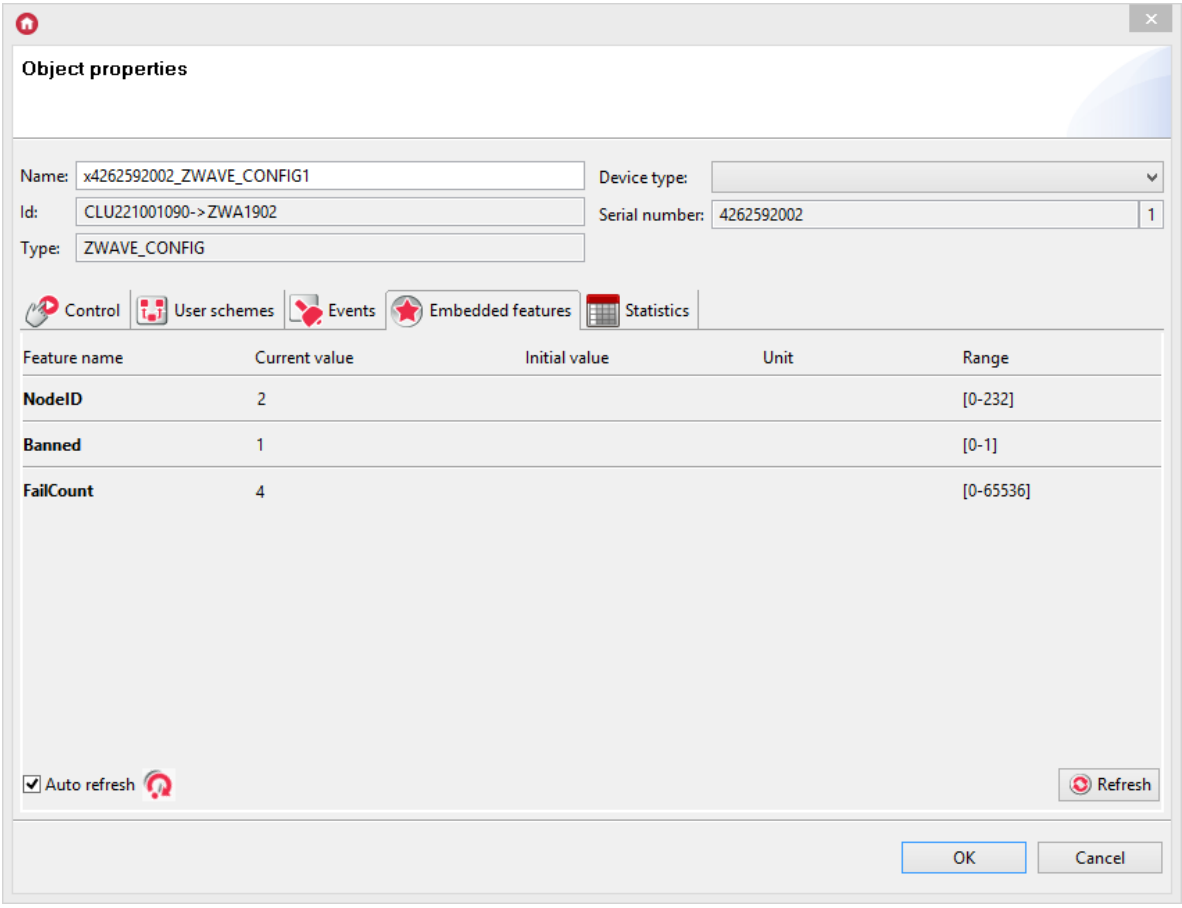
- `NodeID` - Number of the module (node) in the Z-Wave network (transmitted for each Z-Wave module after adding it to the controller)
- `Banned` - Information on blocking Z-Wave communication with the module
- `FailCount` - The number of unsuccessful attempts to communicate with the Z-Wave module

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.

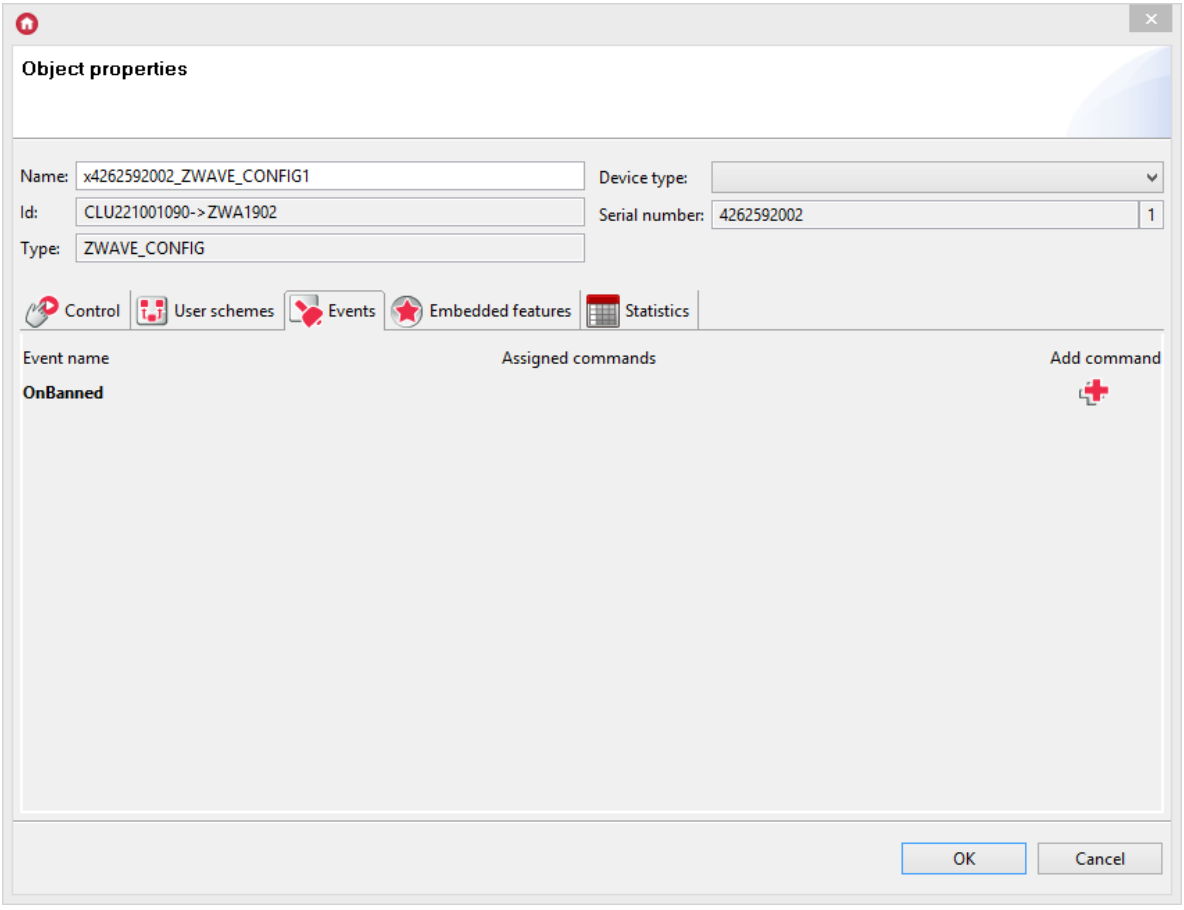


- Another attempt to send an order to the retry device is every 15 seconds - 3 attempts are made to communicate with the device.
- In the case of 3 attempts to communicate with the module, the `Banned` feature is set to 1 and all communication with the module is blocked.



Locking mechanism for communication with the module

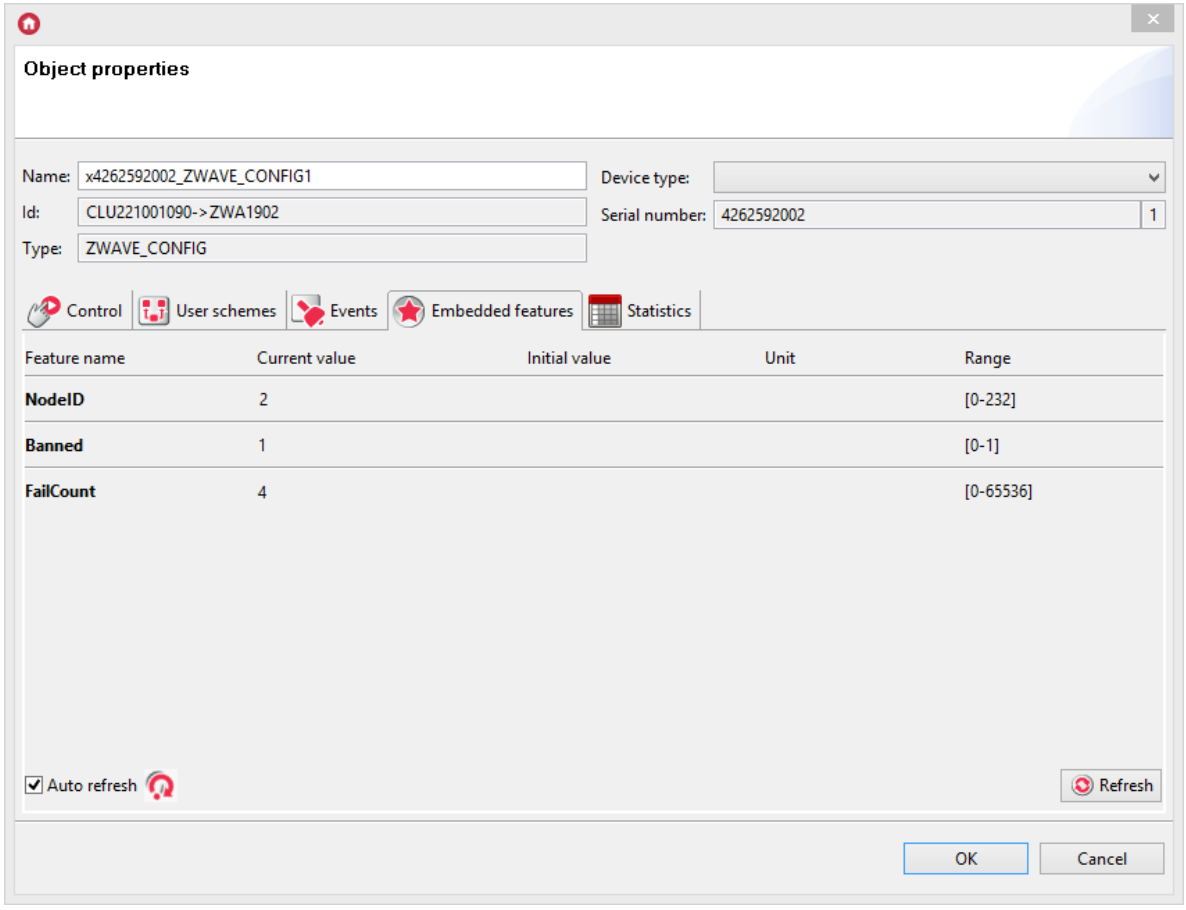
- When the `Banned` feature is set to 1, communication with the Z-Wave device is blocked - this means that all action calls on the device (ie change of output status, query for parameters) are not sent by the CLU to the blocked module.
- You can assign any action when you block communication with a given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



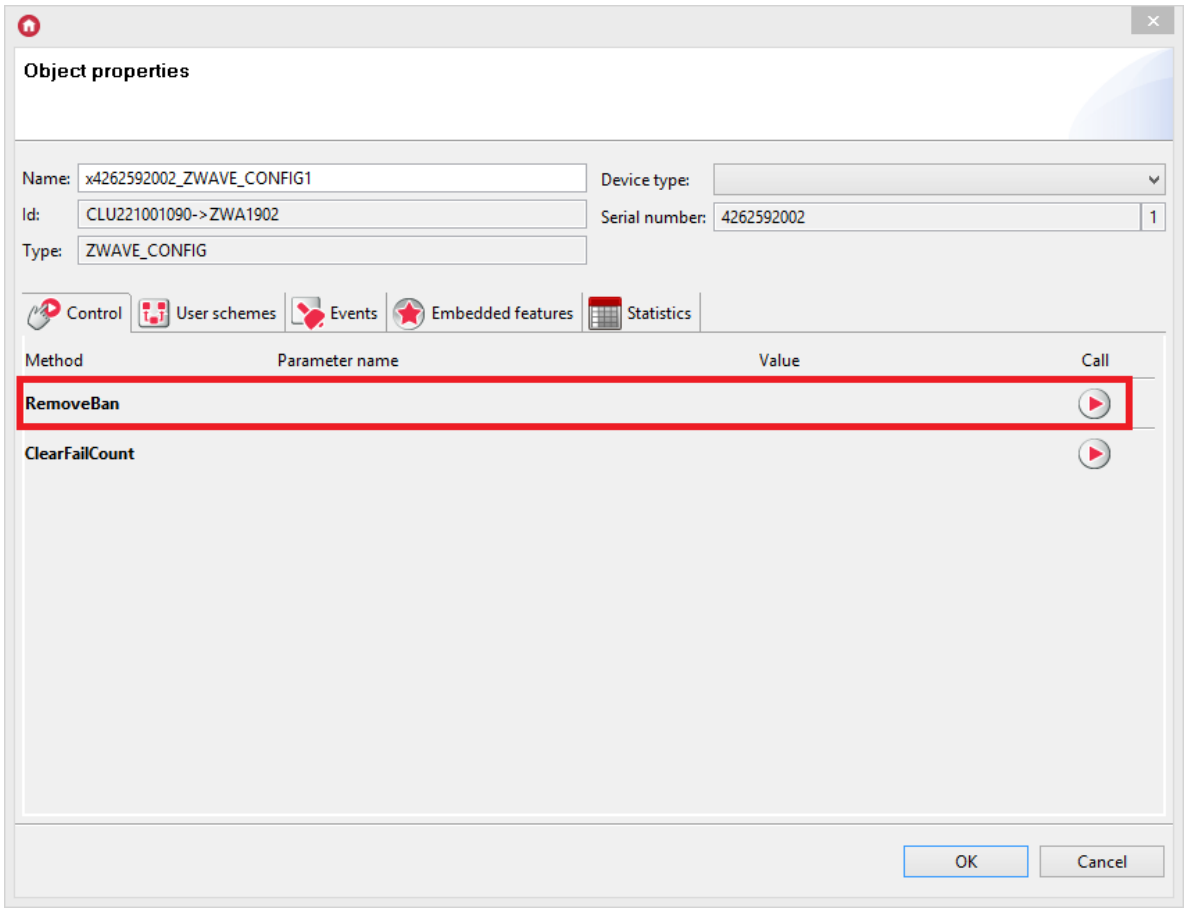
- A short query (NOP) is sent to the banned module every 1.5 minutes:
    - if the module does not confirm receipt of the query, the Banned attribute continues to be 1, and the next query is repeated every 1.5 minutes,

> **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.

> **Note!**
> `RemoveBan` is not synonymous with re-communication with the module - it allows re-sending an order / query to the module! In the event of failure, the entire blocking process is restarted!

- In the event of communication failure with the module, the entire mechanism (counting failures in communication and blocking) starts from the beginning.

It should be remembered that in case of unblocking communication with the module, the `FailCount` feature is not reset - this can be done using the `ClearFailCount` method.

6.4. Z-Wave network configuration tips

When creating a Z-Wave network, it is important to:

- Z-Wave network configuration was carried out after installing devices in the workplace.

The Z-Wave network is defined statically. Z-Wave devices should be linked when they are in their target locations. Changing the position of Z-Wave devices after adding them can cause unexpected problems with communication in the Z-Wave network - with all devices!

- The antenna (in modules that have it) has not been folded or wrapped around the module.

The antenna should be facing the module if possible.

- The battery modules are not woken up at the same time.

Waking modules at the same time leads to delays in operation. In order to avoid the described situation, different awakening times should be used for all devices (in the ZWAVE_WAKEUP object for battery modules) and selected in such a way that the set times have the largest possible "smallest common multiple", e.g. 57min, 58min, 59min , 60min, 61min, etc ...

- There are no inactive modules (damaged or incorrectly removed) in the Z-Wave network.

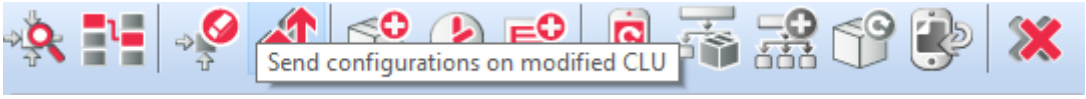
The linked module that is missing in the system causes continuous attempts to renew communication with it, which in turn can introduce temporary delays and deficiencies in communication with other devices as well.

> **Note!**
> For CLU Z-Wave placed in a box/cabinet, it is recommended to use longer antennas and take them out of the switch cabinet.

6.5. Clearing information about nodes

It is possible to simultaneously remove all Z-Wave modules from the CLU. The HardReset function is used for this purpose look up XI.1..

7. Sending the configuration to the CLU



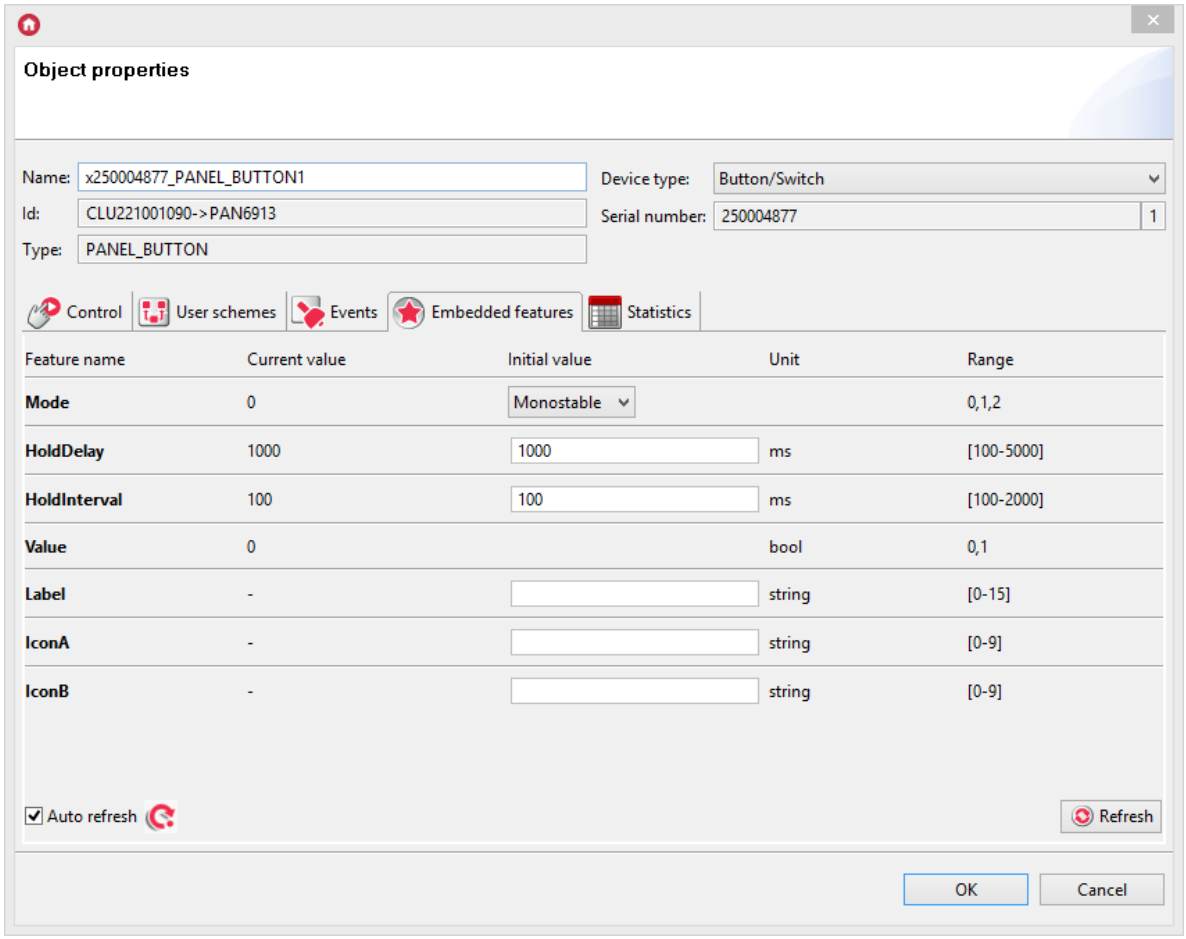
The configuration is stored in the OM and until it is sent to the CLU, it is not taken into account in the operation of the system. To send the configuration to the CLU, press the 'Send configuration' button in the menu.

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):



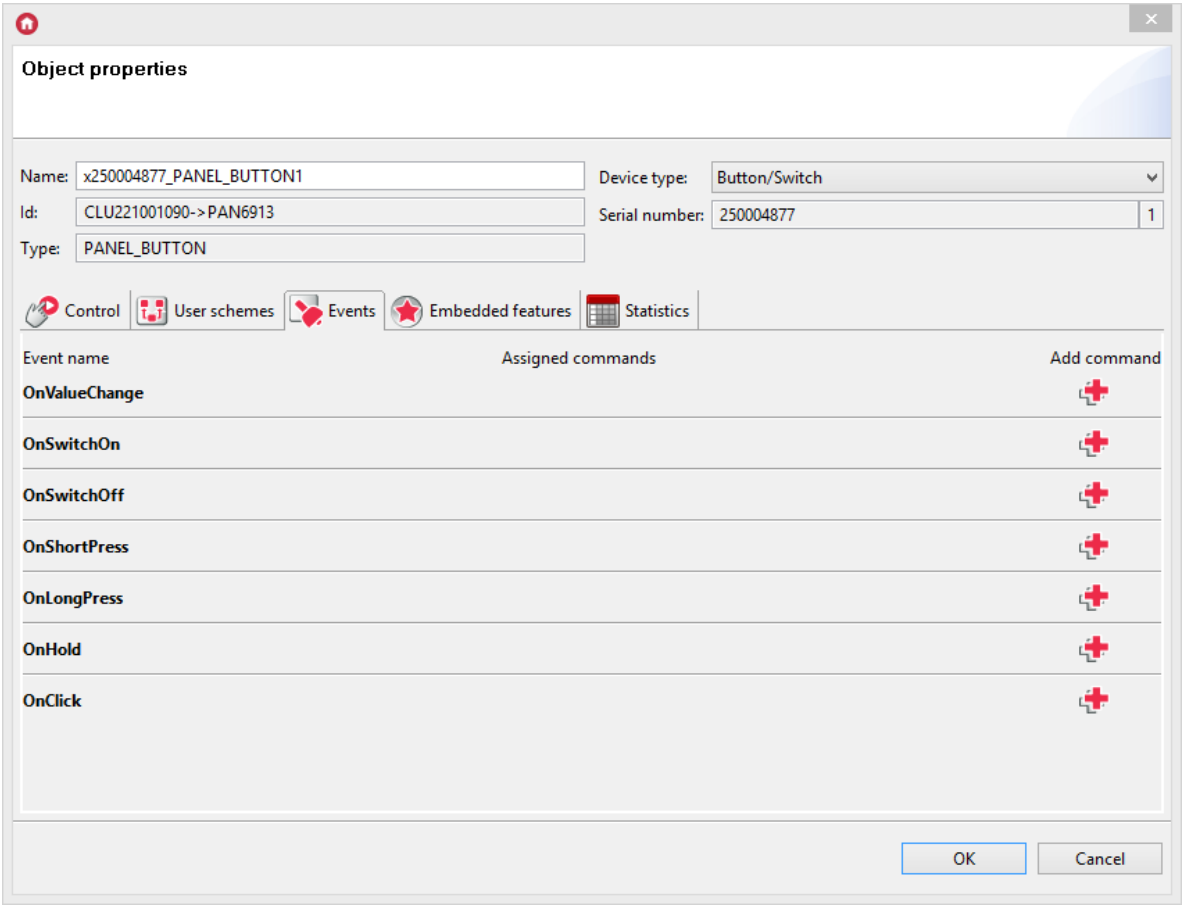
To set the selected feature, in the appropriate field, enter the desired value in the `Initial value` column, and then send the configuration to the CLU.

9. Creating basic connections

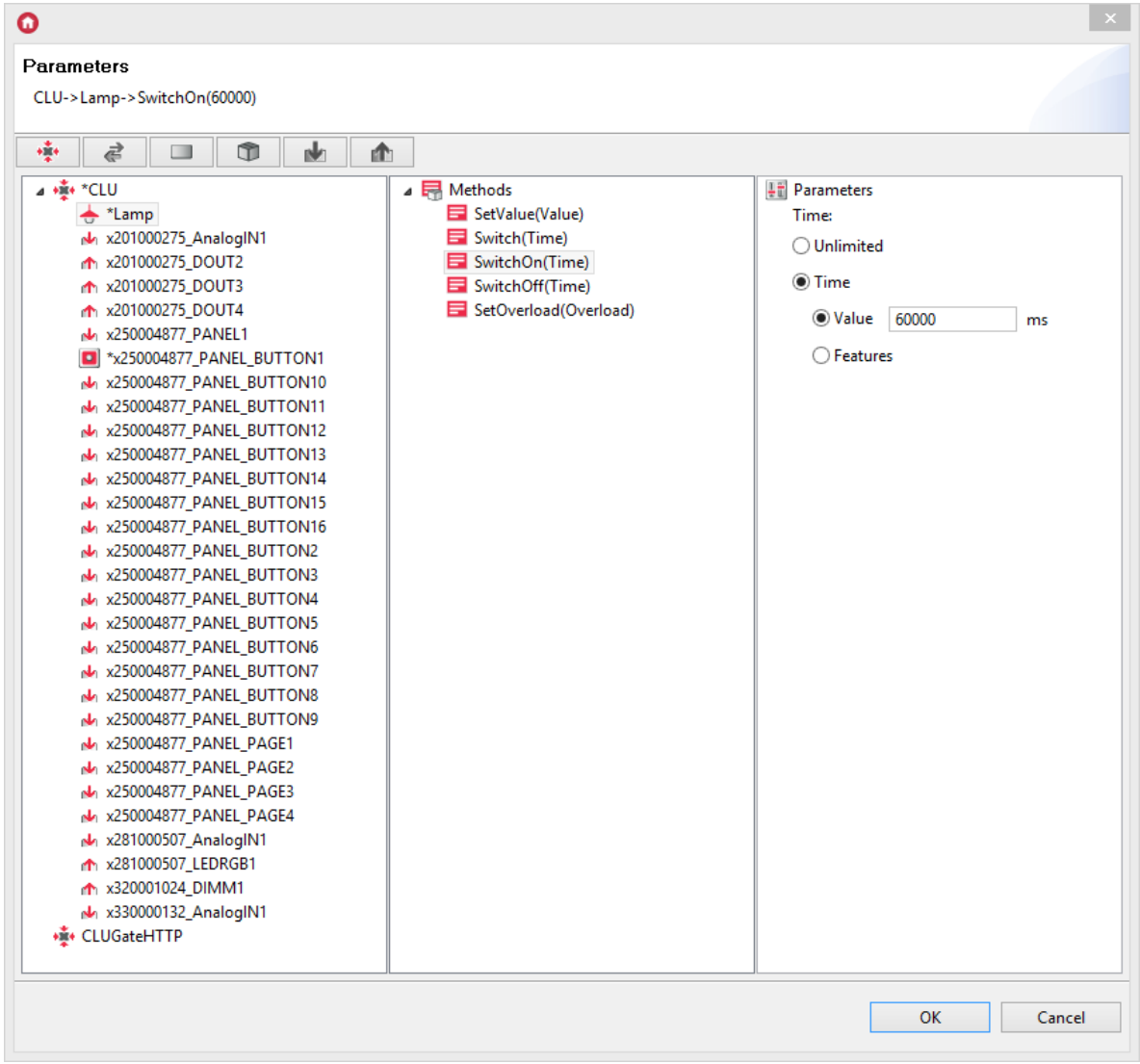
Calling reactions in the system (eg switching on the lighting after pressing the key) is accomplished by creating links between objects. As a rule, these are connections between the entrance (eg switch) and the output (lamp). However, the system does not limit the creation of connections and allows them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

To 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:

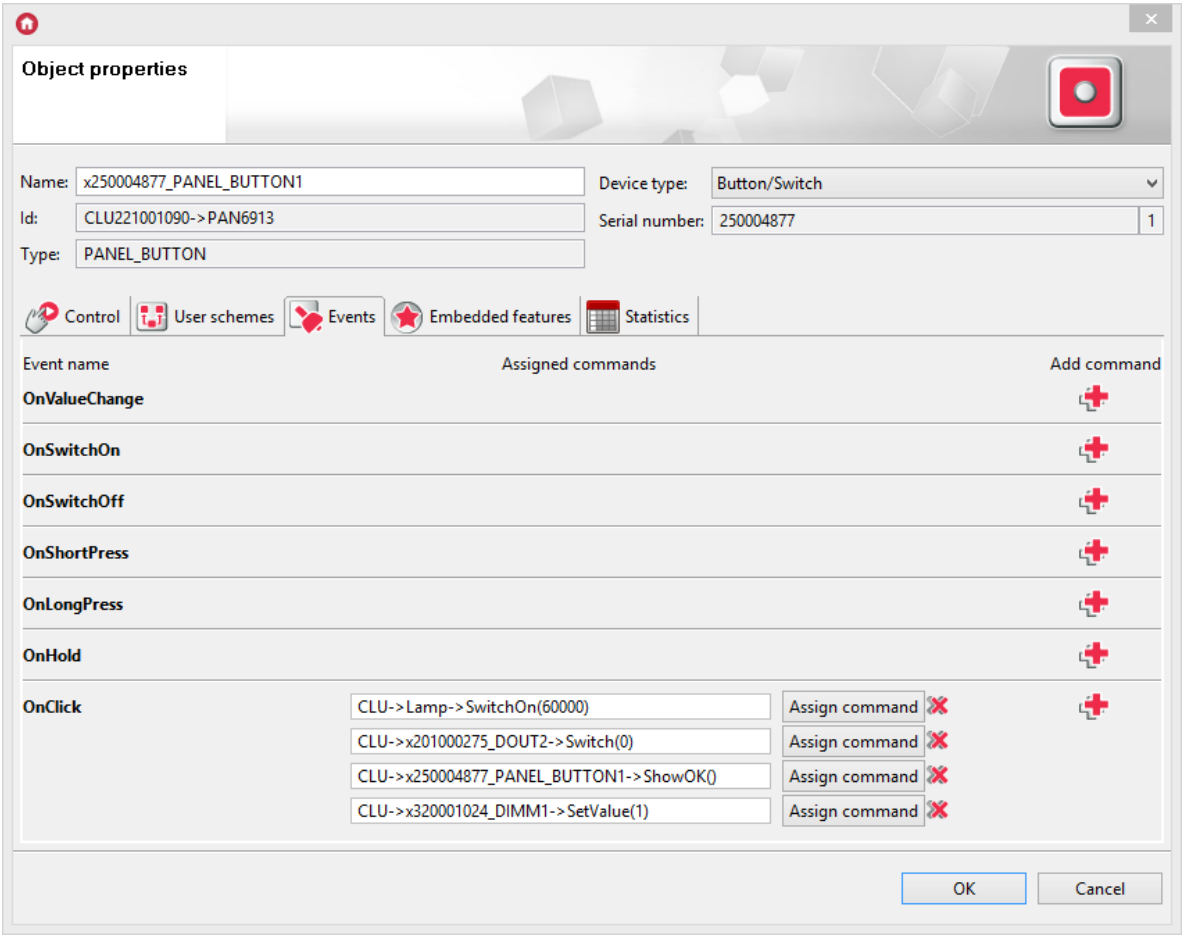


- Find the event to be linked from the list and click `+`;
- In the method selection format, select the object, method and parameters in sequence:



- 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.



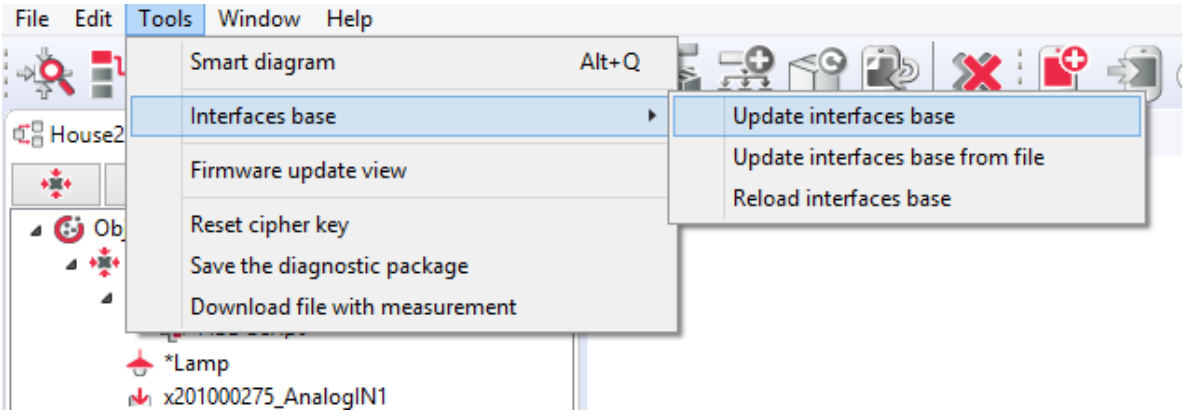
10. Performing an update

10.1. The process of updating the interface database

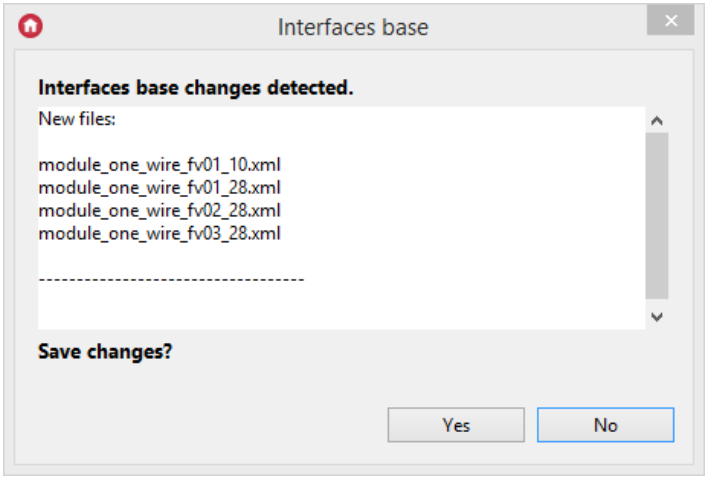
If the option *to automatically update the interfaces database* is marked when the Object Manager is started for the first time, there is no need to run it again. Otherwise, remember to update regularly. Updating the interfaces database should be done always before updating the software of a given Grenton 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:

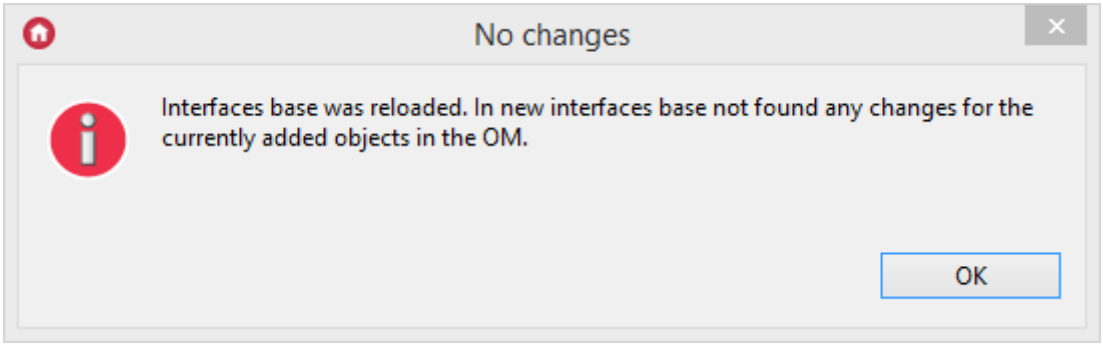
- Select `Tools` from the menu bar.
- Select the item *Interfaces base*.
- Select *Update interfaces database* from the list displayed:



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

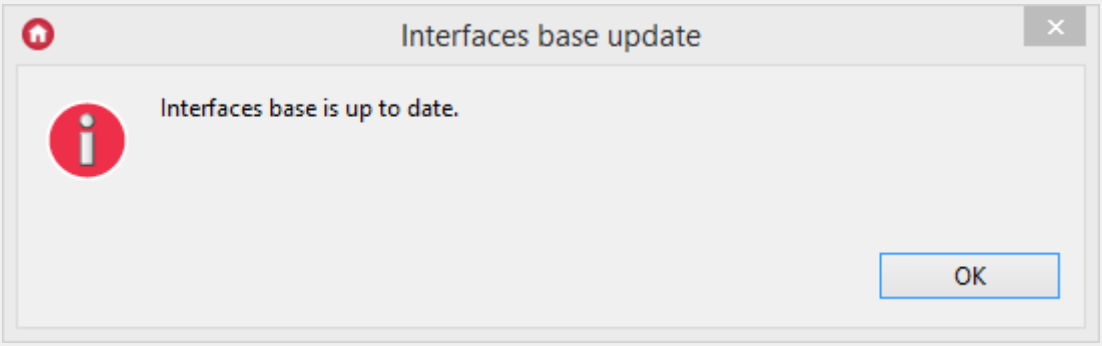


- Then a window will be displayed informing you that the interfaces base has been reloaded:



- 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!

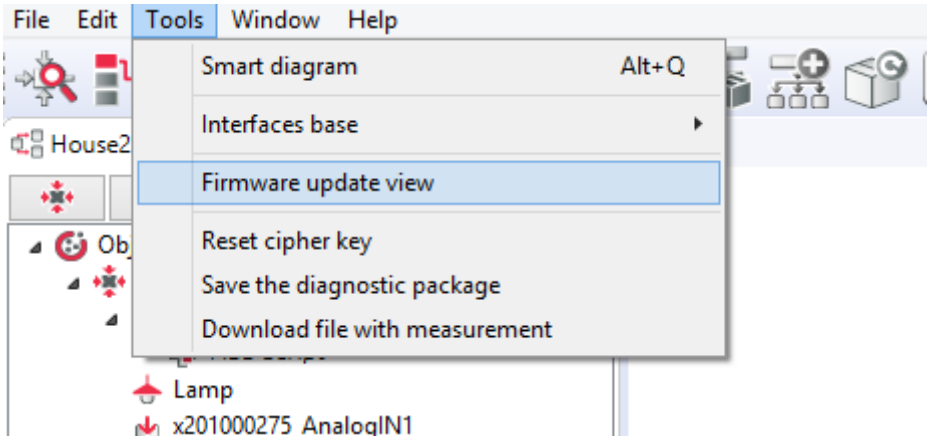
Note!
The following requirements must be met for the update process to run correctly:

- 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.

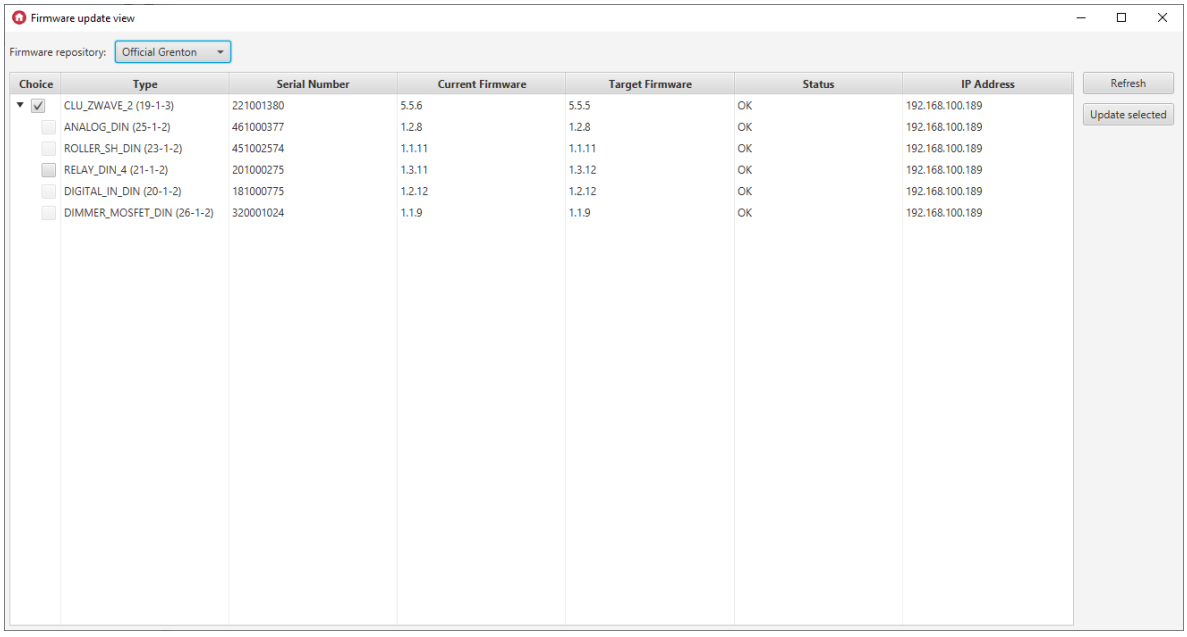
A. Update from Grenton server

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

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

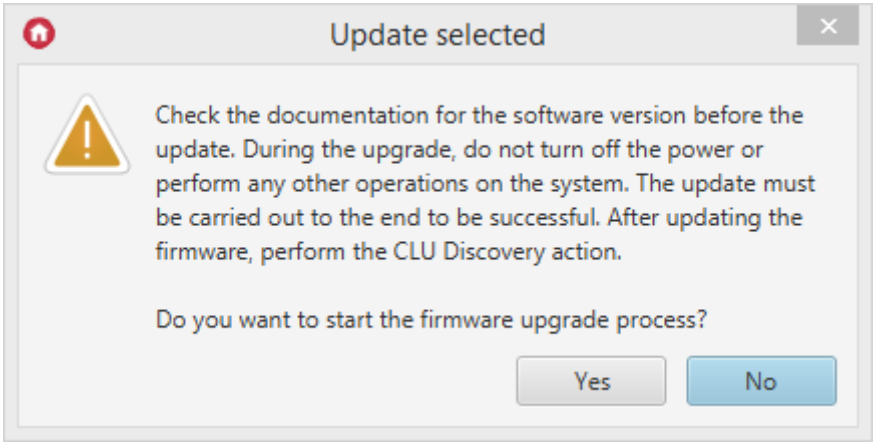


- Select the object type `CLU_ZWAVE_2`. Selecting the check box is only possible if the current firmware on the CLU is out of date:



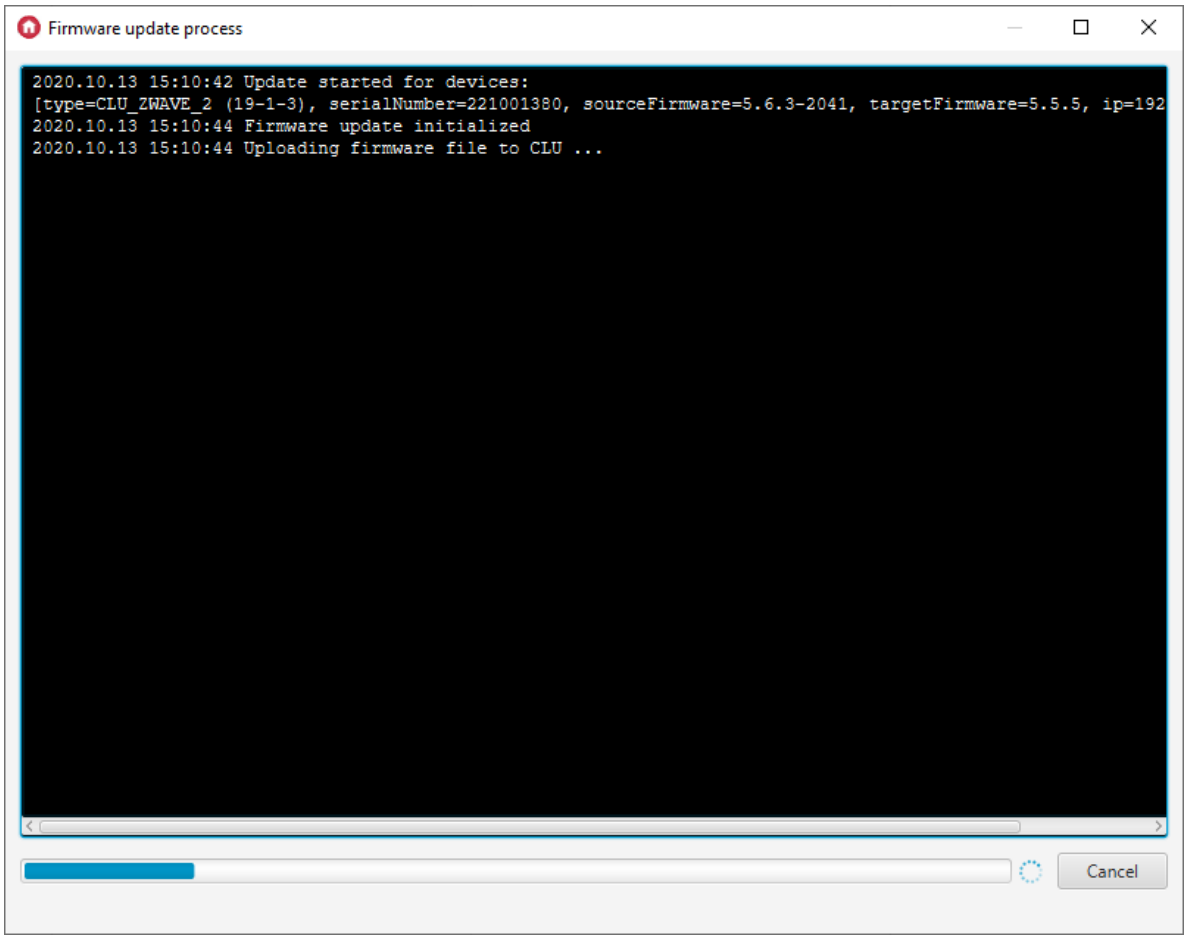
Choice	Type	Serial Number	Current Firmware	Target Firmware	Status	IP Address
☑	CLU_ZWAVE_2 (19-1-3)	221001380	5.5.6	5.5.5	OK	192.168.100.189
☐	ANALOG_DIN (25-1-2)	461000377	1.2.8	1.2.8	OK	192.168.100.189
☐	ROLLER_SH_DIN (23-1-2)	451002574	1.1.11	1.1.11	OK	192.168.100.189
☐	RELAY_DIN_4 (21-1-2)	201000275	1.3.11	1.3.12	OK	192.168.100.189
☐	DIGITAL_IN_DIN (20-1-2)	181000775	1.2.12	1.2.12	OK	192.168.100.189
☐	DIMMER_MOSFET_DIN (26-1-2)	320001024	1.1.9	1.1.9	OK	192.168.100.189

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

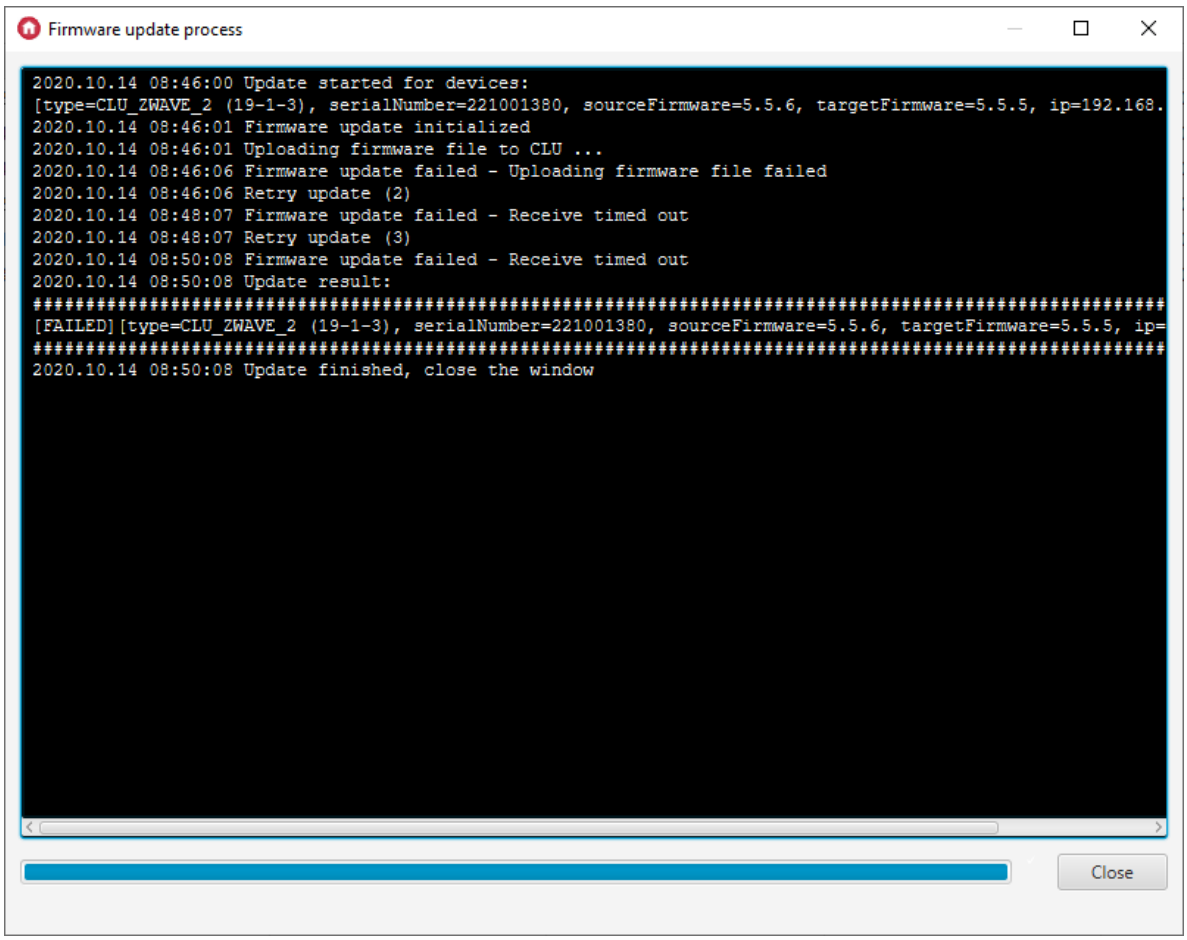


- Once accepted, the upgrade process will begin:

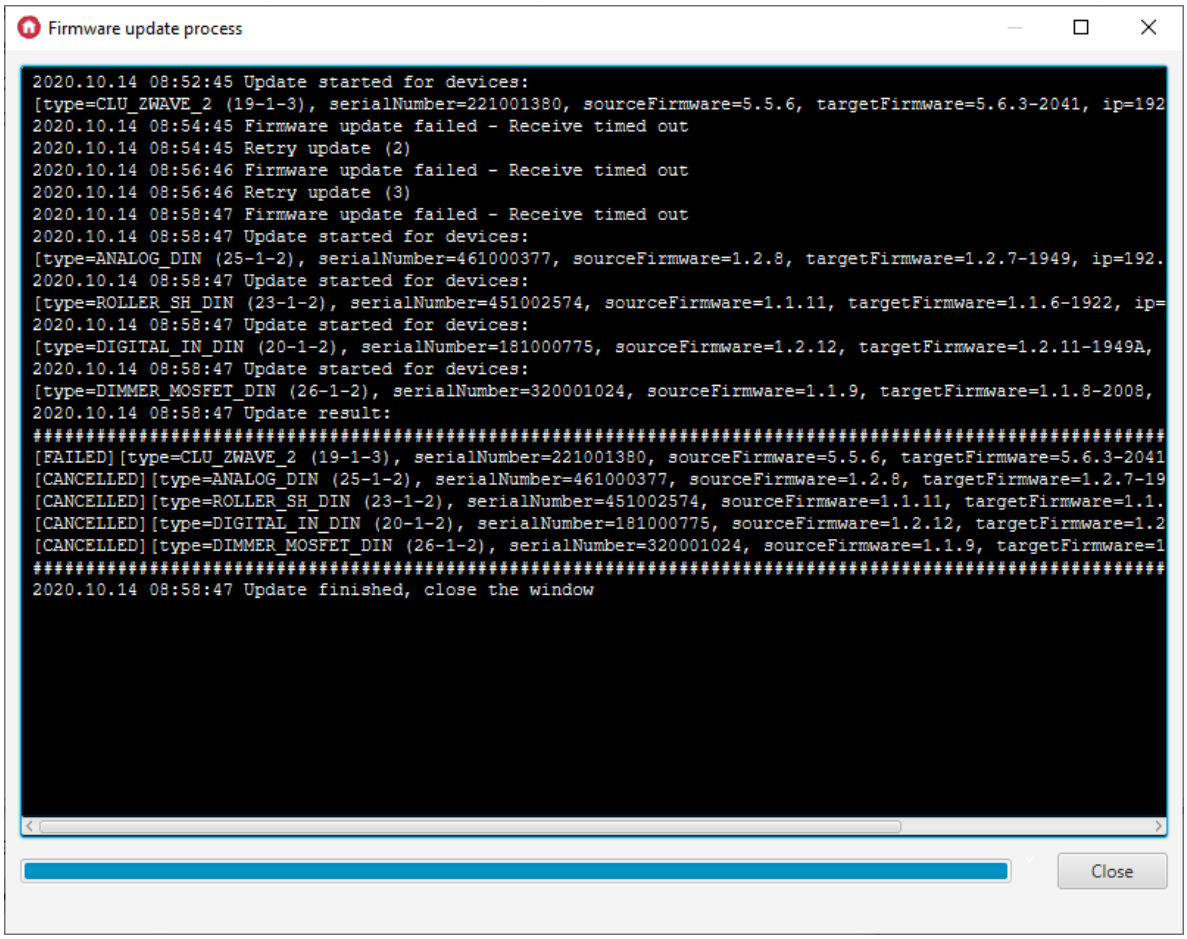
Note!
During the upgrade process, do not turn off the power or perform other activities on the system.



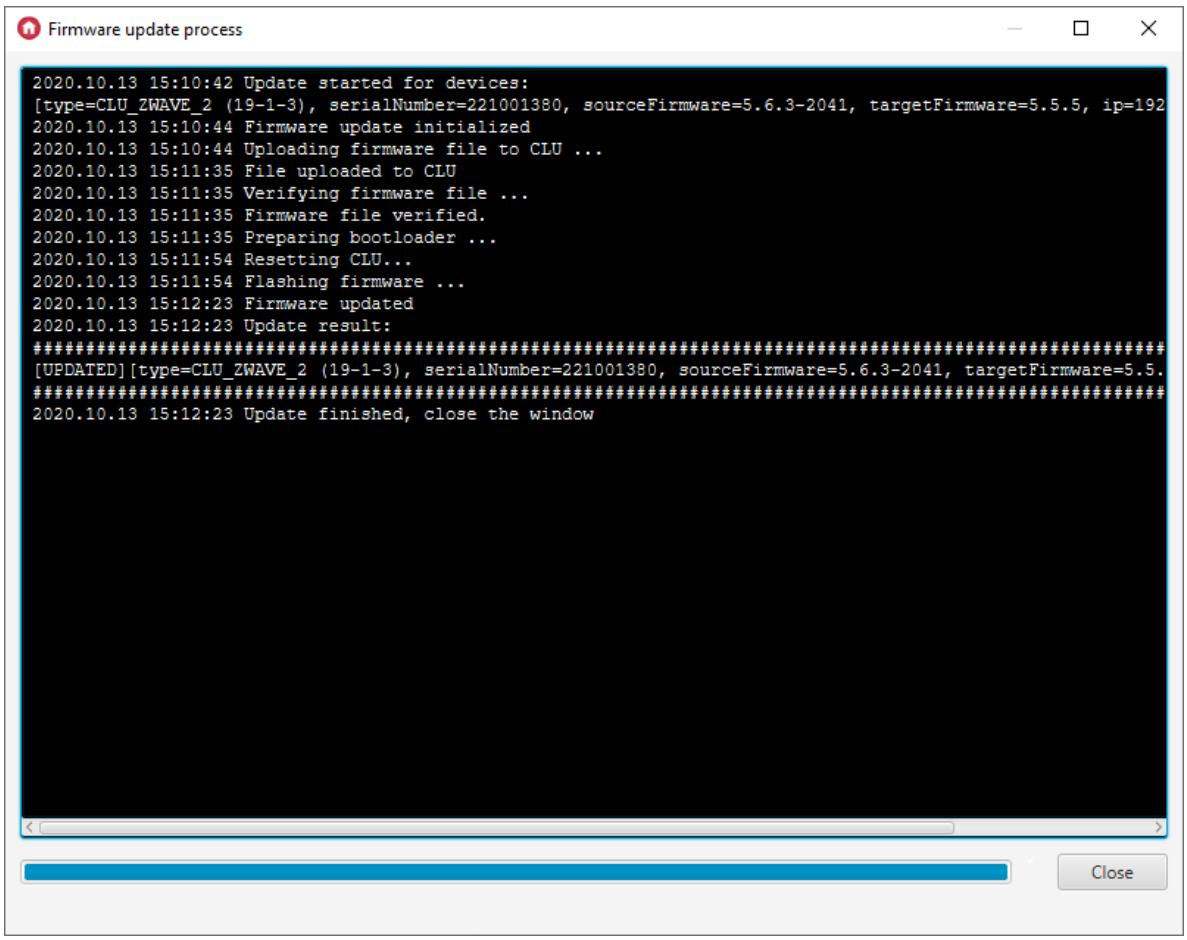
- If the firmware update fails, there will be two attempts to update the module. If they also fail, the message [FAILED] will appear next to the module:



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



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



- To complete the update process, click `Close` 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.

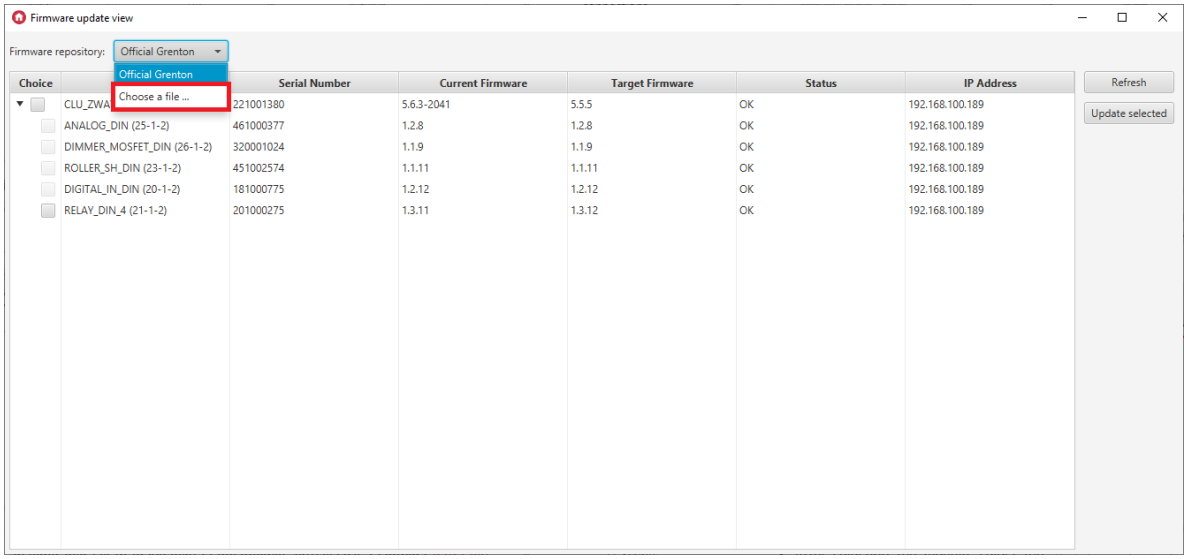
B. Update with a .ZIP file

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.

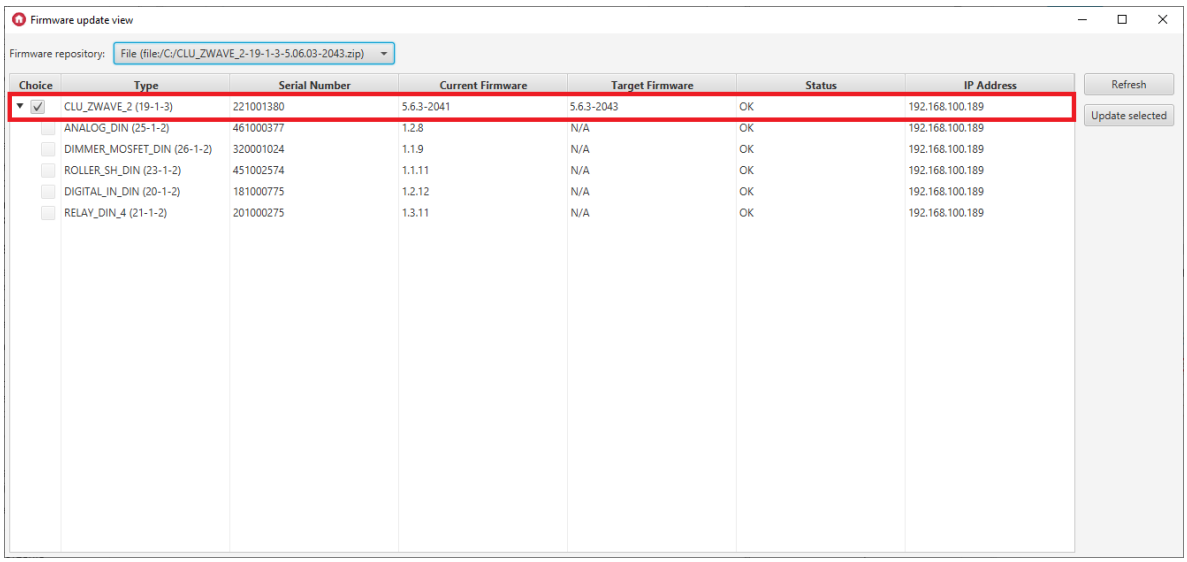
> **Note!**
> Do not rename the .zip file provided by Grenton. The file must have an appropriate name to be properly loaded.

To update from a file you need to:

- Expand the *Firmware repository* and use the `Choose a file` 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

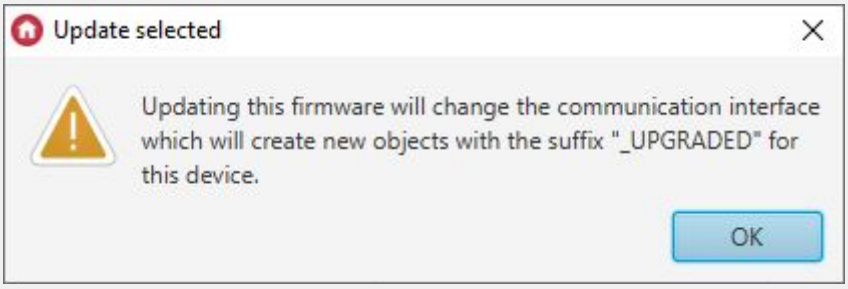
- The computer must be connected to the AC adapter, it cannot be on battery power.
- The network connection between the CLU<->router<->computer must be wired, the WIFI connection cannot be used.
- Do not perform any actions on the Grenton system while updating the firmware.
- You should start with the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

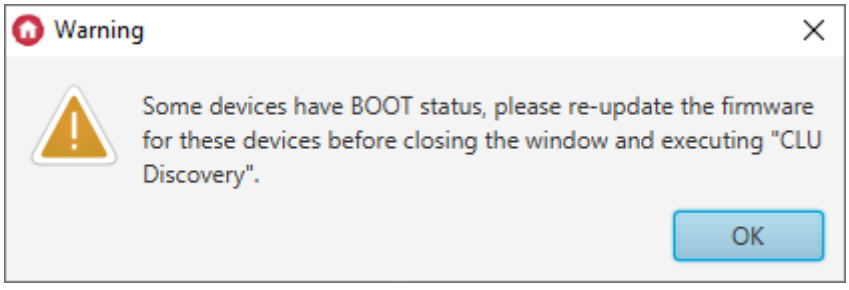
Note!

For updating modules in Object Manager version 1.5.1 or lower 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.



In the case of Object Manager version 1.6.1 or higher, objects are updated automatically during CLU Discovery. It is recommended to use the latest available version of Object Manager.

- 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

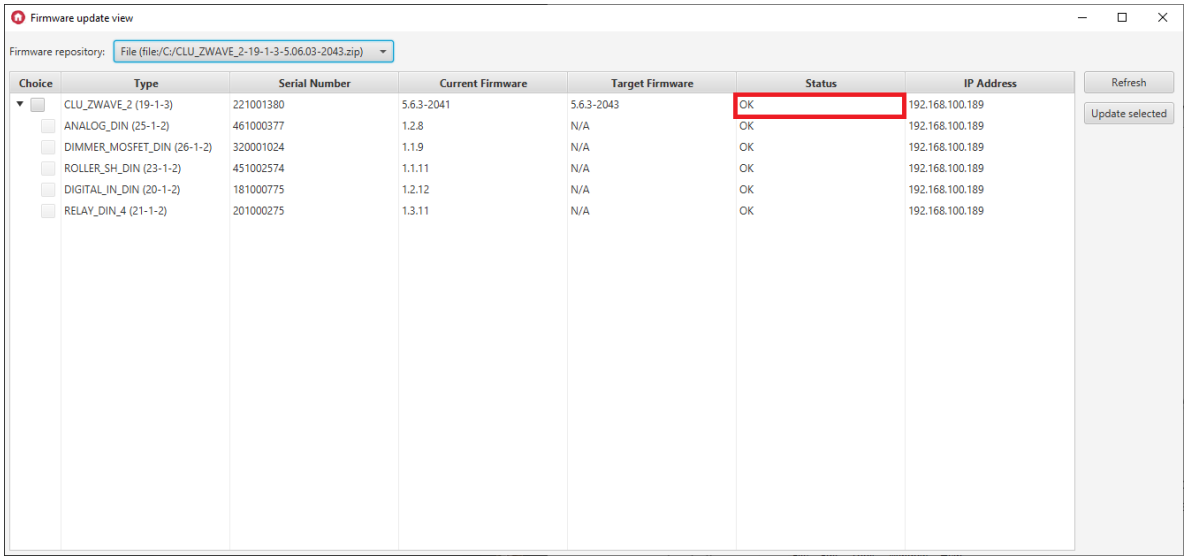
The modules status is displayed in the firmware update view table. If there is a status change while the update window is open, the list must be refreshed using the "Refresh" button.

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

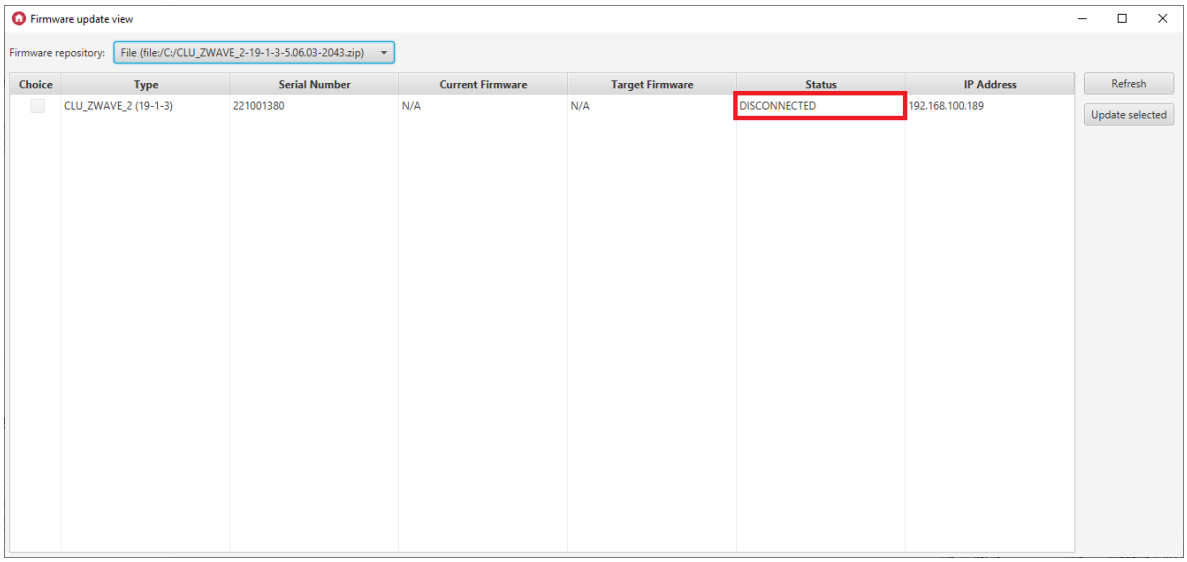
> **Note!**
> The functionality of device statuses is available for CLUZ fw. version 5.06.03-2043 or higher!

A. CLU status

Status: OK - Correct connection with the CLU.

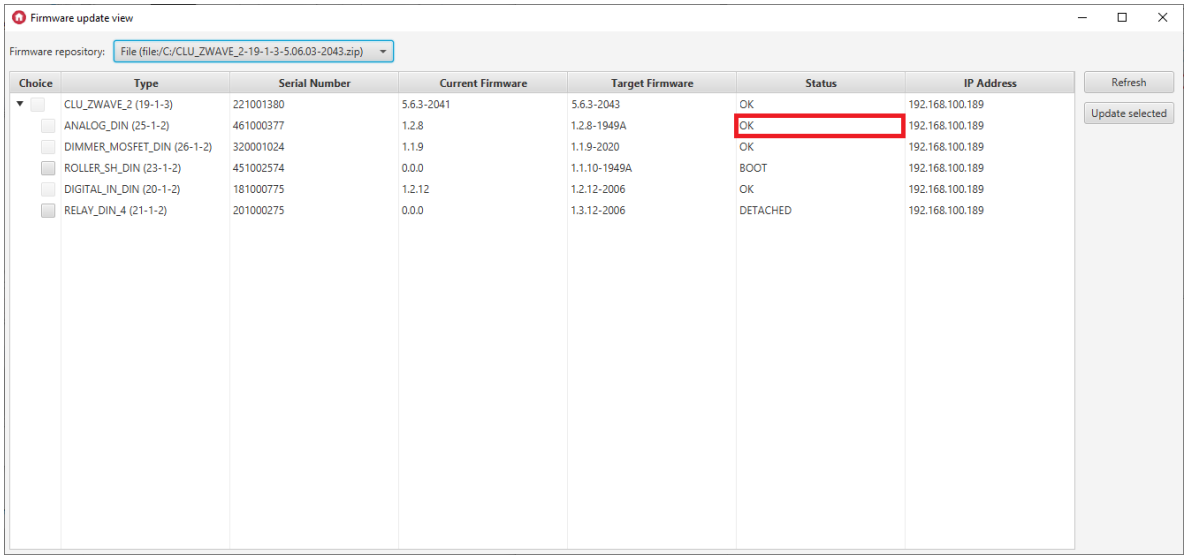


Status: DISCONNECTED - The OM cannot connect to the CLU. This status is when the OM does not get a response from the CLU. In this case, check if the network cable is properly connected to the CLU / router / switch or perform a voltage reset of the CLU.



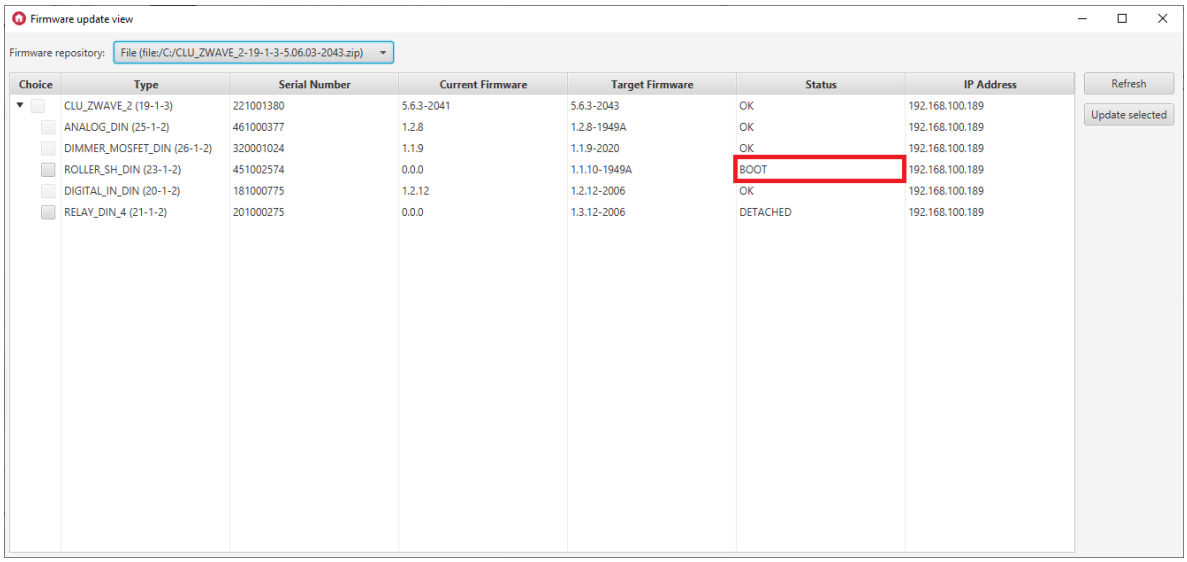
B. TF-Bus modules status

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

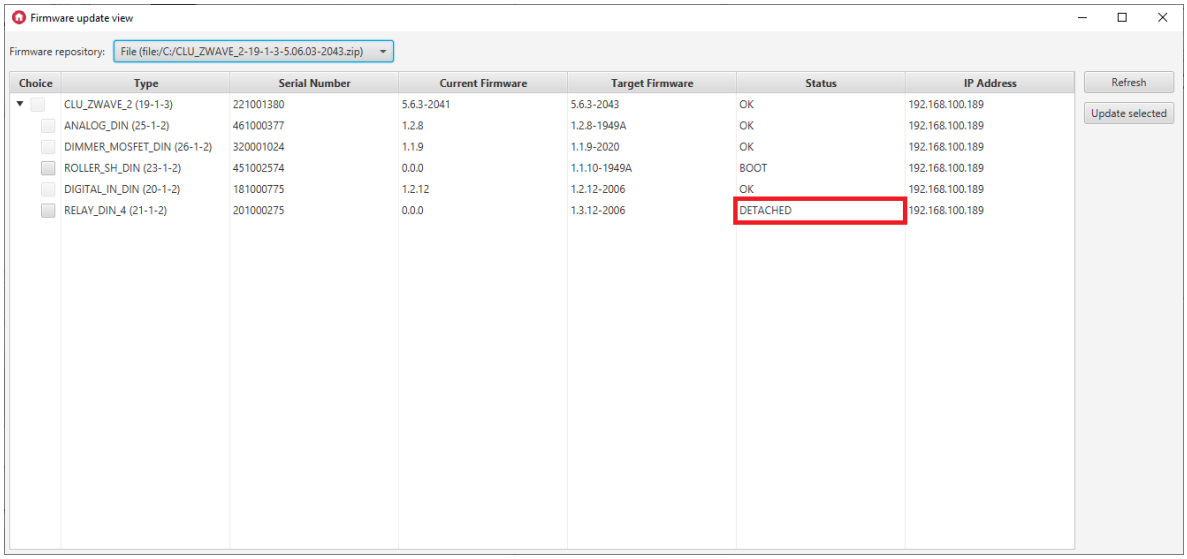


Status: BOOT - The module is currently in the bootloader. This state appears when the module update is interrupted. In this case, the module must be updated again.

> **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..



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.

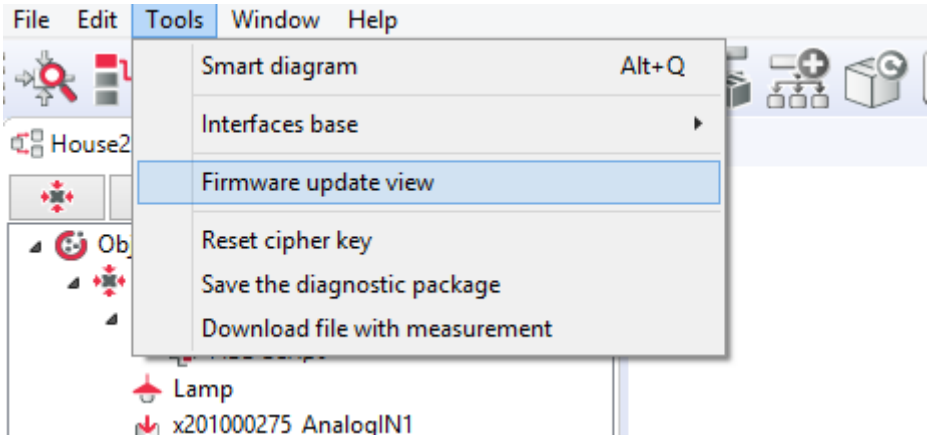


10.5 Forcing the module update

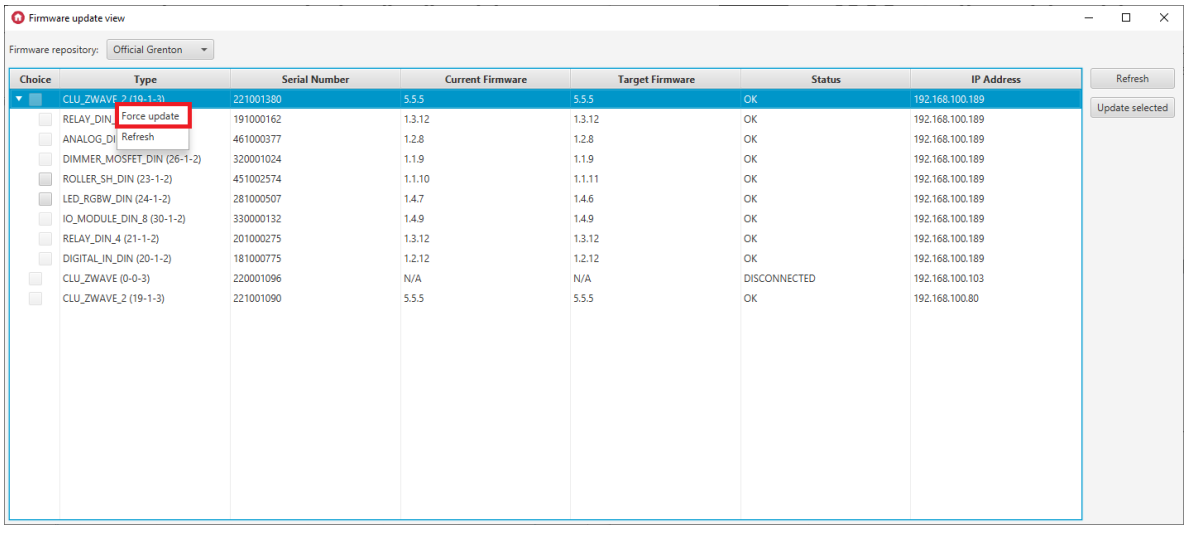
If a properly connected module is not detectable by the Discovery process, it is possible that the firmware for that module has not been properly loaded. In such a situation, it is necessary to force the module update.

In order to force a module update, you should:

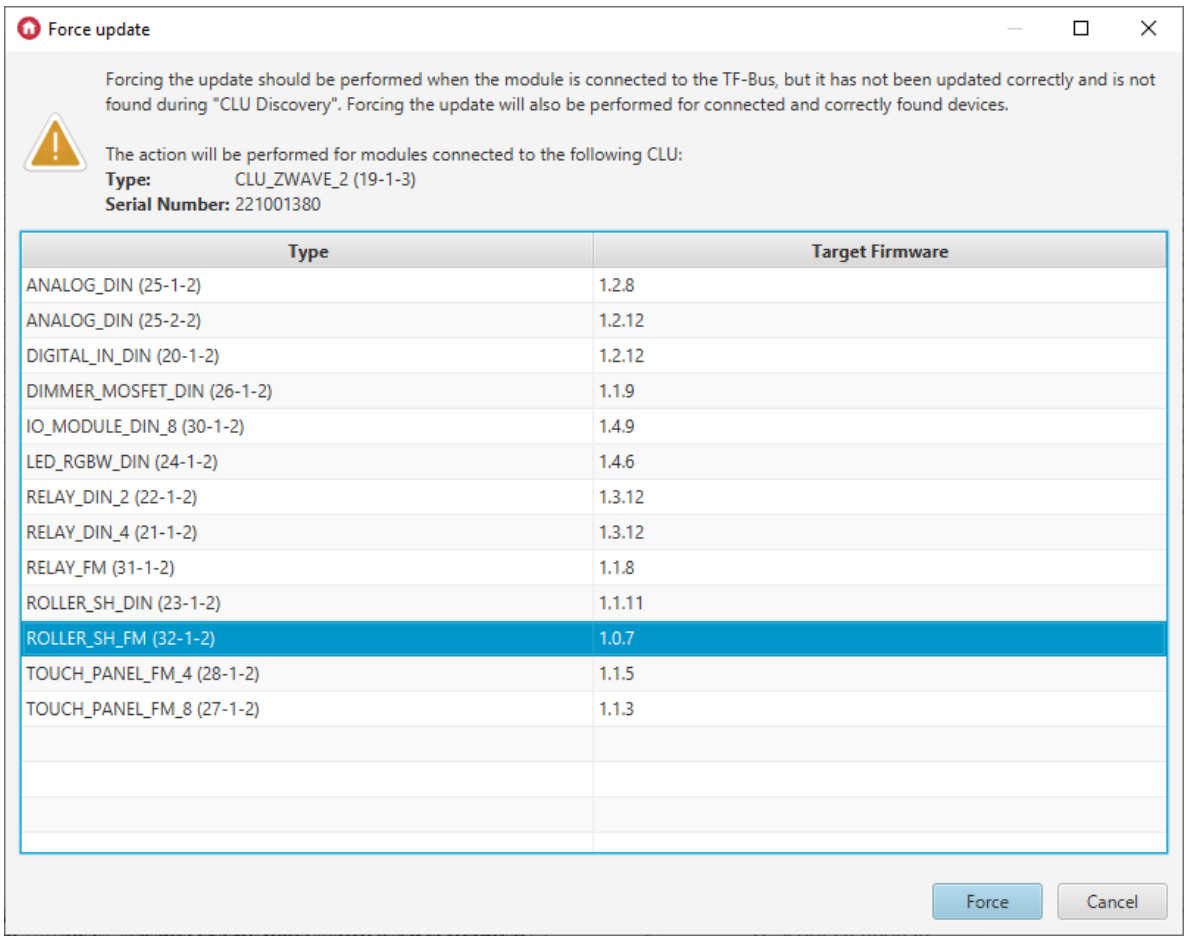
- Select `Tools` from the menu bar.
- Select item *Firmware update view*:



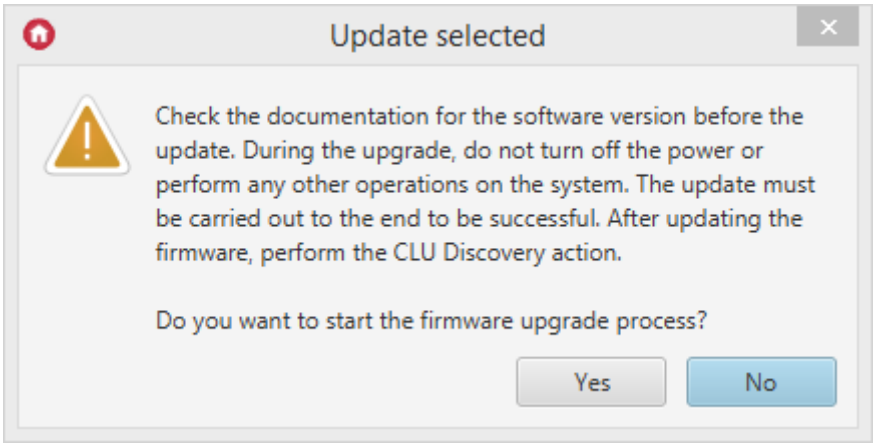
- Right-click on the CLU module and select the option "Force update":



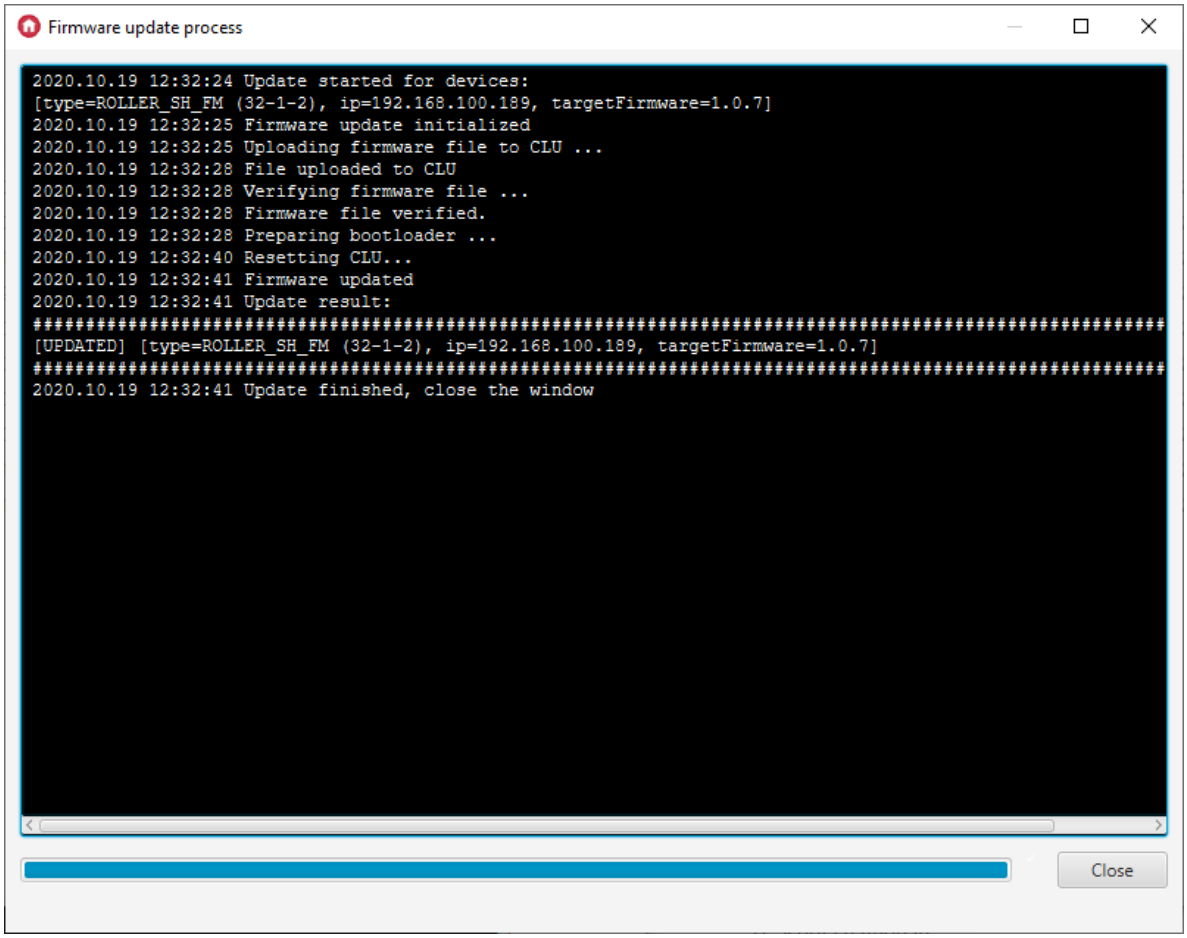
- Select one module for which the forced update is to be performed and press the `Force` button:



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



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



- CLU Discovery must be performed when finished.

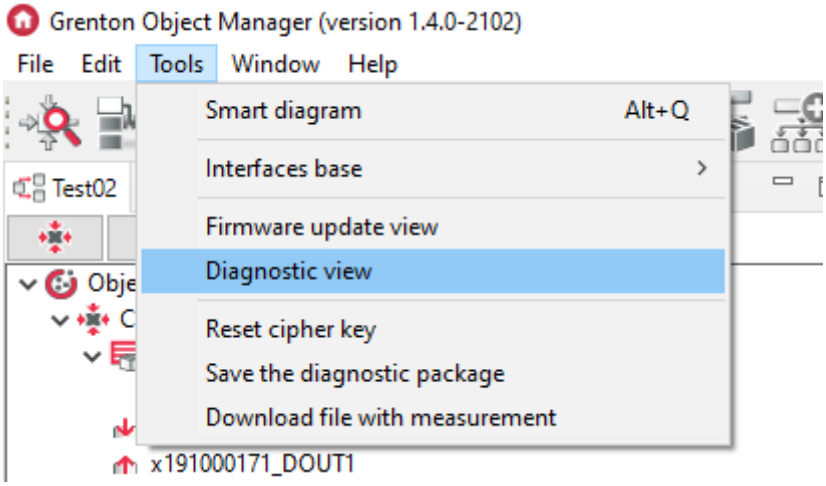
11. Diagnostic view

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

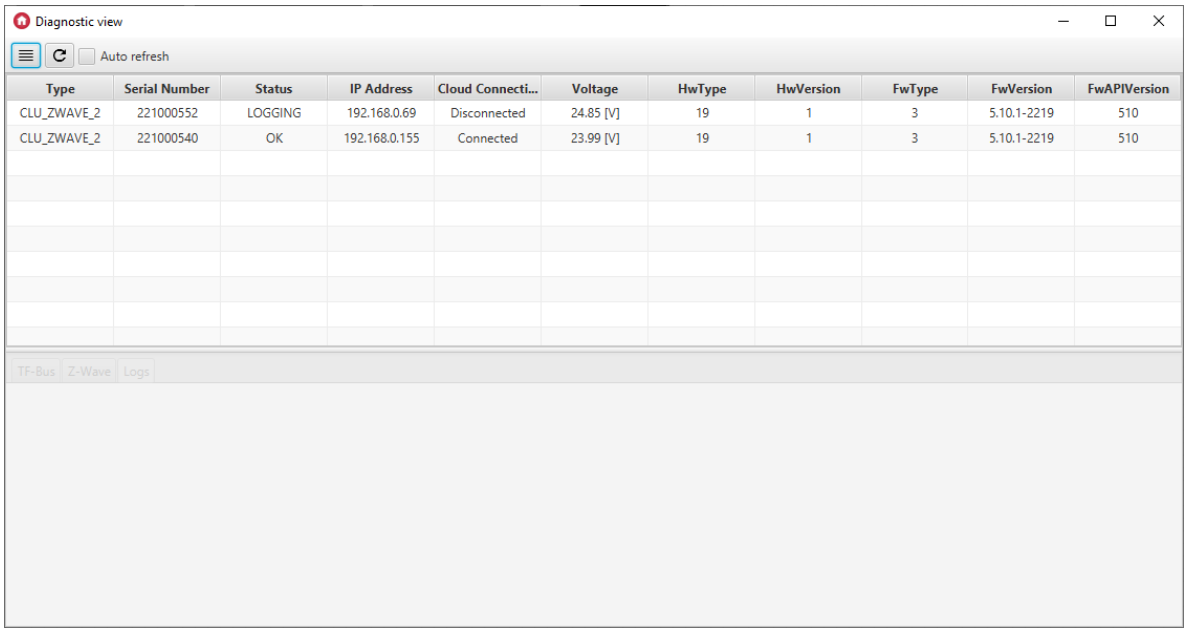
The diagnostic view presents information about all CLU in the project and the modules connected to them.

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:



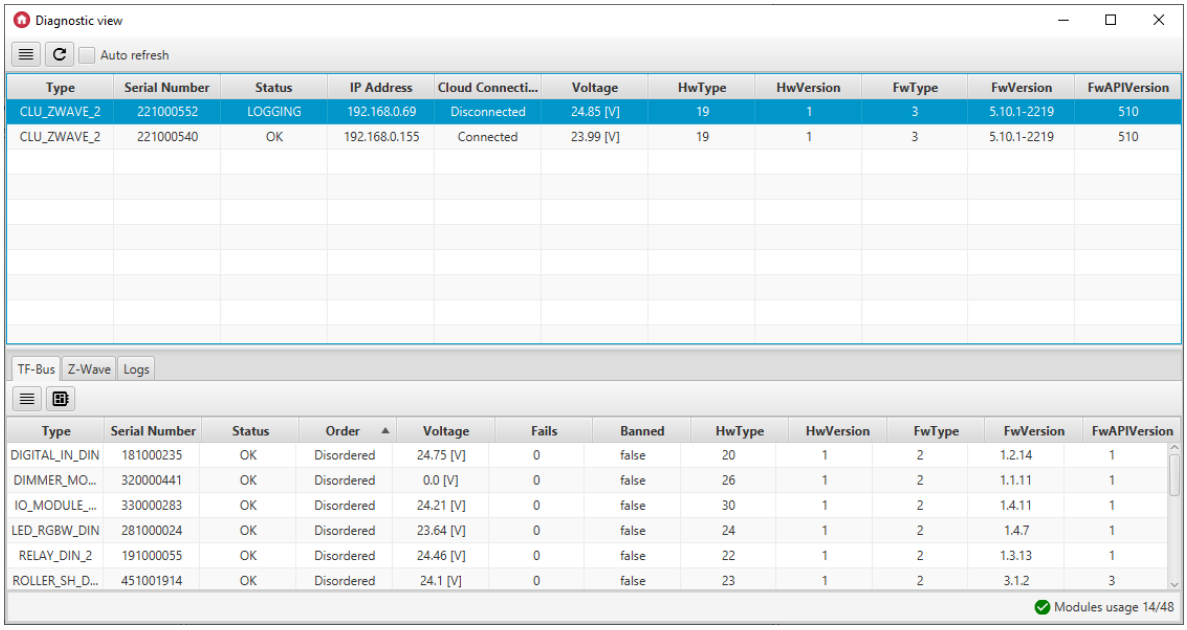
The view presents the following information for the CLU:

- `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,
- `LOGGING` - CLU in logging mode,
- `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:



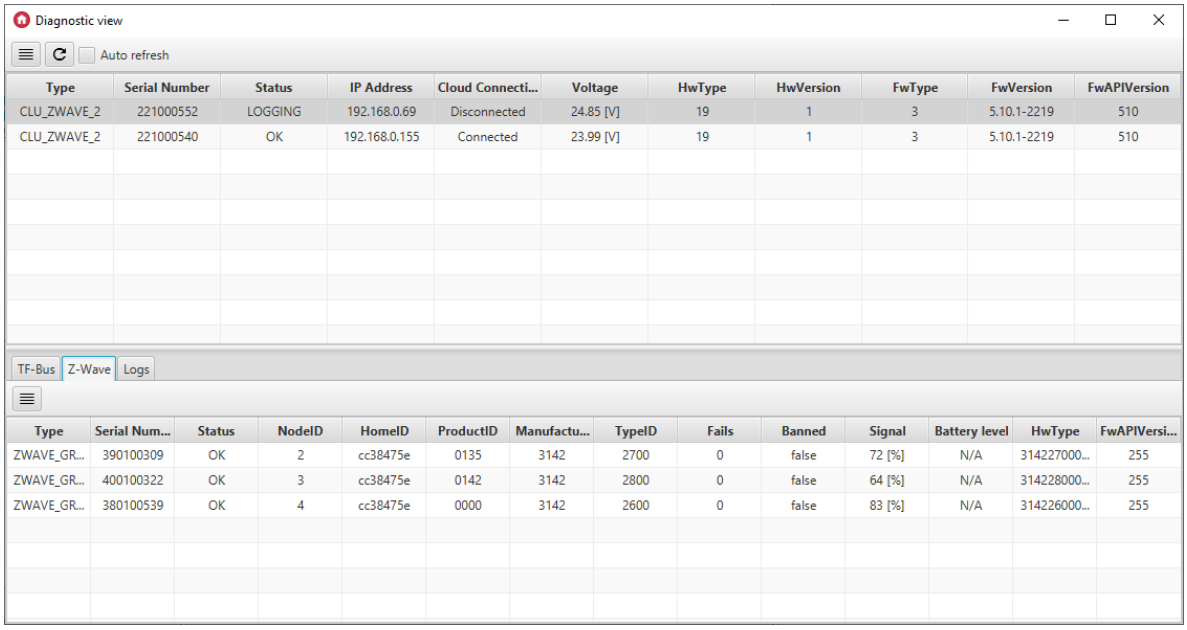
The view shows the following information for TF-Bus modules:

- `Type` - module type name,
- `Serial Number` - serial number of the module,
- `Status` - connection status with TF-Bus,
- `Order` - the sequence of connection to the TF-Bus (set manually),
- `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:



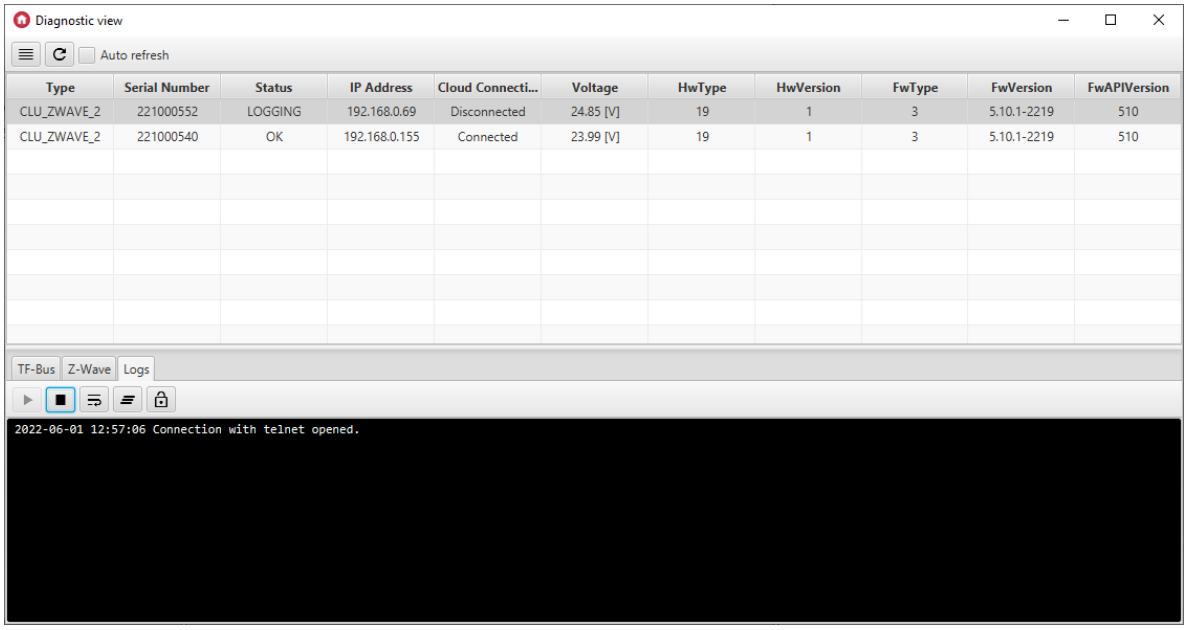
The view presents the following information for Z-Wave modules:

- `Type` - module type name,
- `Serial Number` - serial number of the module,
- `Status` - connection status with CLU,
- `NodeID` - Node ID of the module,
- `HomeID` - Home ID of the module,
- `ProductID` - Product ID of the module,
- `ManufacturerID` - Manufacturer ID of the module,
- `TypeID` - Type ID of the module,
- `Fails` - 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.

For CLU/GATE with the `TelnetLogLevel` embedded feature, a `Logs` tab is available to record and display the logs of a given CLU:



Note!

To fully use the described logging functionality, you must have Object Manager version 1.7.0 or higher and CLU with firmware 5.10.01 or higher / GATE with firmware 1.4.2 or higher.

A textbox is available in the view to present the logged logs. Logging level specified via CLU/GATE embedded feature `TelnetLogLevel` or `SetTelnetLogLevel` method:

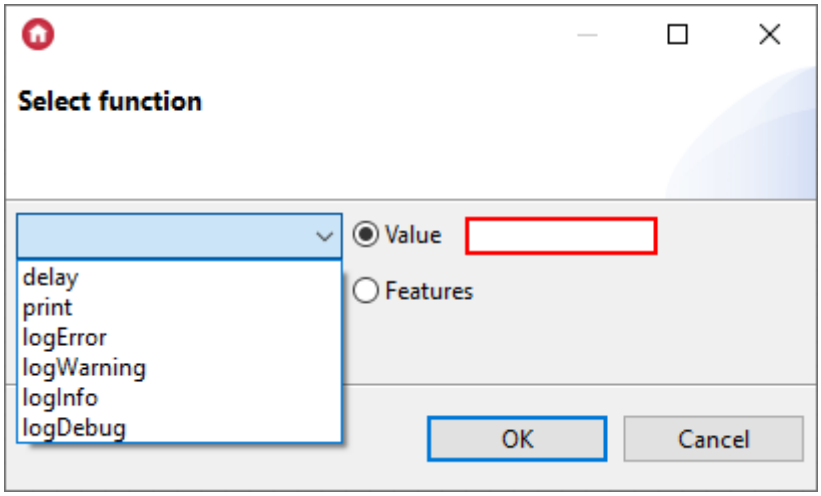
- OFF,
- ERROR,
- WARNING,
- INFO,
- DEBUG.

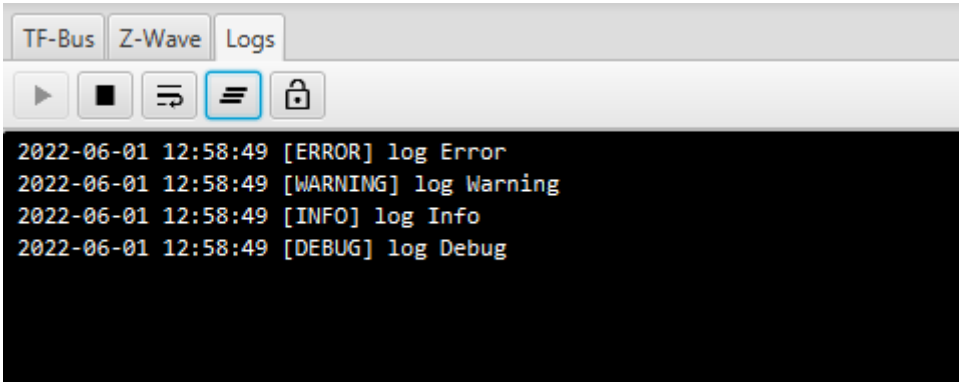
Note!

Only one telnet connection is supported within one CLU/GATE.

The maximum number of characters for one log line is 1000.

The functionality of sending logs is available in scripts, using function blocks. It enables to display logs with the designations: [ERROR], [WARNING], [INFO], [DEBUG]. When using the `print` function the value is displayed in logging at the login [DEBUG] level.





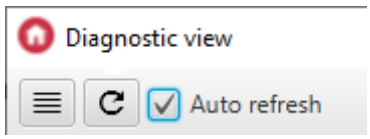
The logging functionality is also available whenever the CLU/GATE is in the Emergency mode. It is then possible to display the reason for entering Emergency mode.

> **Note!**
>
> It is not recommended to expose TELNET logs by port forwarding on the router, due to system security considerations.

11.1 Configuration of the diagnostic view

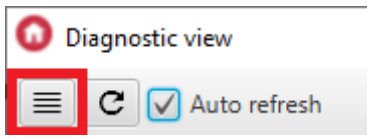
A. Refreshing the view

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

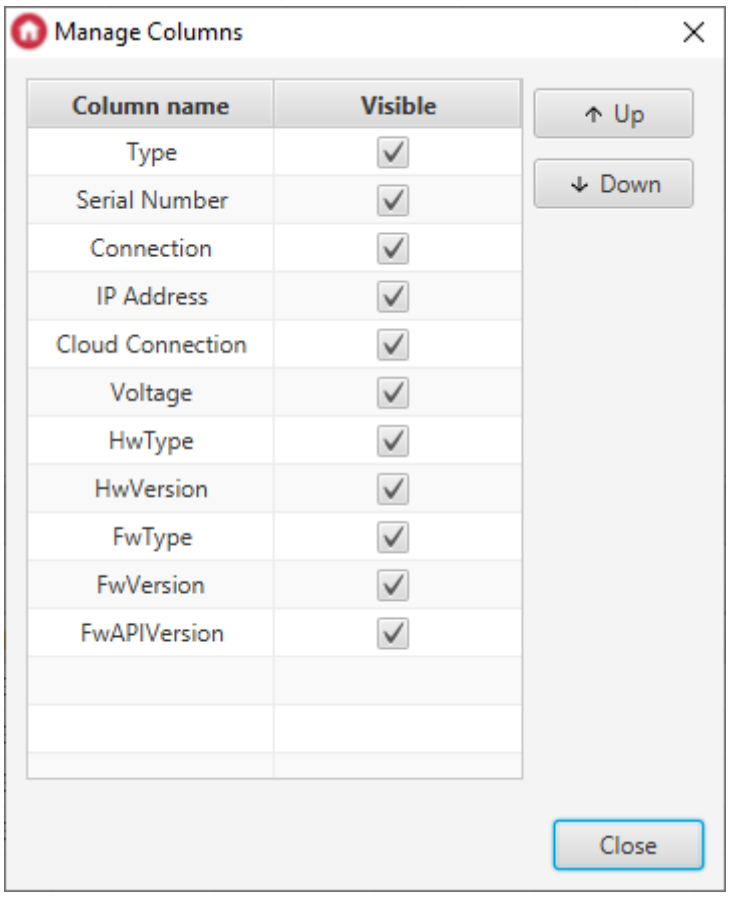


B. Table configuration

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.



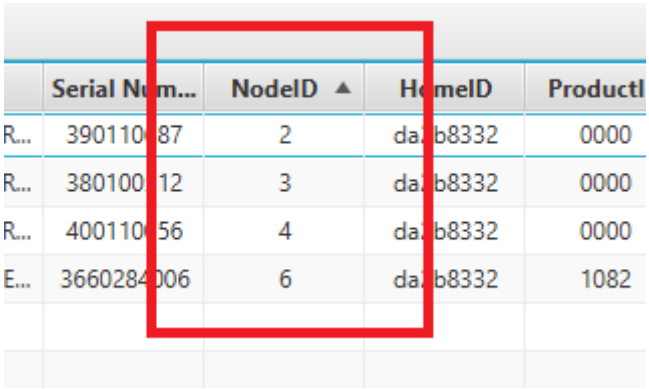
The configuration window will appear:



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

C. Sorting rows

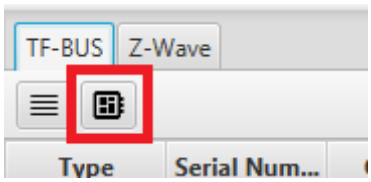
By clicking on a column name, you can sort rows in ascending or descending order or return to the default display.



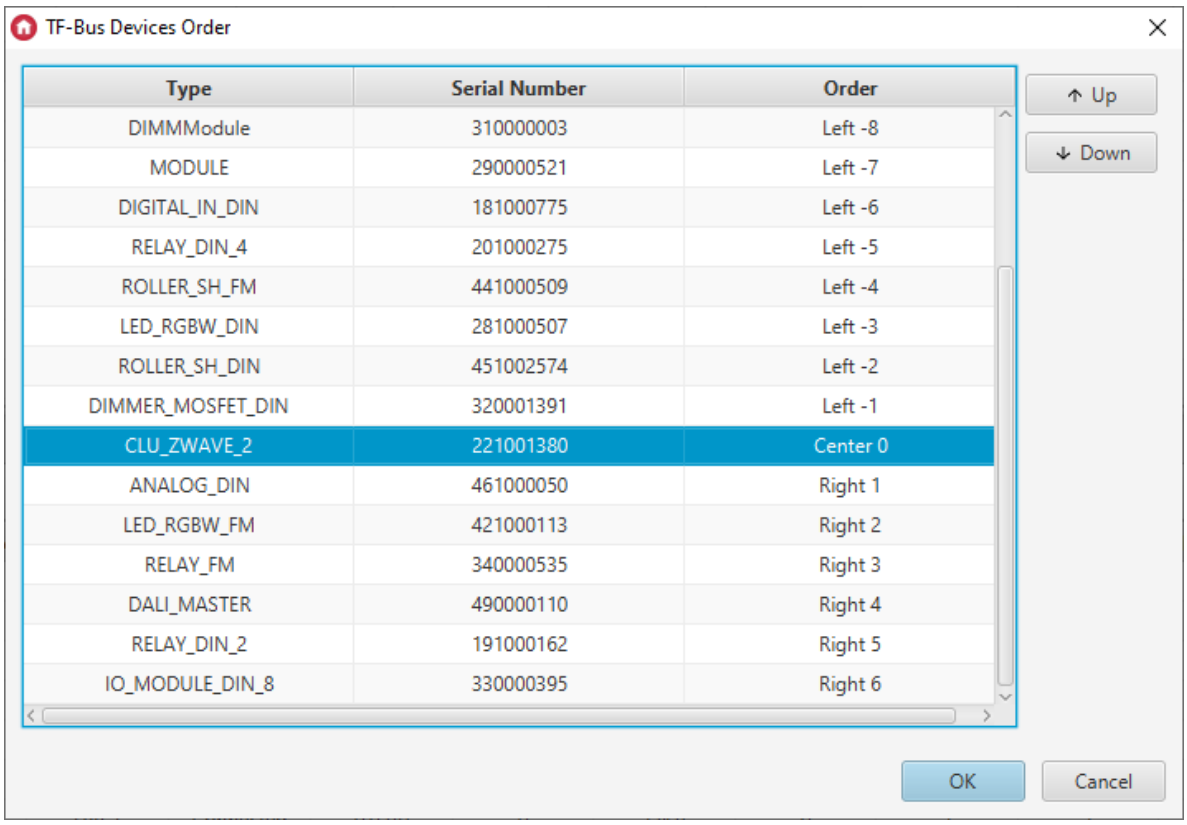
D. Setting the sequence of connection to the TF-Bus

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:



E. Configuration of the login tab

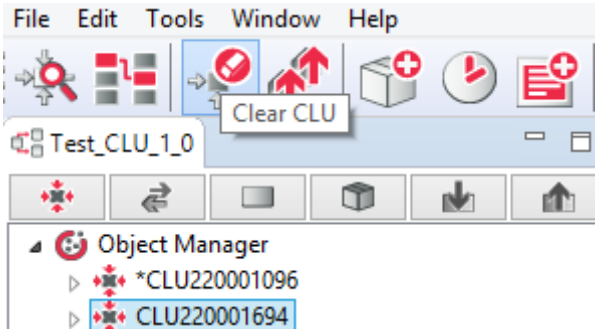
Available buttons in the tab:

- Start log,
- Stop log,
- Word wrap,
- Clear log,
- Scroll lock.

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

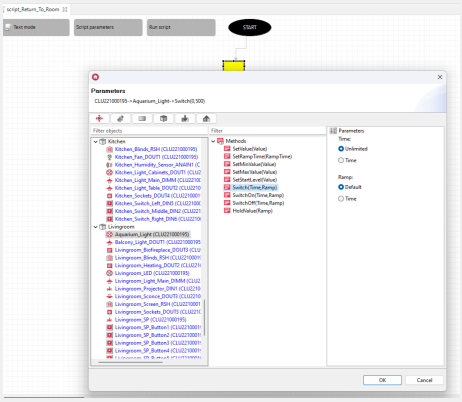
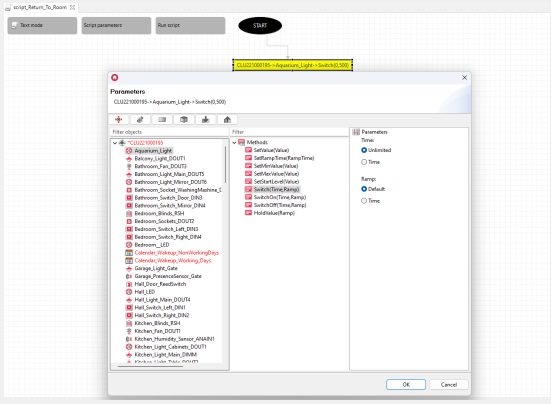
1. Containers

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*.

Note!
Selecting a given object from a container causes the reference to be assigned to a given CLU, not to the container. After reopening the window, the object from the CLU branch is selected.

Selected object from container branch	After reopening - reference from CLU
	

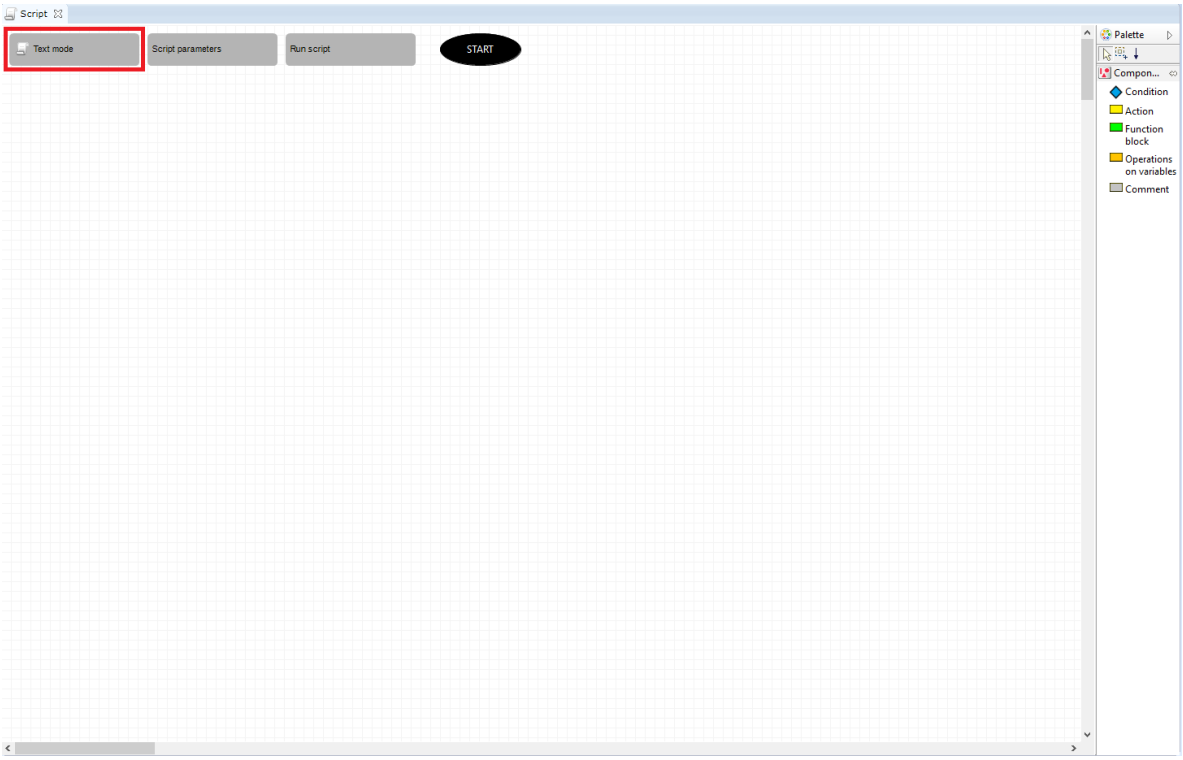
2. Scripts

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:

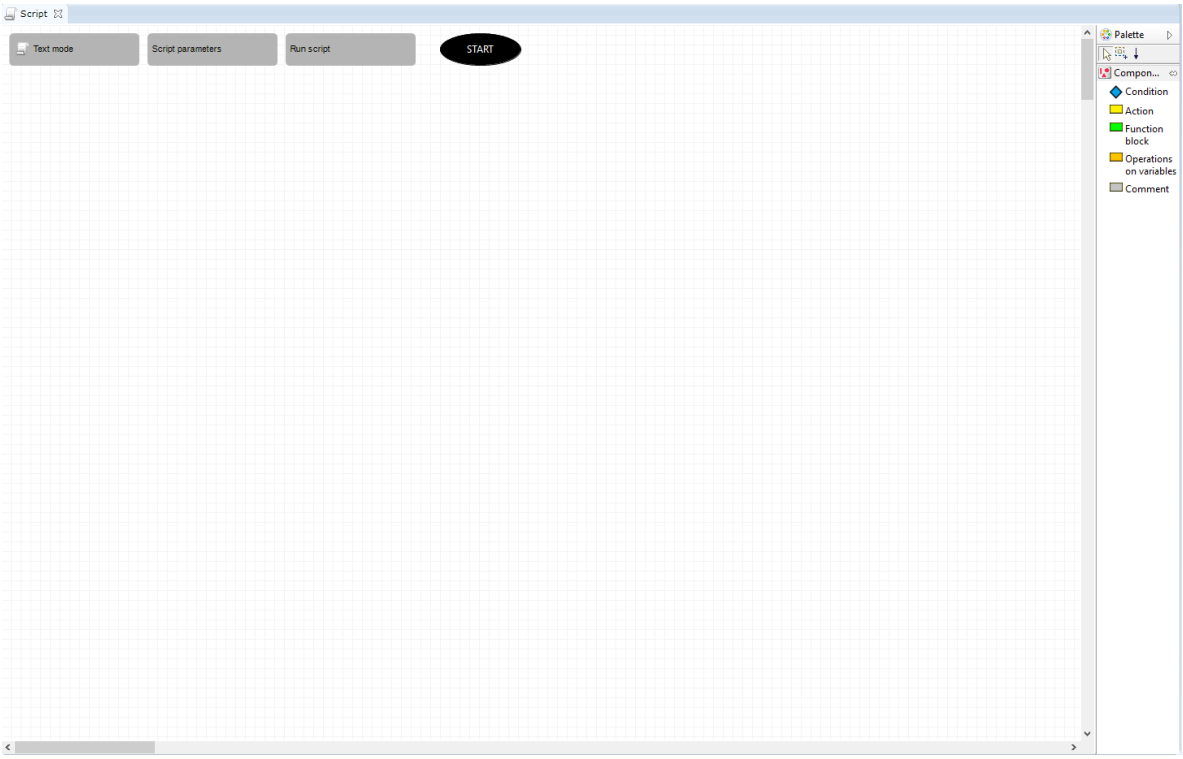
After naming the script (no Polish letters allowed), script builder that enables script creation will open in a tab. Script builder can work in two modes: graphic and text. After new script creation, script builder automatically launches in the graphic mode. You can switch to text mode by clicking `Text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



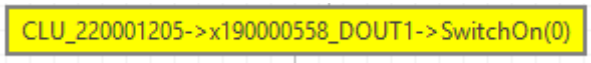
There is components list on the right of the worksheets. Drag commands from the list to the worksheet to add them. After dropping a command on the worksheet, a dialogue box open which allows to determine command parameters and conditional instructions. After adding a new component to the worksheet, a 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.

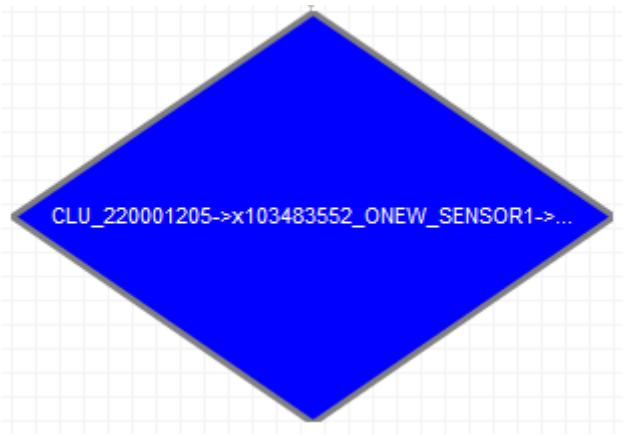
Script Builder uses the following components:

A. Action



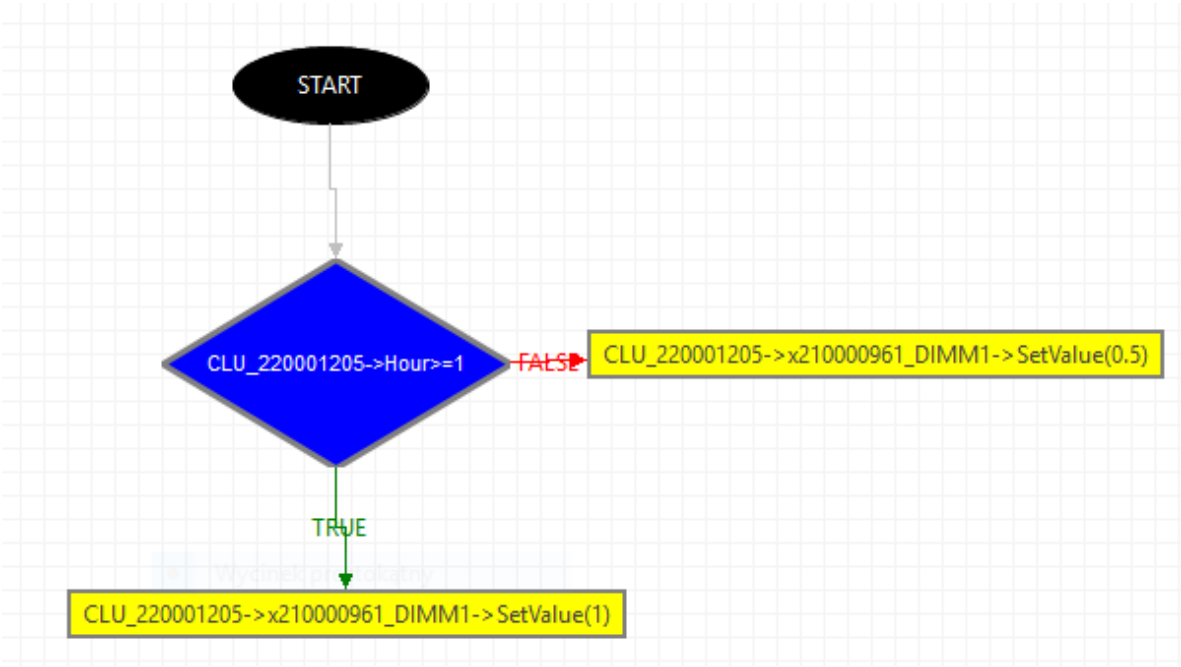
The block in which the order to be executed is entered. The command might be not only method invocation, but also value change or script invocation. After dragging action icon into the worksheet, a window with objects list and their methods opens. Scripts are available on the list as CLU methods after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

B. Condition

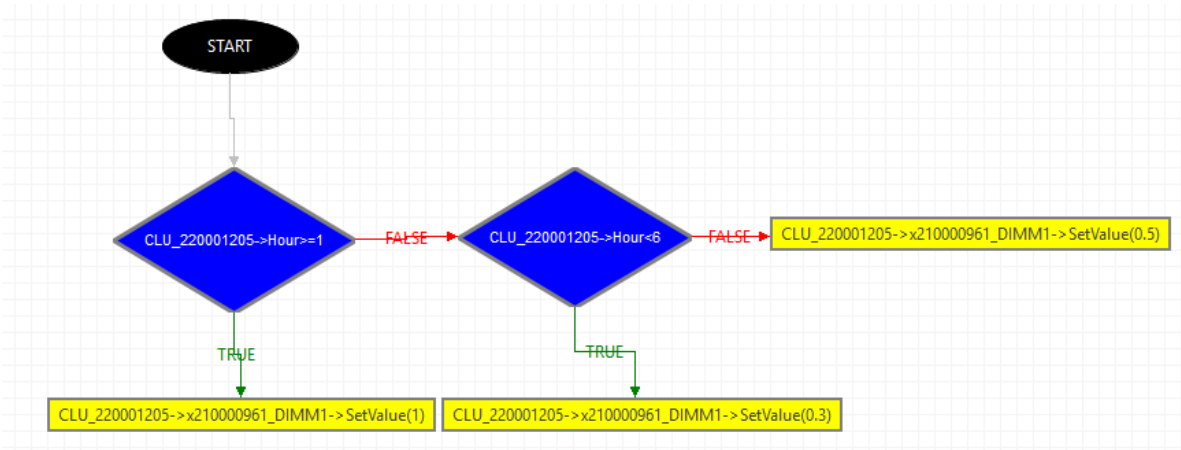


Logic block realizing `IF then ELSE` function. Using this block makes it possible to make the action dependent on the conditions, eg if it is dark, turn on the light, if not, turn it off. After dragging block to the worksheet, enter condition which needs to be fulfilled in its parameters. After adding "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:



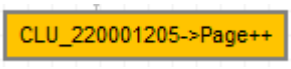
Conditions can compare any object feature or script parameter with a number, a text, other feature, or other script parameter.

C. Function block

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:

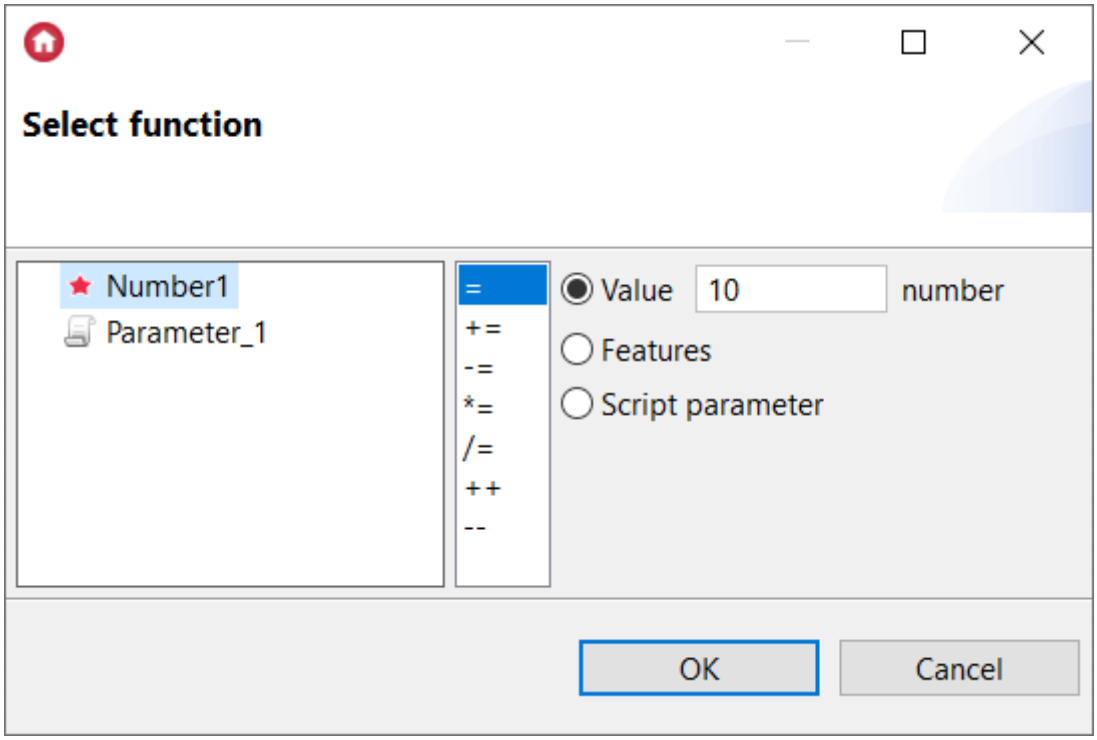
- **DELAY**
  Allows to set time delay between consecutive commands during script realization
- **PRINT**
  The command that causes the display of the declared text in debugging.

D. Operation on variables



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):

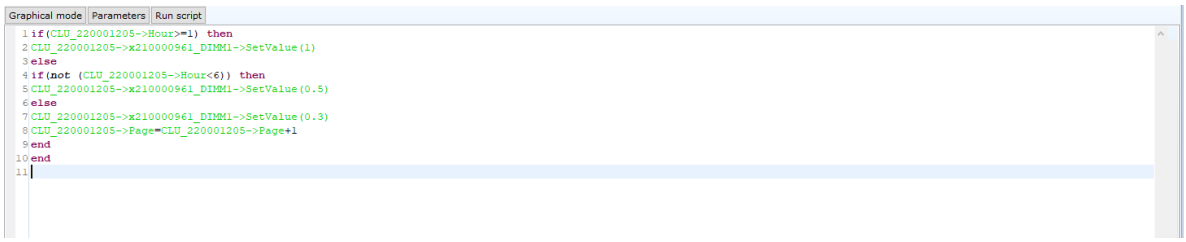
Actions	Description
=	Setting the number 10 as the value of the user feature / script parameter.
+=	Adding the number 10 to the value of the user feature / script parameter.
-=	Subtracting the number 10 from value of the user feature / script parameter.
*=	Multiplying the number 10 by the value of the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

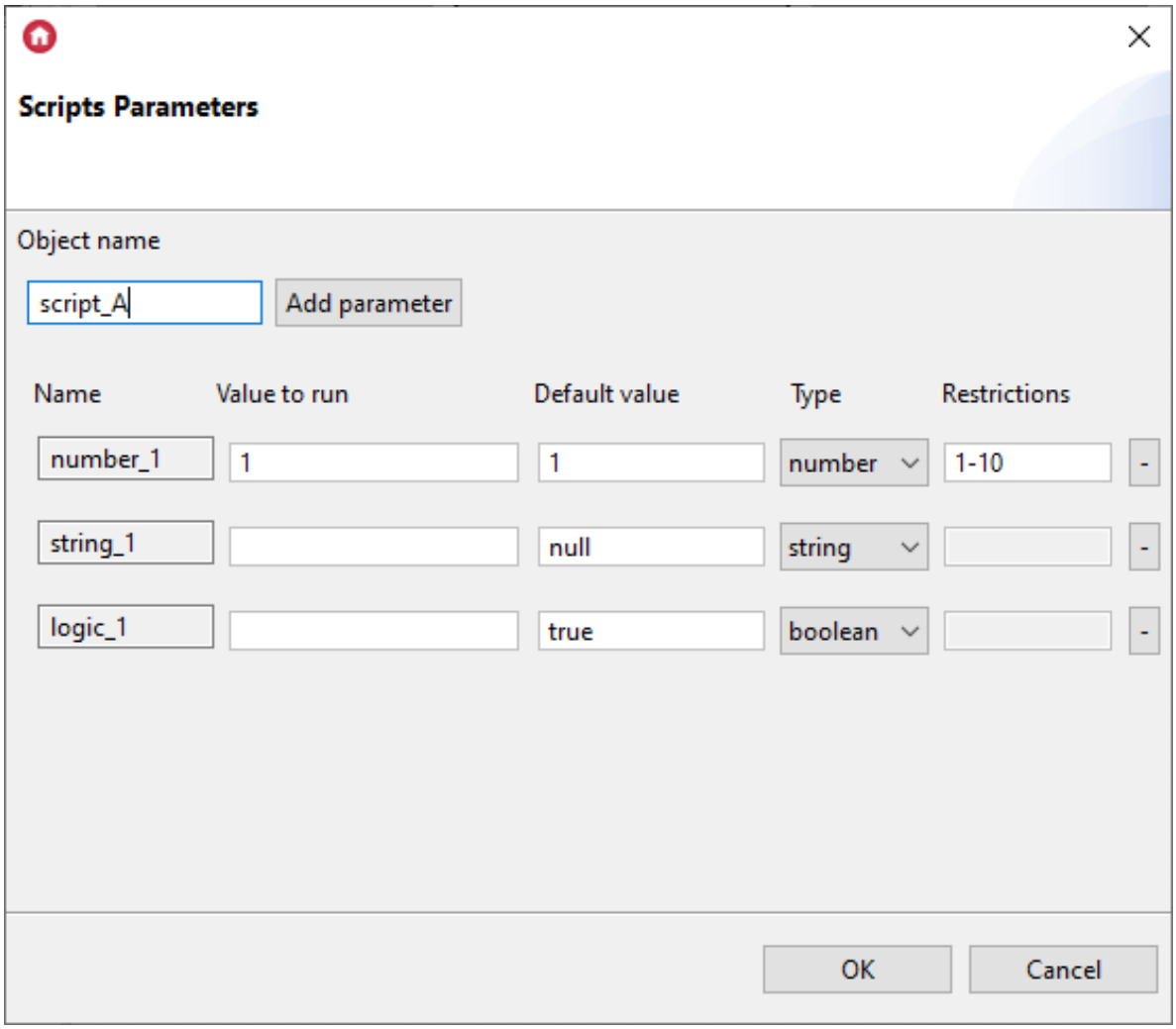
Logical interface addresses are treated as functions and can be invoked and used as parameters in conditional instructions, loops, etc.

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



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.



> **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.

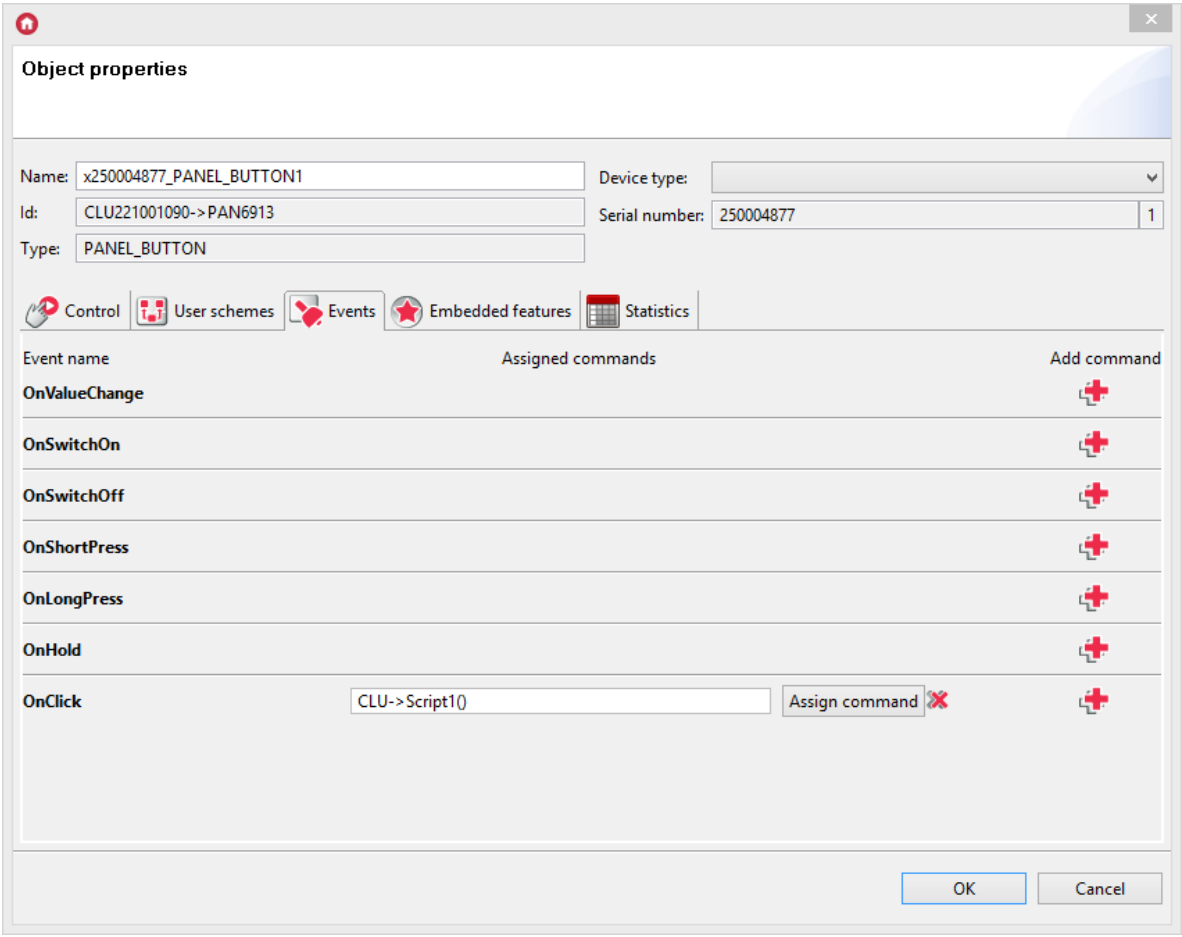
> **Note!**
> Local variables in one script may only be used by the CLU on which the script was created. In order for the variable to be used by other CLUs, the **user feature** must be defined on the CLU, e.g .:
>
> (CLU_A - CLU on which the script was created)

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

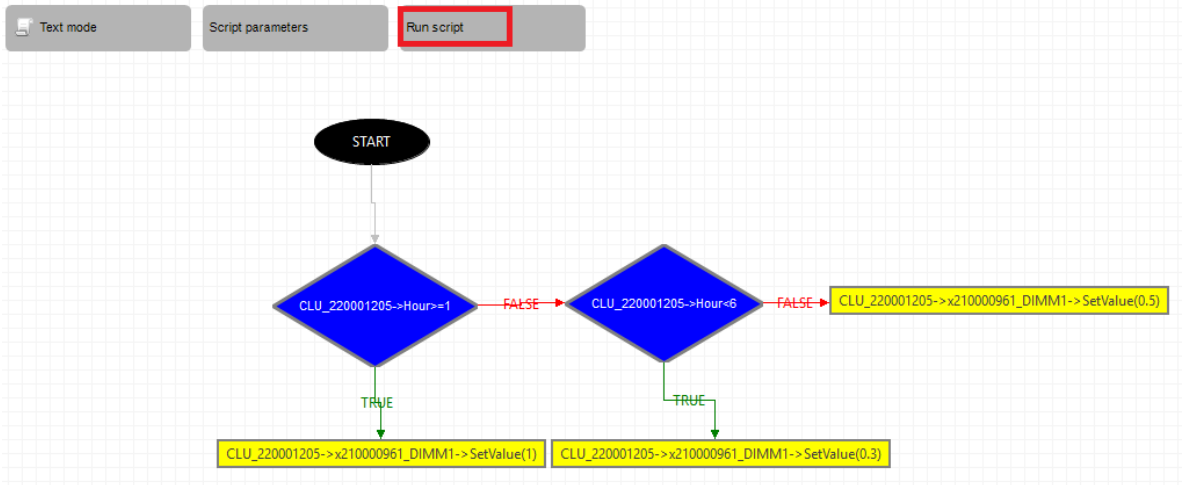
2.4. Scripts invocation

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.

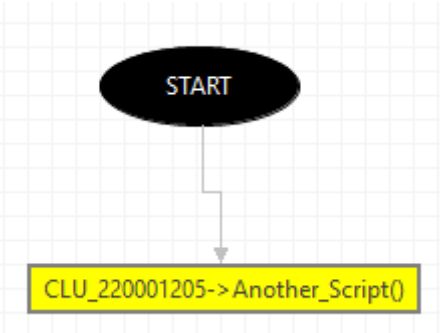
- **Invocation by an event**
  Picture below shows adding script to a switch. The script will be started after pressing the switch.



- **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)
```

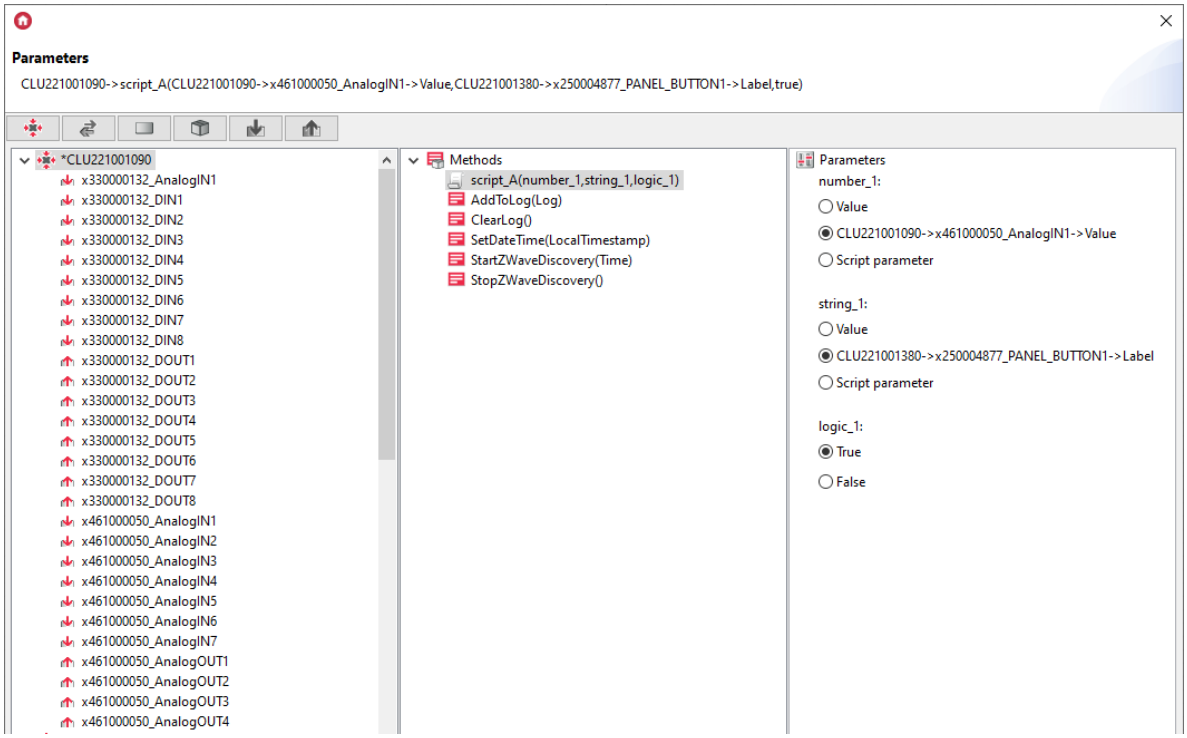
where: parameters were specified for a variable of type number, string and boolean.

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.

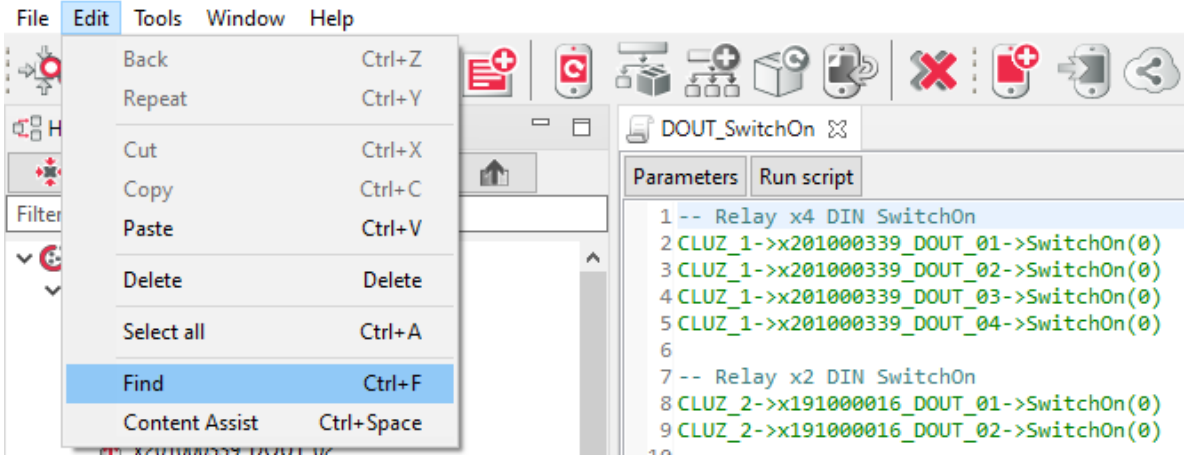
The input parameters can be easily defined using the Parameters window:



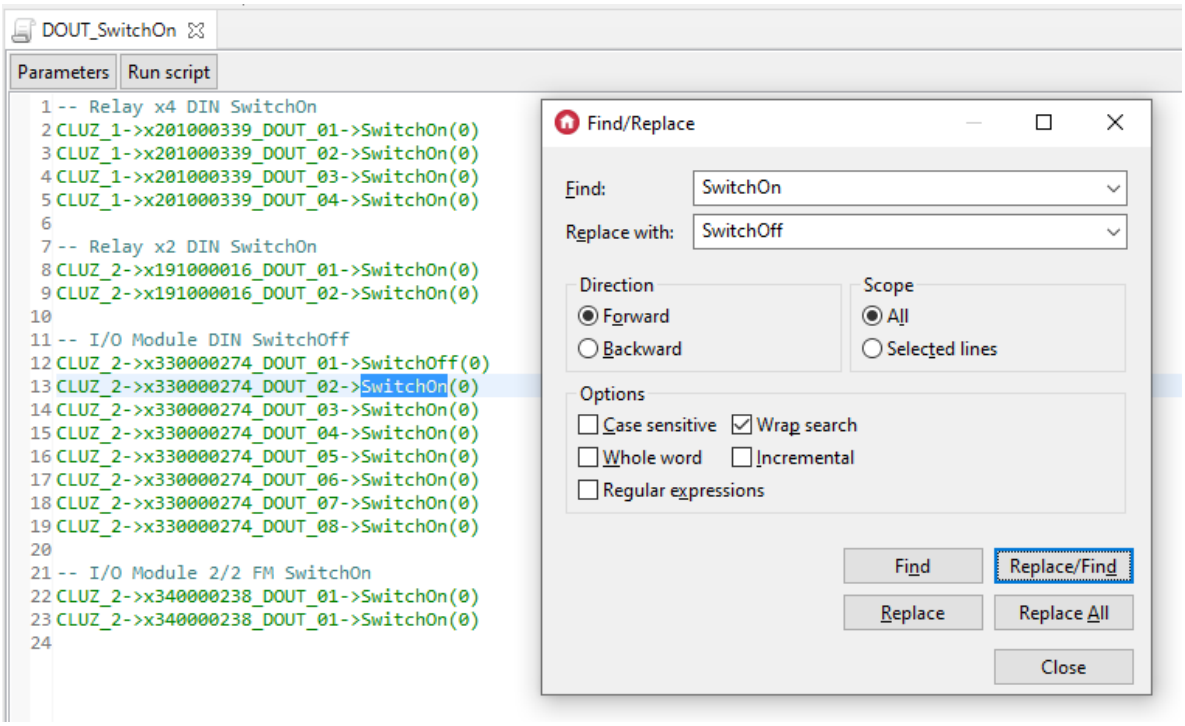
2.4. Find / Replace function

> **Note!**
> The Find / Replace function is available in Object Manager version 1.6.0 or higher.

In the case of scripts created in the editor, it is possible to perform the operation of finding and replacing data strings in the script. The Find / Replace function is available from the `Edit -> Find / Replace` menu or it can be called in the script using the `Ctrl - F` keyboard shortcut.



When searching for a given string, it should be placed in the `Find` text box, while when searching and replacing, the new phrase should be enter also in the `Replace with` text box. The window contains options that define the specificity of searching for a given phrase, e.g. specifying the direction of the search, the scope for which the search is being performed, or the case-sensitivity of the letters.



Actions available:

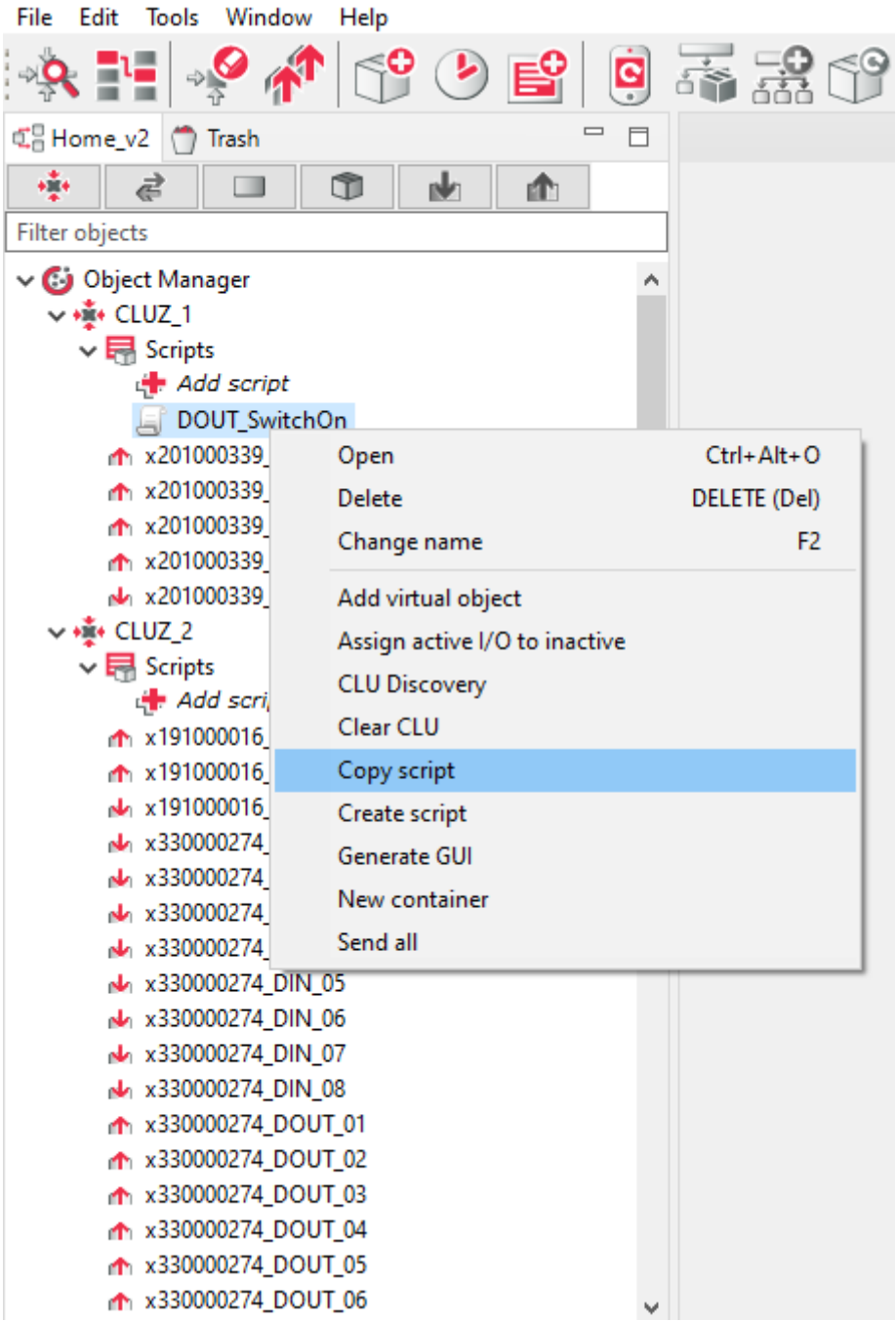
- *Find* - the phrase entered in the *Find* field is searched;
- *Replace* - the found phrase is replaced with the text entered in the *Replace with;*
- *Replace/Find* - the indicated phrase is replaced and then the next text (entered in the *Find* field) is searched;
- *Replace All* - the replacement occurs throughout the script for the given phrase.

> **Note!**
> The *Find / Replace* feature is not available for scripts in graphic mode.

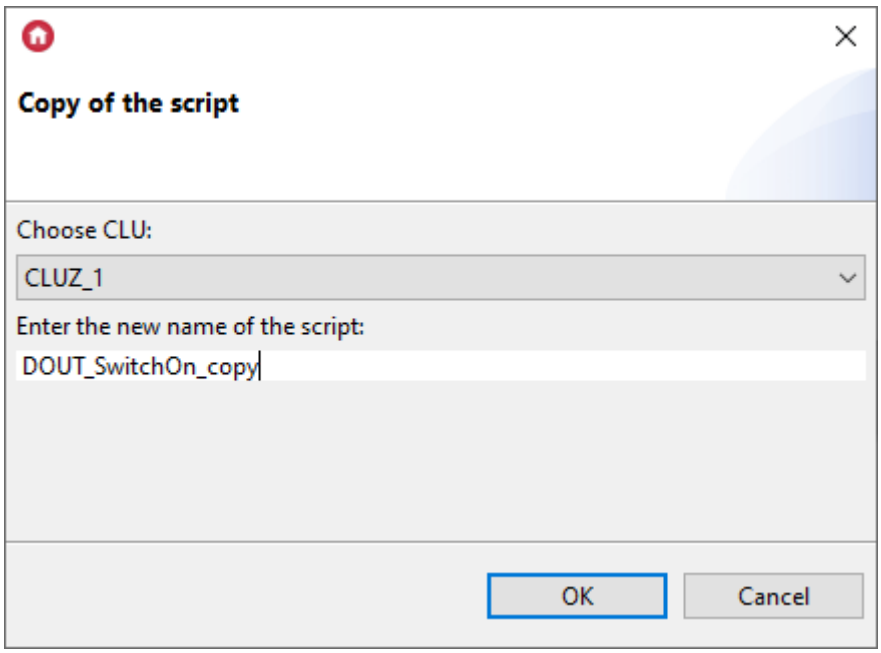
2.5. Copying scripts

> **Note!**
> The option to copy text scripts is available in Object Manager version 1.6.0 or higher. The option to copy graphic scripts is available from version 1.9.0.

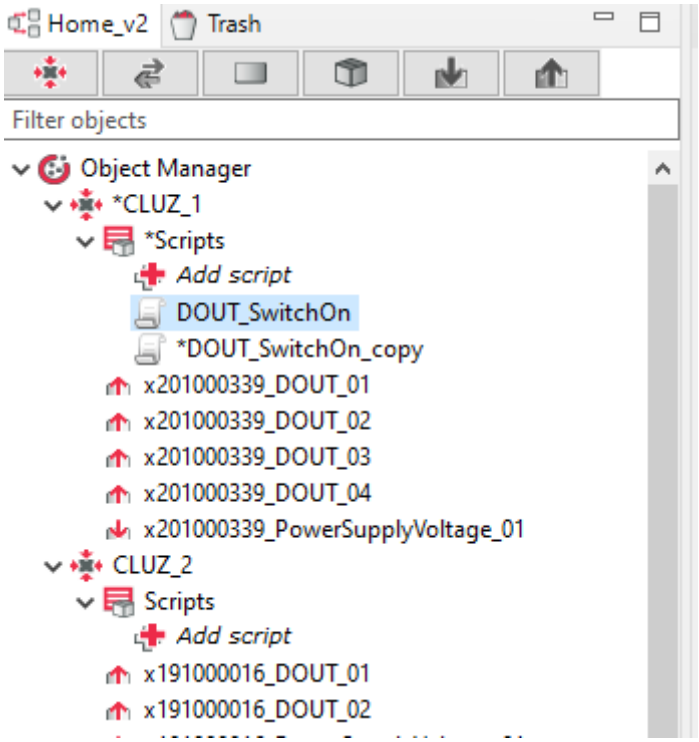
In the case of scripts created in editor, it is possible to make a copy of the script. The script can be copied in the area of a given CLU as well as it is possible to make a copy of the script to another CLU.



After calling the option, the *Copy of the script* window is displayed in which you must select the CLU to which the selected script will be copied and a new script name.



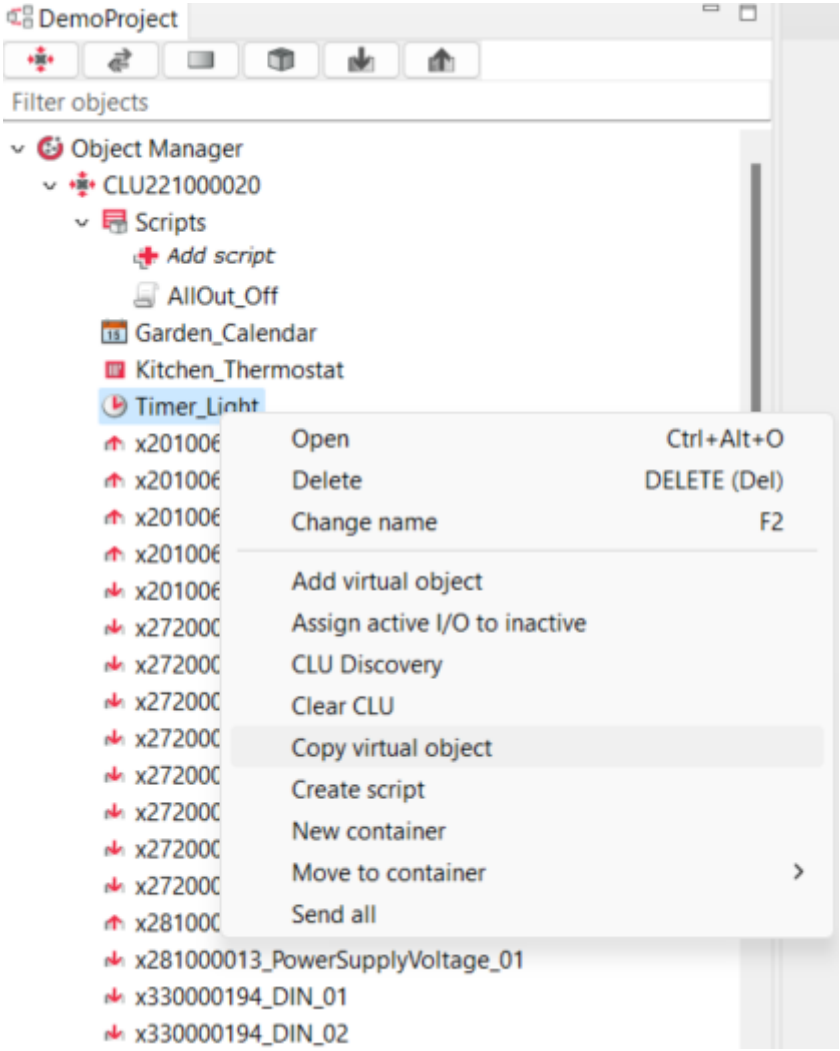
The copied script is added to the list of scripts of a given (selected) CLU.



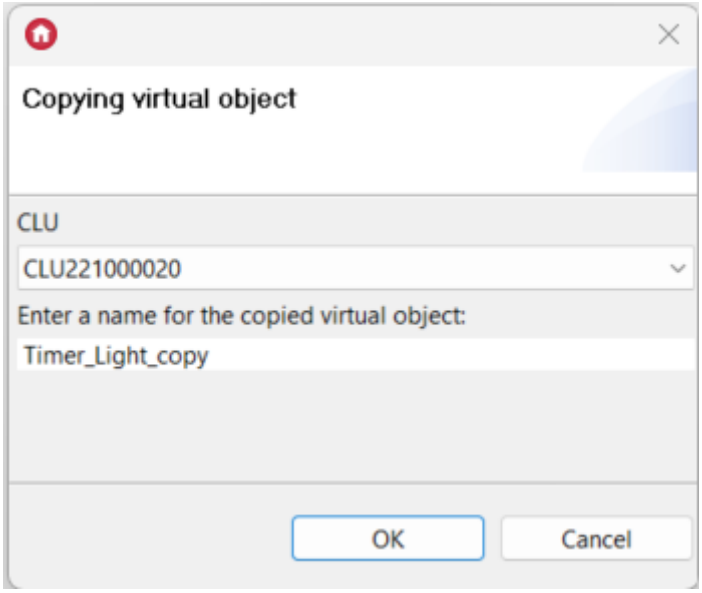
3. Copying virtual objects

> **Note!**
> The option to copy virtual objects is available from version 1.9.0.

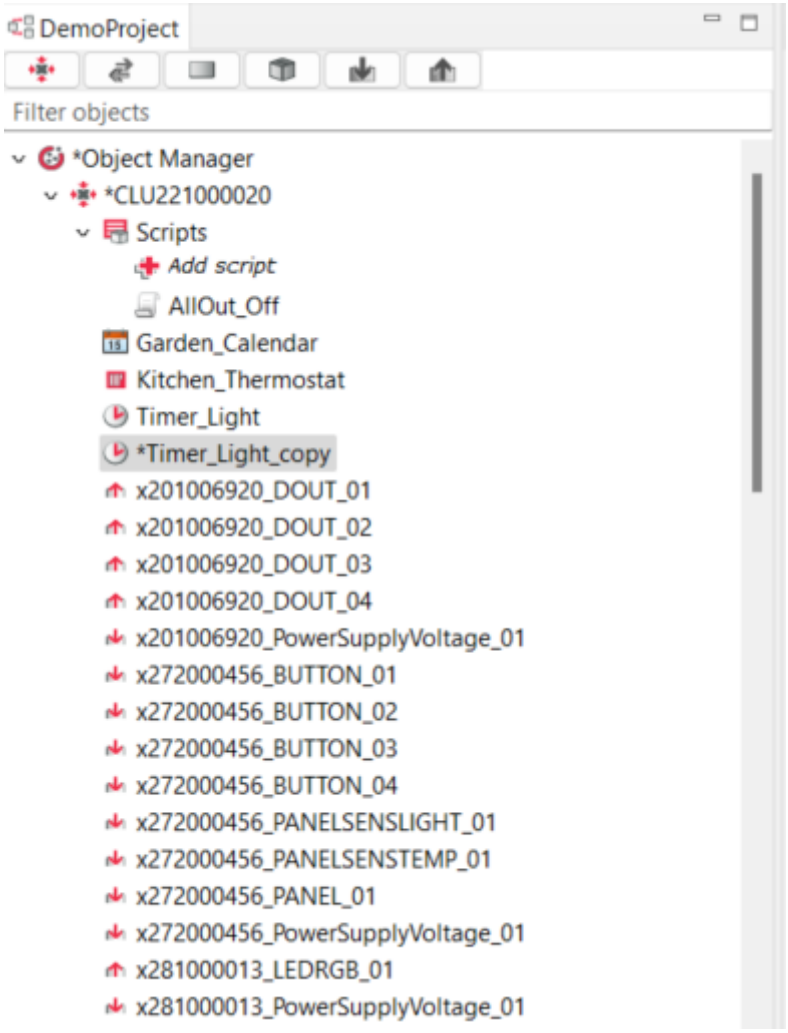
In the case of objects, it is possible to make a copy of a given virtual object. The object can be copied within a given CLU and it is also possible to make a copy to another CLU, if the CLU also supports a given virtual object (in a given version).



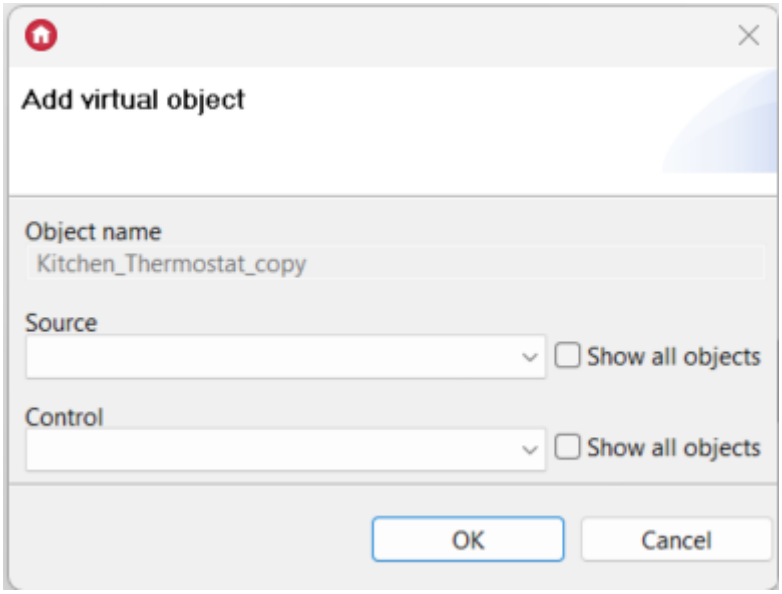
After activating the option, the *Copy virtual object* window appears in which you must select the CLU to which the selected object will be copied and provide a new name for the object.



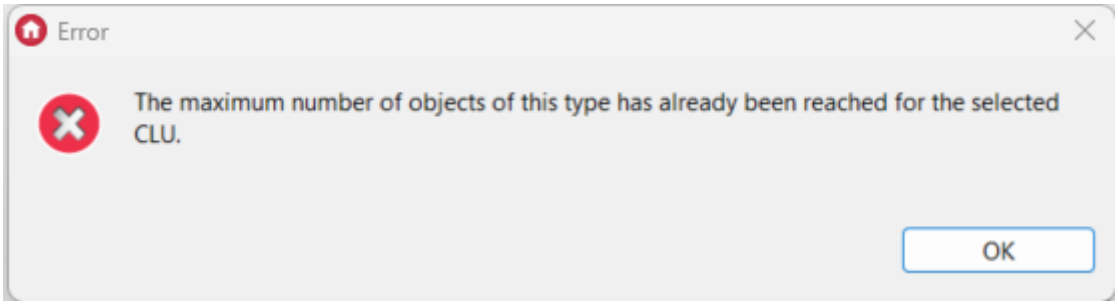
The copied object is added to the list of the given (selected) CLU.



When copying a thermostat, after confirming the object name, a window with the selection of the source/receiver for the copied object is displayed.



If the number of objects of a given type for CLU is exceeded during copying, the following message will be displayed:



4. Date and time

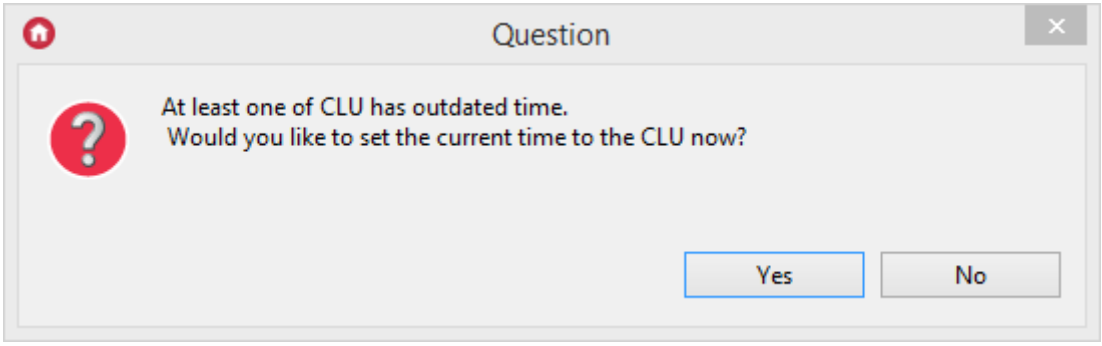
CLU is equipped with real time clock (RTC) powered by built-in battery. CLU provides several features which can be used in the script. The full list of time-related features reads as follows:

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.


Question
At least one of CLU has outdated time.
Would you like to set the current time to the CLU now?
Yes
No

VIII. Visual Builder – Smartphone control

Important information - end of support for functionality of Visual Builder

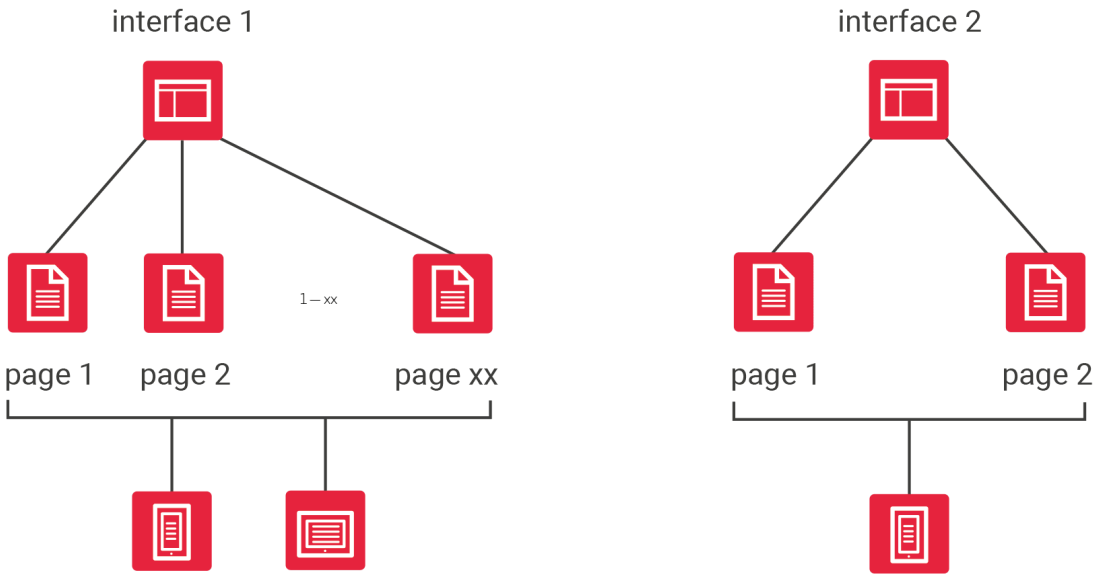
Note!
Support for Visual Builder functionality has been ended in Object Manager version 1.9.0 and higher. The Home Manager interface wizard has been removed and it is impossible to open/edit interfaces created in the project. Saving your project using the current version of OM will result in the loss of any data associated with Visual Builder - including interfaces created in the project.

Note!
Using Visual Builder is only possible with Object Manager version 1.8.1 or lower.

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.

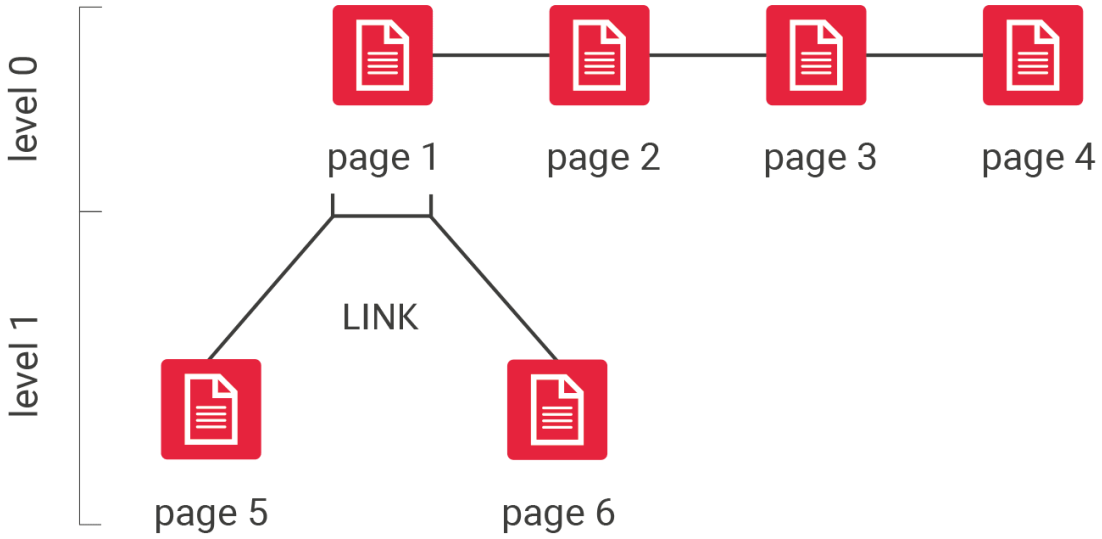


2. Interface structure

Note!
Support for Visual Builder functionality has been ended in Object Manager version 1.9.0 and higher. The Home Manager interface wizard has been removed and it is impossible to open/edit interfaces created in the project. Saving your project using the current version of OM will result in the loss of any data associated with Visual Builder - including interfaces created in the project.

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.

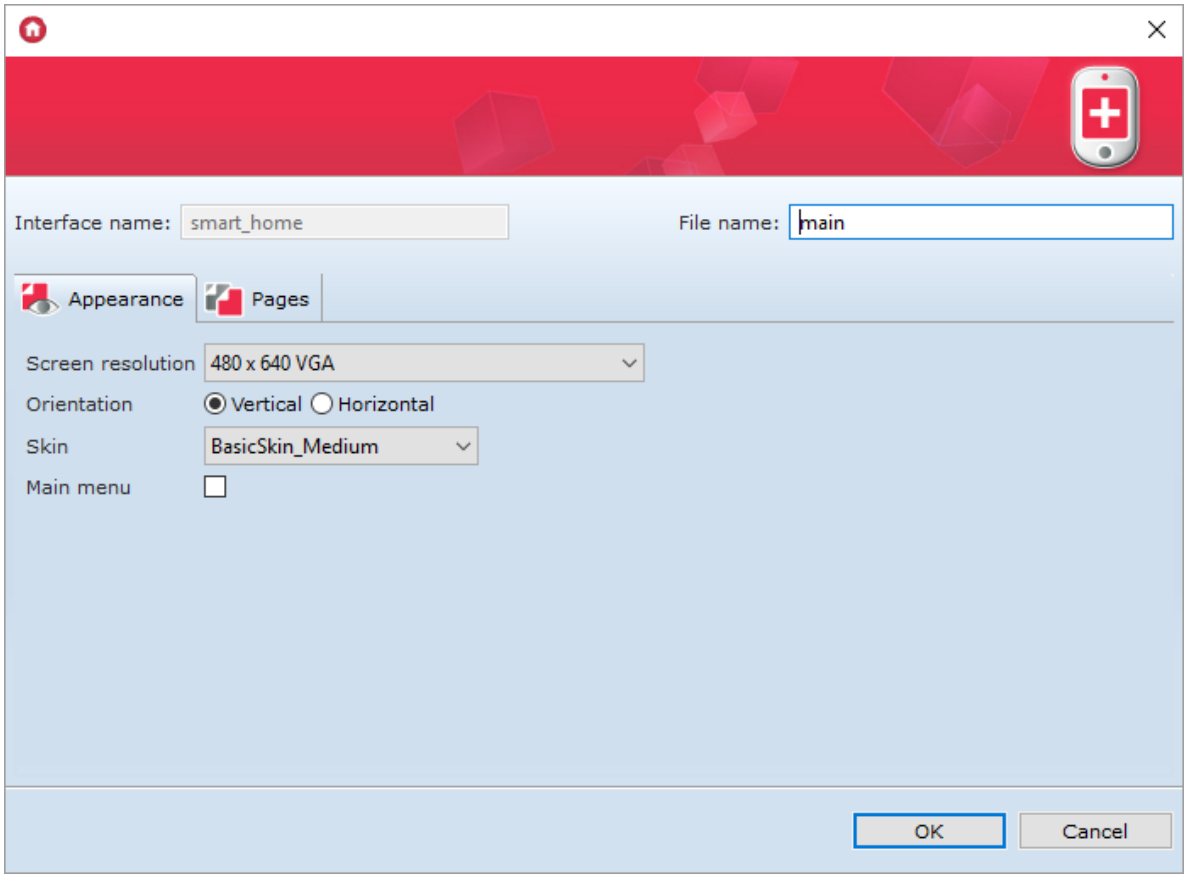
4. New interface creation

Note!
Support for Visual Builder functionality has been ended in Object Manager version 1.9.0 and higher. The Home Manager interface wizard has been removed and it is impossible to open/edit interfaces created in the project. Saving your project using the current version

of OM will result in the loss of any data associated with Visual Builder - including interfaces created in the project.

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

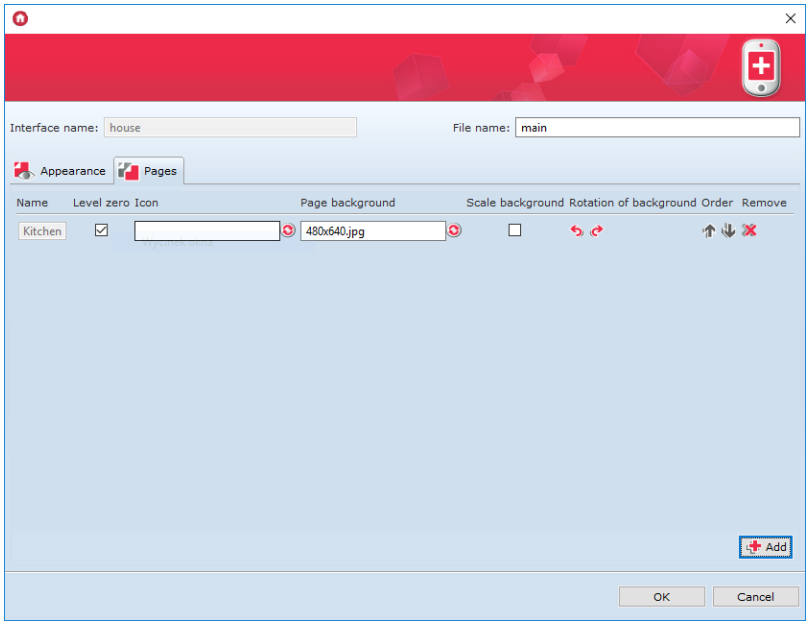
After entering name, new interface window will open. It contains two tabs: appearance and pages (interface window is also available through double-clicking icon of created interface in the objects menu). `Appearance` tab:



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.

In addition, there is possibility of background scaling. Selecting this option adjusts any background resolution to the resolution of the interface being created.

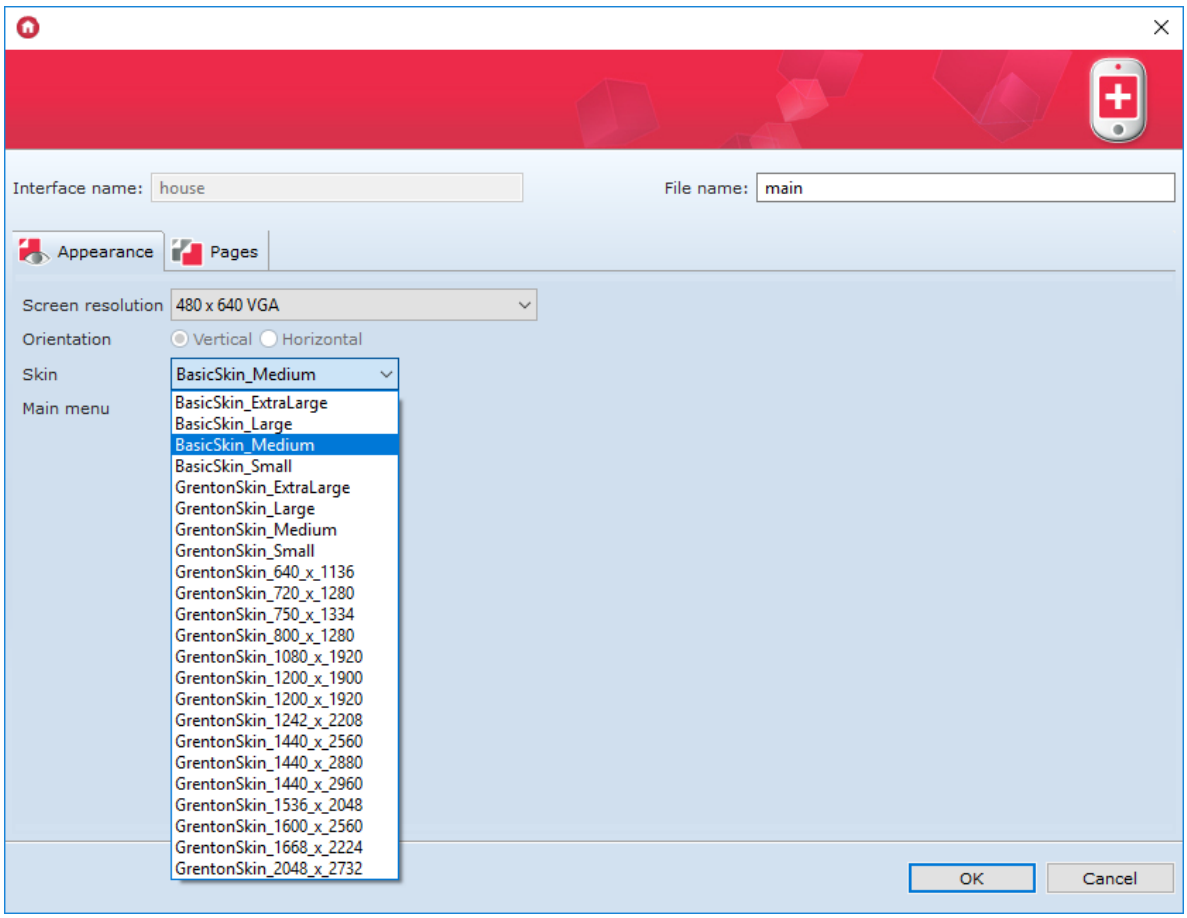
> **Note!**
> Newly created project has no information in the *Pages* tab, new information appears whenever interface page is created.

4.1. Graphic skin selection

Skins are graphic settings sets for mobile device interface.

GRENTON skins

The user can use skins provided with OM in created interfaces. List of available skins is located in the mobile interface parameters.

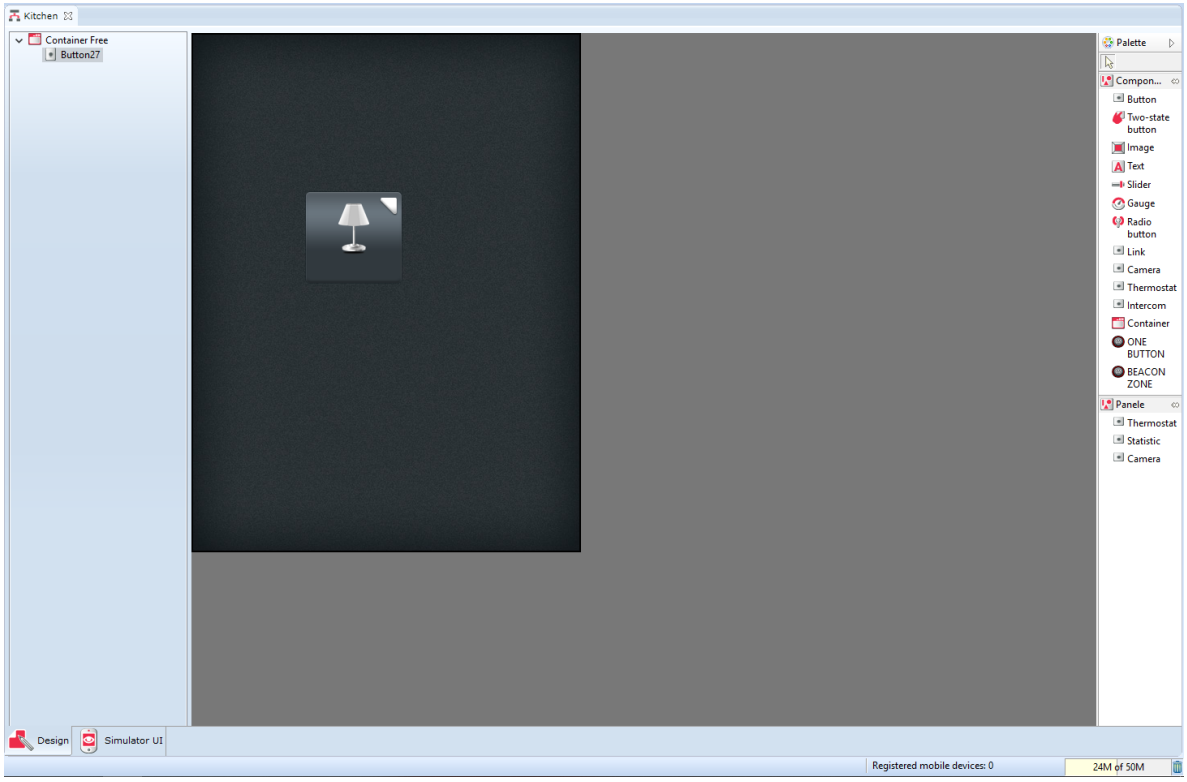


4.2. Interface pages creation

After interface creation, new pages should be added to it. Page creation is launched from the actions menu:

After creating the new page and naming it, edition worksheet will open. The worksheet contains two tabs: Design and UI Simulator (the tabs are located at the bottom part of the page).

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

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

  Two-state button

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

  Image

- **Text** - enables adding text box

  Text

- **Slider** - enables fluid regulation

  Slider

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

  Gauge

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

  Radio button

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

  Link

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

  Container

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

  Camera

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

  Thermostat

- **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)

  Intercom

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

  ONE BUTTON

- **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).

  BEACON ZONE

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

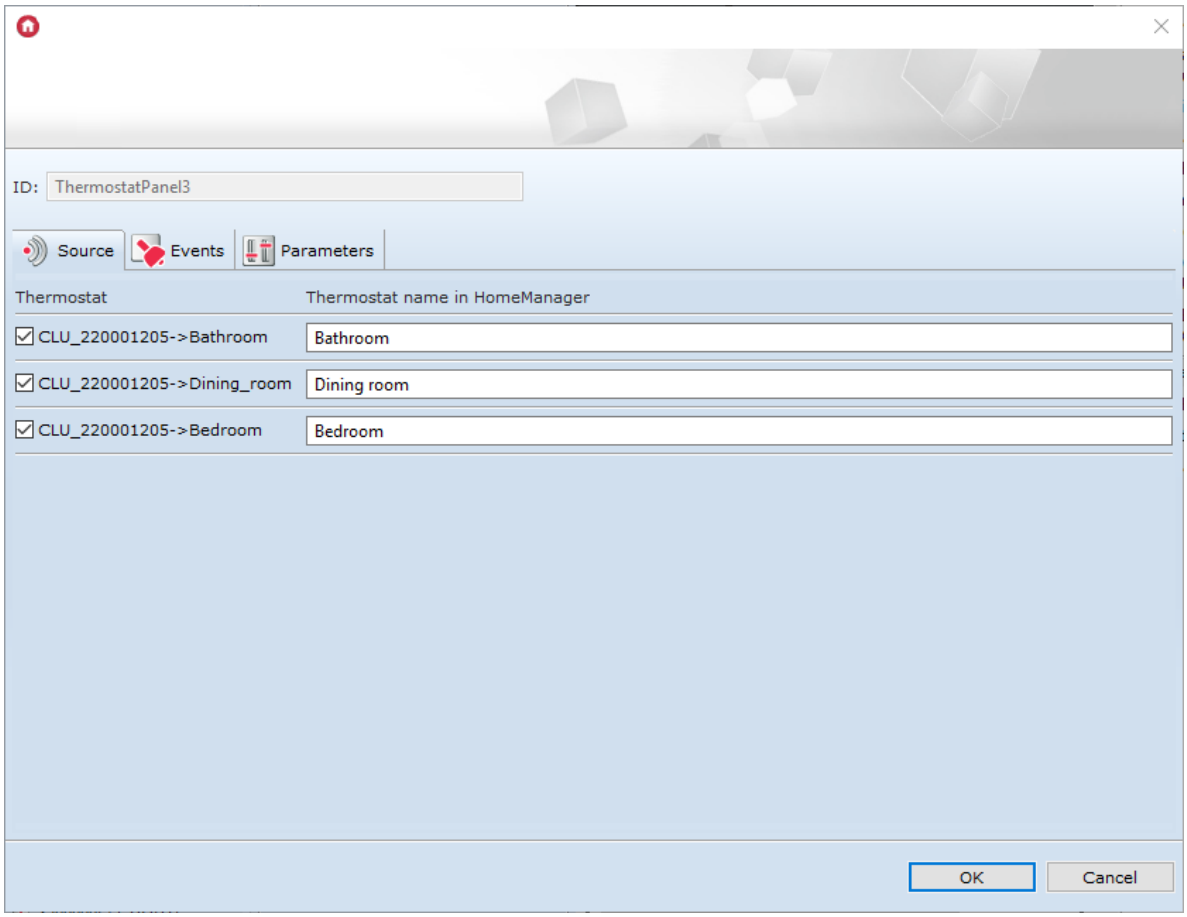
4.4. Panels

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

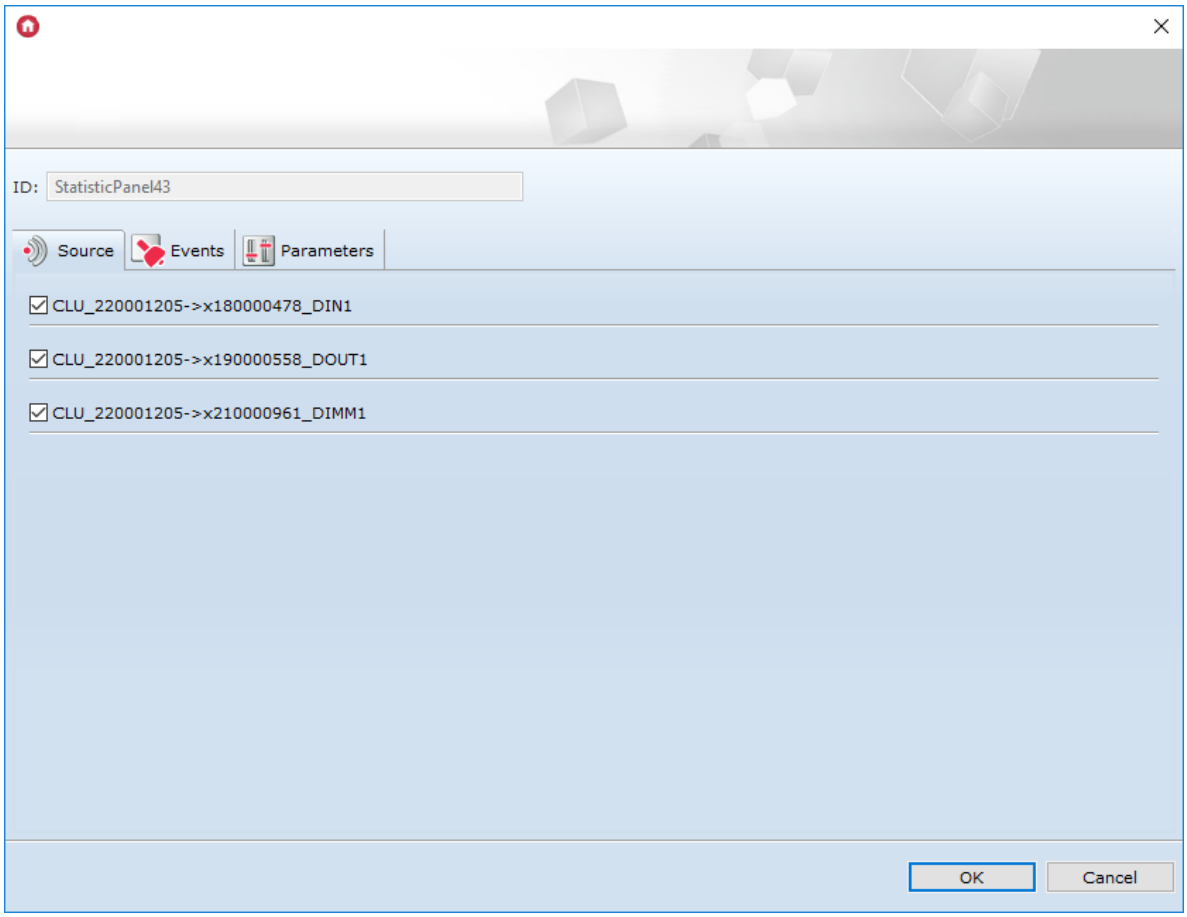
The previously created virtual object `Thermostat` is set as the thermostat panel source.



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

  Statistic

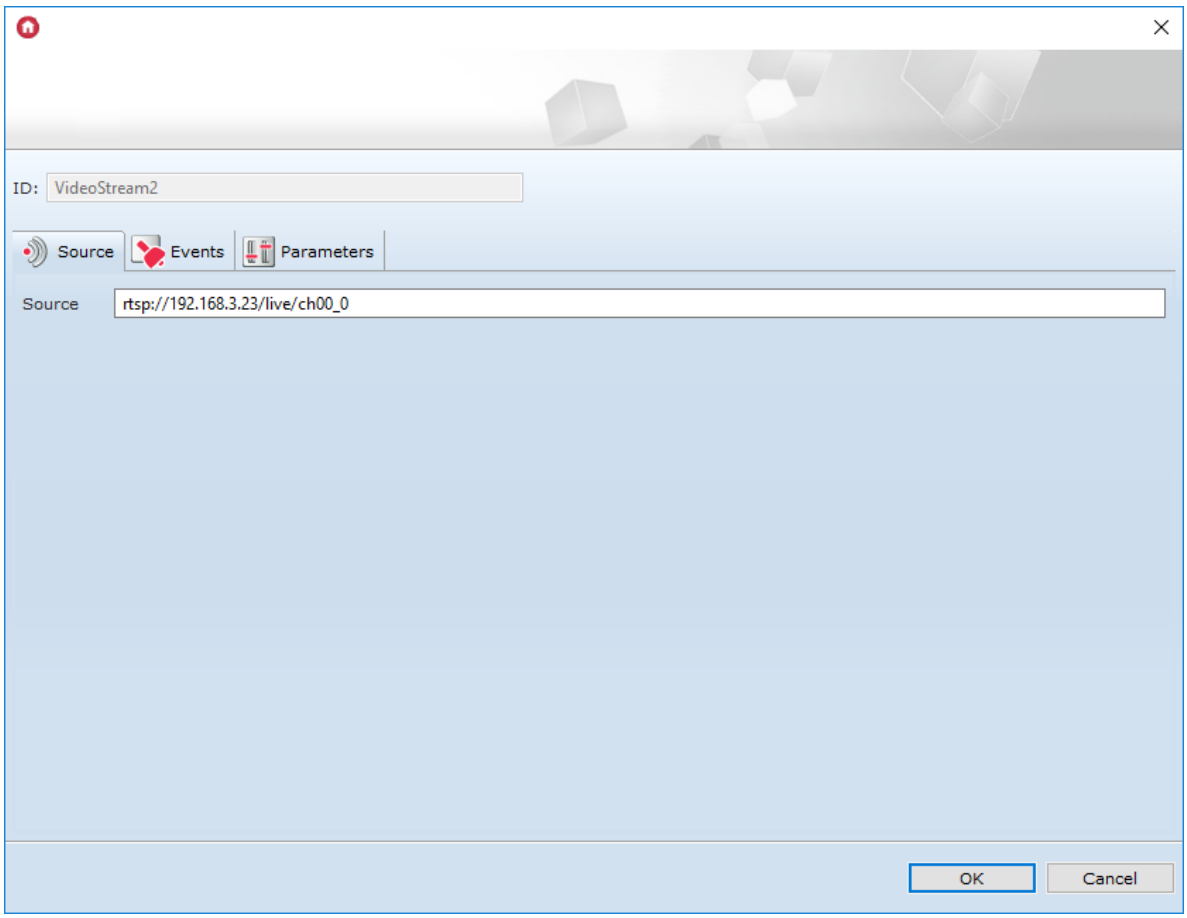
After dragging the panel to the interface page, select the objects for which the media measurement will be presented in the HM. The window will display only objects for which *Media Measurement* was previously attached.



- **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.

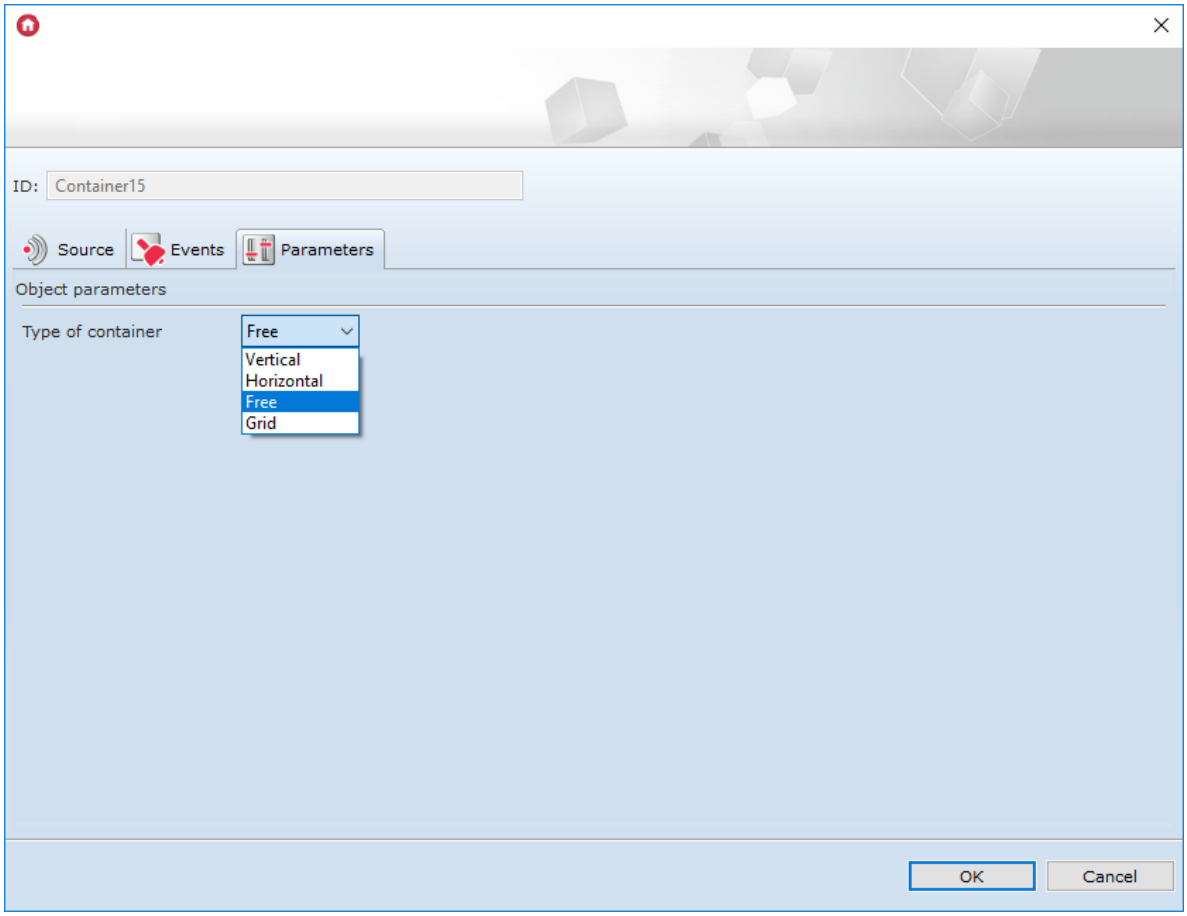


4.5. Containers

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

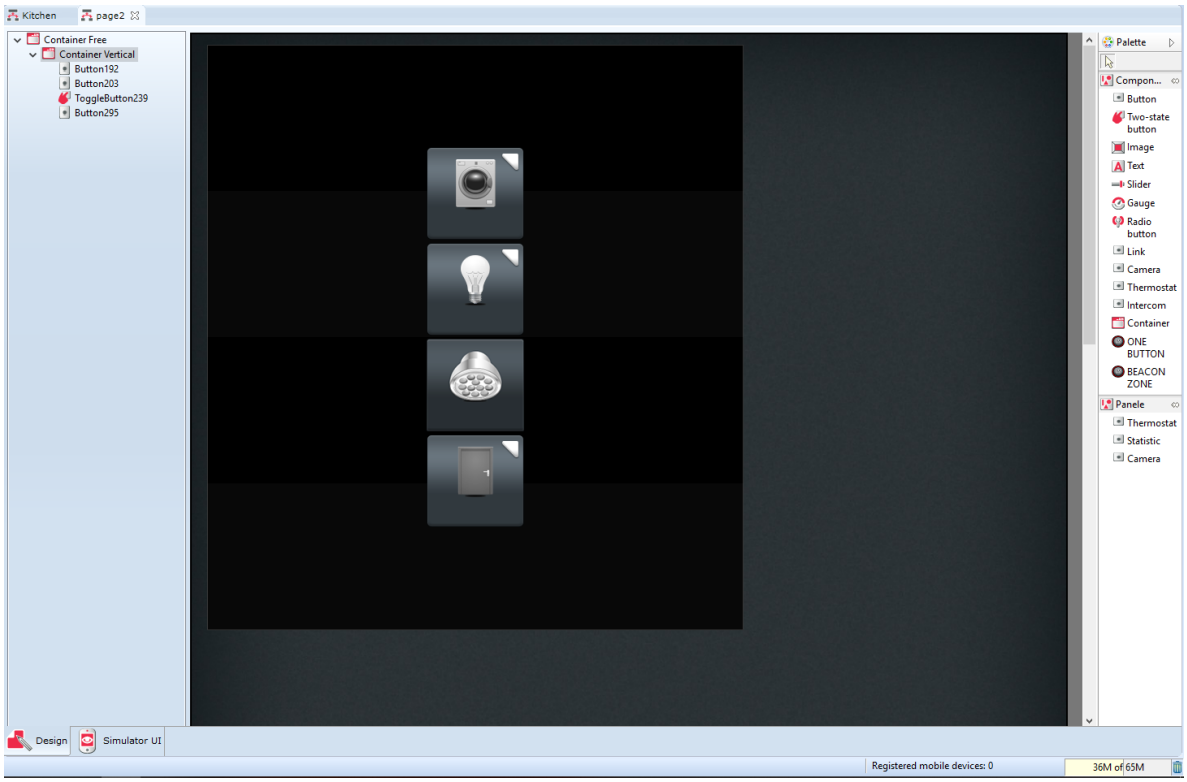
Objects within the workspace are arranged accordingly to the type of the selected container.

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.

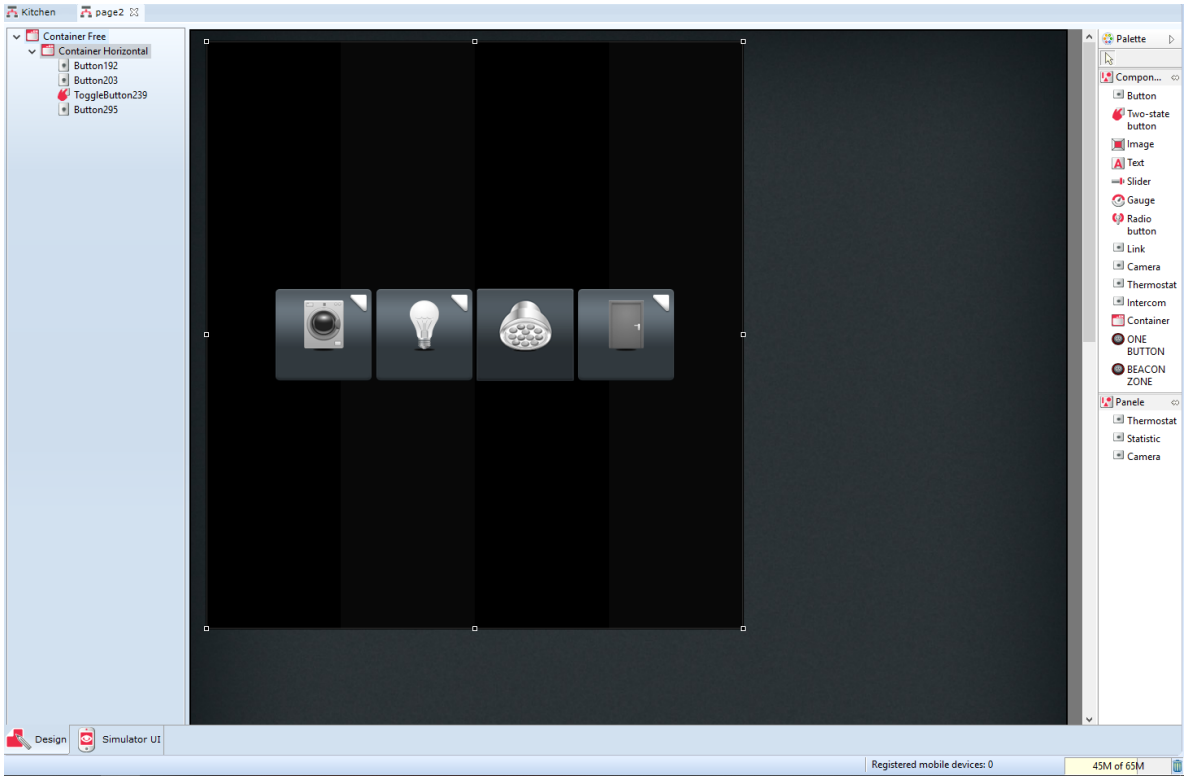


There four types of containers:

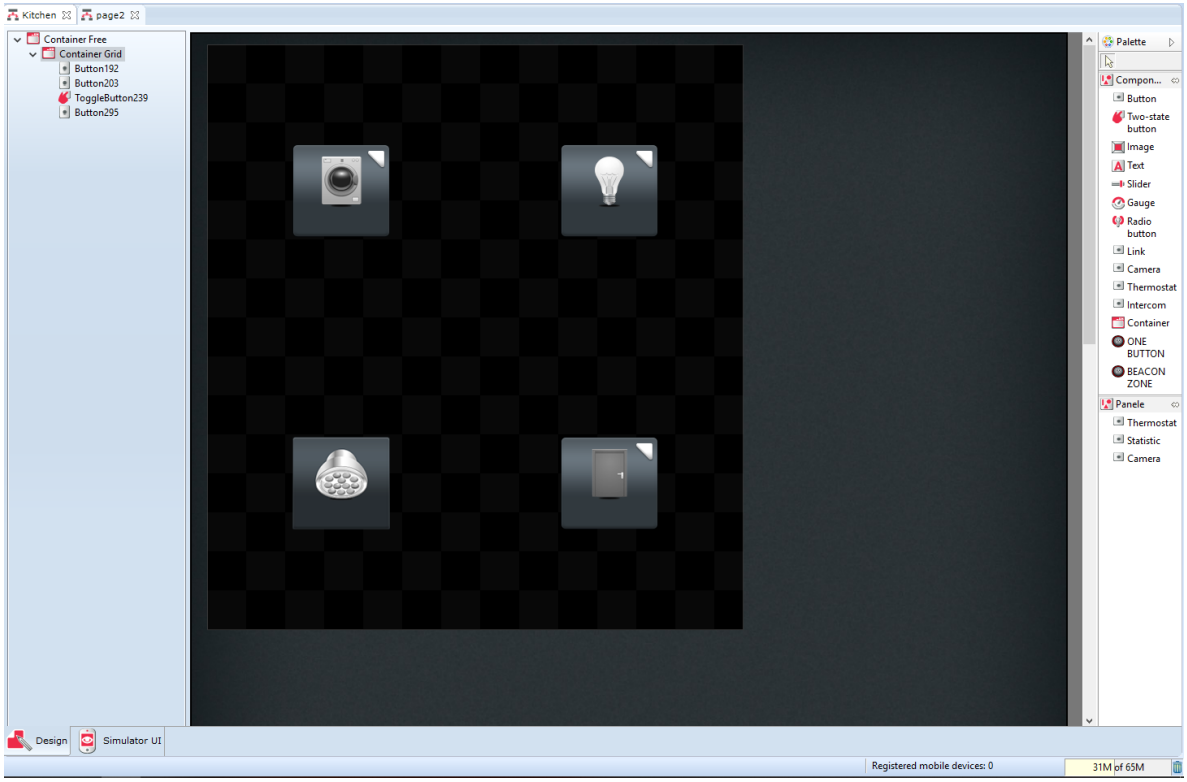
1. **Vertical** - the elements are arranged vertically in equal, automatically created sections.



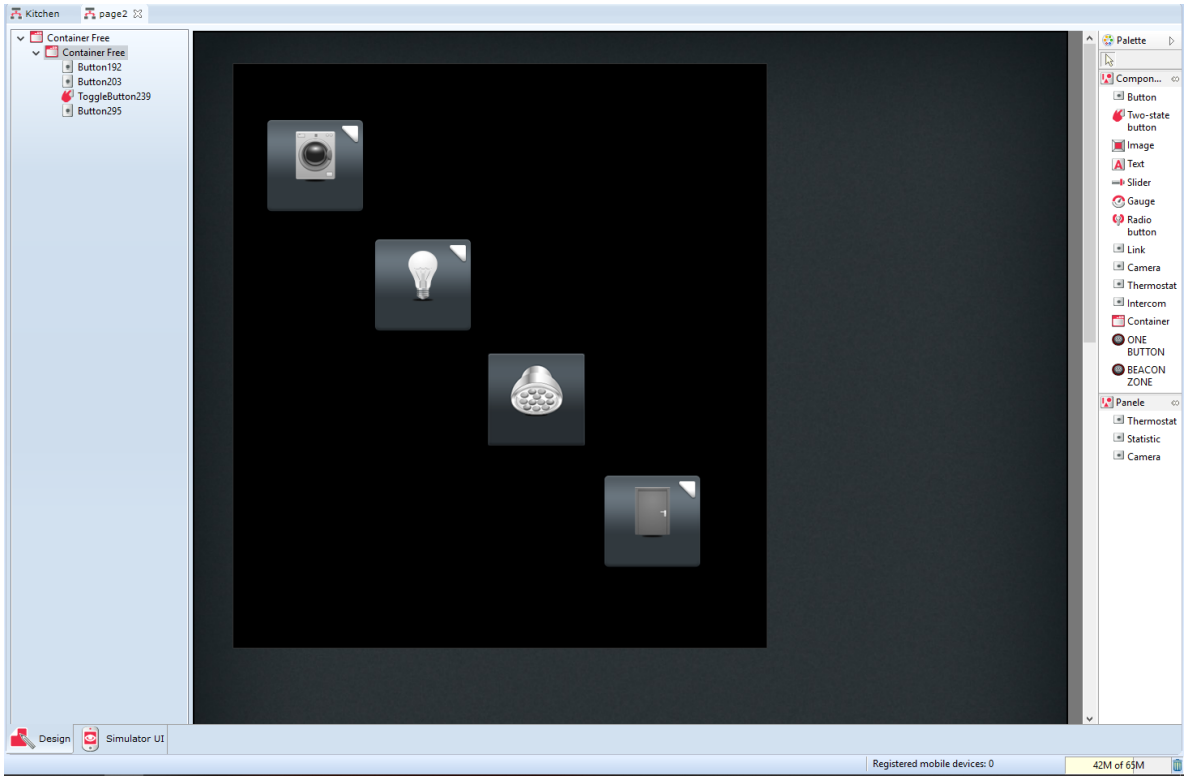
2. **Horizontal** - the elements are arranged in horizontal sections



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



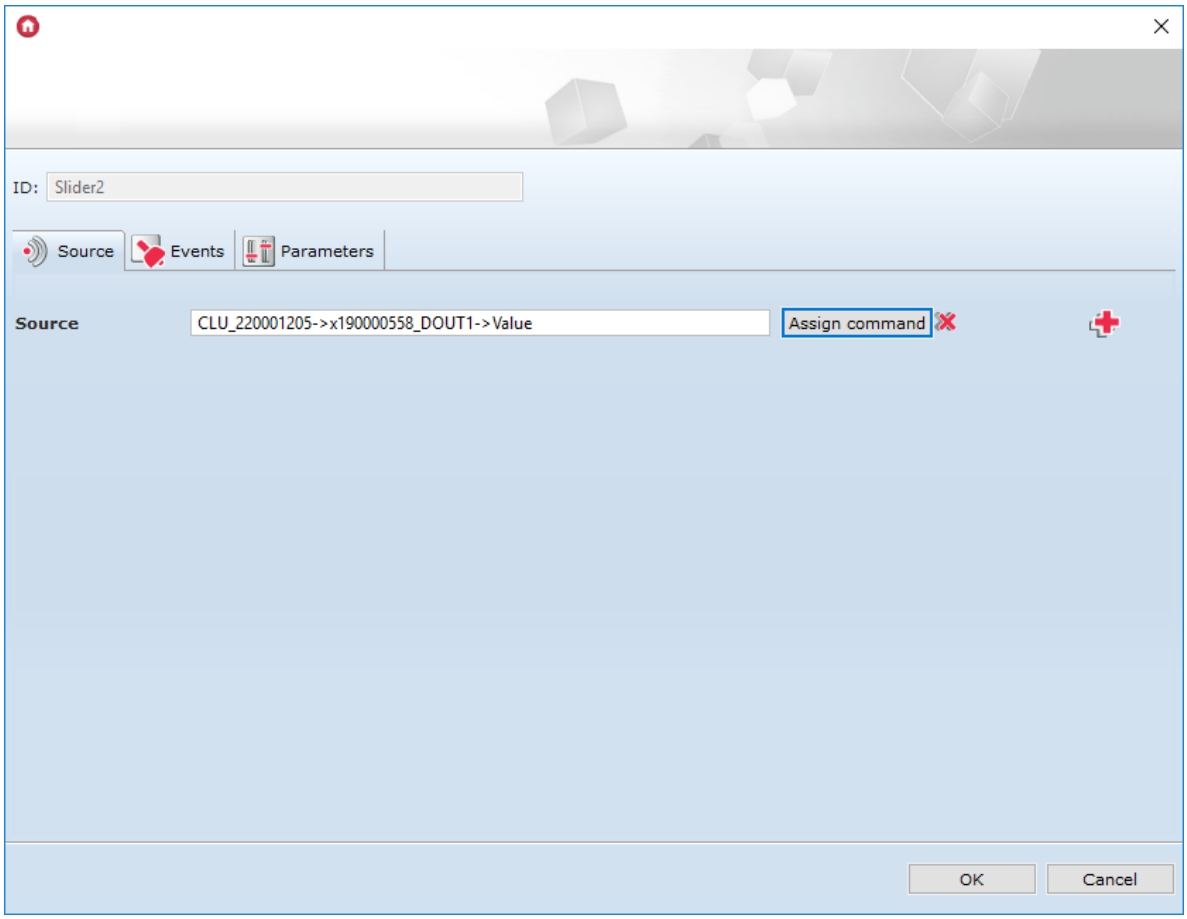
4. **Random** – enables any arrangement of the elements within the whole container area



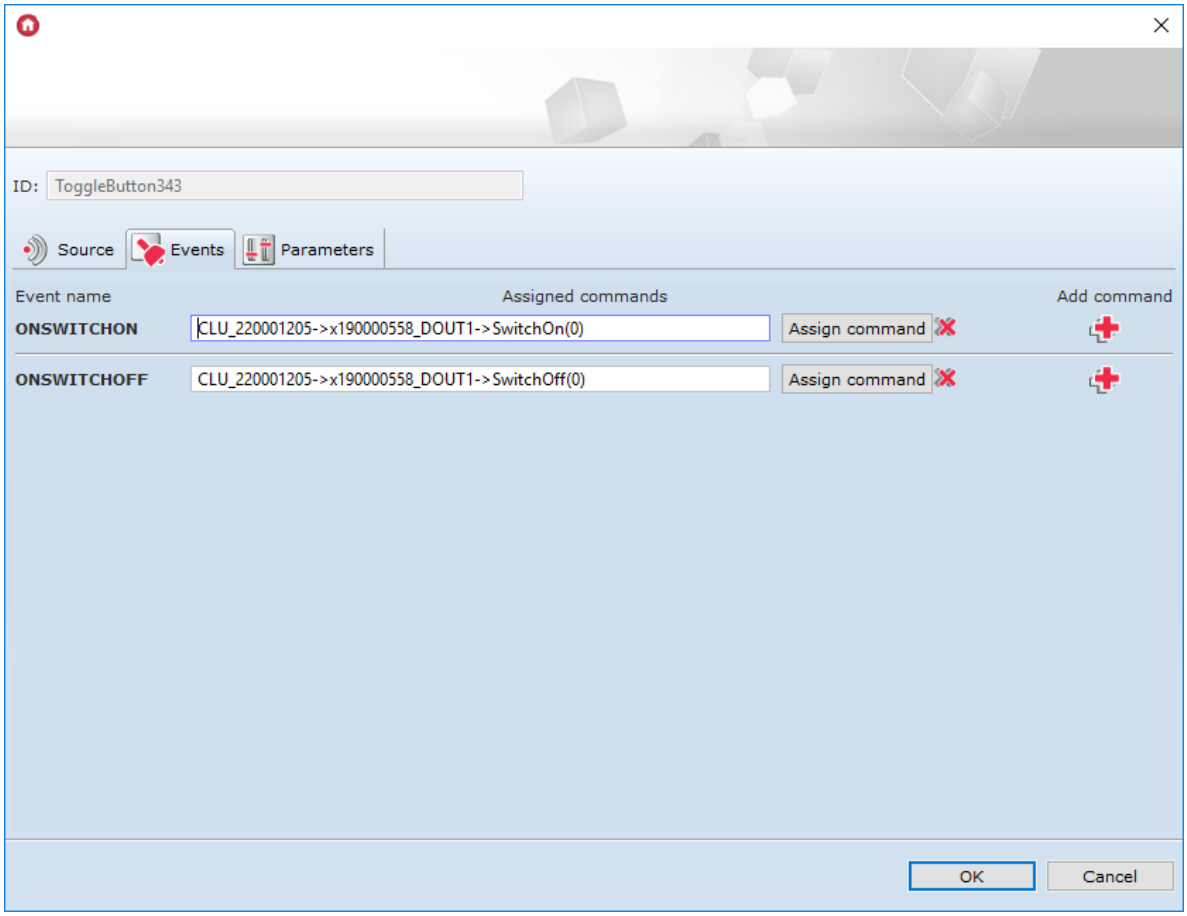
4.6. Adding components and connecting to the system objects

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:

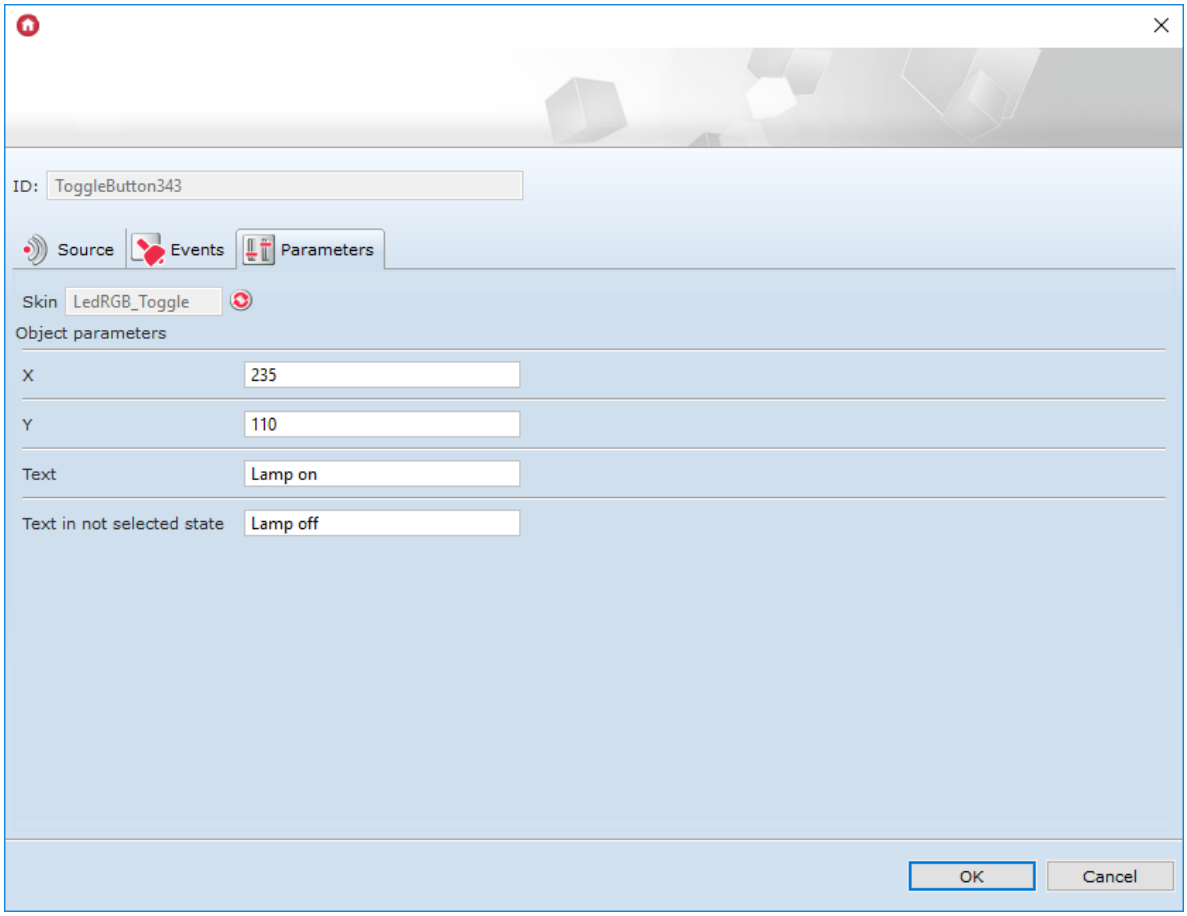
1. In the `Source` tab, select an object which value should be mirrored, and time of refreshing the value, e.g. if you put a slider controlling the dimmer in the interface, then the controlled dimmer must be set as a source so the current value of light can be displayed on the smartphone.



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.

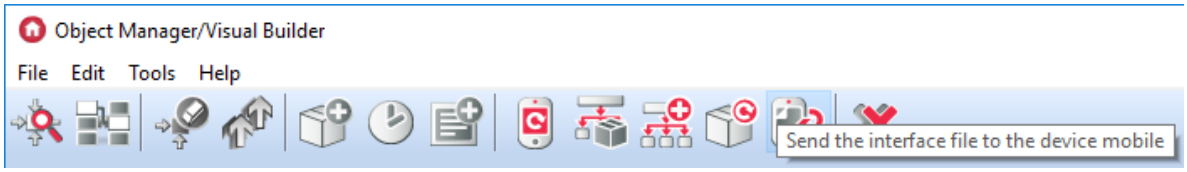
After and during interface creation the user has an option of checking its functionality and appearance. To do that, launch UI Simulator (second tab on the bottom of the page).

4.7. Sending interface to mobile device

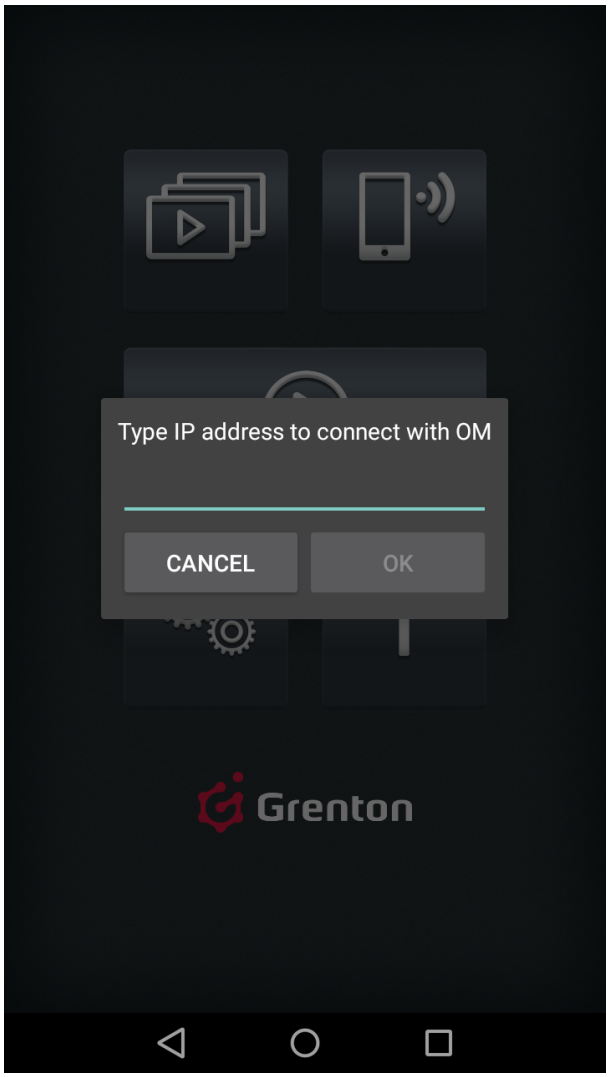
To enable system control by a mobile device, the created interface must be sent to GRENTON HOME MANAGER application installed on the device.

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:



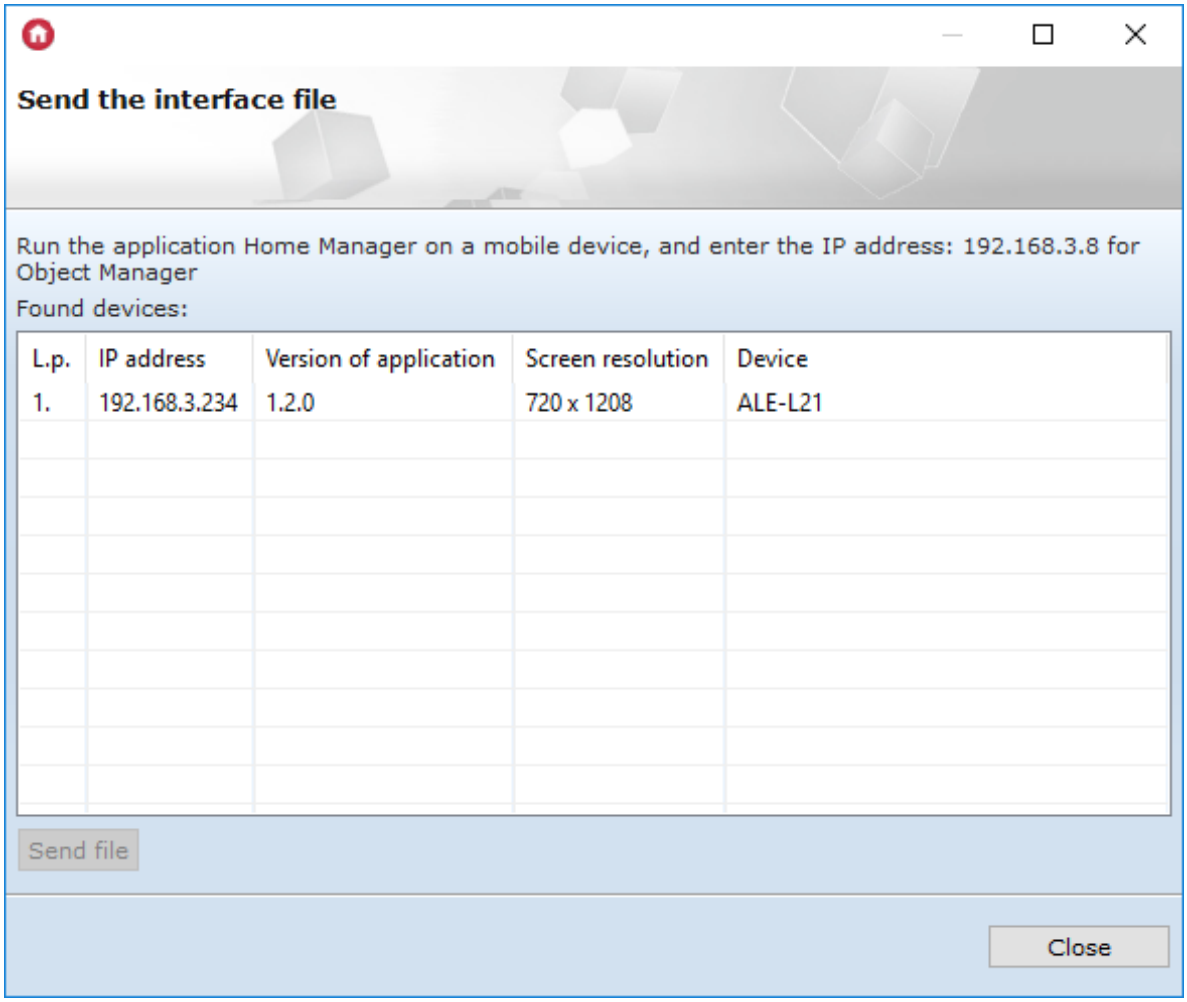
- On the mobile device, connect to the network in which the CLU is located (after displaying the send window in the Object Manager);
- In the open Home Manager application, select **Connect to OM from the main menu**;
- Enter the IP address of the Object Manager and choose *OK*:



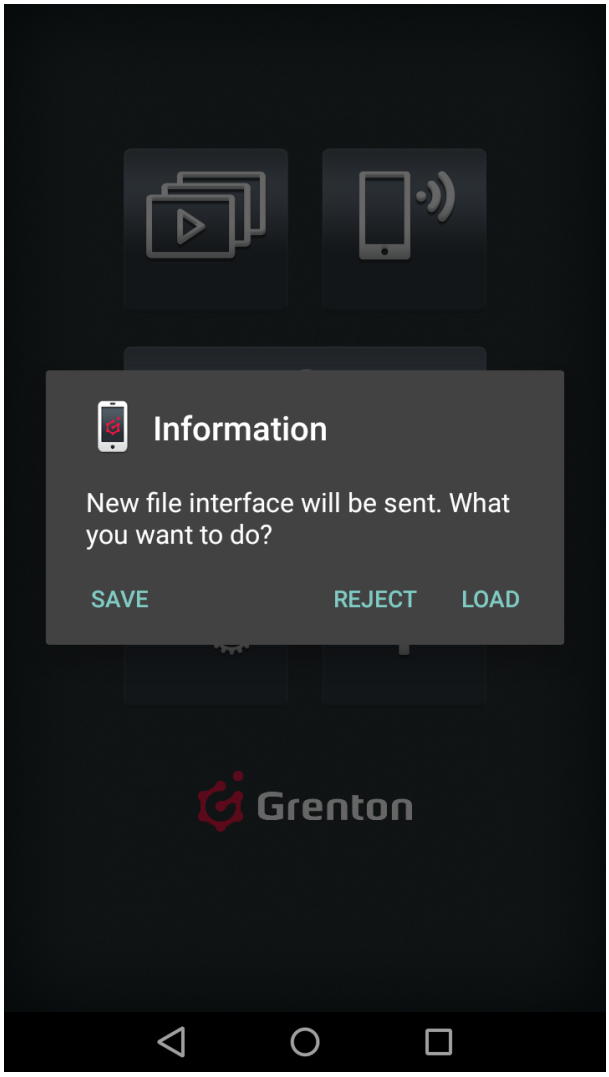
- The mobile device will appear in the send window that was displayed in the Object Manager;

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*:



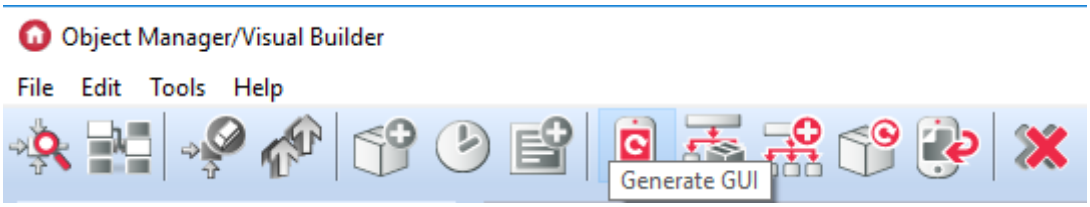
- The transfer status bar will appear on the screen. When finished, the information on the correct completion of the transfer will be displayed on the upper bar of the program.
- After sending the file with the interface to the mobile device, for the remote control to be possible, the uploaded interface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

5. Automatic interface creation - GUI generator

> **Note!**
> **Support for Visual Builder functionality has been ended in Object Manager version 1.9.0 and higher. The Home Manager interface wizard has been removed and it is impossible to open/edit interfaces created in the project. Saving your project using the current version of OM will result in the loss of any data associated with Visual Builder - including interfaces created in the project.**

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:

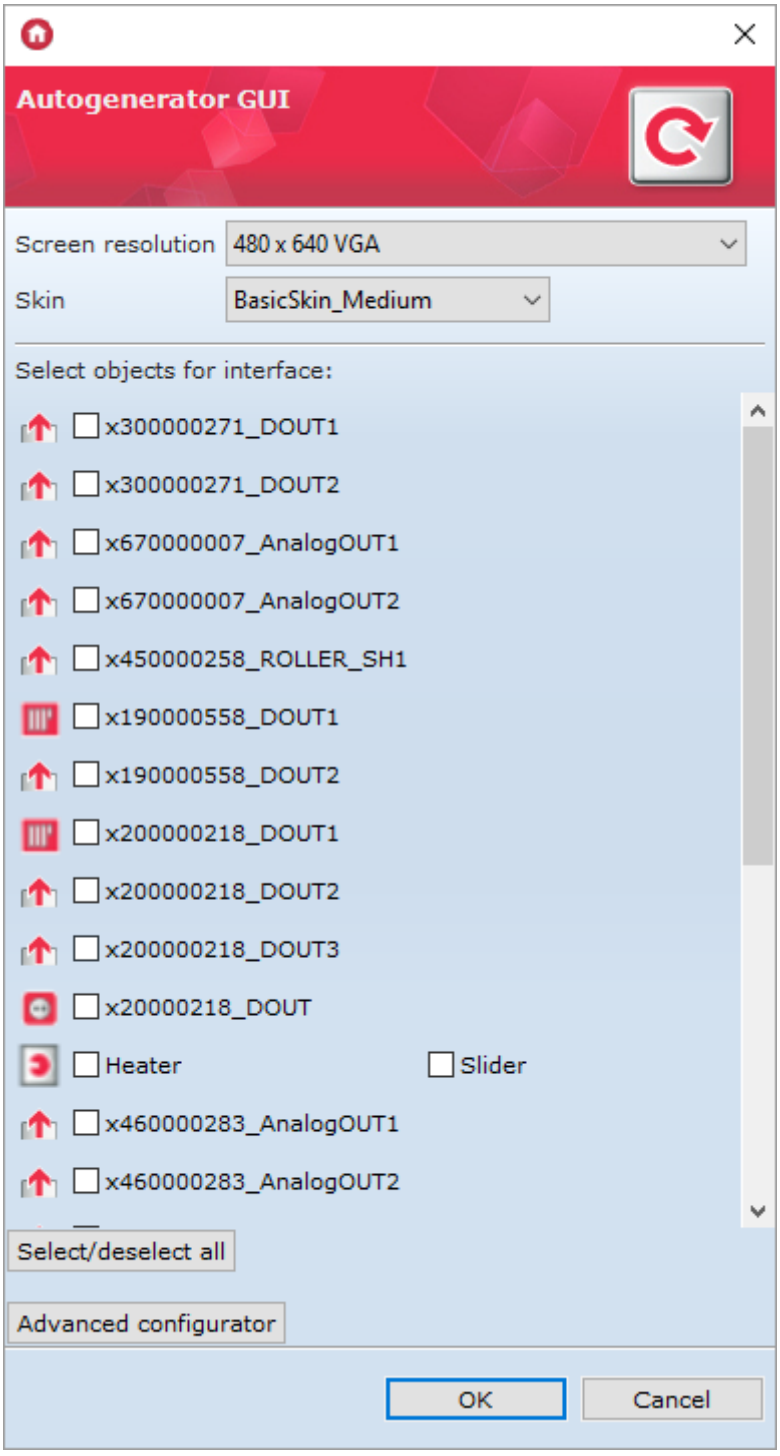


5.1. Creating an interface with available resolution

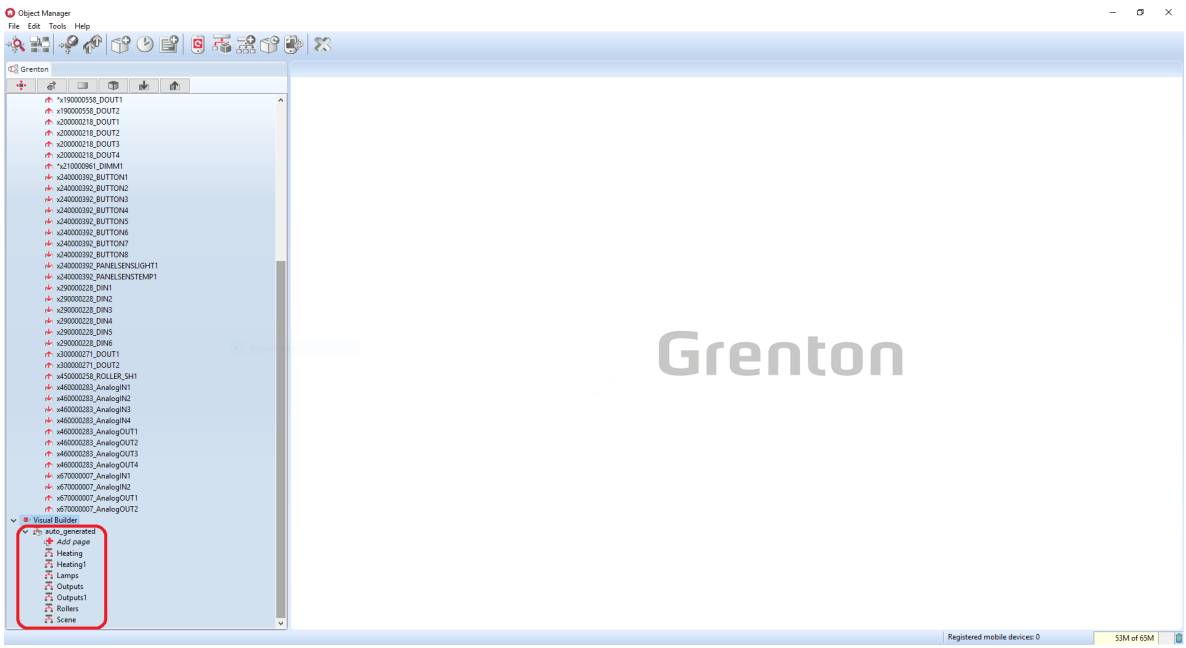
A. Simple configurator

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

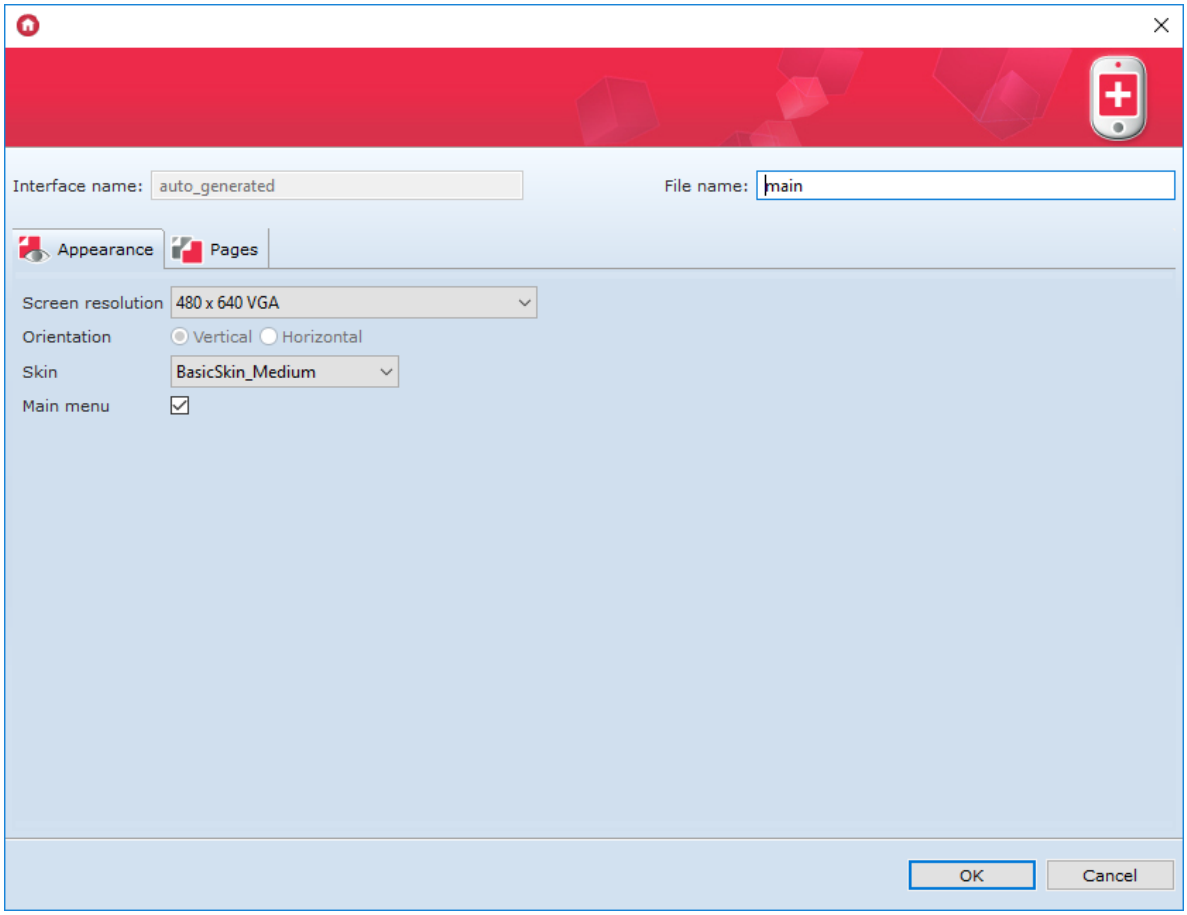


After selecting the objects of interest, click `OK`. As a result, the newly created pages appear in the list of objects (under the icon of the created interface) according to the following figure:



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`.

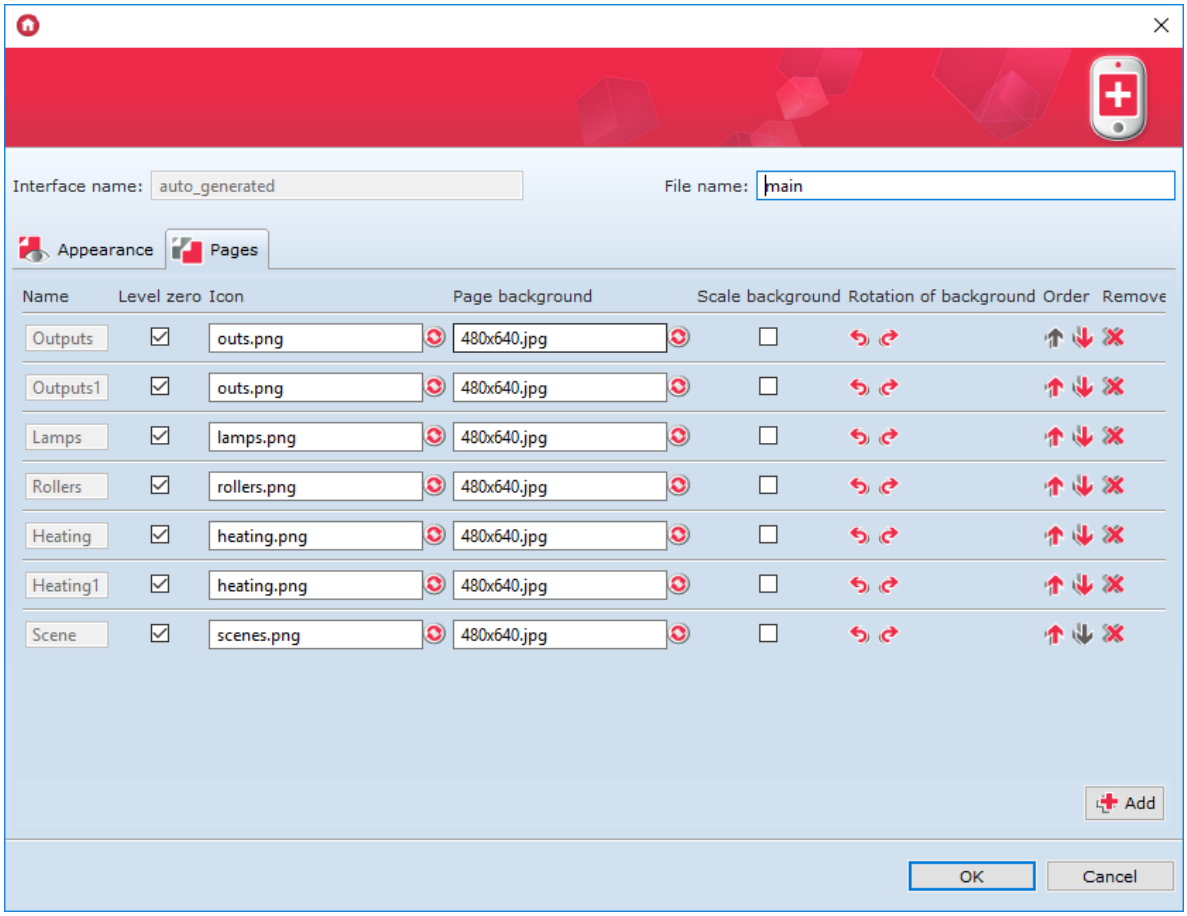
In the `Appearance` tab, the user can select skins visible in the interface. In this view there is also a field *Main menu*. After selecting it, a menu will be created containing all available and selected pages.



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

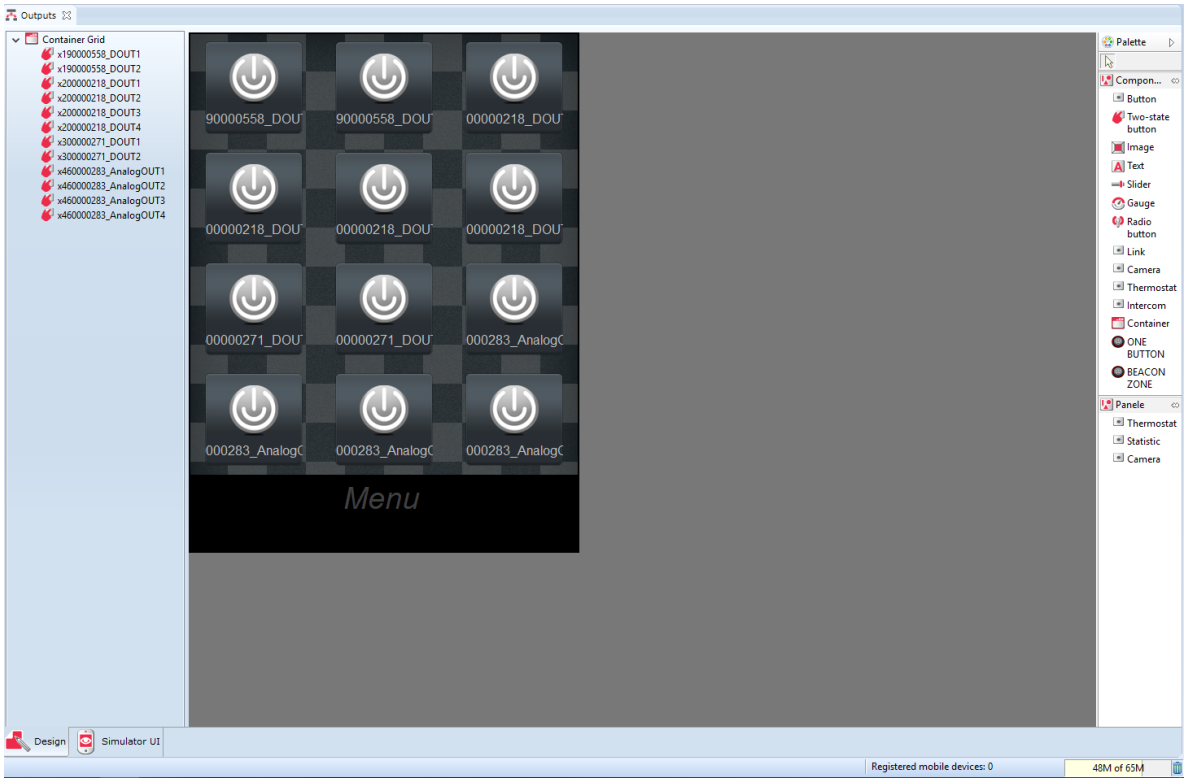
- **Level zero** - whether or not the page should be displayed in the menu;
- **Icon** – icon displayed in the menu (by default, it is icon from the selected skin);
- **Background** – background of the displayed page. By default, background from the selected skin is displayed, but the user can define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

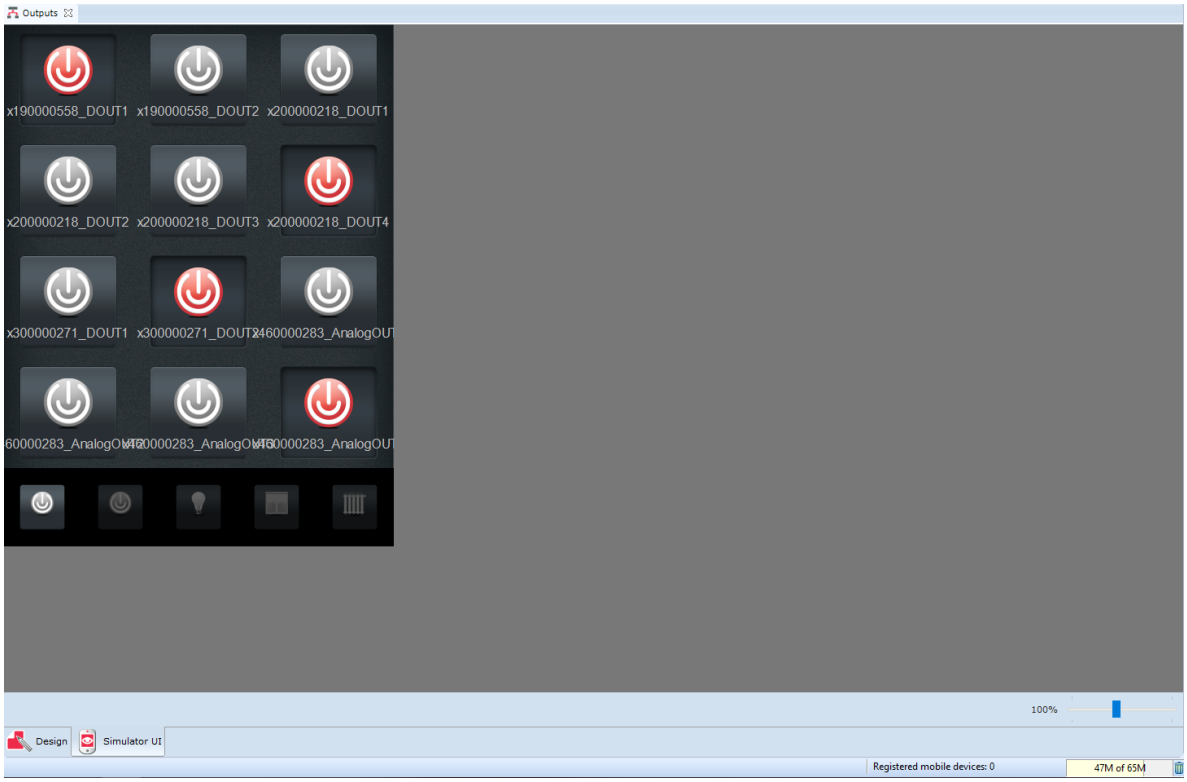


The user also has the option of making changes to the generated pages. Double-clicking on the page icon will open the edit sheet, containing the two tabs `Design` and `Simulator`.

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



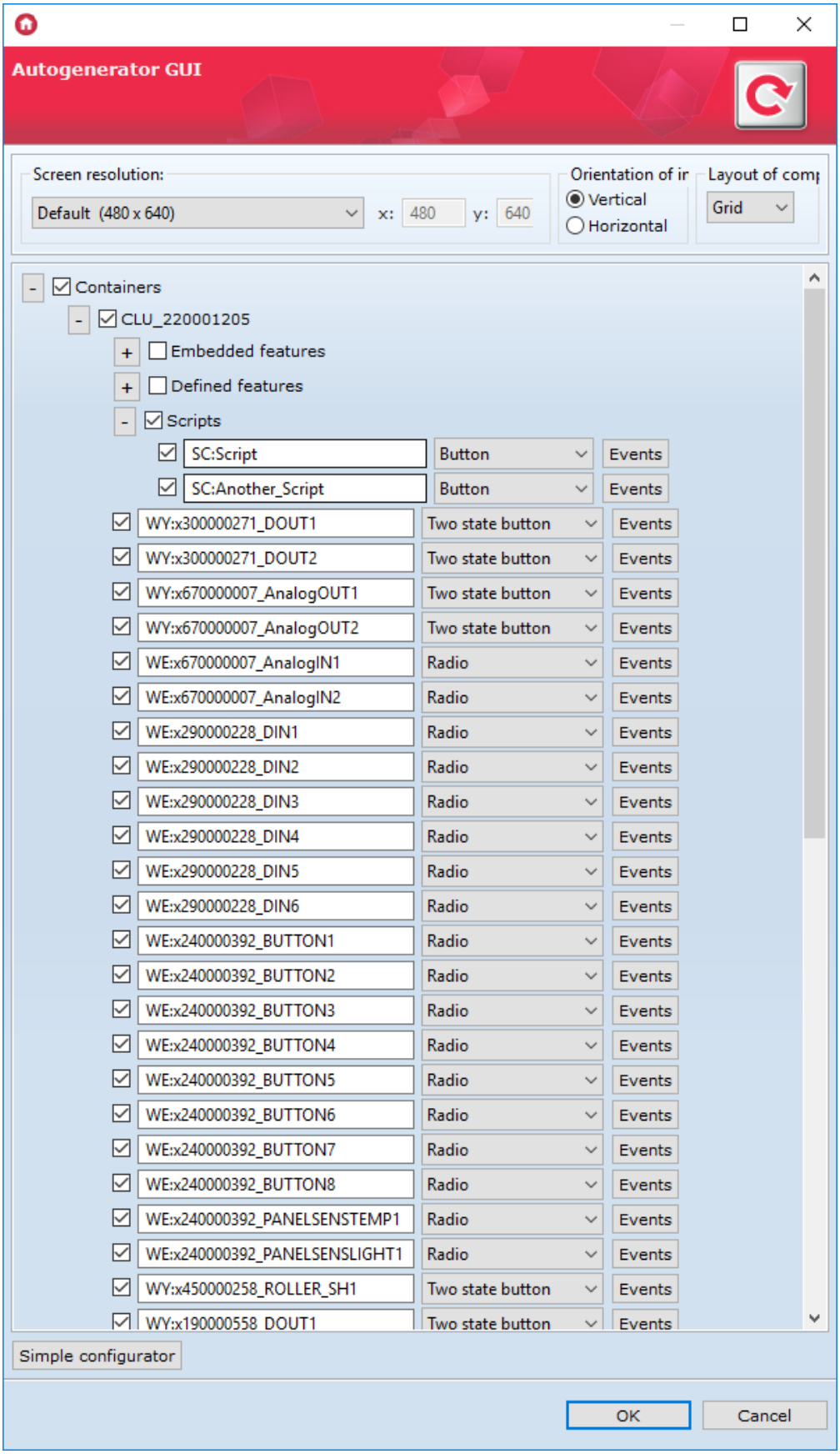
The `Simulator` tab - gives the user the ability to check the appearance and operation of the created interface from the computer screen (before it is sent to the mobile device).



B. Advanced configurator

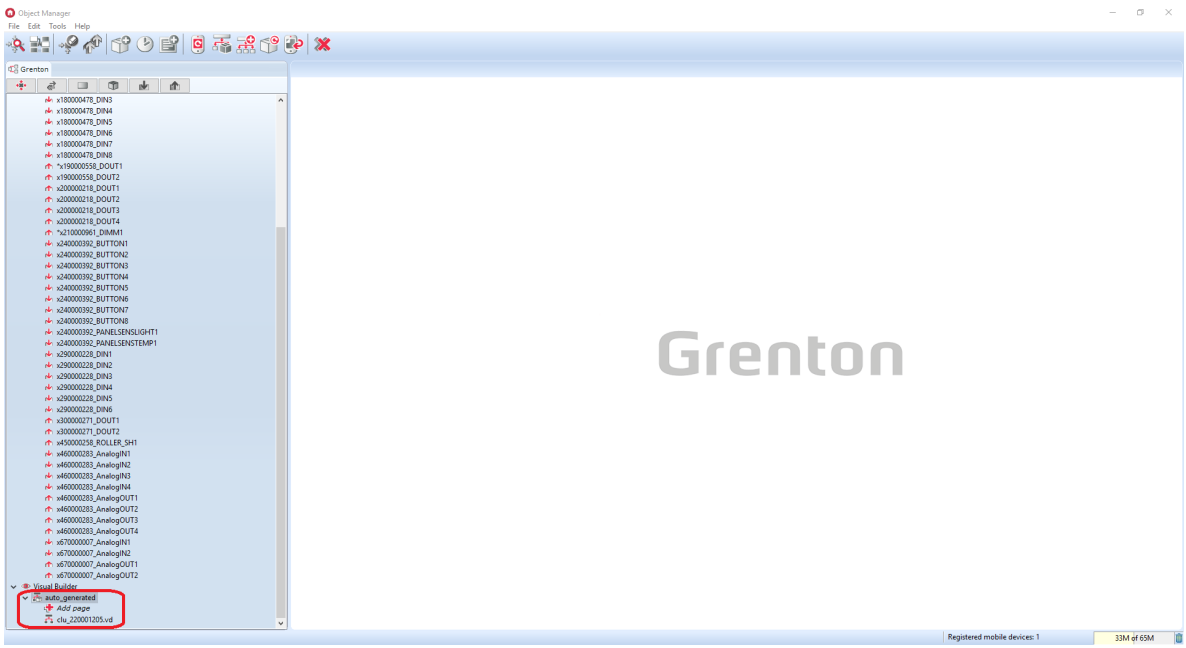
After clicking the `Generate GUI` icon in the `Autogenerator` window, you can select the `Advanced configurator` option. Selecting this option will open a new window in which you should select:

- 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 `OK`, the window of the created interface opens. The window, in addition to the name field of the created interface, contains two tabs: `Appearance` and `Pages`. 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 `OK`, the newly created pages appear in the list of objects (under the icon of the created interface) as shown in the following figure:

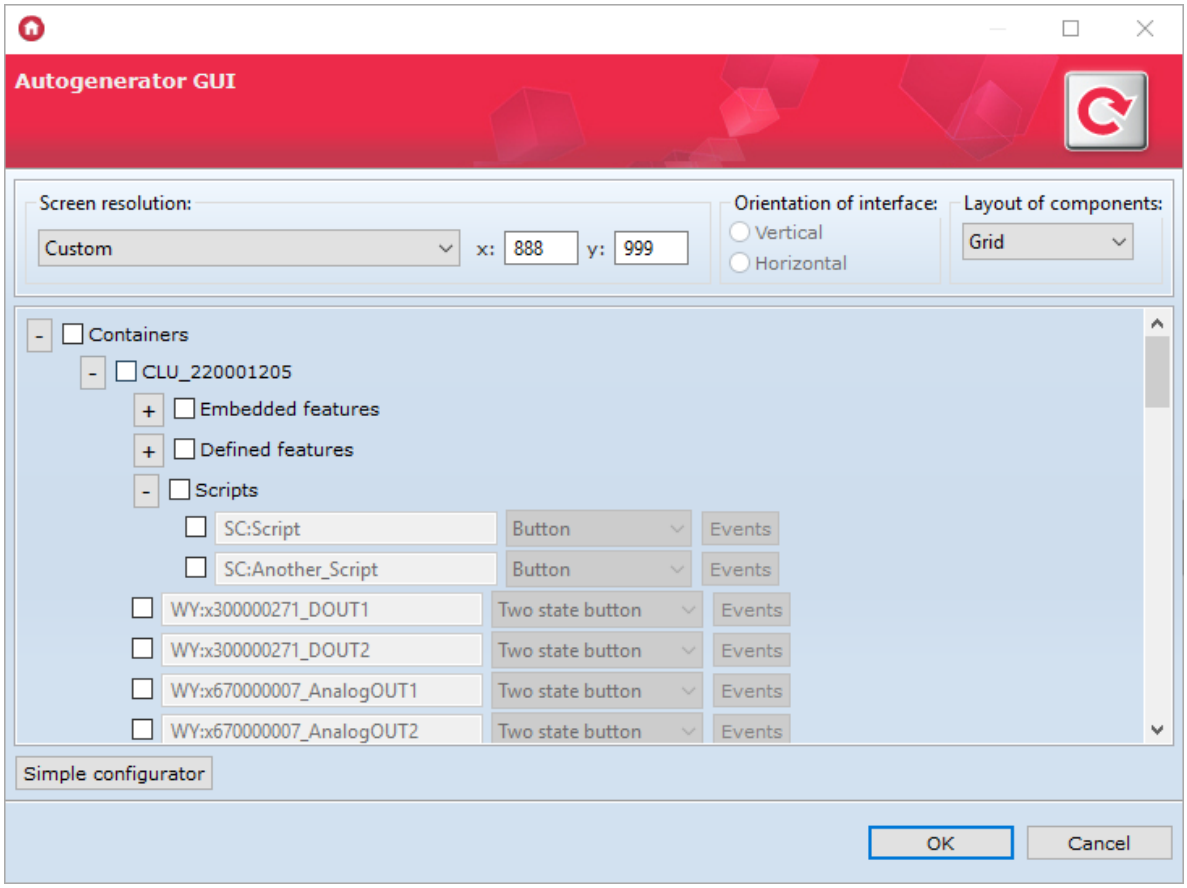


As with the simple configurator - the user has the option of making changes to the generated pages. Double-clicking on the page icon will open the edit sheet, containing the two tabs `Design` and `Simulator`.

5.2. Creating an interface with its own resolution

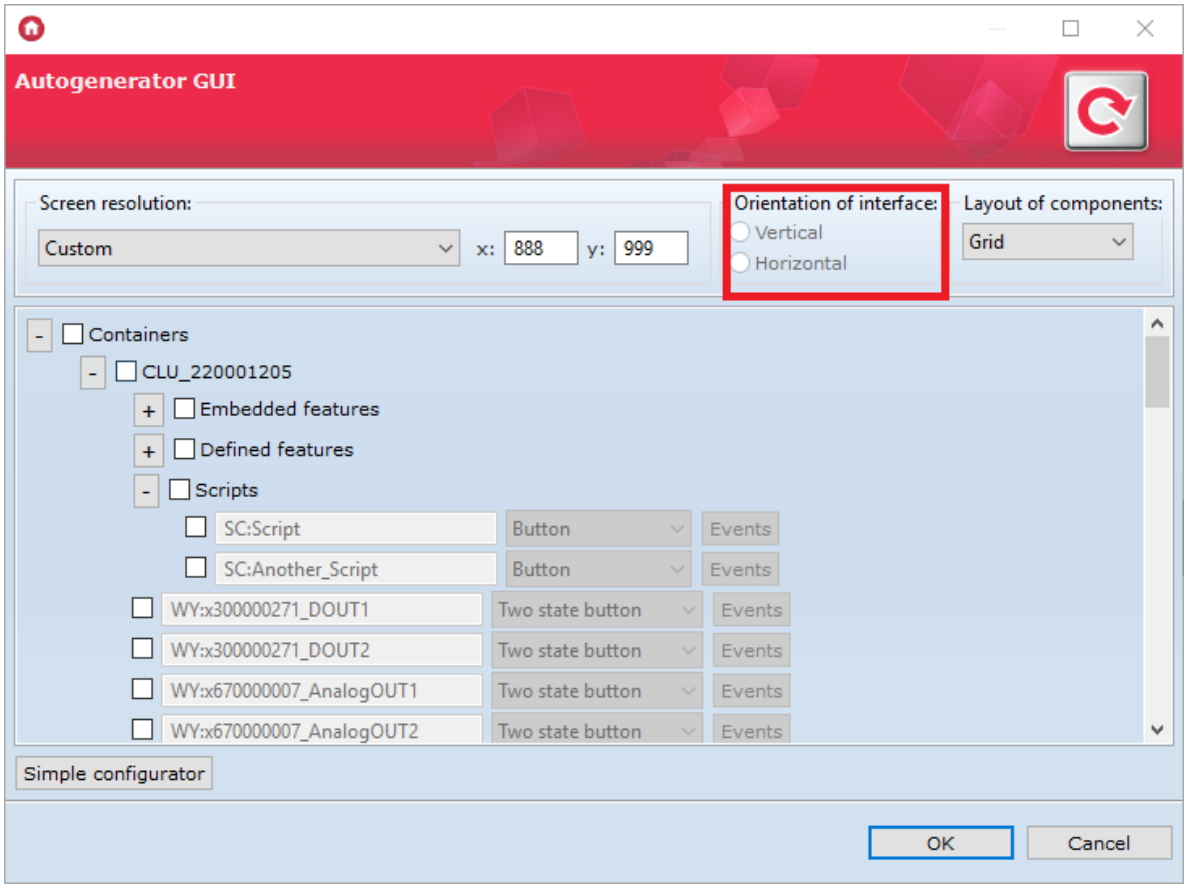
In the case of the advanced configurator, it is possible to create an interface with its own, selected resolution. To do this:

- 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 in Object Manager version 1.9.0 and higher. The Home Manager interface wizard has been removed and it is impossible to open/edit interfaces created in the project. Saving your project using the current version of OM will result in the loss of any data associated with Visual Builder - including interfaces created in the project.**

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 [1];
- Find the port settings;
- Set the triggering and forwarding port to **554** [2] 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 door phone - 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 [3];
- It is necessary to find the *SIP* account settings [4];
- 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.

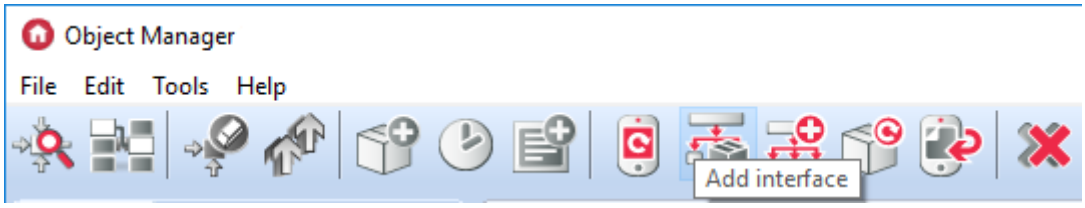
Note!
If in the setting of Intercom, it is necessary to select one account from several configurable, select the previously selected one - in the example **Account_1**!

6.2. Creation and configuration of the application interface

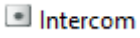
A. Adding a door intercom to the application interface in the Object Manager

In order to add a video intercom to the interface:

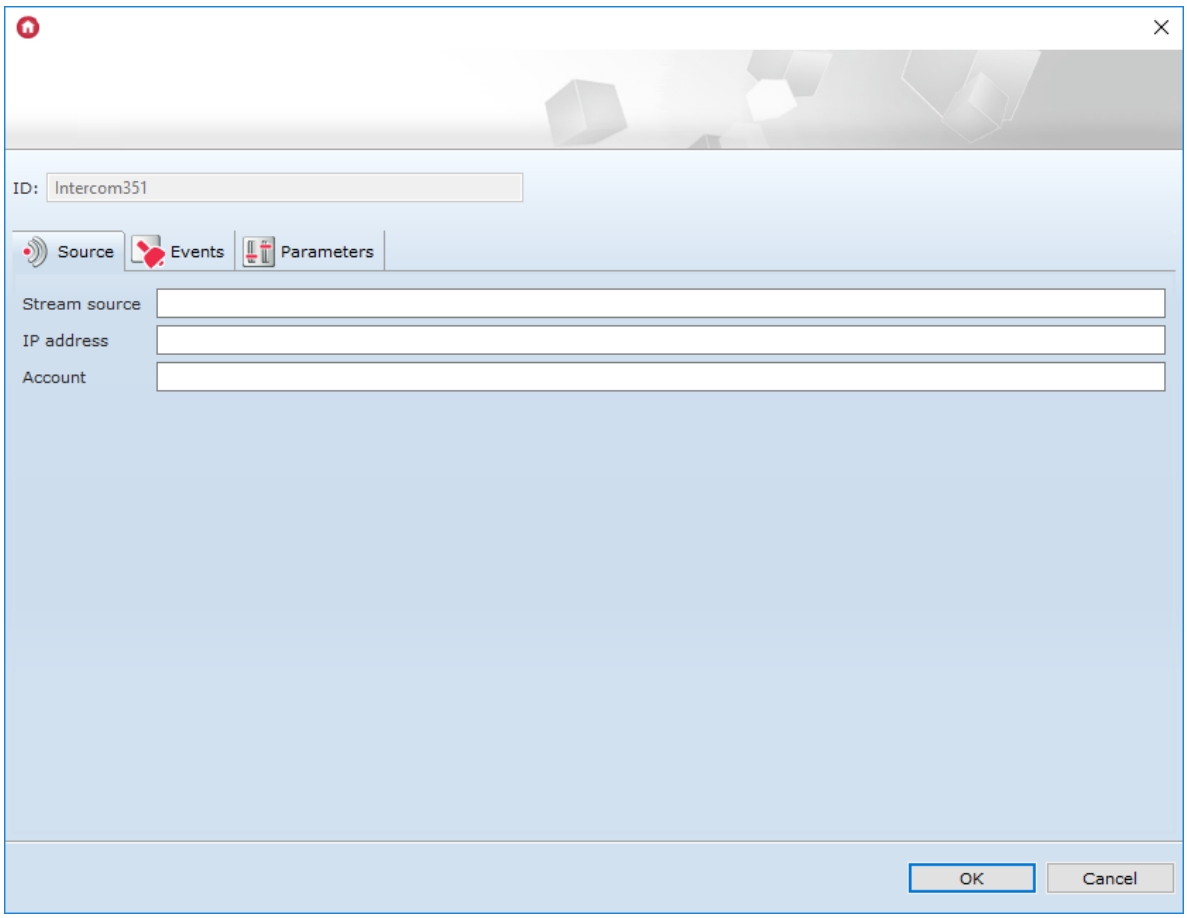
- From the main menu, click *Add interface*:



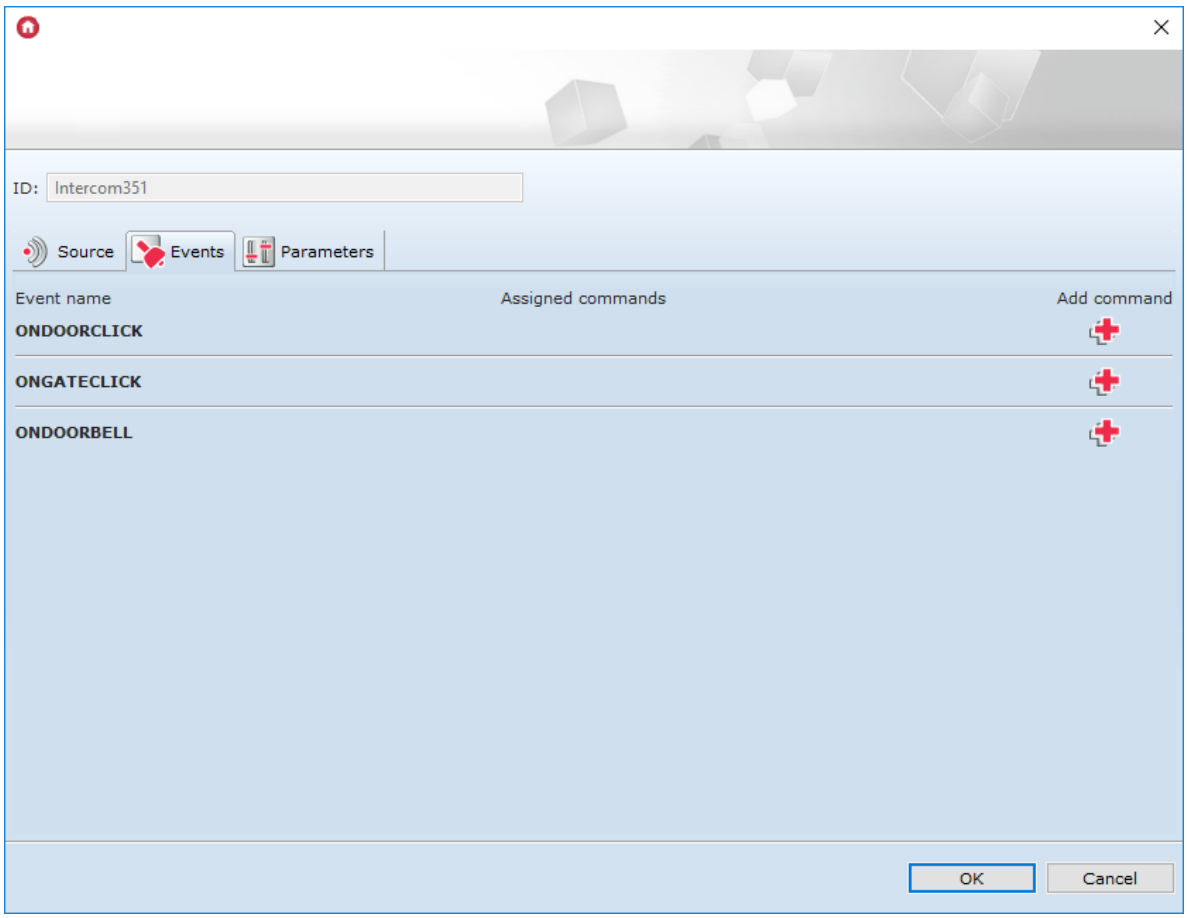
- Configure interface settings - choose: resolution, name, skin, add at least one page;
- To the created page - from the component palette - add the button *Intercom*:



- 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*"):



- Go to the tab *Events*:
  - To the `OnDoorClick` event assign a method to be called after pressing the wicket opening button in the door phone 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:

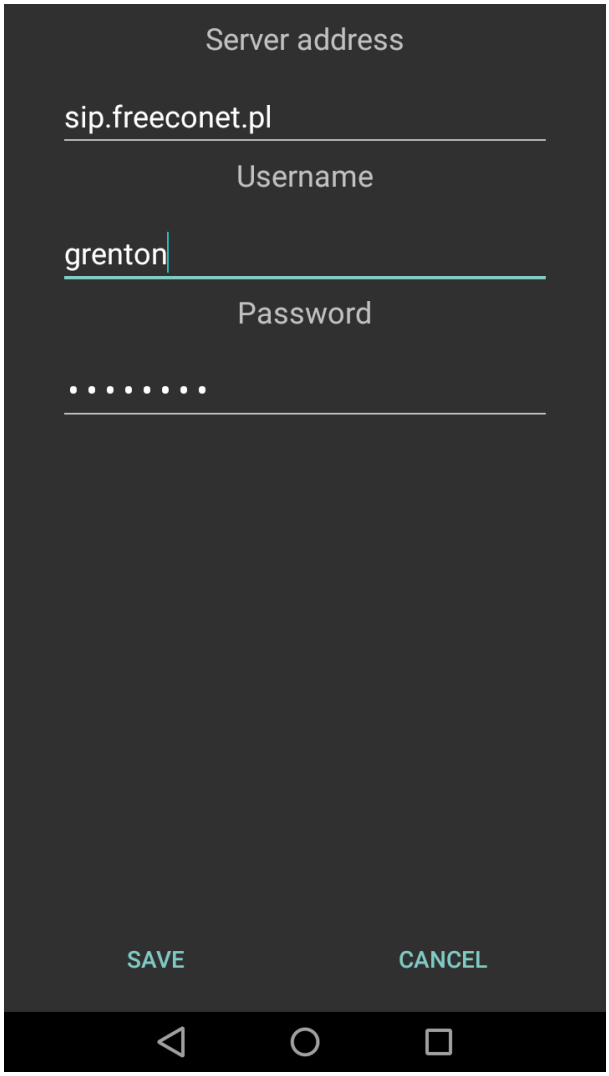


- Click OK;
- Send the interface to the mobile device - look up VIII.4.7.

B. Home Manager application configuration

In order to carry out the configuration:

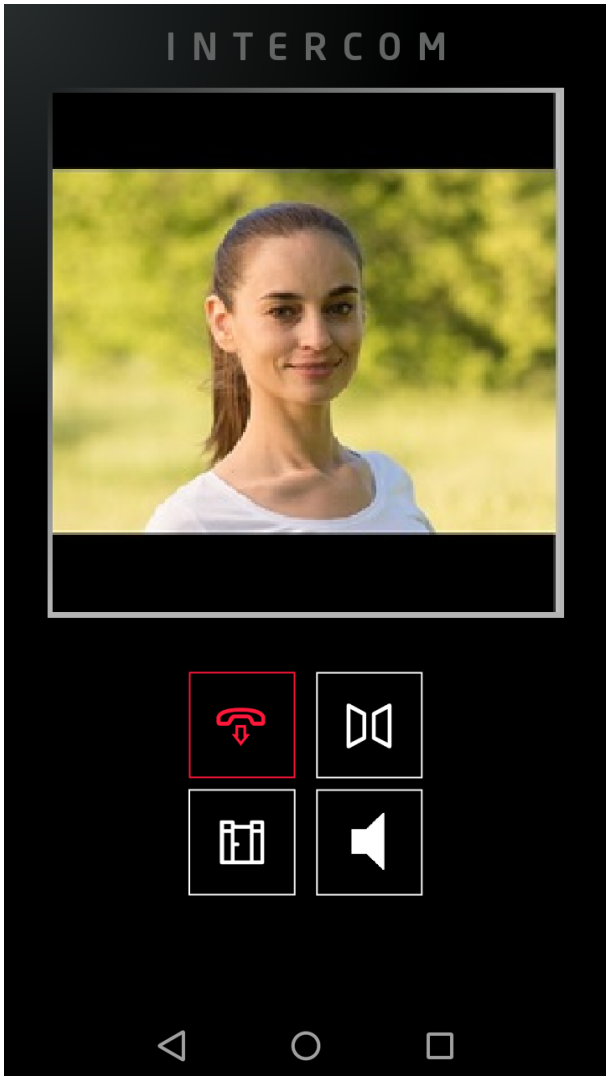
- Open the Home Manager application;
- Select *Settings* from the main menu (gearstick pictogram);
- From section *Intercom* select *SIP configuration* [5] ;
- In the settings specify:
  - **Server address** - server IP address *SIP* on which the accounts were created;
  - **User name** - number / account 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:



- 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 door phone panel there is also a button for switching on / off the hands-free mode.



7. IP cameras image operation

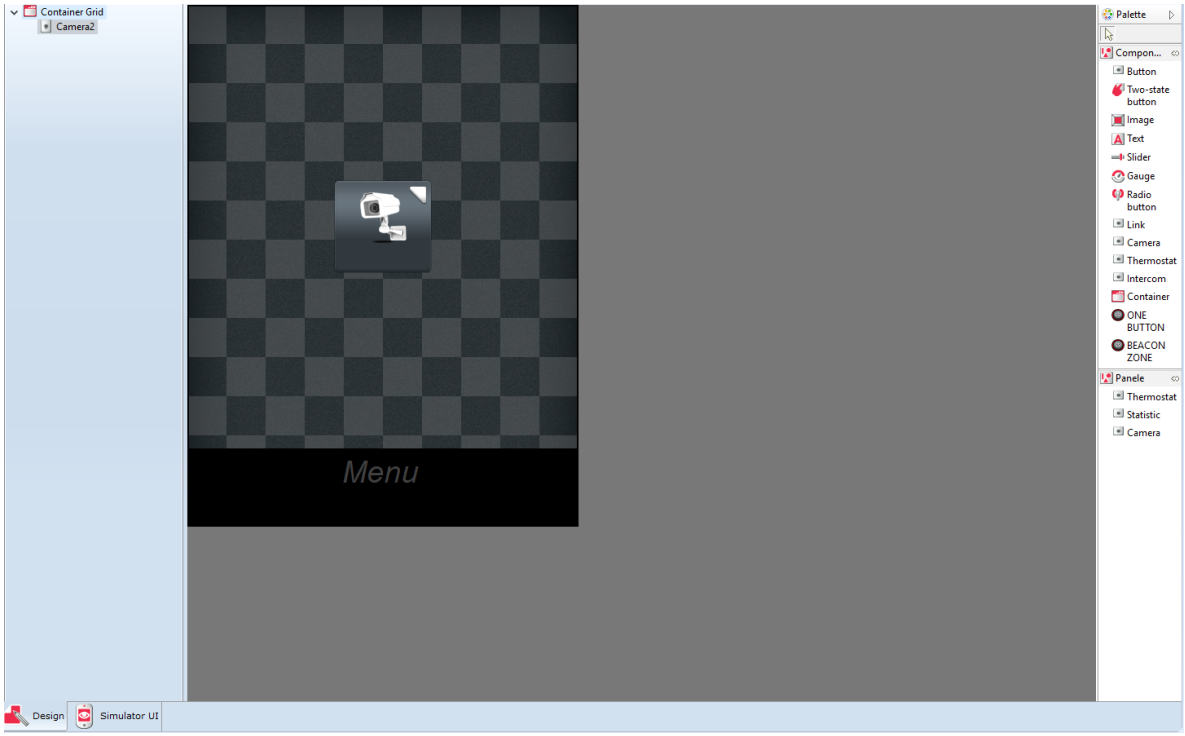
> **Note!**
> **Support for Visual Builder functionality has been ended in Object Manager version 1.9.0 and higher. The Home Manager interface wizard has been removed and it is impossible to open/edit interfaces created in the project. Saving your project using the current version of OM will result in the loss of any data associated with Visual Builder - including interfaces created in the project.**

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.

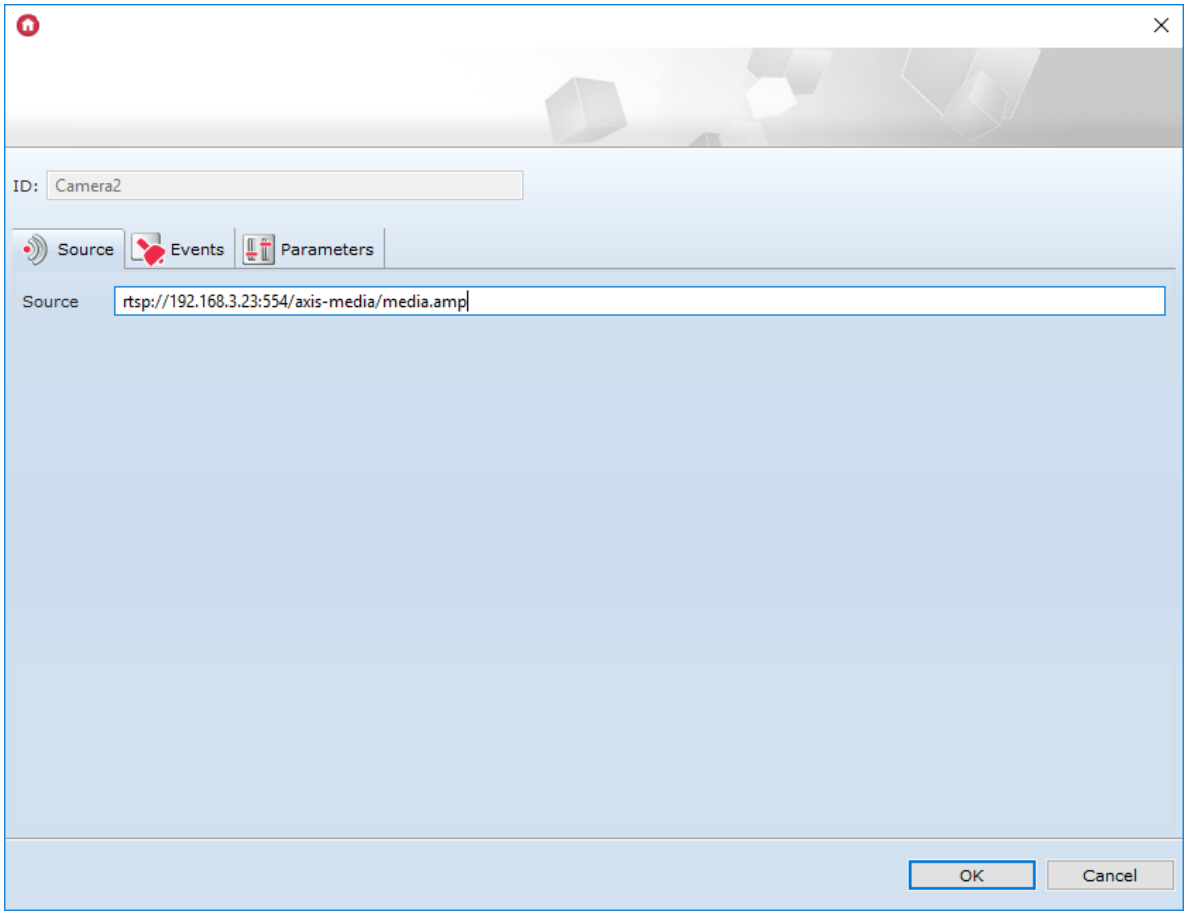
> **Note!**
> Home Manager application displays correctly images of cameras supporting RTSP protocol and h264 codec in MPEG standard.

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.

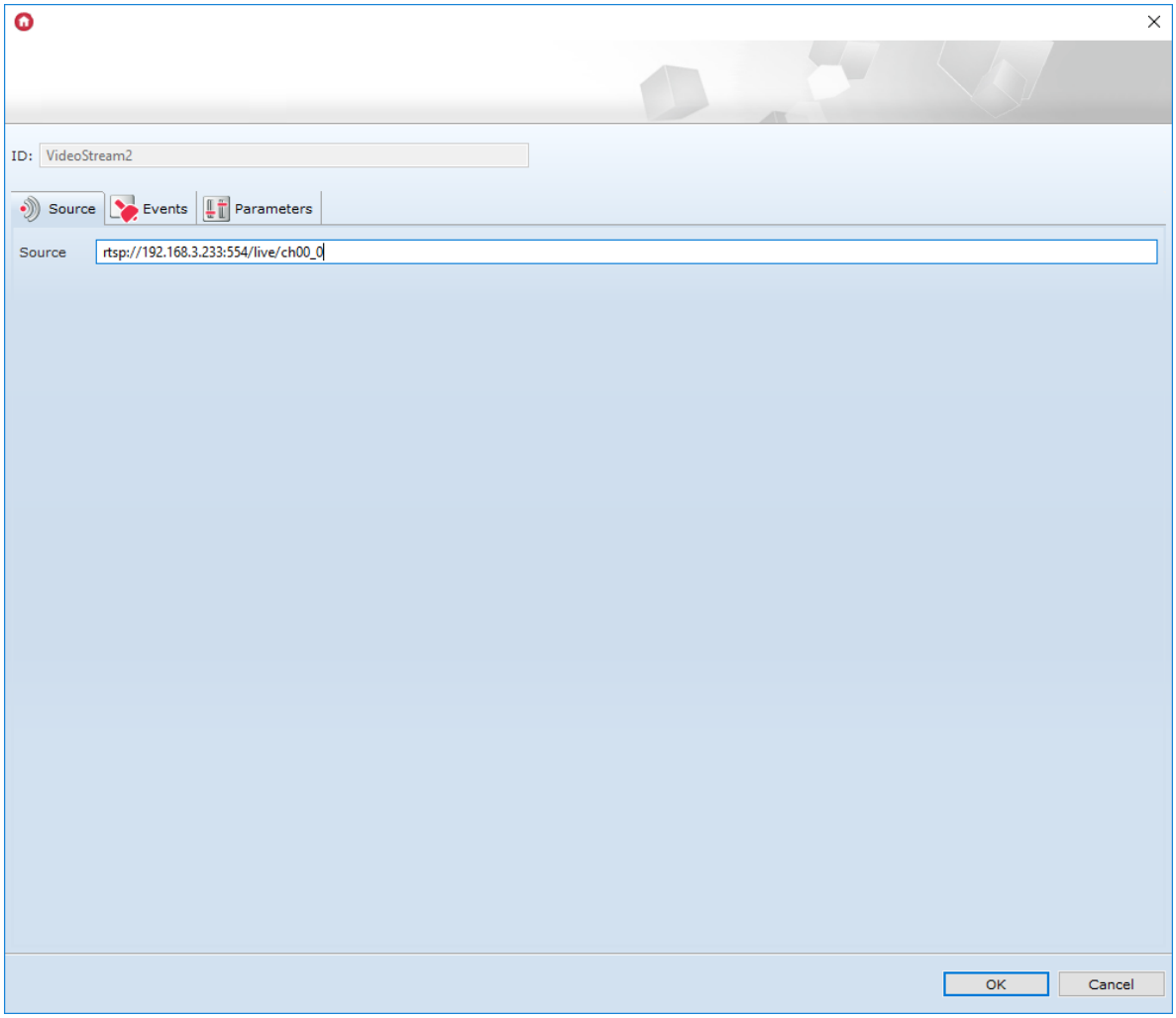


After sending created interface, image from the camera will be displayed on the screen of the mobile device after clicking added object.

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

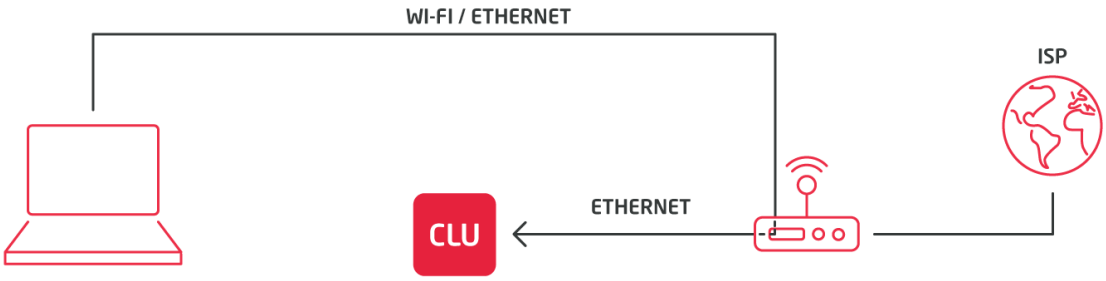
The Grenton system gives you the freedom to control your home from anywhere in the world. Sitting at work, or being on a business trip, we can easily control the state of our investment and manage its functions in a very simple way.

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.

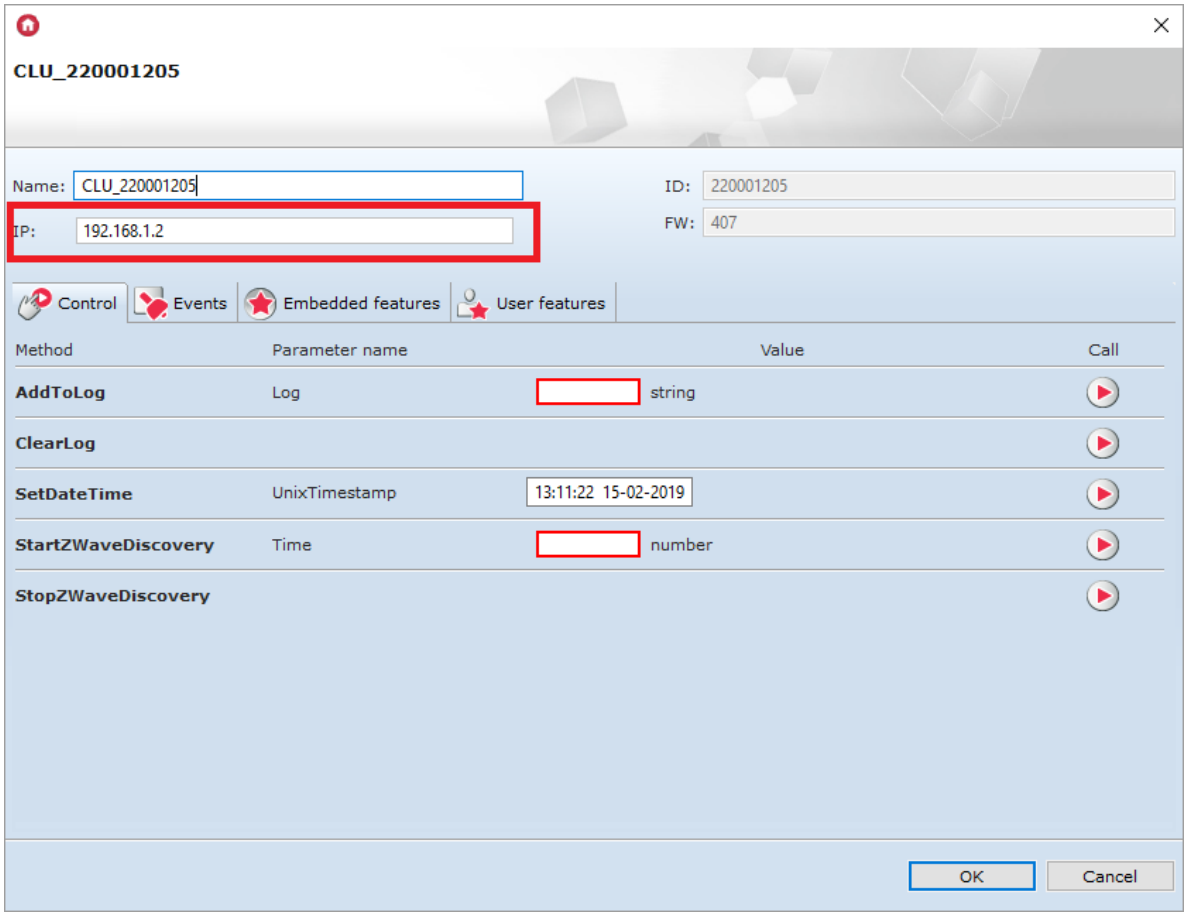
8.1. System configuration

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:



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.

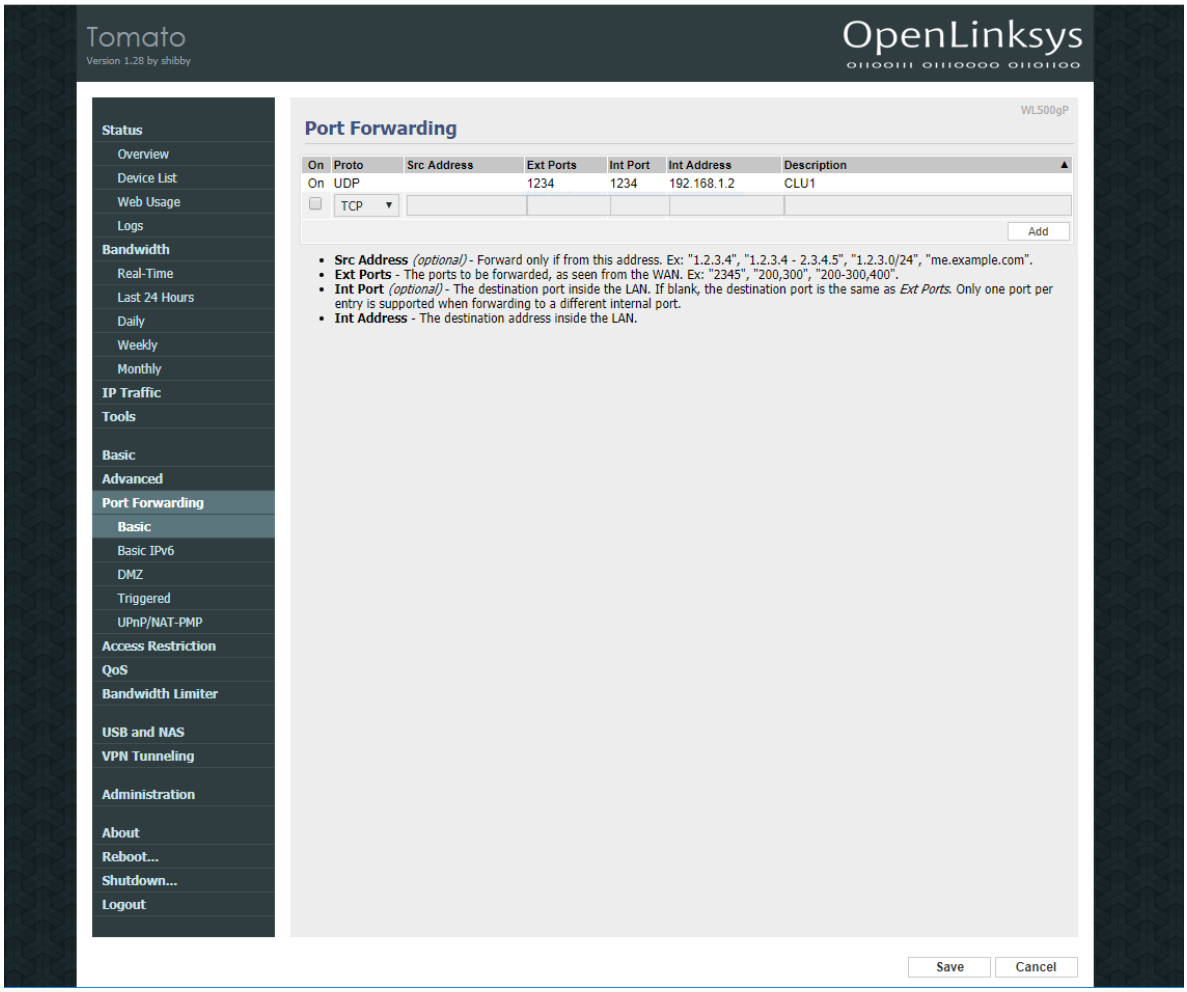
In order to set up port forwarding it is necessary to:

- 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:



- 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.

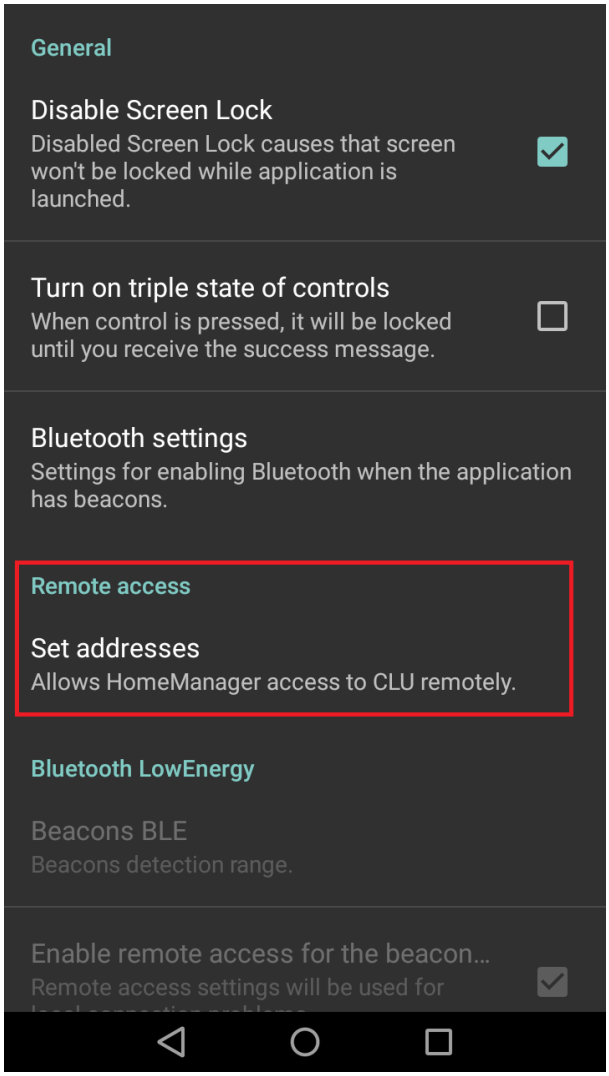
8.3. Configuration of the Home Manager mobile application

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;

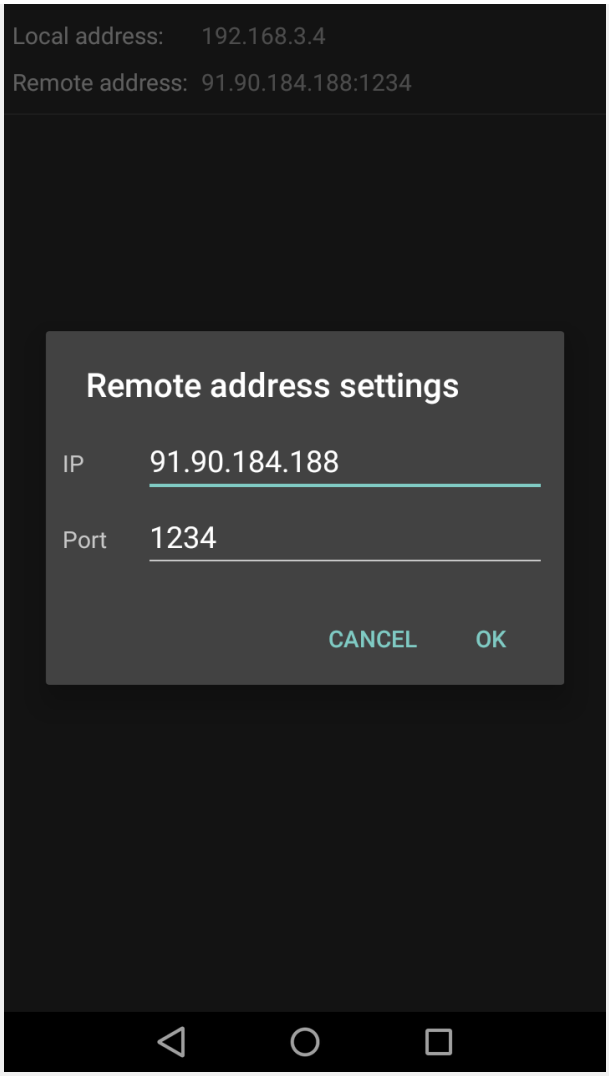
- go to the main screen of the mobile application and enter *Settings* (by clicking on the gear icon in the lower left corner of the screen):

- 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

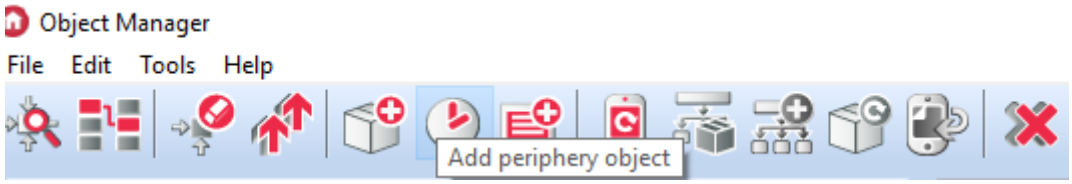
1. Timers

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

METHODS

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

EVENTS

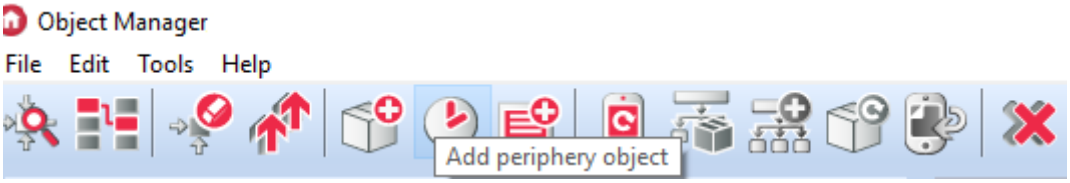
Name	Description
`OnTimer`	The event is called when the timer is counted
`OnStart`	The event is triggered when the timer starts
`OnStop`	Event triggered when the timer stops
`OnPause`	The event is called when the timer is paused

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.

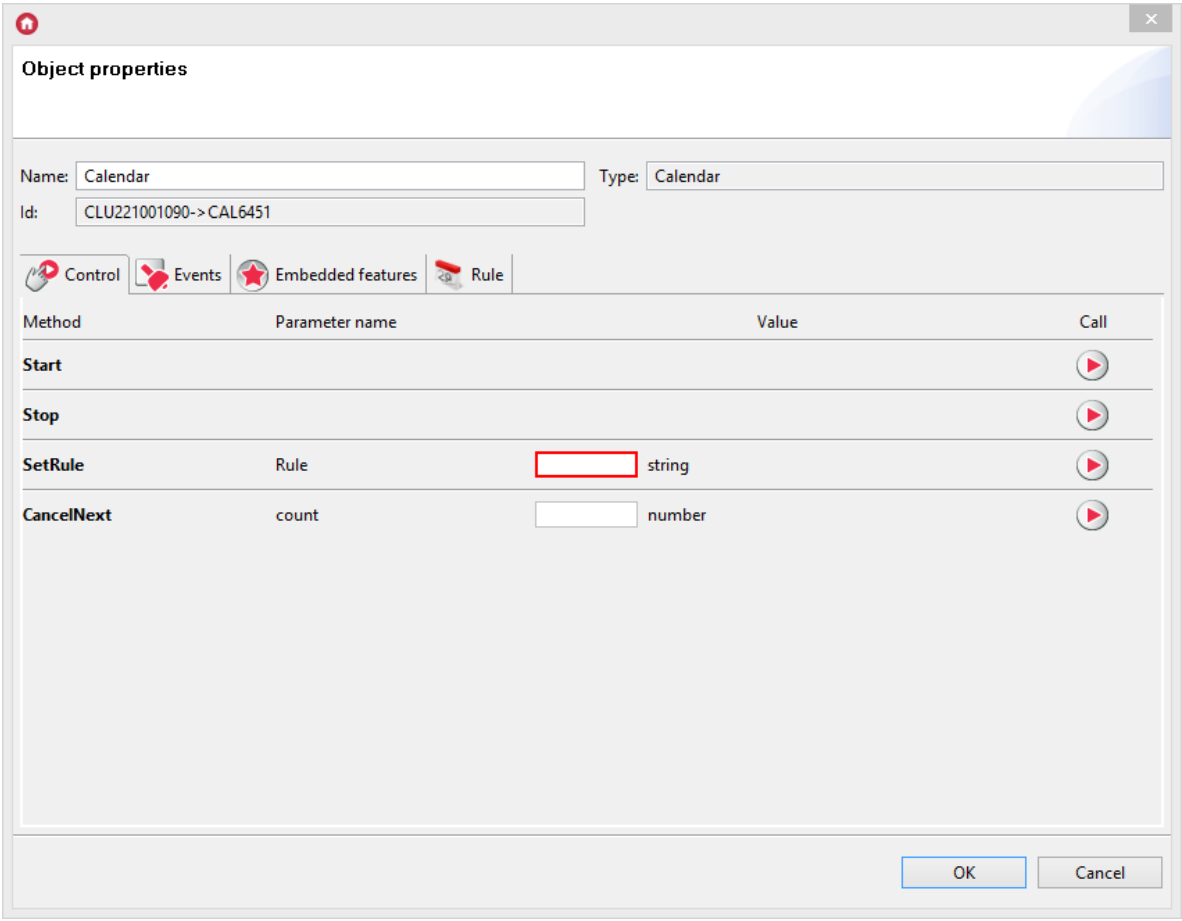


In the window that opens, select `Calendar`. After clicking `OK`, enter the name for the calendar you are creating. The Object Manager will display the properties window of the created object.

> **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

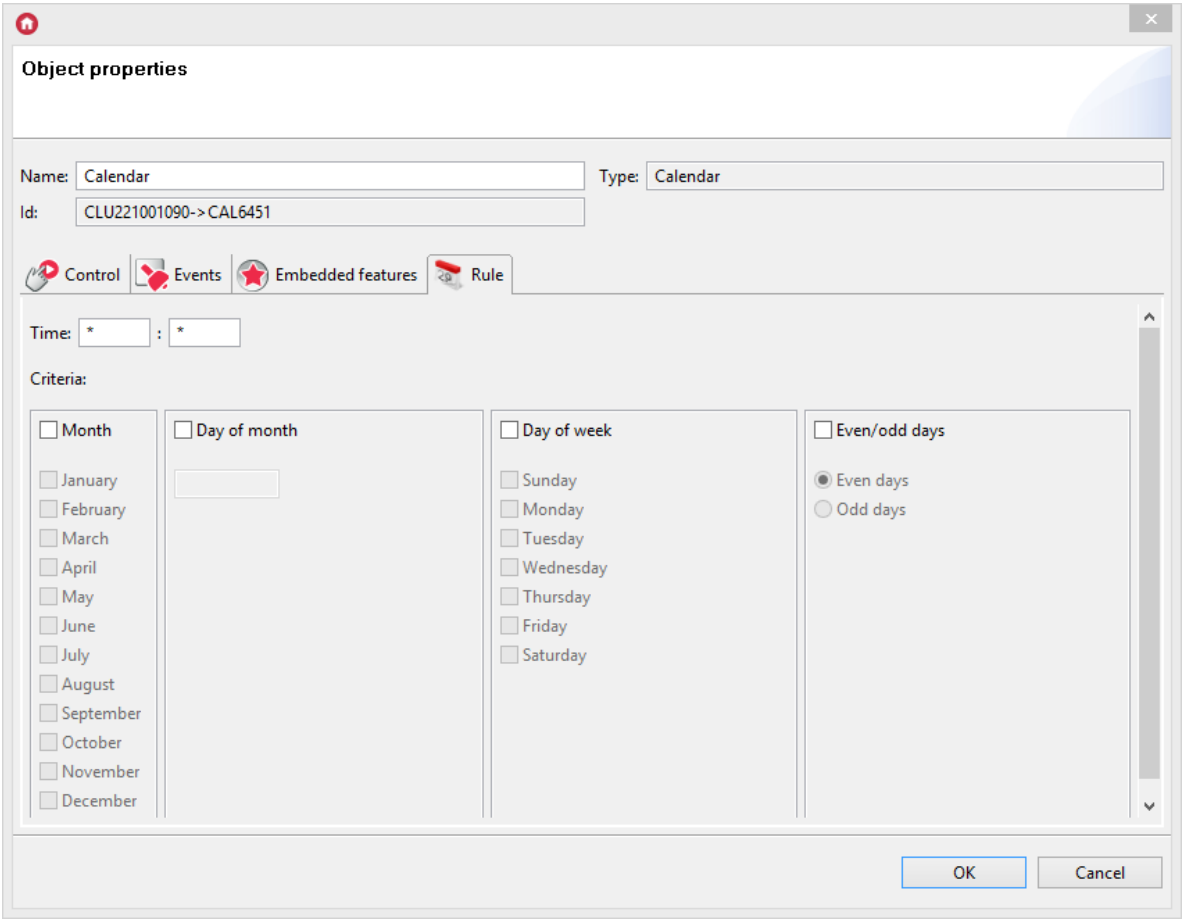
There are two ways of entering rules for the calendar:

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

METHODS

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

EVENTS

Name	Description
`OnCalendar`	Events since calendar action invocation
`OnStart`	Events since restarting calendar work
`OnStop`	Events since stopping calendar work
`OnCancel`	Events since cancelling the nearest actions

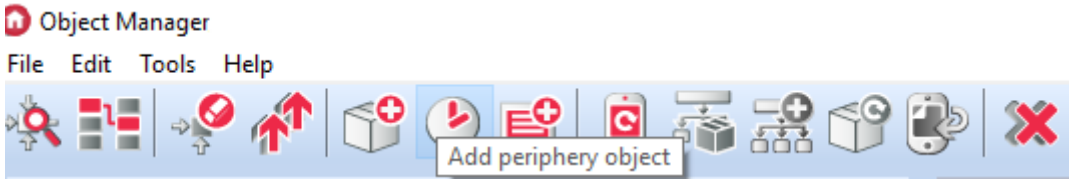
3. Schedule

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.

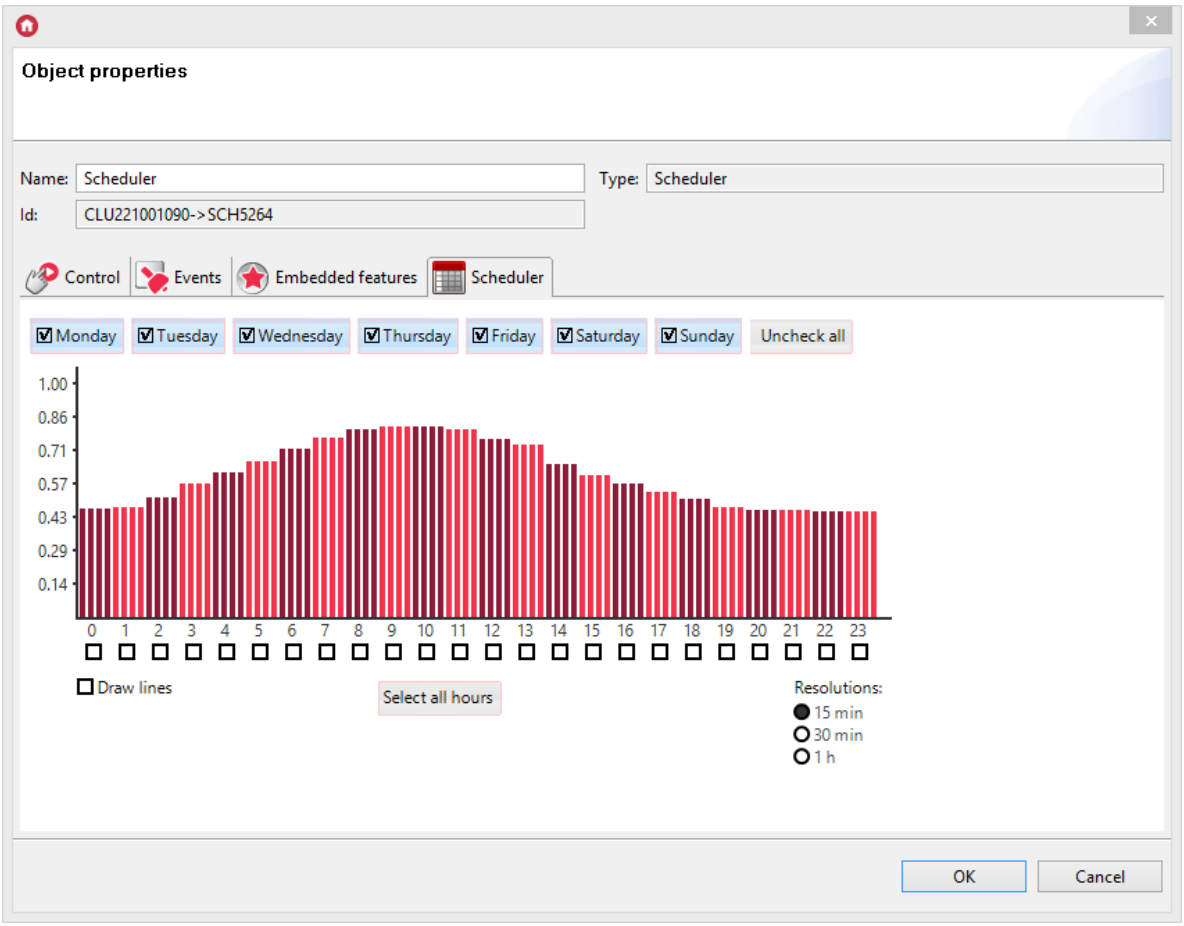


In the opened selection window, it is necessary to find and select the `Scheduler` object. After entering the name, the schedule properties window will open on the screen.
In this window there are four tabs:

- **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.

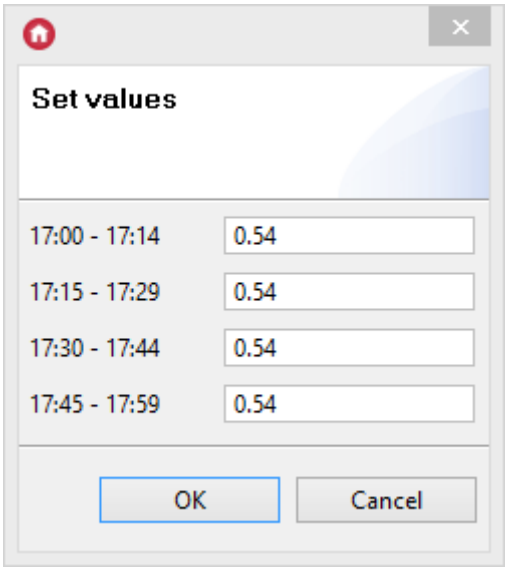
B. Setting values for the schedule

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



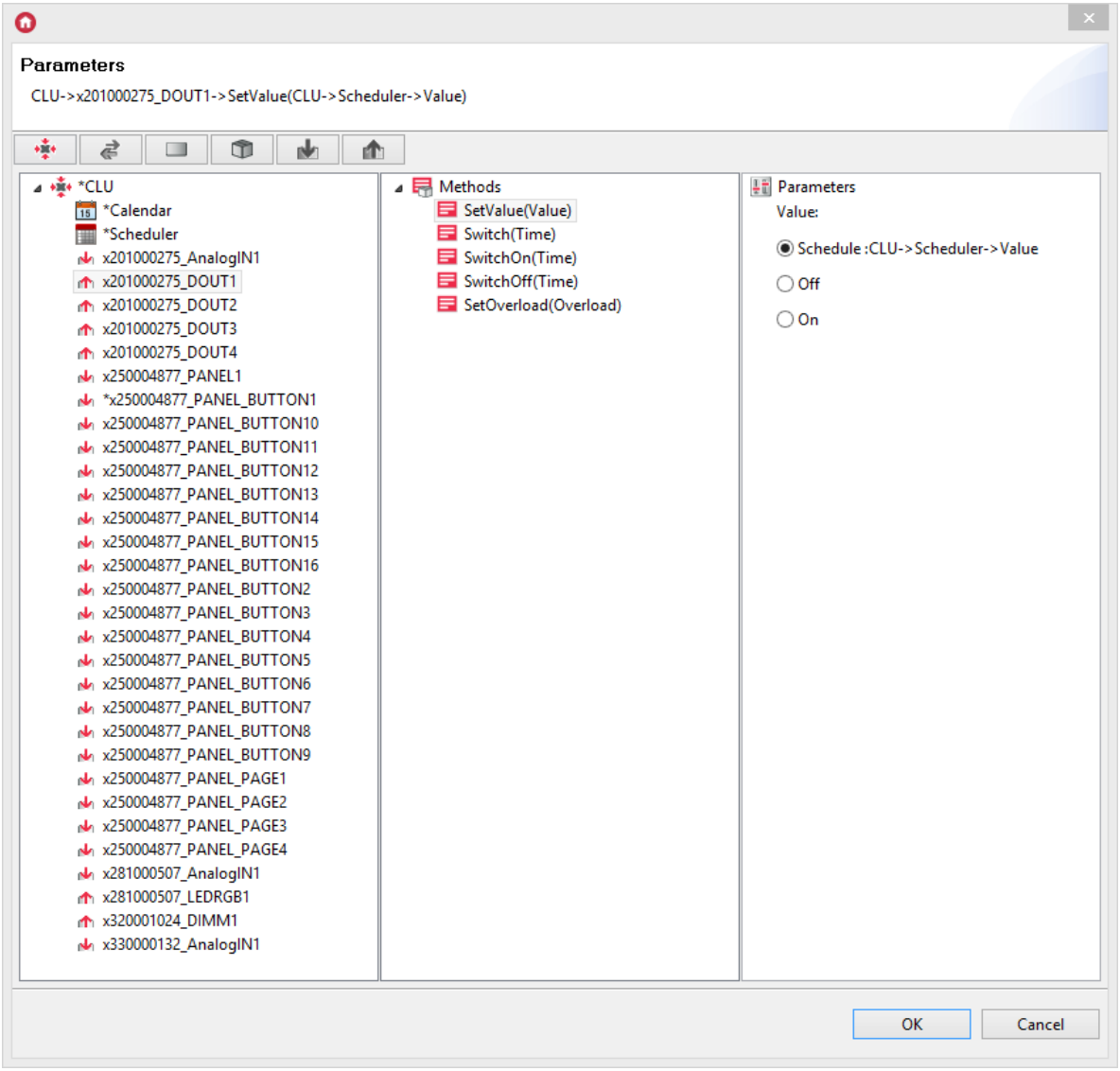
The schedule allows to enter values for 7 days (within one week) with 15-minutes frequency. You can set values for every day individually or for several days at once. Day of which the values are currently being set is discerned by black mark on the right of its name. Switching to another day happens 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
`Data`	String defining value changes schedule (see: Data format)
`State`	Schedule work status: 1 (active) or 0 (not active)
`Value`	Output value, changed every 15 minutes according to the schedule
`Min`	Minimum value for setting graphic interface value range
`Max`	Maximum value for setting graphic interface value range

METHODS

Name	Description
`Start`	Switching to active state (`State`=1)
`Stop`	Switching to paused state (`State`=0)
`SetData`	Setting weekly schedule

EVENTS

Name	Description
`OnHarmonogram`	Events since change of `Value` feature
`OnStart`	Events since restarting work
`OnStop`	Events since stopping work

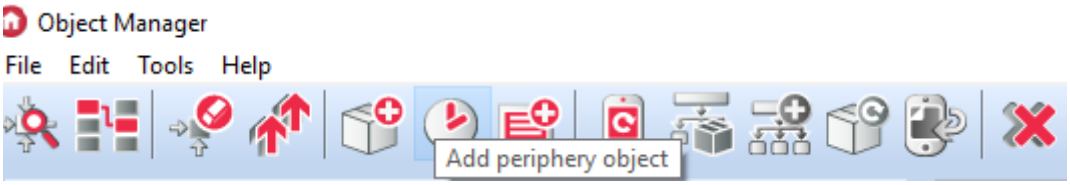
4. Thermostat

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`.

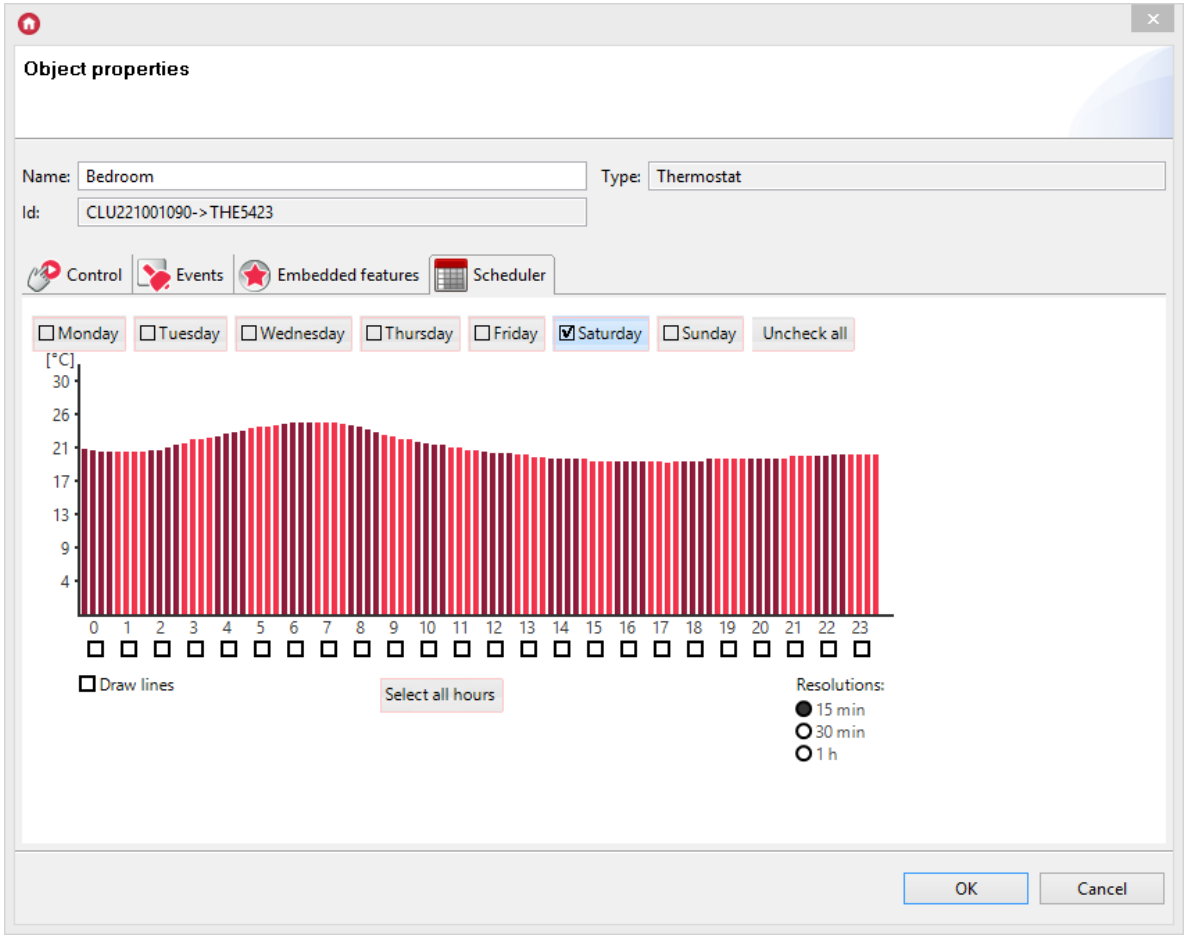


In the opened selection window, search for and select `Thermostat`. After entering the name, entering the source (which should be the temperature sensor responsible for a given heating zone) and selecting the receiver (which is the output to which the device responsible for a given heating zone is connected - 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:

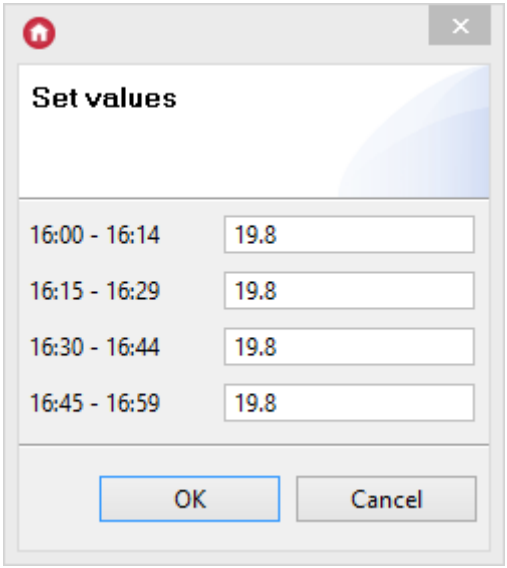
- **Control** - includes schedule methods;
- **Events** - contains schedule events;
- **Built-in features** - contains a list of schedule features;
- **Scheduler** - includes a graphical interface that allows you to easily formulate values for the entire scope of the schedule.

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 (`State` = 1)
`Stop`	Switching the thermostat to an inactive state (`State` = 0)
`IncreaseDegree`	Increase `PointValue` by 1 ° C
`DecreaseDegree`	Decrease `PointValue` by 1 ° C
`HeatUp`	Increasing `PointValue` by a given value at a specified time
`HolidayModeStart`	Starting `holiday mode`
`HolidayModeStop`	Stopping the `holiday mode`
`AutoModeStart`	Starting the `AutoMode` mode (downloading temperature from the schedule)
`AutoModeStop`	Stop the `AutoMode` mode
`SetData`	Setting the weekly schedule
`SetOutputType`	Output type setting (`Auto` - auto detection, `Digital` - digital output, `Analog` - 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 `PointValue` 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 `OutValue` is set to a value greater than zero
`OnOutOff`	An event that is dispatched when the value of `OutValue` is less than zero
`OnHolidayModeOn`	An event generated when starting holiday mode
`OnHolidayModeOff`	An event generated when the holiday mode is turned off

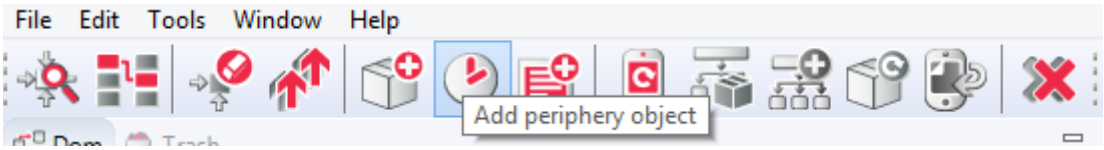
5. Push

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
`Message`	Notification content
`Title`	Notification title
`LastSendTime`	The time when the last notification was sent
`Interval`	Interval between successive notifications (in seconds)

METHODS

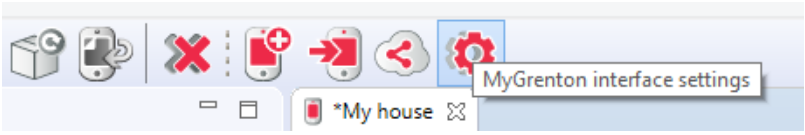
Name	Description
`SetMessage`	Setting the notification content
`ClearMessage`	Clearing the notification content
`SetTitle`	Setting the notification title
`ClearTitle`	Clearing the notification title
`Send`	Sending notification to device
`SetInterval`	Setting the interval between successive notifications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

EVENTS

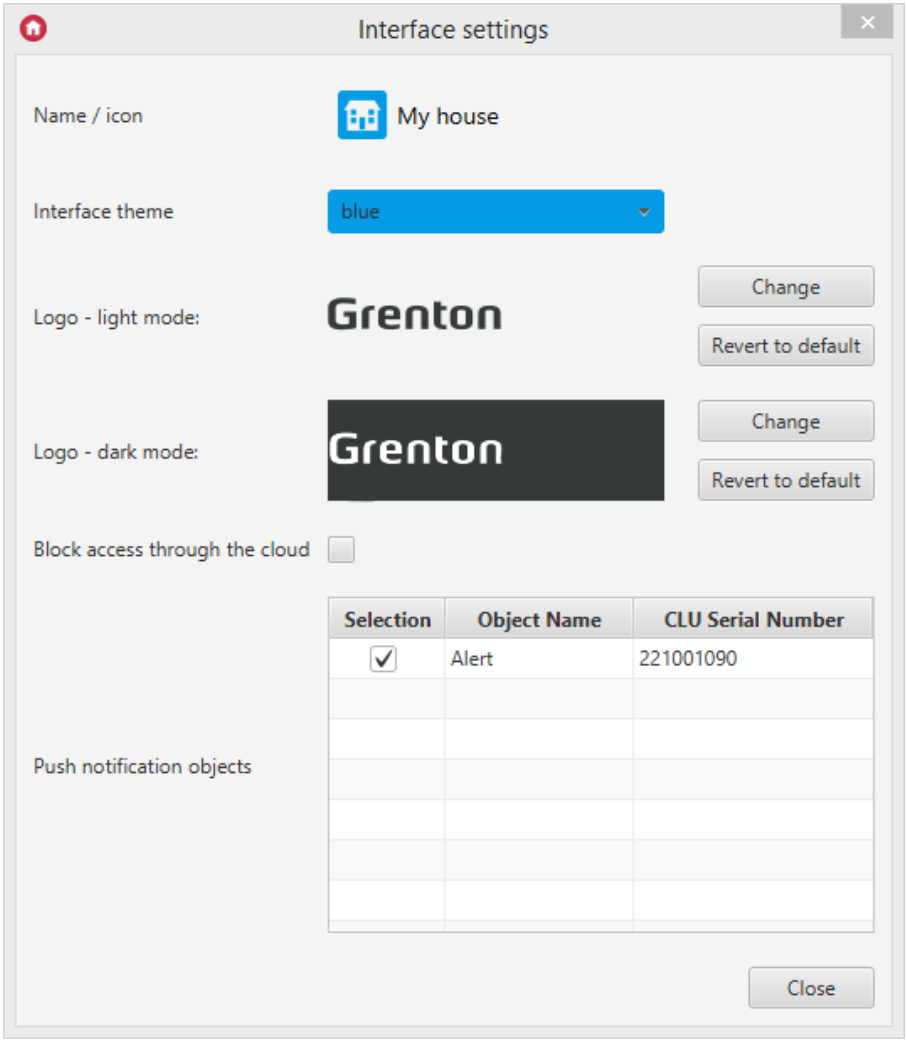
Name	Description
`OnSend`	An event generated when the message is sent
`OnOverflow`	An event generated when the queue overflows

C. MyGrenton configuration

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



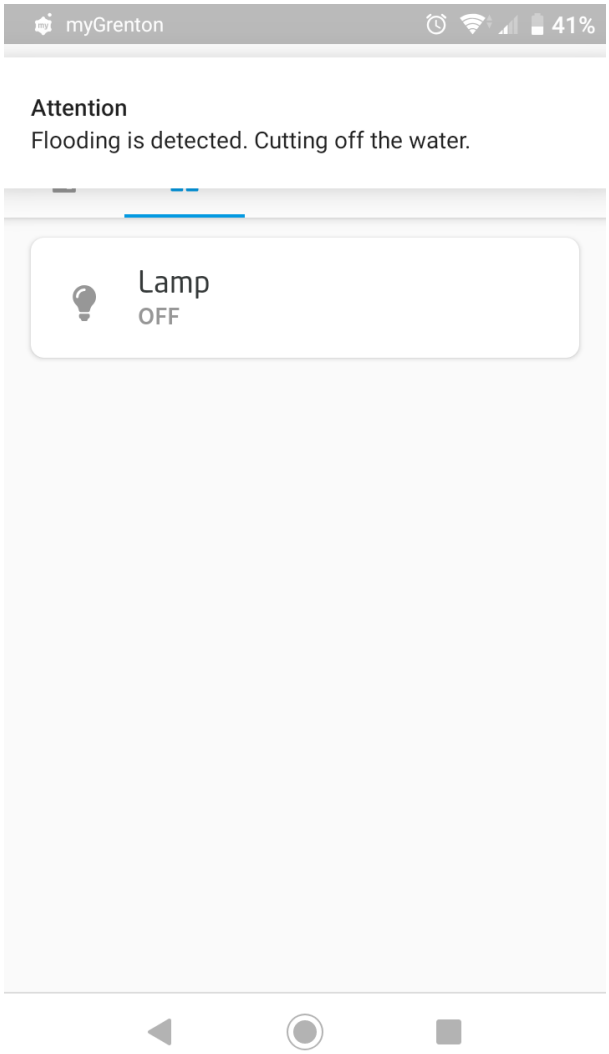
The window with interface settings will appear:



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.

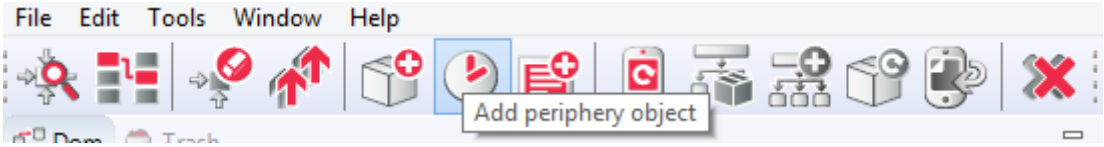
6. Presence Sensor

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!**
> To fully use the described functionality of the Presence Sensor object, you must have a CLUZ with firmware 5.10.01 or higher.

A. Presence sensor creation

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 `PresenceSenor` 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. Operation mode of presence sensor

The presence sensor can work in two modes depending on the motion sensor used:

- **Impulse input** - a mode, in which the `PresenceDetected` = 1 is maintained for a `Timeout` period - after this time, the `PresenceDetected` change to 0;
- **State input** - a mode, in which after the detection of movement by the sensor `PresenceDetected` = 1 is maintained until `UndetectedPresence` method is started - after its activation the high state of the `PresenceDetected` is maintained during the `Timeout`.

C. Presence sensor configuration parameters

FEATURES

Name	Description
`Timeout`	Time (in seconds) from last movement detection. When this time elapses PresenceDetected sets on 0
`State`	Current state of presence sensor: `1` - On `0` - Off
`PresenceDetected`	Movement detection state
`TimeFromLastPresence`	The time since the last motion detection (from the sensor or switching on the light from the button). Reset after calling: - DetectPresence() - regardless of Locked, DetectionDelay - SwitchLocked() when changing Locked to 1 - SetLocked(On) In state mode after calling DetectPresence(), TimeFromLastPresence = 0, until UndetectedPresence() is called
`DetectionDelay`	Time to ignore DetectPresence after changing Locked On->Off
`Locked`	Presence blocked status: `0` - reacting to DetectPresence `1` - maintain PresenceDetected as 1
`Mode`	Operation mode depending on the type of motion sensor used: `0` - impulse `1` - state

METHODS

Name	Description
`Start`	Start a presence sensor
`Stop`	Stop a presence sensor
`DetectPresence`	Method called while presence detection. Sets PresenceDetected value to `1` and resets TimeFromLastPresence timer
`UndetectedPresence`	Used in state mode (Mode = 1). Ends keeping the PresenceDetected parameter (after Timeout)
`SwitchLocked`	Changes the value of the Locked parameter to the opposite. Cases: - changing Locked from `0` to `1` - setting to 1 and blocking PresenceDetected, triggering the OnSwitchOn event (if PresenceDetected = 0 before), resetting TimeFromLastPresence - changing Locked from `1` to `0` - setting to 0 and unlocking PresenceDetected, triggering the OnSwitchOff event
`SetLocked`	Sets the value of the Locked parameter. Cases: - changing Locked from `0` to `1` - setting to 1 and blocking PresenceDetected, triggering the OnSwitchOn event (if PresenceDetected = 0 before), resetting TimeFromLastPresence - changing Locked from `1` to `0` - setting to 0 and unlocking PresenceDetected, triggering the OnSwitchOff event - SetLocked(On) if Locked = 1 - resets TimeFromLastPresence - SetLocked(Off) if Locked = 0 - no reaction
`SetTimeout`	Set a Timeout parameter (in seconds)
`SetDetectionDelay`	Sets the DetectionDelay parameter (in seconds)
`SetMode`	Sets the Mode parameter

EVENTS

Name	Description
`OnStart`	A event called when sensor starts
`OnStop`	A event called when sensor stops
`OnSwitchOn`	A event called if a presence detected (change of PresenceDetected parameter value from `0` to `1`)
`OnSwitchOff`	A event called when a timer timeouts (change of PresenceDetected parameter value from `1` to `0`)

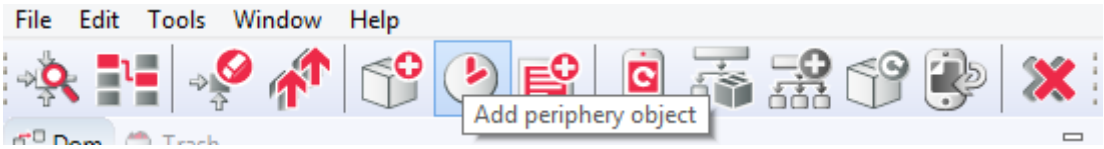
7. Sunrise and Sunset Calendar

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!**
> To fully use the described functionality of the Sunrise Sunset Calendar object, you must have a CLUZ with firmware 5.11.01 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
`Longitude`	Longitude in decimal degrees (DD), range -180 to 180
`Latitude`	Latitude in decimal degrees (DD), range -90 to 90
`State`	Current state of the sunrise and sunset calendar: 1 - On, 0 - Off
`SunriseUTC`	UTC sunrise time for a selected location (± 5 minutes) `N\A` - Unable to calculate sunrise time
`SunsetUTC`	UTC sunset time for a selected location (± 5 minutes) `N\A` - Unable to calculate sunset time
`SunriseLocal`	Sunrise local time for a selected location (± 5 minutes) `N\A` - Unable to calculate sunrise time for a selected location
`SunsetLocal`	Sunset local time for a selected location (± 5 minutes) `N\A` - Unable to calculate sunset time for a selected location
`SunriseUTCTimestamp`	UTC sunrise time for a selected location (± 300 seconds) `-1` - Unable to calculate sunrise time
`SunsetUTCTimestamp`	UTC sunset time for a selected location (± 300 seconds) `-1` - Unable to calculate sunset time
`SunriseLocalTimestamp`	Sunrise local time for a selected location (± 300 seconds) `-1` - Unable to calculate sunrise time for a selected location
`SunsetLocalTimestamp`	Sunset local time for a selected location (± 300 seconds) `-1` - Unable to calculate sunset time for a selected location"
`IsDayNow`	Determines the current part of the day based on local sunrise/sunset (± 5 minutes): `0` - nighttime, `1` - daytime, `-1` - Unable to calculate for a selected location
`SunriseOffset`	Offset for the sunrise (in minutes), range -1439 to 1439
`SunsetOffset`	Offset for the sunset (in minutes), range -1439 to 1439
`NextSunrise`	Time left until sunrise (in minutes) `-1` - Unable to calculate for a selected location
`NextSunset`	Time left until sunset (in minutes) `-1` - Unable to calculate for a selected location

METHODS

Name	Description
`Start`	Starts sunrise and sunset calendar
`Stop`	Stops sunrise and sunset calendar
`SetLongitude`	Set a longitude in decimal degrees (DD), range -180 to 180
`SetLatitude`	Set a latitude in decimal degrees (DD), range -90 to 90
`SetSunriseOffset`	Set the offset for the sunrise (in minutes), range -1439 to 1439
`SetSunsetOffset`	Set the offset for the sunset (in minutes), range -1439 to 1439

EVENTS

Name	Description
`OnStart`	A event called when calendar starts
`OnStop`	A event called when calendar stops
`OnSunrise`	A event called when the sun rises
`OnSunset`	A event called when the sun sets
`OnSunriseSunsetChange`	A event called when the sun rises or sets
`OnDay`	A event called when IsDayNow changes from 0 (nighttime) to 1 (daytime)
`OnNight`	A event called when IsDayNow changes from 1 (daytime) to 0 (nighttime)

8. Event Scheduler

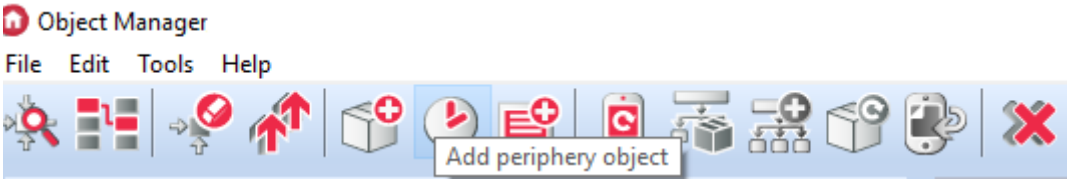
The event scheduler is a virtual object created within a given CLU. It is used to trigger a given event for a given hour and day of the week. In order to define the time and day of the week on which the event is to be executed, an appropriate rule should be added using the `AddRule` method. Rules are created using the syntax compliant with the CRON rules of the LINUX system.

> **Note!**
> To fully use the described functionality of the Event Scheduler, you must have a CLUZ with firmware 5.09.02 or higher.

To configure and manage rules it is recommended to use the `EVENT_SCHEDULER` widget from the level of myGrenton application - see section XVIII.3.18.

A. Create a event scheduler

In order to create a event scheduler, 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 `EventScheduler` object. After selecting, pressing `OK`, it is necessary to name the new event scheduler. The created event scheduler will appear on the list of objects of the selected CLU.

B. Event scheduler rules

There are two ways of entering rules for the event scheduler:

- Through the myGrenton application and the EVENT_SCHEDULER widget - a detailed description can be found in chapter see section XVIII.3.18.
- By entering CRON rule using `AddRule` method in the control tab - detailed information on this process can be found in CRON calendar documentation.

Note!
The Event Scheduler virtual object uses only the three parts of the CRON record "**minute hour** day month **week_day**". The day and month values are ignored, so you must enter `*` in place of these values. The target CRON syntax looks like this: `minute hour * * week_day`

C. Event scheduler configuration parameters

FEATURES

Name	Description
`RuleList`	List of all rules in the format `{{id, rule_state, "crone"},{id, rule_state, "crone"},...}` `rule_state:` 0 - rule disabled, 1 - rule enabled
`CurrentRule`	Rule from the list that is responsible for the current event `{id, rule_state, "crone"}` Returns a run rule for 1 minute, then "N\A" `rule_state`: 0 - rule disabled, 1 - rule enabled
`NextRule`	Rule from the list for the next event in `{id, rule_state, "crone"}` format `rule_state`: 0 - rule disabled, 1 - rule enabled
`RuleCount`	Number of rules added
`RuleAvailableCount`	The number of rules that can be added to an existing list (free space)
`State`	Current state of event scheduler

METHODS

Name	Description
`Start`	Start Event Scheduler
`Stop`	Stop EventScheduler
`AddRule`	Adds a rule to the list. Provide a crone rule `minute hour * * day_of_week`. Returns the id number of the assigned rule. When return 0 - error
`DeleteRule`	Removes the rule with the given id from the list. Returns 0 - ok, 1 - error
`GetRule`	Returns the rule in the format `{id, rule_state, "crone"}` for the given `rule_state`: 0 - rule disabled, 1 - rule enabled
`EnableRule`	Change state of rule to activated. Returns 0 - ok, 1 - error
`DisableRule`	Change state of rule to deactivated. Returns 0 - ok, 1 - error
`GetRules`	List of all rules in the format `{{id, rule_state, "crone"},{id, rule_state, "crone"},...}` `rule_state`: 0 - rule disabled, 1 - rule enabled
`GetNextRule`	Rule from the list for the next event in `{id, "crone"}` format

EVENTS

Name	Description
`OnStart`	A event called when Event Scheduler starts
`OnStop`	A event called when Event Scheduler stops
`OnEvent`	Target event triggered on the basis of the set rules and the current time of the device
`OnRuleAdd`	The event is raised when adding a rule to the list
`OnRuleDelete`	The event is raised when removing a rule from the list

9. MultiFanACThermostat

`MultiFanACThermostat` is a virtual object used for creating cooling or heating control configurations based on temperature sensor values and entered cooling or heating schedules on a weekly basis. It features additional functions such as managing a three-stage fan and the control output protection.

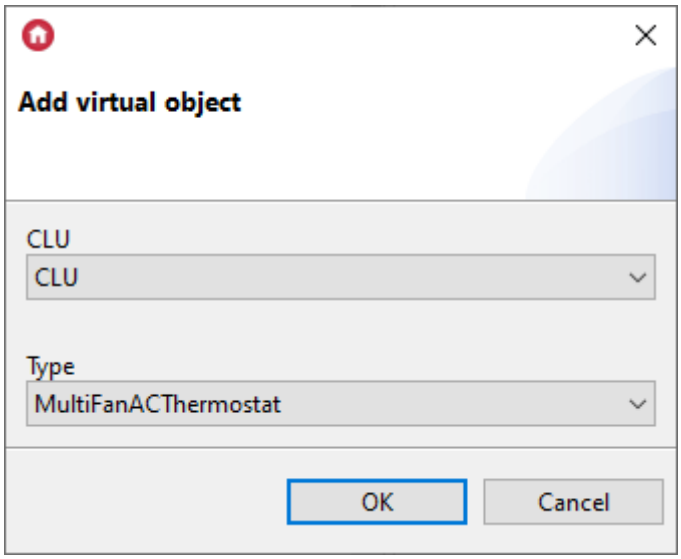
In one CLU, up to 64 objects can be created.

> **Note!**
>
> After creating the `MultiFanACThermostat` object (after sending a new configuration to the CLU), it becomes active automatically. To stop its operation, you need to invoke the `Stop` method.

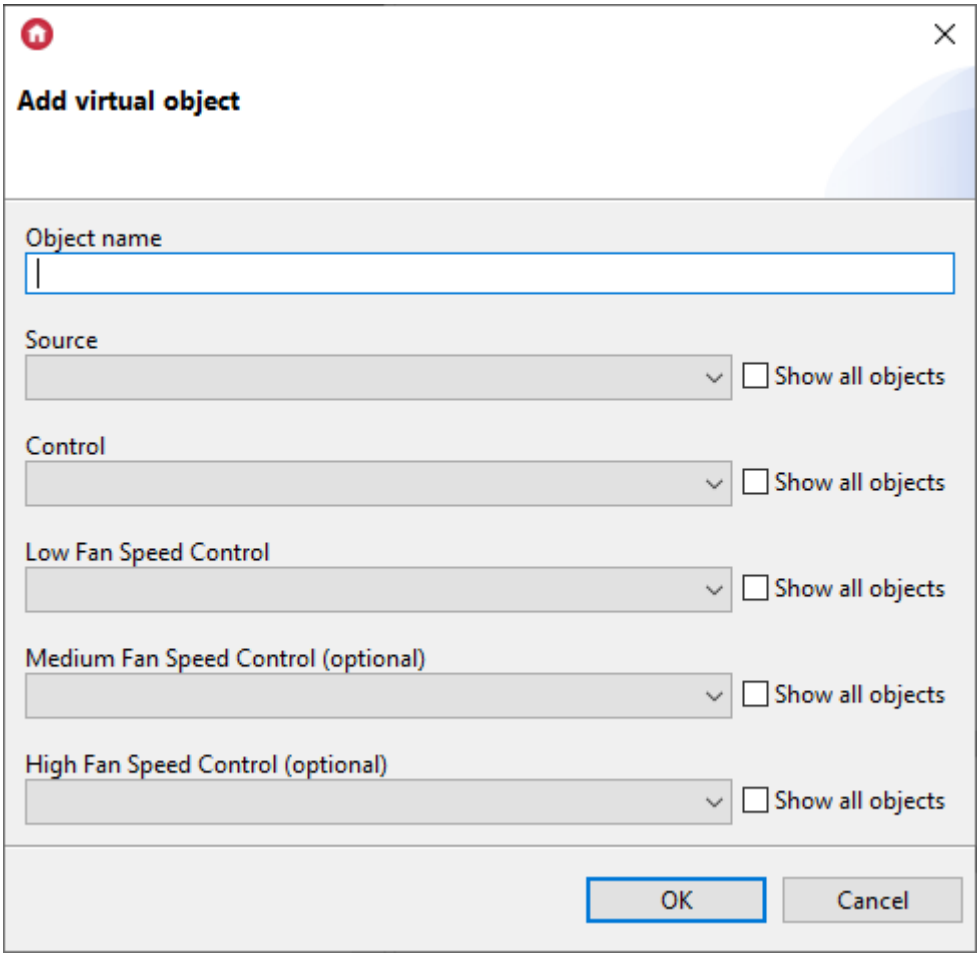
A. Creating object MultiFanACThermostat

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 opening selection window, find and select `MultiFanACThermostat`:



After confirming the window, the window for assigning a name and thermostat inputs and outputs will open:



Assign a name to the virtual object and specify the temperature source (e.g., OneWire sensor), control output (e.g., DOUT object), and fan stages - single, two, or three-stage depending on the device. Assigning medium and high-speed settings is optional.

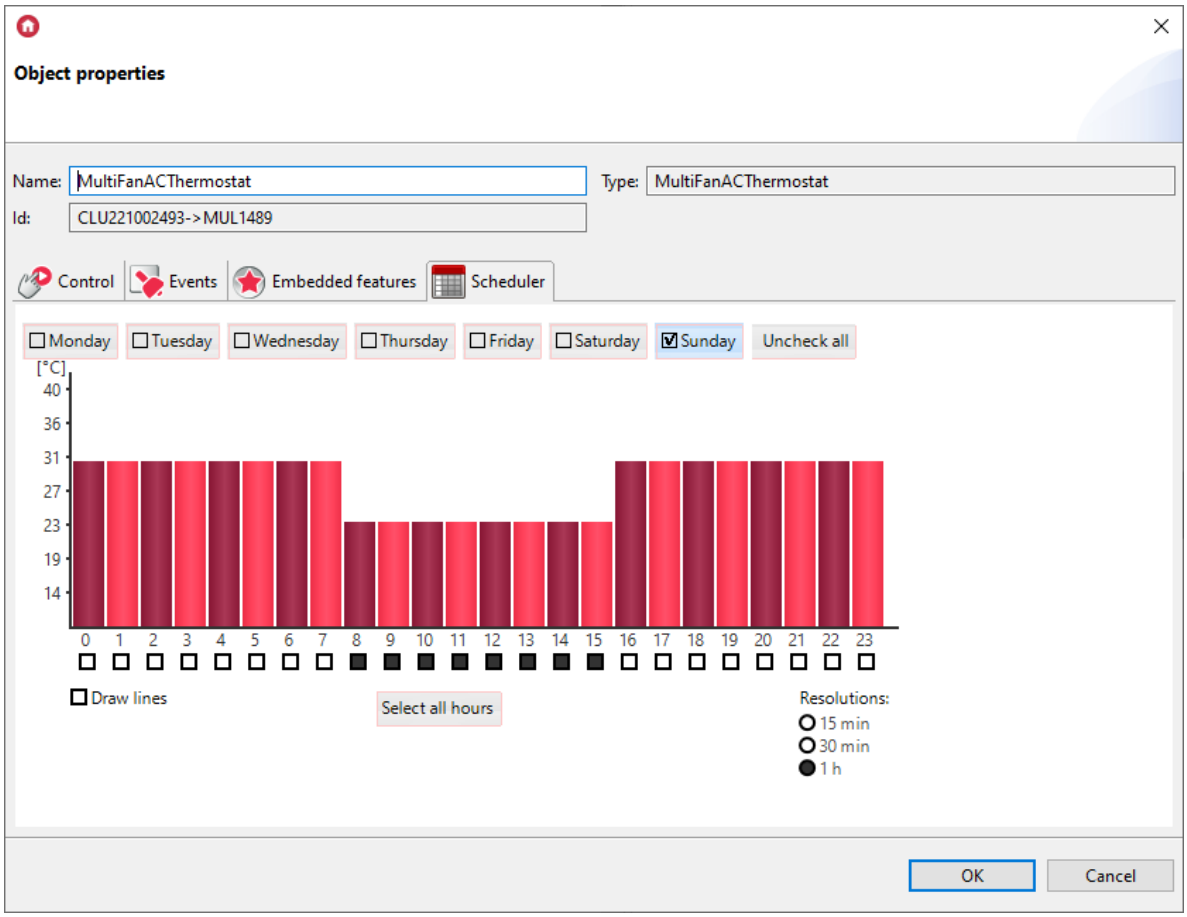
After clicking the `OK` button, the thermostat properties window will open.

In this window there are four tabs:

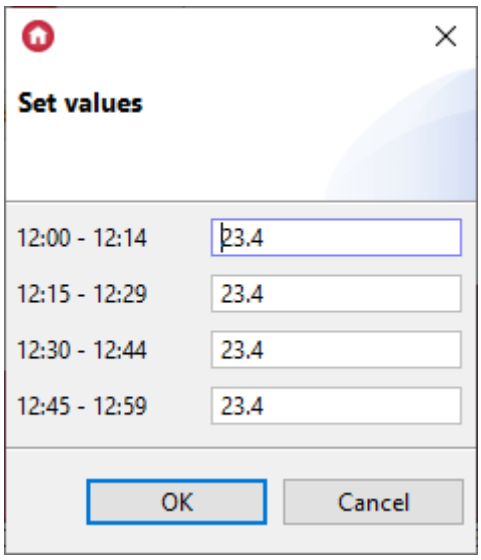
- **Control** - includes thermostat methods;
- **Events** - contains thermostat events;
- **Embedded features** - contains a list of thermostat features;
- **Scheduler** - includes a graphical interface that allows you to easily formulate values for the entire scope of the thermostat.

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 schedule can be edited directly in the myGrenton application.

C. Functionality of the virtual object MultiFanACThermostat

The virtual object `MultiFanACThermostat` has the following functions:

- `ControlSwitchDelay` - the possibility of defining a minimum time difference after which the control output is switched on, after the fan has been switched on. The fan switches on before the control output is switched on and switches off after a delay with respect to the control output;
- `ProtectionDelayOn` - the possibility of defining the time for which the control output output cannot be switched on after any change in the state of the control output output (`ControlOutValue`). After each reboot of the system, switching on can occur at the earliest after the `ProtectionDelayOn` time (compressor protection function);
- `ProtectionDelayOff` - the possibility of defining the time for which the control output output cannot be switched off after any change in the state of the control output output (`ControlOutValue`) (control output protection function).
- `MediumFanSpeedDelta` - specifies the temperature difference between the source temperature (`CurrentTemp`) and the target temperature (`TargetTemp`) to activate the second stage of the fan (`FanMediumControl`).
- `HighFanSpeedDelta` - specifies the temperature difference between the source temperature (`CurrentTemp`) and the target temperature (`TargetTemp`) to activate the third stage of the fan (`FanHighControl`).

> **Note!**
>
> The value of the `MediumFanSpeedDelta` feature must be less than the value of `HighFanSpeedDelta`.

The fan can be controlled by one of the four available `FanMode` modes:

- `FanMode` = 0 (Auto) - the fan switches on before the control output is switched on and switches off with a delay in relation to the control output according to the `ControlSwitchDelay` time;
- `FanMode` = 1 (Low) - if the thermostat is turned on (`State` = 1), the output `FanLowControl` of the fan is continuously activated regardless of the control output output.
- `FanMode` = 2 (Medium) - if the thermostat is on (`State` = 1), the output `FanMediumControl` of the fan is continuously activated regardless of the control output output.

- `FanMode` = 3 (High) - if the thermostat is turned on (`State` = 1), the output `FanHighControl` of the fan is continuously activated regardless of the control output output.

> **Note!**
>
> In the case of changing the state of the thermostat in mode `FanMode` = 1, 2, or 3, the `ControlSwitchDelay` time is not ignored.

D. Configuration parameters of the MultiFanACThermostat object

FEATURES

Name	Description
`Source`	Thermostat input, connection to a temperature sensor
`Control`	Thermostat output, thermostat linking to control element (Y)
`FanLowControl`	Thermostat fan output, connection with the fan (Gl)
`FanMediumControl`	Thermostat fan output, connection with the fan (Gm)
`FanHighControl`	Thermostat fan output, connection with the fan (Gh)
`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: 1 - cooling mode
`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 target temperature value depending on the set `Mode`
`CurrentTemp`	Returns the temperature values at the sensor
`ControlOutValue`	The value assigned to the control output
`FanControlOutValue`	The value assigned to the fan control output 1 - Off 2 - On (Low) 3 - On (Medium) 4 - On (High)

Name	Description
`FanMode`	Fan mode: 0 - automatic mode (automatic activation and speed adjustment), 1 - always-on mode (low), 2 - always-on mode (medium), 3 - always-on mode (high)
`ControlSwitchDelay`	Delay for turning the control output on or off in relation to the fan output
`ProtectionDelayOn`	The time for which the control output cannot be turned on after it has been turned off (compressor protection function)
`ProtectionDelayOff`	The time for which the control output cannot be turned off after it is turned on (compressor protection function)
`MediumFanSpeedDelta`	The difference between the source and target temperatures to activate the second stage of the fan
`HighFanSpeedDelta`	The difference between the source and target temperatures to activate the third stage of the fan

METHODS

Name	Description
`Start`	Switching thermostat to active state (`State` = 1)
`Stop`	Switching the thermostat to an inactive state (`State` = 0)
`IncreaseDegree`	Increase `PointValue` by 1 ° C
`DecreaseDegree`	Decrease `PointValue` by 1 ° C
`HeatUp`	Increasing `PointValue` by a given value at a specified time
`HolidayModeStart`	Starting `holiday mode`
`HolidayModeStop`	Stopping the `holiday mode`
`AutoModeStart`	Starting the `AutoMode` mode (downloading temperature from the schedule)
`AutoModeStop`	Stop the `AutoMode` mode
`SetData`	Setting the weekly schedule
`SetOutputType`	Output type setting: `Auto` - auto detection, `Digital` - digital output, `Analog` - analog output
`SetPointValue`	Setting the manually set temperature
`SetHolidayModeValue`	Setting the temperature value for the holiday mode
`SetHysteresis`	Setting the hysteresis value
`SetFanMode`	Sets the fan mode
`SetControlSwitchDelay`	Sets the delay for turning the control output on or off in relation to the fan output
`SetProtectionDelayOn`	Sets the time for which the control output cannot be turned on after it has been turned off (compressor protection function)
`SetProtectionDelayOff`	Sets the time for which the control output cannot be turned off after it is turned on (compressor protection function)
`SetMediumFanSpeedDelta`	Specifies the difference between the source and target temperatures to activate the second stage of the fan
`SetHighFanSpeedDelta`	Specifies the difference between the source and target temperatures to activate the third stage of the fan

EVENTS

Name	Description
`OnPointValueChange`	An event generated when the value of the `PointValue` property is changed
`OnStart`	An event generated when the thermostat is restarted
`OnStop`	An event generated when the thermostat stops working
`OnControlOutValueOn`	An event generated when output `ControlOutValue` is switched on
`OnControlOutValueOff`	An event generated when output `ControlOutValue` is switched off
`OnHolidayModeOn`	An event generated when starting holiday mode
`OnHolidayModeOff`	An event generated when the holiday mode is turned off

X. Media measurement

Important information - end of support for the Virtual Media measurement functionality

> **Note!**
> **Support for the virtual media measurement functionality has been ended in Object Manager version 1.9.0 and higher. Assumes Statistics (in the features of built-in objects) and options related to virtual measurement have been removed.**

> **Note!**
> Using virtual media measurement is only possible for Object Manager version 1.8.1 or lower.

1. Virtual media measurement

1.1. Launching media measurement on the Object Manager page

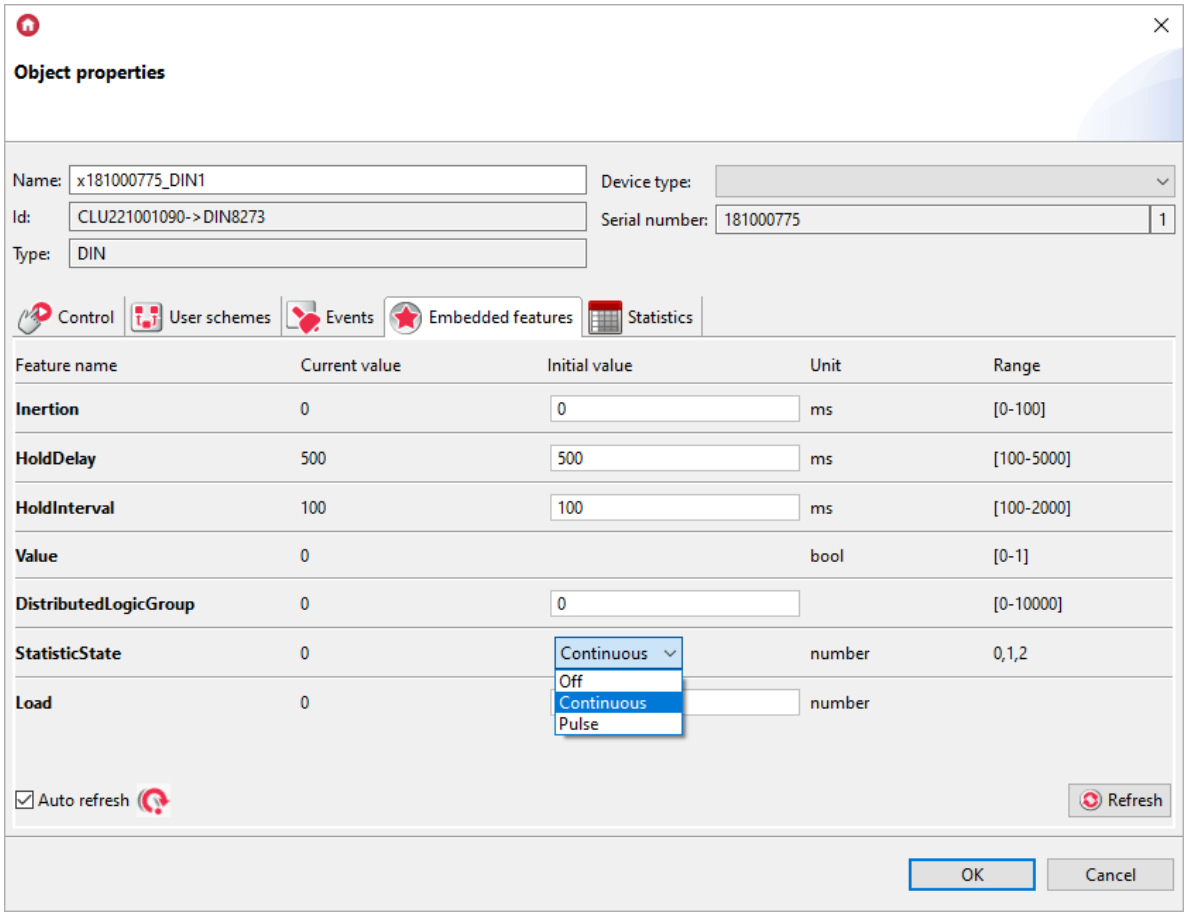
The Object Manager allows you to perform a media measurement, which allows the estimated presentation of the energy consumed (based on the time the device is turned on and the receiver power specified in the configuration). The media measurement configuration takes place in OM and it must be run for 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.

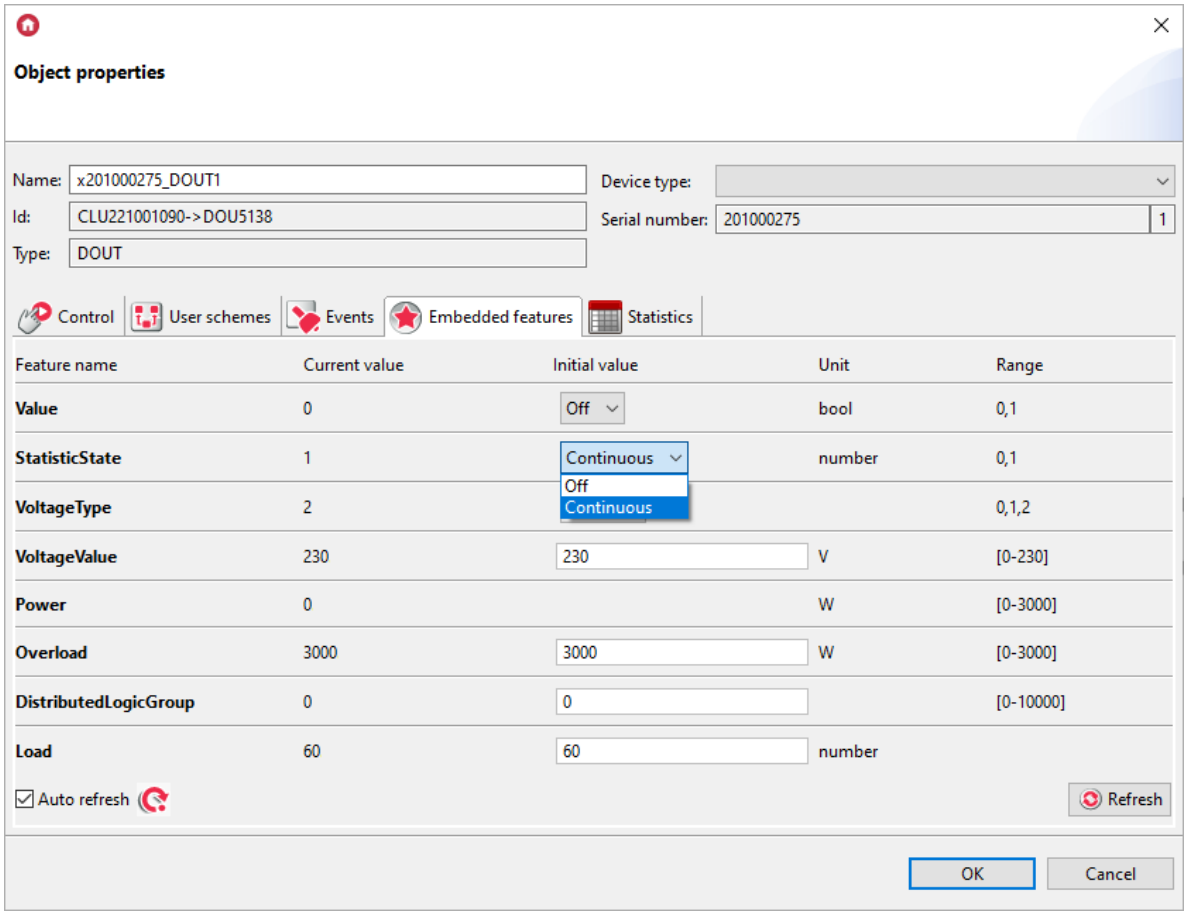
> **Note!**
> **Support for the virtual media measurement functionality has been ended in Object Manager version 1.9.0 and higher. Assumes Statistics (in the features of built-in objects) and options related to virtual measurement have been removed.**

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):



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



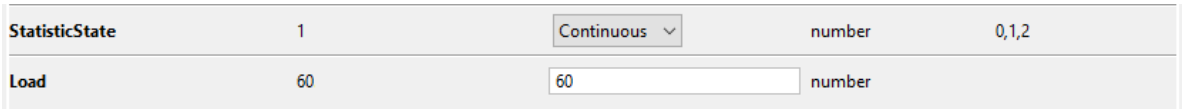
Note!

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!

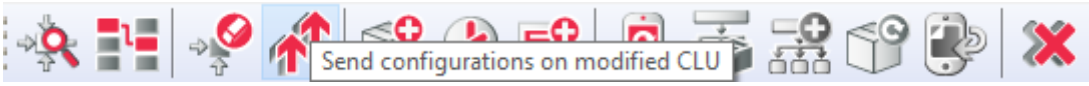
A. Creating a configuration

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):



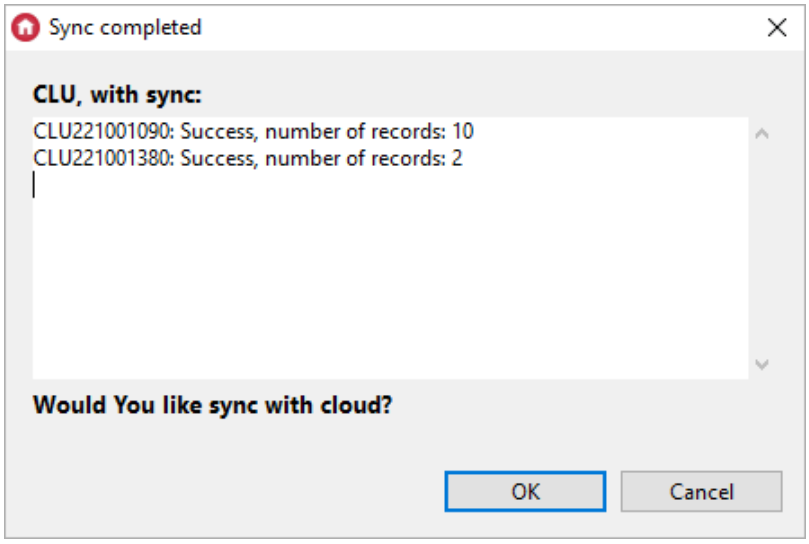
- Confirm with *OK*;
- Add media measurement settings for subsequent modules - repeat the above steps;
- Send the configuration to the CLU.



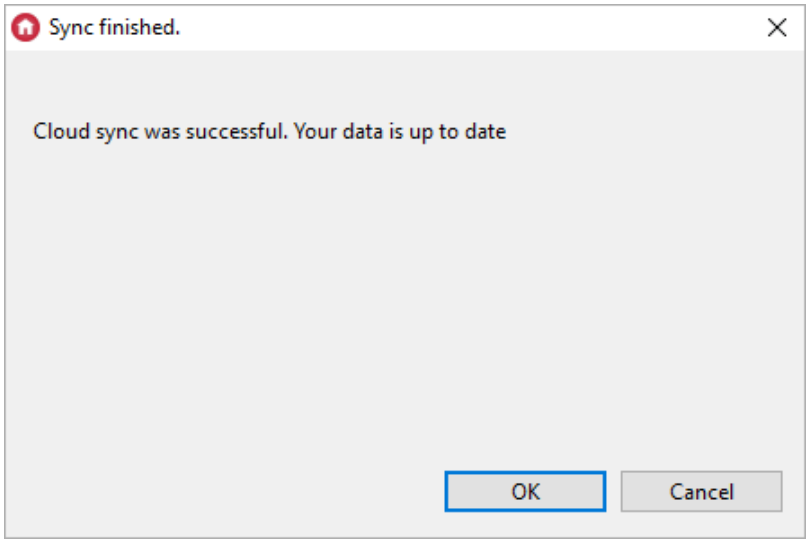
B. Read media measurement in the Object Manager

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:

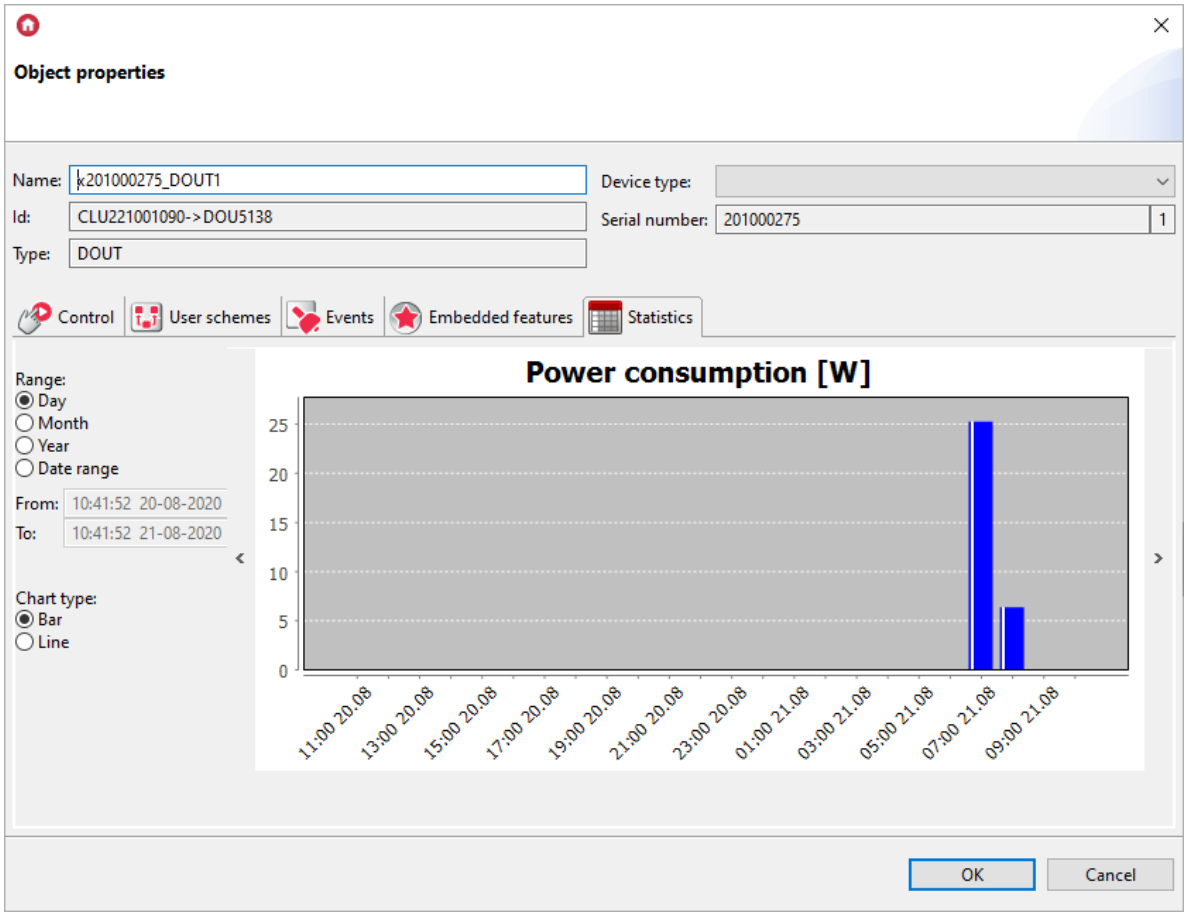


- Click *OK*;
- The Object Manager will then synchronize the downloaded data with the cloud;
- After the synchronization is completed, press *OK*:



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.



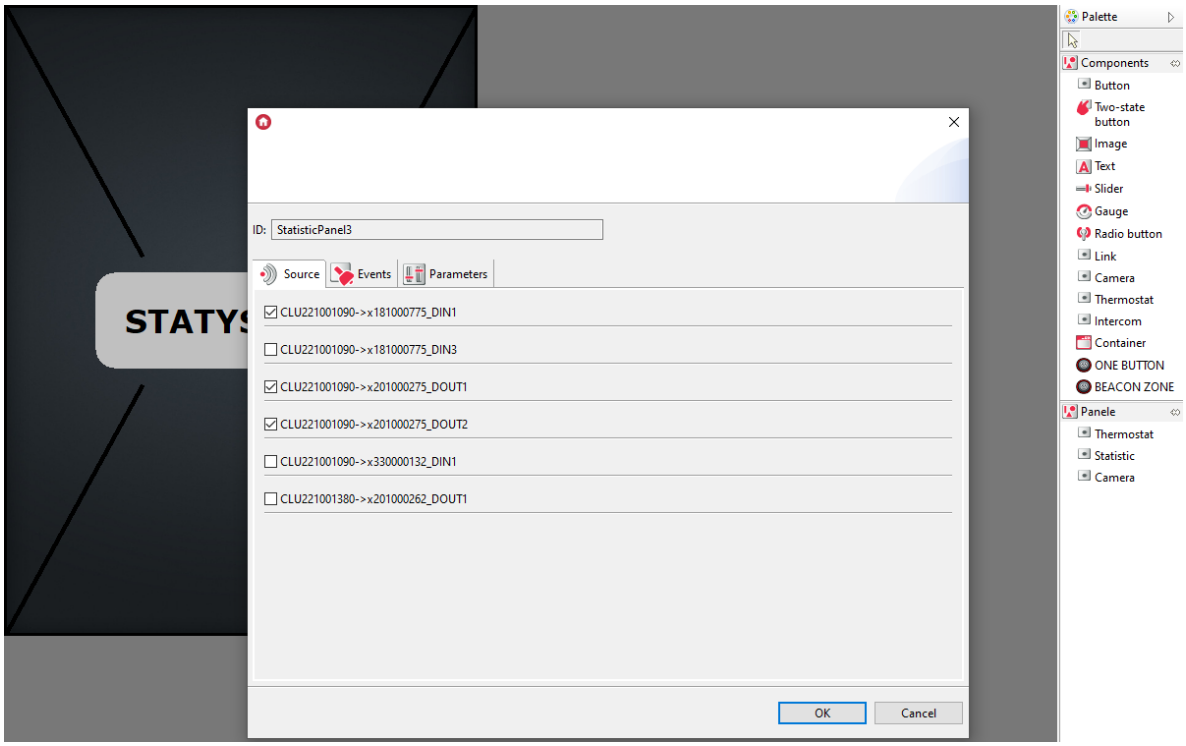
C. Configure the media measurement for the Home Manager application interface

Note!
Support for Visual Builder functionality has been ended in Object Manager version 1.9.0 and higher. The Home Manager interface wizard has been removed and it is impossible to open/edit interfaces created in the project. Saving your project using the current version of OM will result in the loss of any data associated with Visual Builder - including interfaces created in the project.

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.

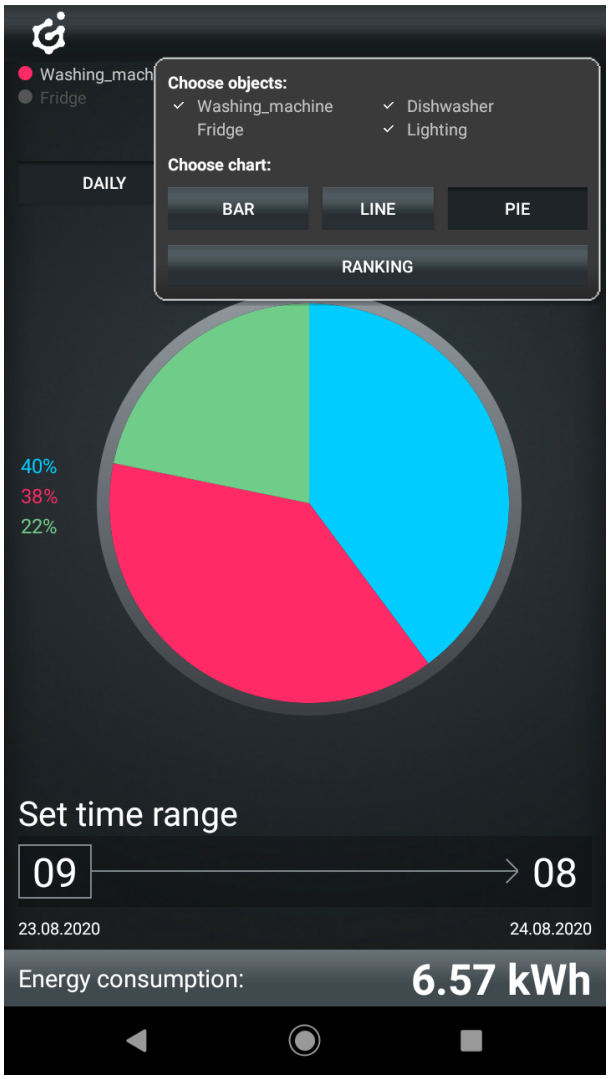
To properly use the media measurement in the mobile application, first take measurements from the CLU and - if necessary - synchronize the measurements with the cloud.

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 [6].
- Launch the application interface - the measurements should be updated and displayed on the chart.

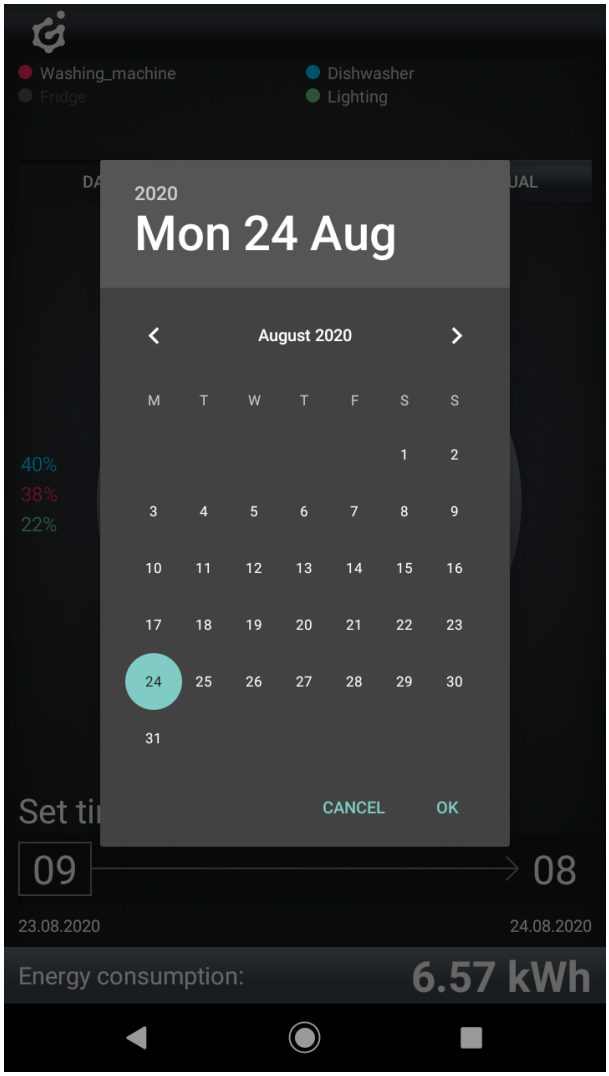
B. Media panel view options

- 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*.

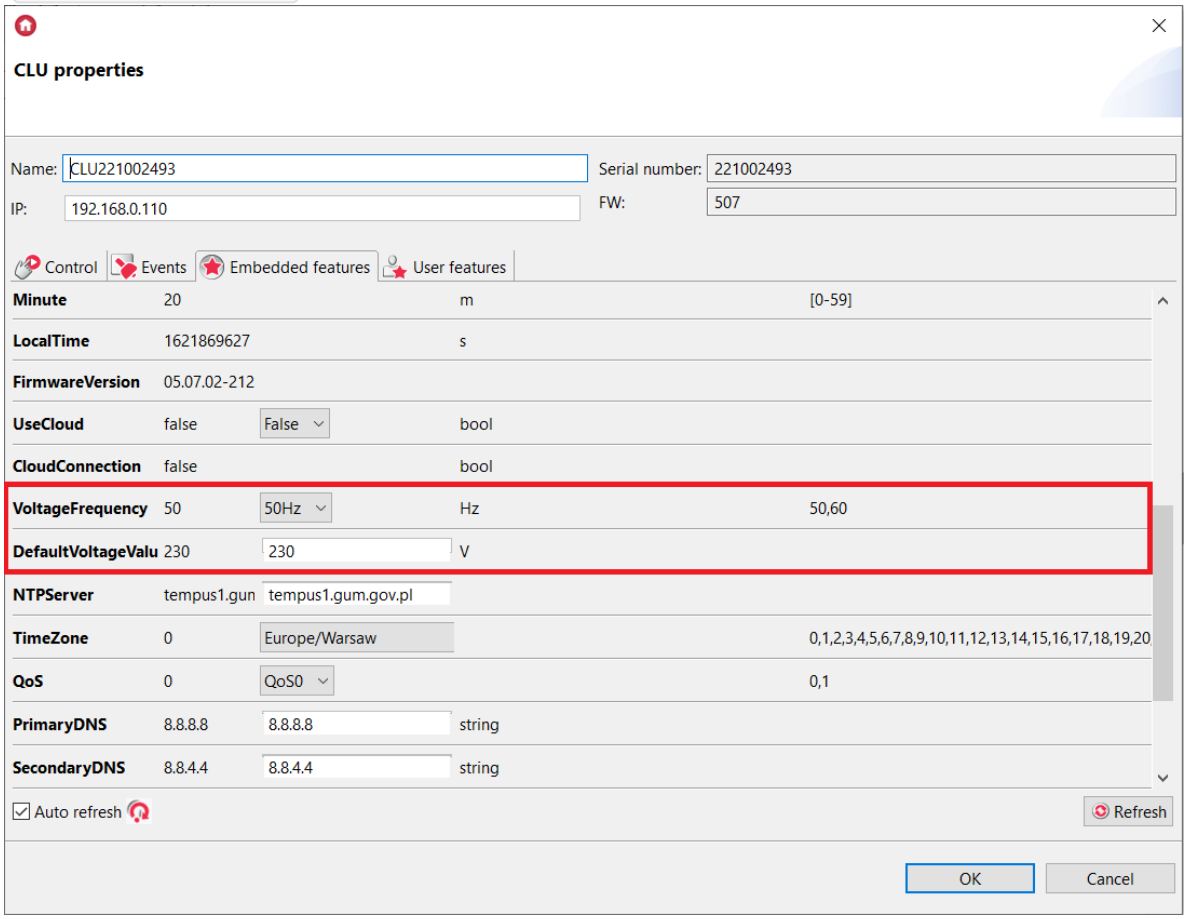
2. Real media measurement

> **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

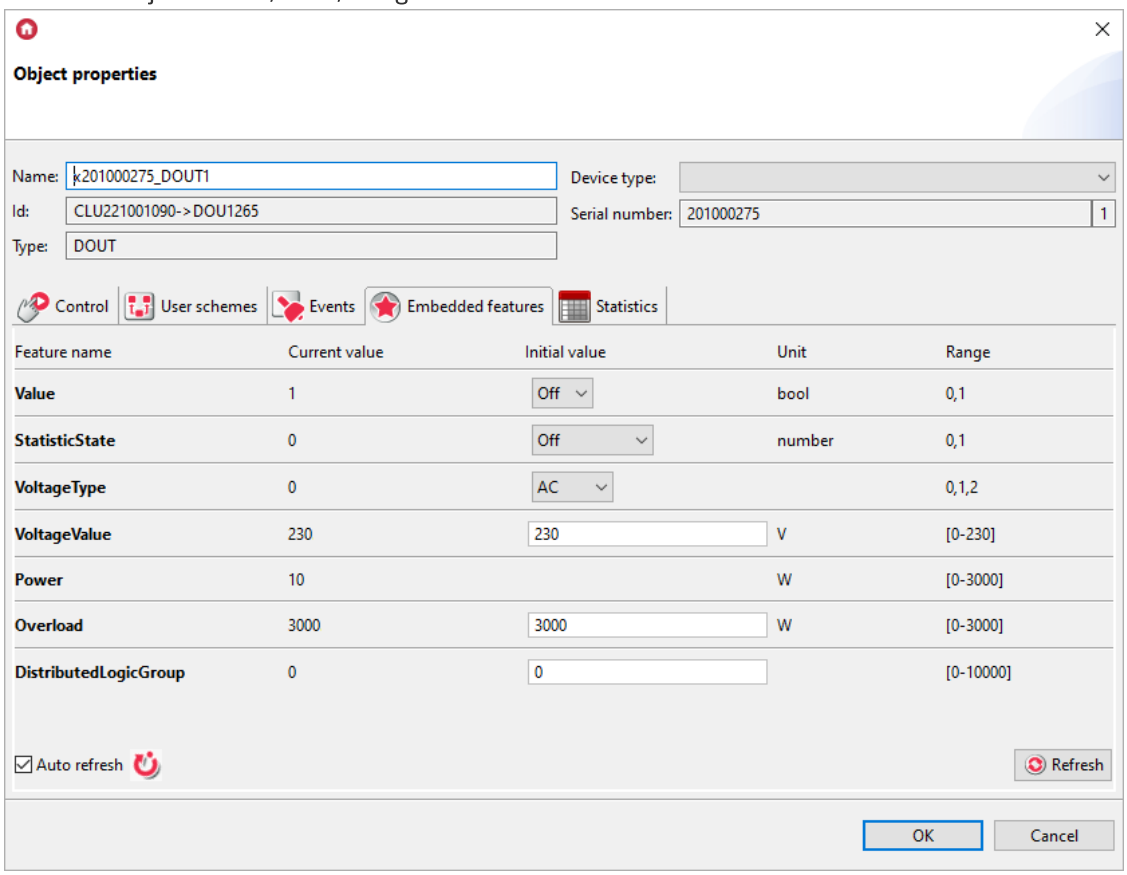
(`DefaultVoltageValue`) of the network.



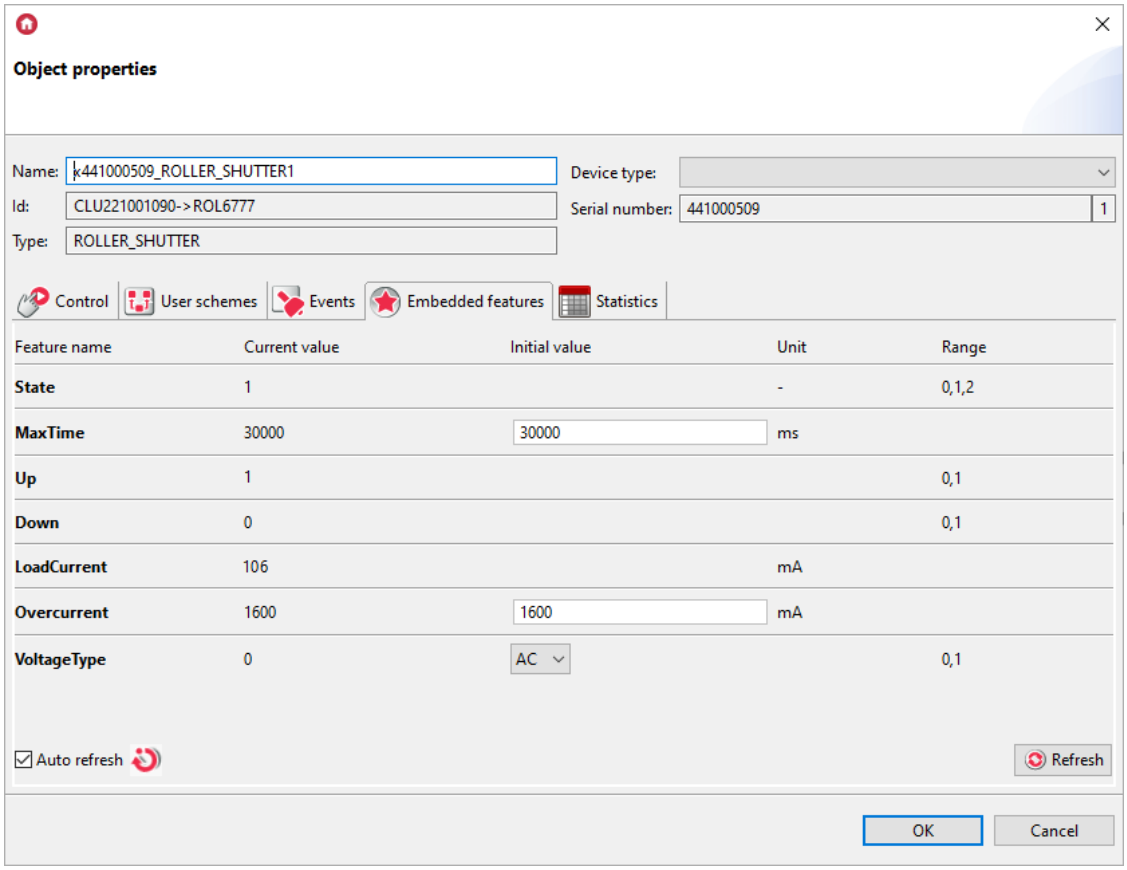
Media measurement is recorded in real time. `Power` features are expressed in watts for relay output modules and `LoadCurrent` are expressed in milliamperes for blind drive control modules.

The `VoltageType` feature takes the values:

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



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



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.

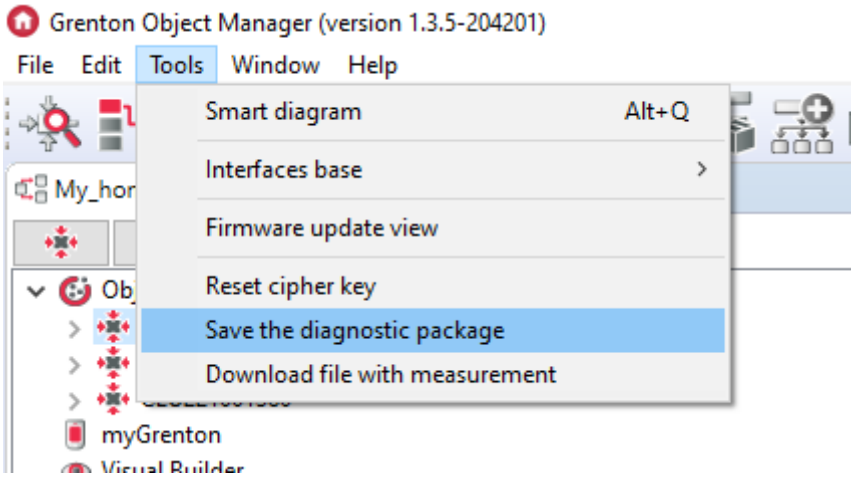
> **Note!**
> If Z-Wave modules were added to the CLU before starting the *Hard Reset* function, after performing the reset it will be necessary to perform the procedure of deleting and re-adding each Z-Wave module!

2. System diagnostics - *Save the diagnostic package*

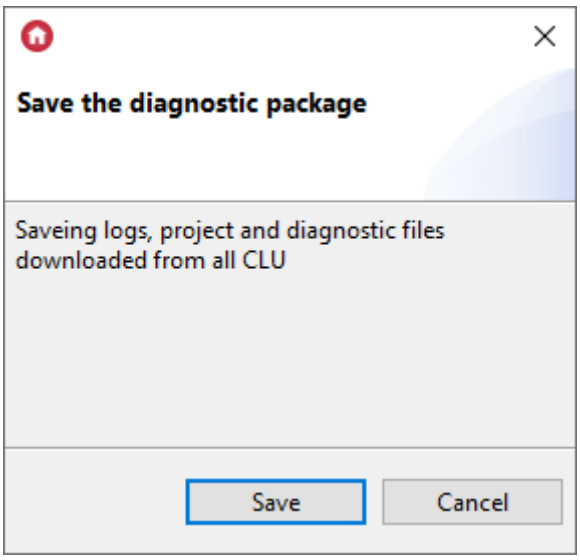
Diagnostic package is used for CLU diagnostics and for quick finding of problems in the created project.

In order to diagnose the system, you should:

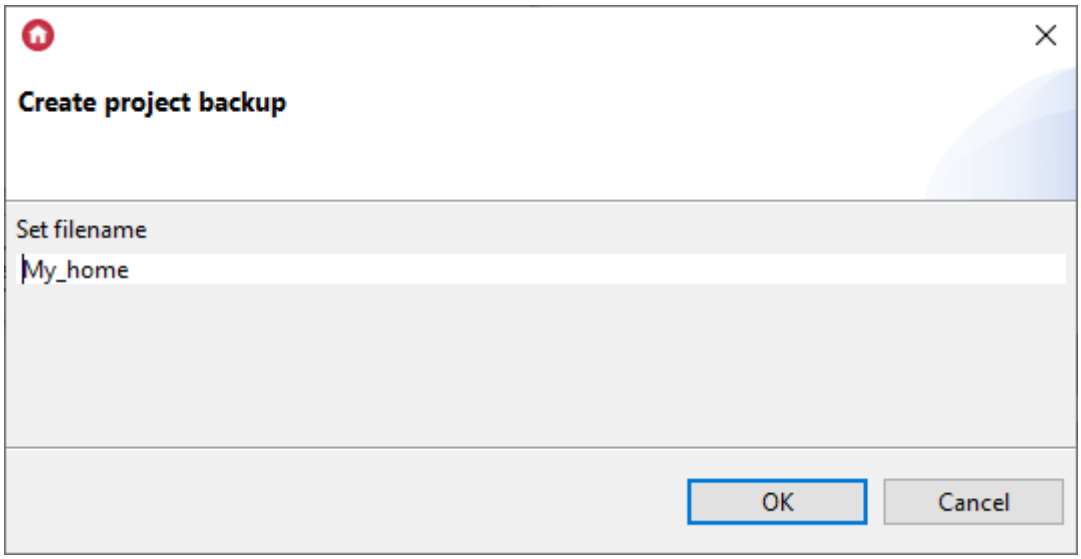
- Open the project in the Object Manager;
- Select `Tools` from the taskbar and then *Save the diagnostic package*:



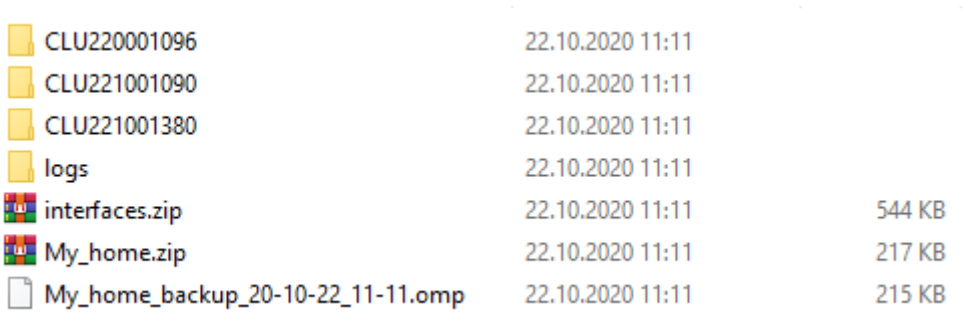
- In the window that opens, select *Save*:



- 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:



- 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.

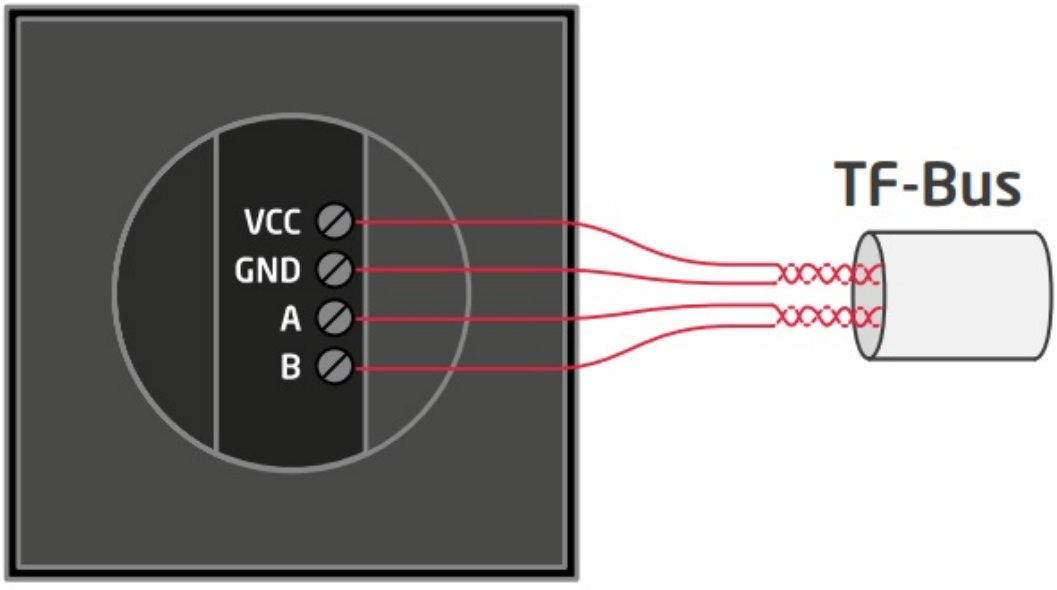
2. Connection of the Smart Panel to the CLU

> **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.

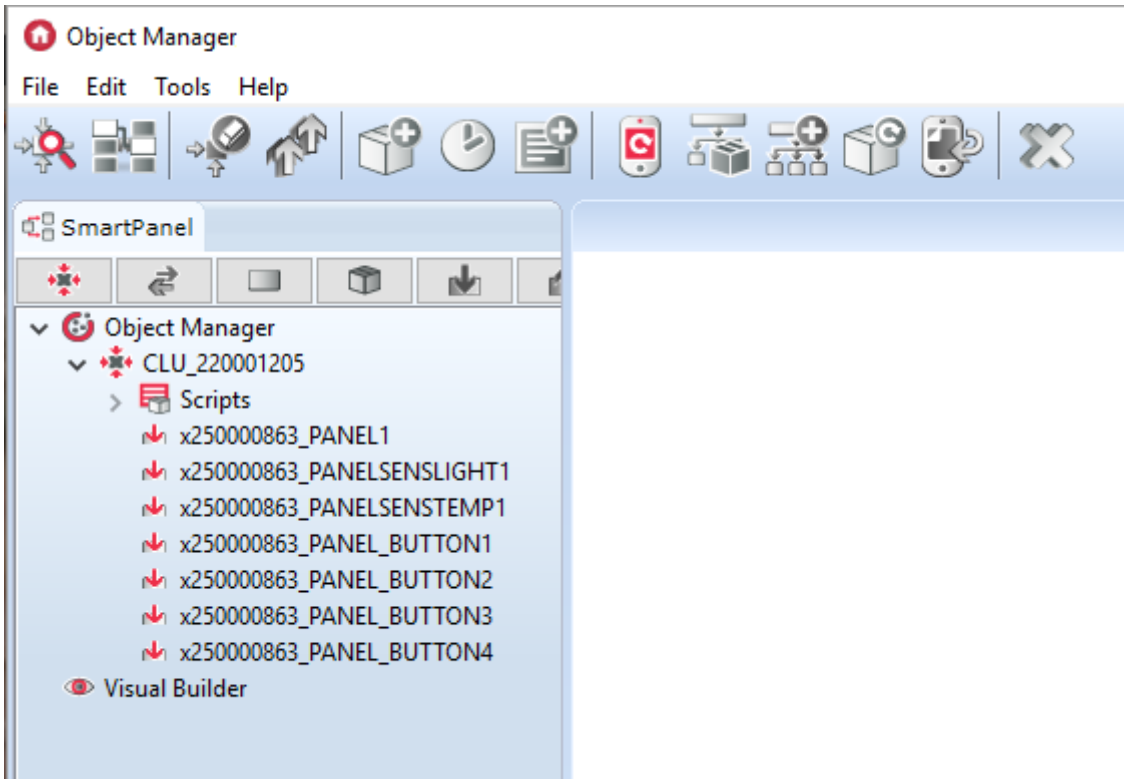
> **Note!**
> Smart Panel version v4 is available for Object Manager in version 1.2.1.190201 and higher, and for CLU with firmware 04.07.49-1912 and higher.

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:

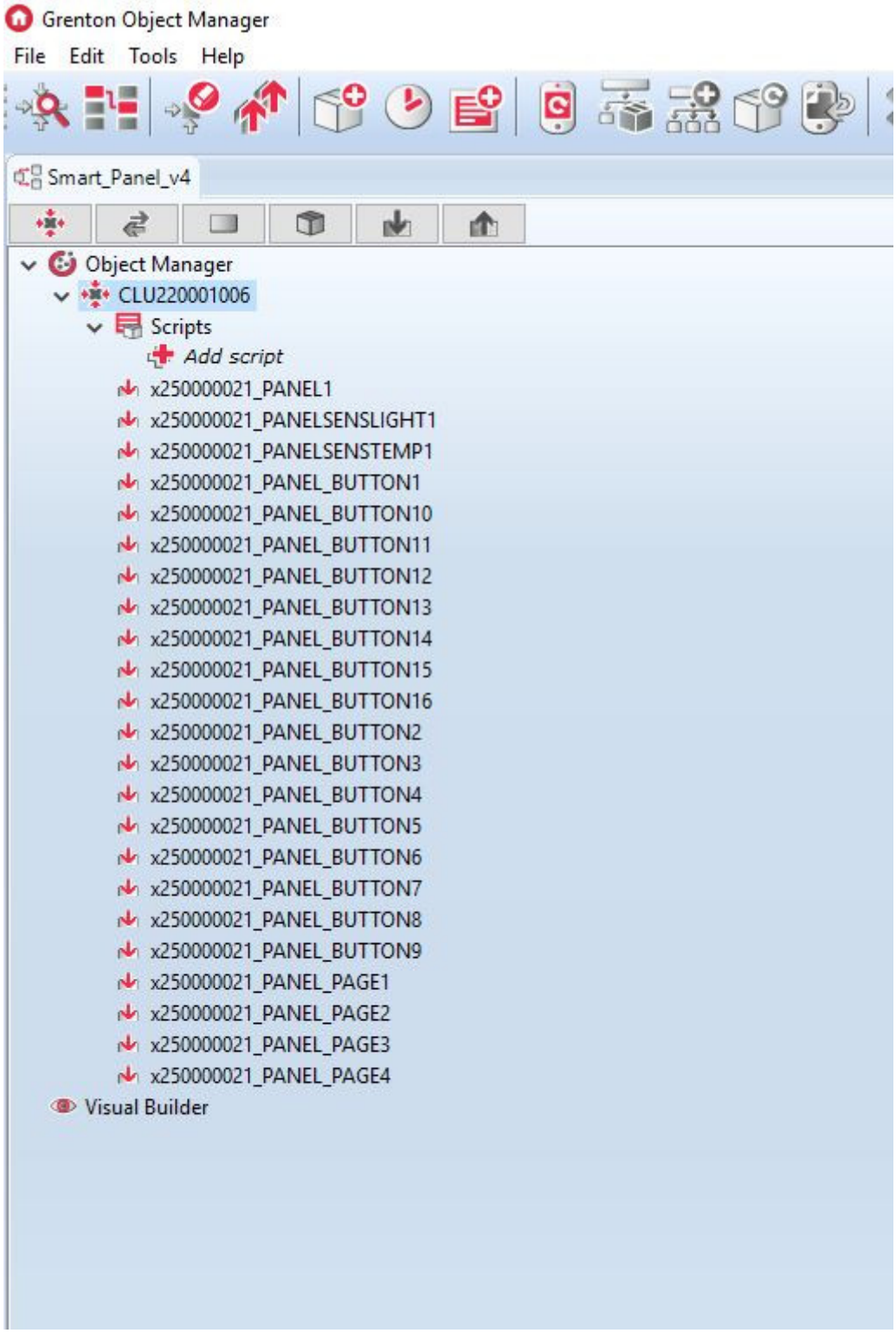
- Connect one lead from the first twisted pair (eg UTP cable) to the Vcc terminal;
- Connect the other wire from the pair to the GND terminal;
- Connect one cable from the other pair to terminals A and B.



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

METHODS

Name	Description
`SwitchOnDisplay`	It wakes the display from sleep mode
`ShowButtons`	Changes the display mode to *buttons*
`ClearScreen`	Cleans the display content in *freedraw mode*
`PrintText`	Display text in *freedraw*
`PrintFloat`	Displays the number in *freedraw*
`DrawLine`	Draws the line in *freedraw*
`DrawPoint`	Draws a point in *freedraw*
`DrawIcon`	Draws the icon (bmp) in *freedraw*
`DisplayContent`	Displays the contents of the graphic memory buffer; changes the display mode to *freedraw*
`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
`OnChange`	An event dispatched when the state changes (regardless of the value)
`OnSwitchOn`	An event that is triggered when the high state on input is set
`OnSwitchOff`	An event dispatched when the low state on input is set
`OnShortPress`	The event is triggered after pressing the button for 500 - 2000 ms
`OnLongPress`	The event is called after pressing the button for the 2000 - 5000 ms
`OnHold`	An event that is called for the first time after the `HoldDelay` time has elapsed, and then cyclically every time `HoldInterval`
`OnClick`	The event is triggered after pressing the button for less than 500ms

C. Temperature and lighting sensors

FEATURES

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

EVENTS

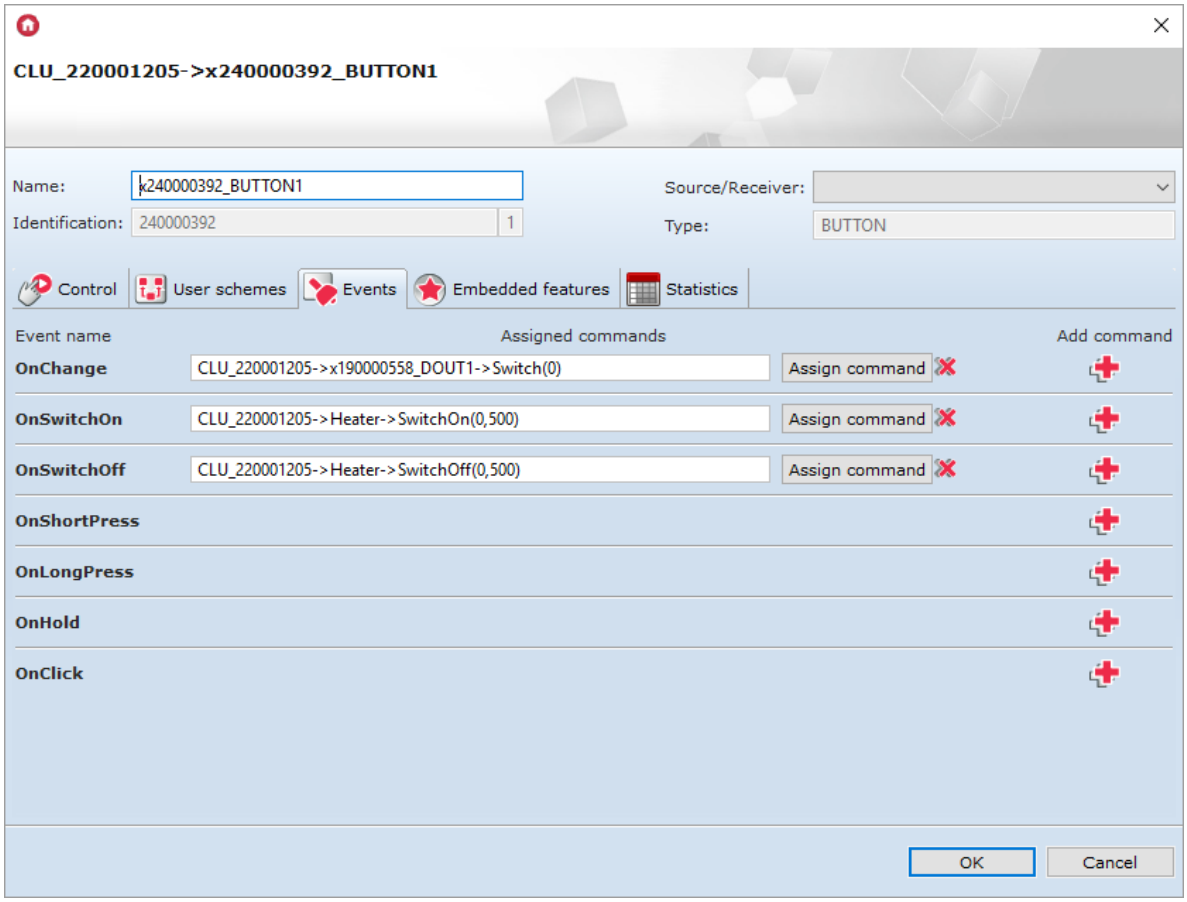
Name	Description
`OnChange`	Event triggered when the input state changes (regardless of value)
`OnRaiseValue`	An event triggered when the upper hysteresis threshold is exceeded
`OnLowerValue`	An event triggered when the hysteresis threshold is exceeded
`OnOutOfRange`	The event is dispatched when the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

4.2 Creating button and display configurations

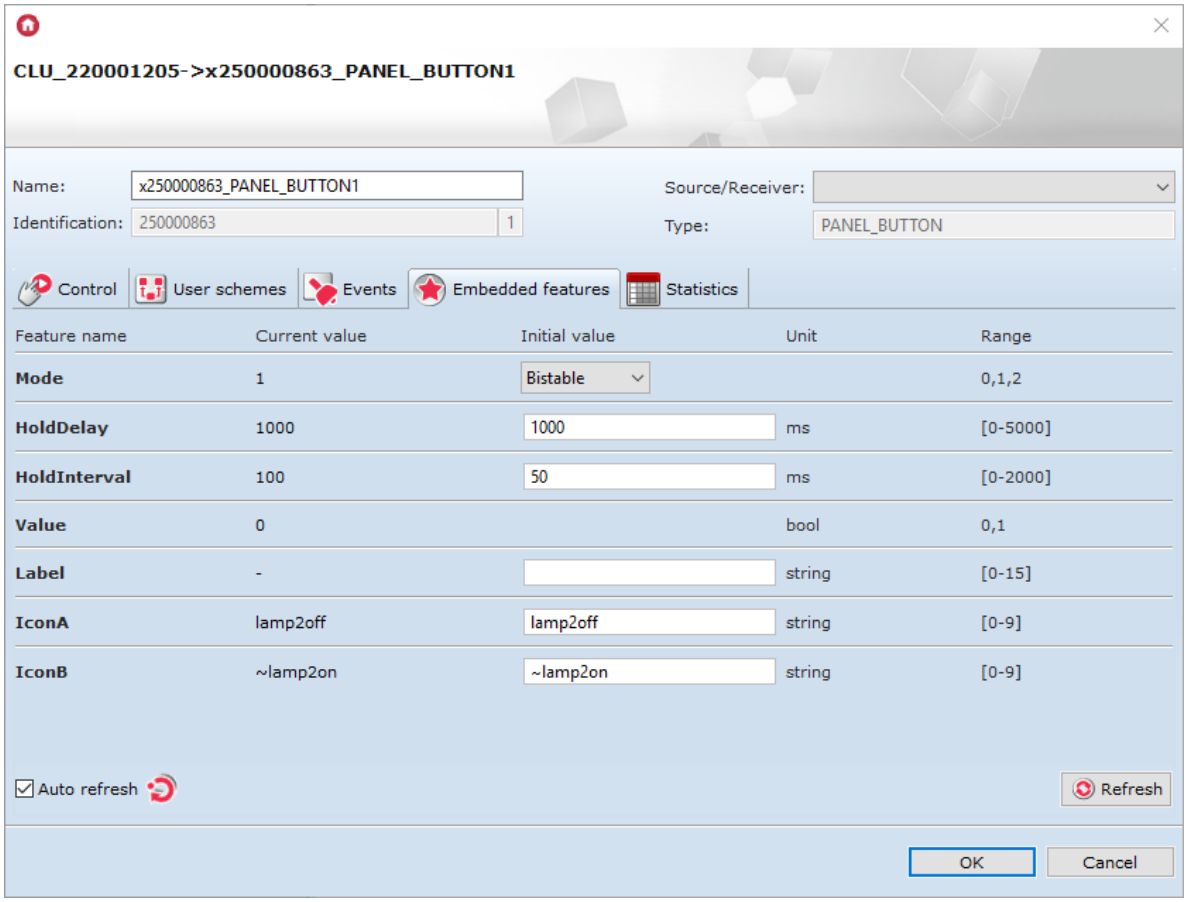
In order to create a configuration:

- 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):



- 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`):



The above features can be set both in the tab *Built-in features*, as well as via the methods: `SetLabel`, `SetIconA`, `SetIconB`.

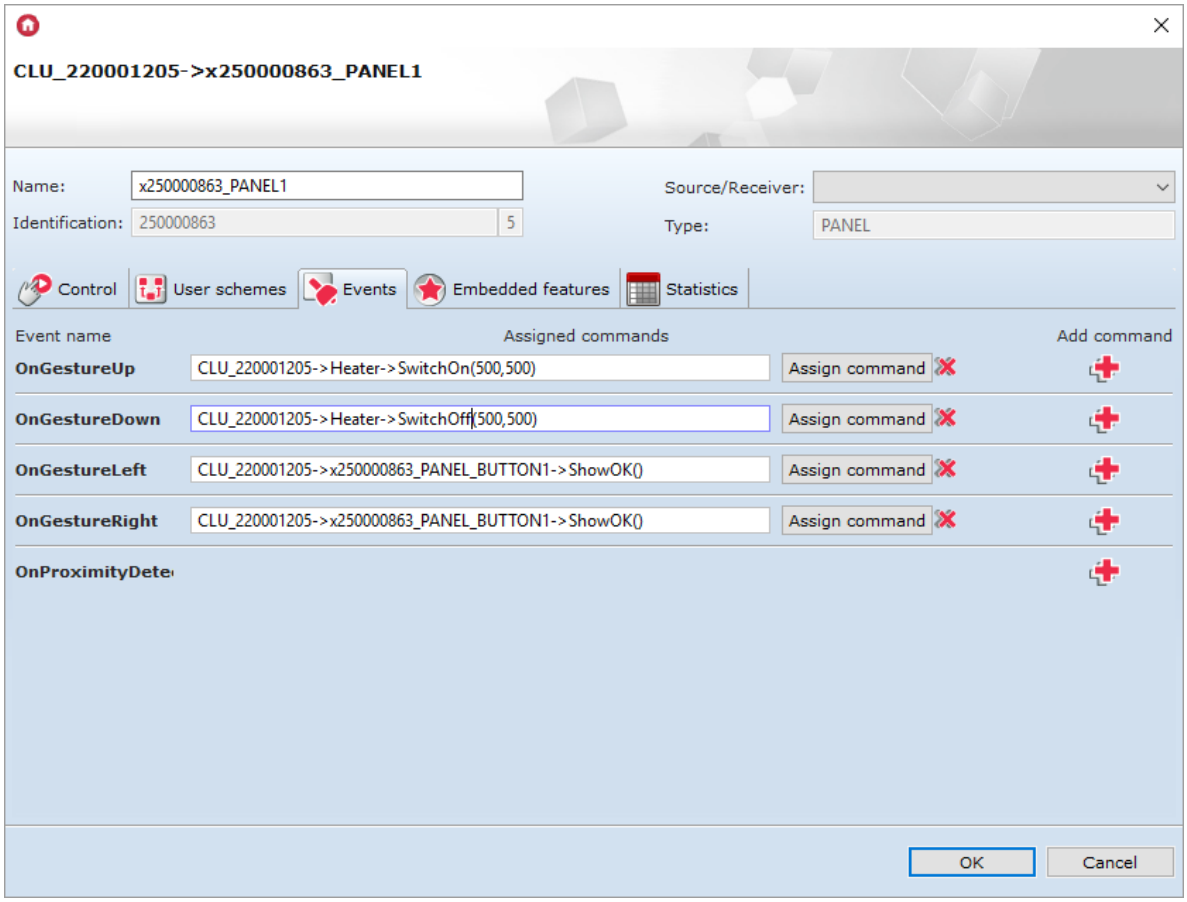
> **Note!**
> The `SetIcon` method has a higher priority in the system than the `SetLabel` method!

- Send the configuration to the CLU.

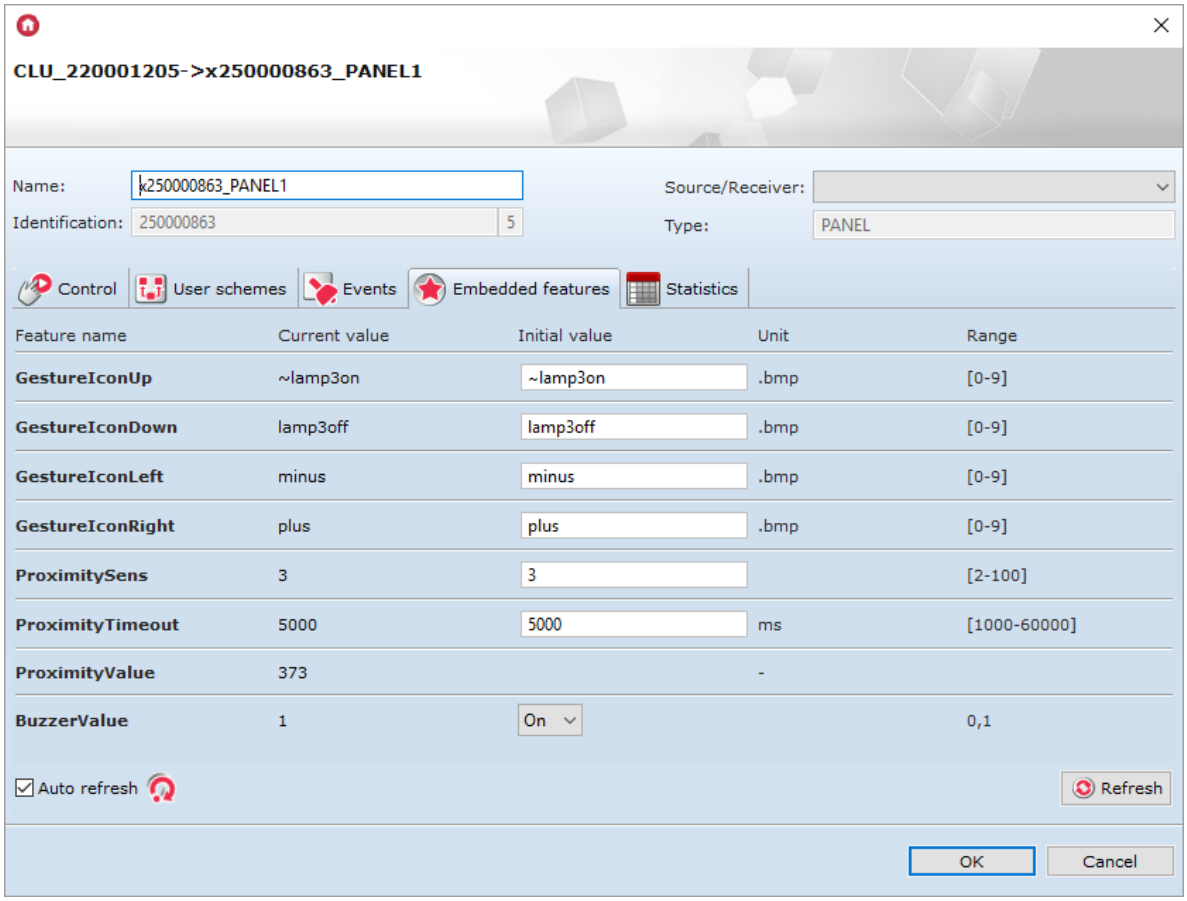
4.3 Creating a gesture sensor configuration

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):



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:



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

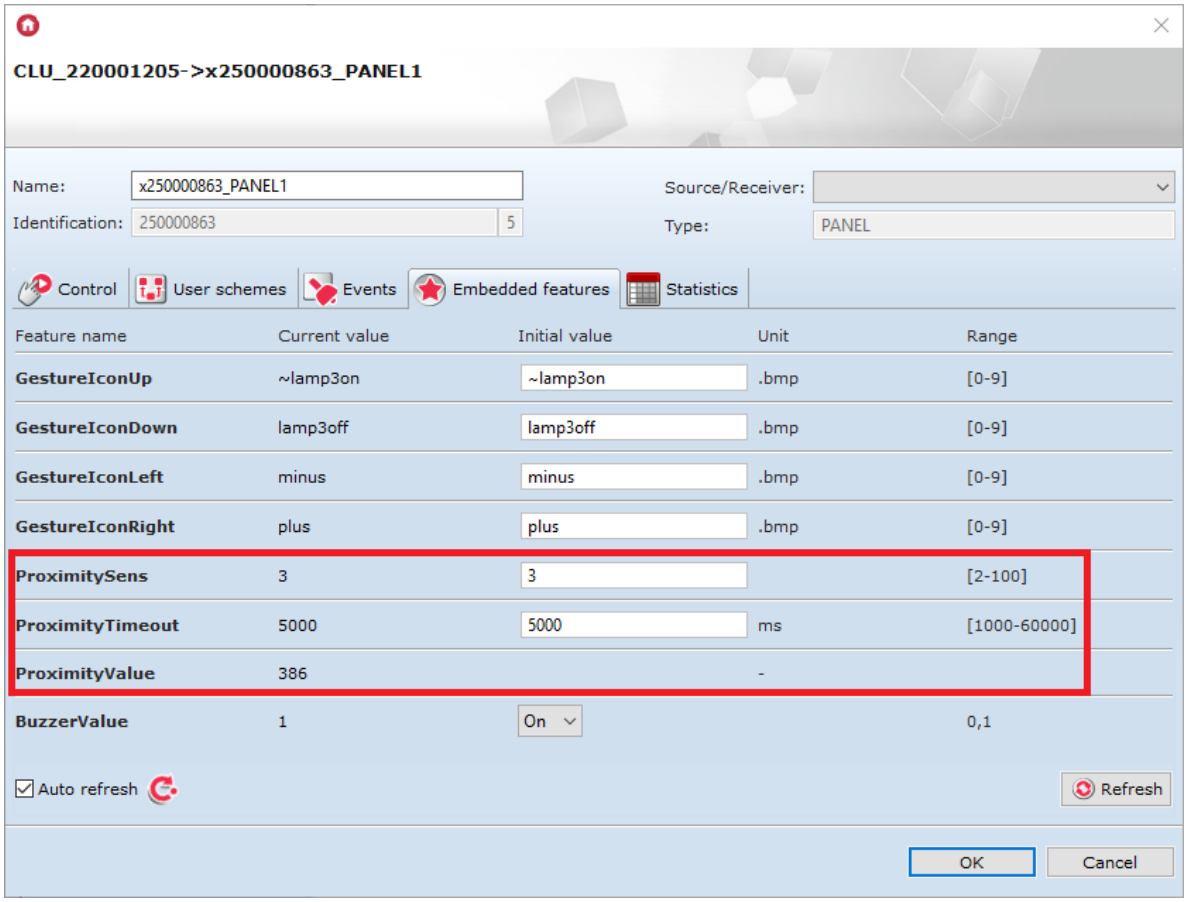
- Confirm the configuration window with *OK*;

- Send the configuration to the CLU.

4.4 Configuration of the proximity sensor

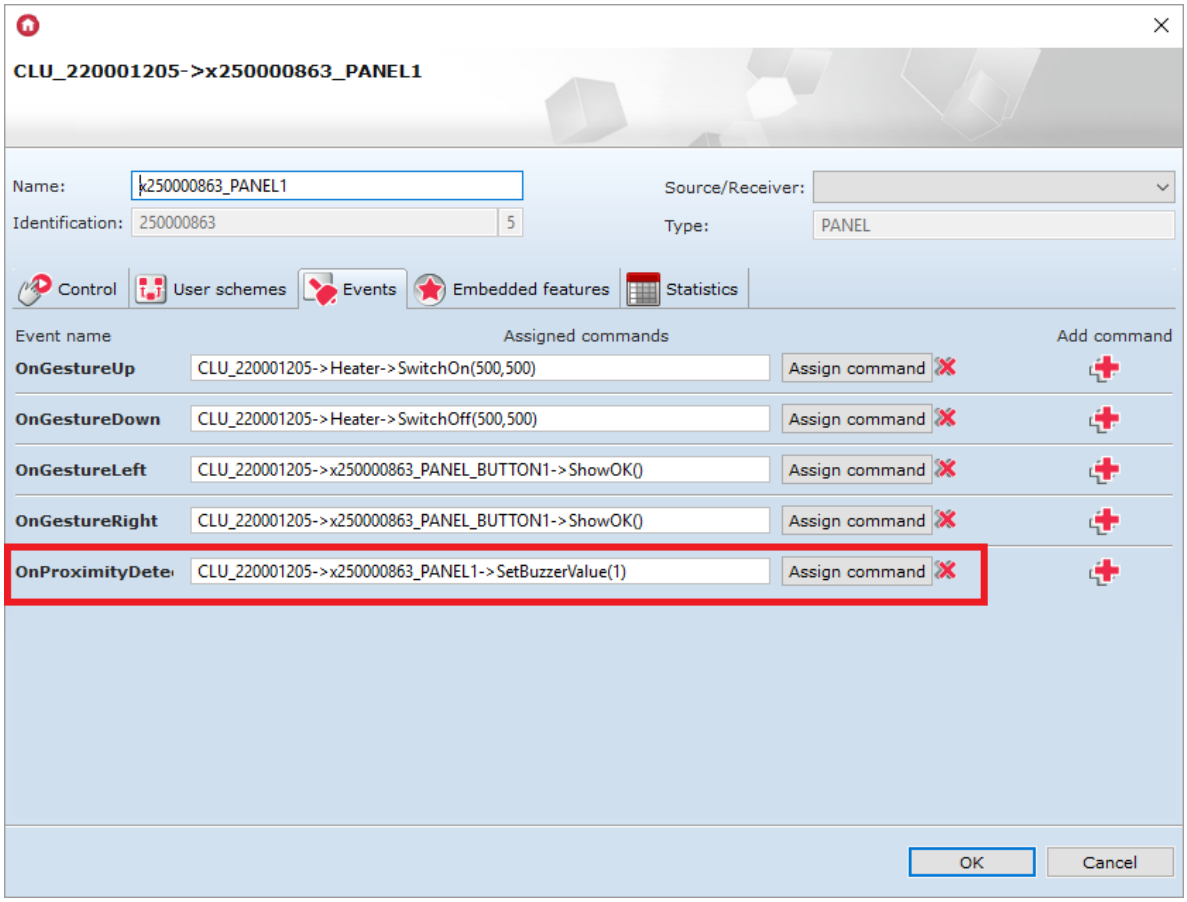
To set the proximity sensor parameters:

- 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:



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:

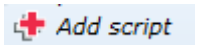


- Send the configuration to the CLU.

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!

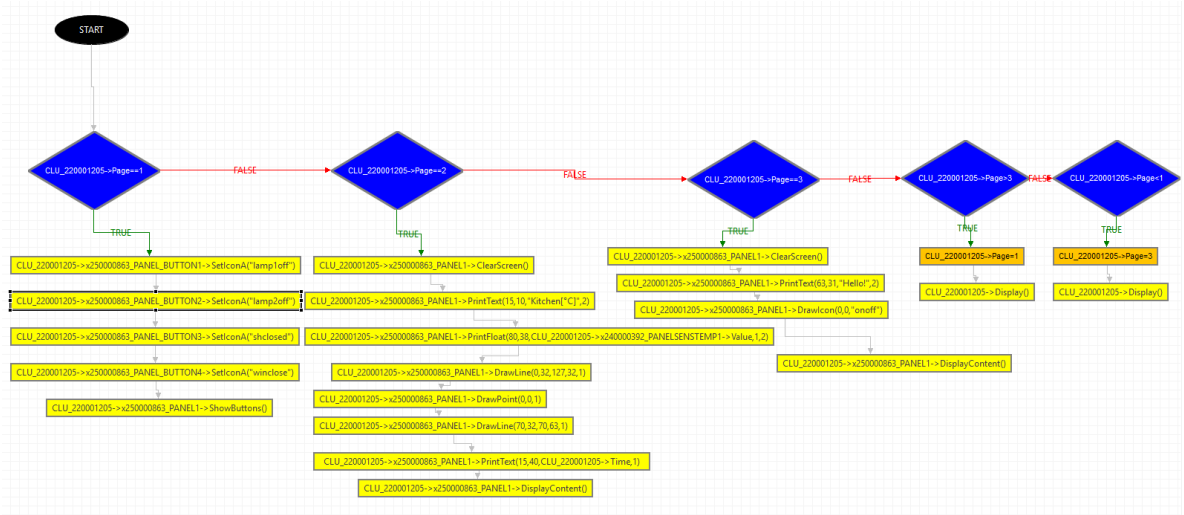
- **PAGE WITH THE BUTTONS** - Add a condition checking the current page number (value *User features*: *page*) to the script and for the given condition - for a specific page - add the icon allocation action for all 4 buttons (`SetIconA` methods for elements *PANEL_BUTTON1-4*) and the method `ShowButtons` 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.

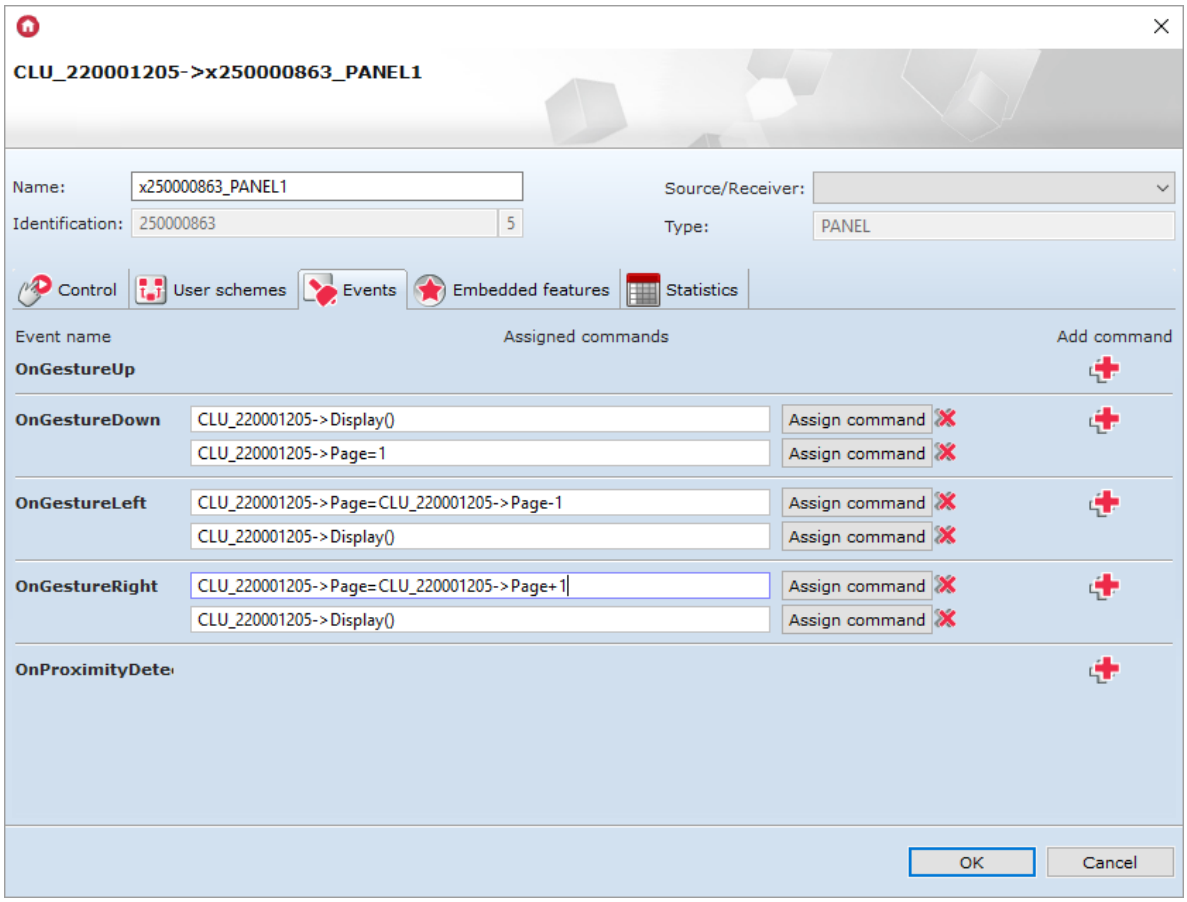
The implementation of all the methods described above is presented in the screen shot of the sample script:



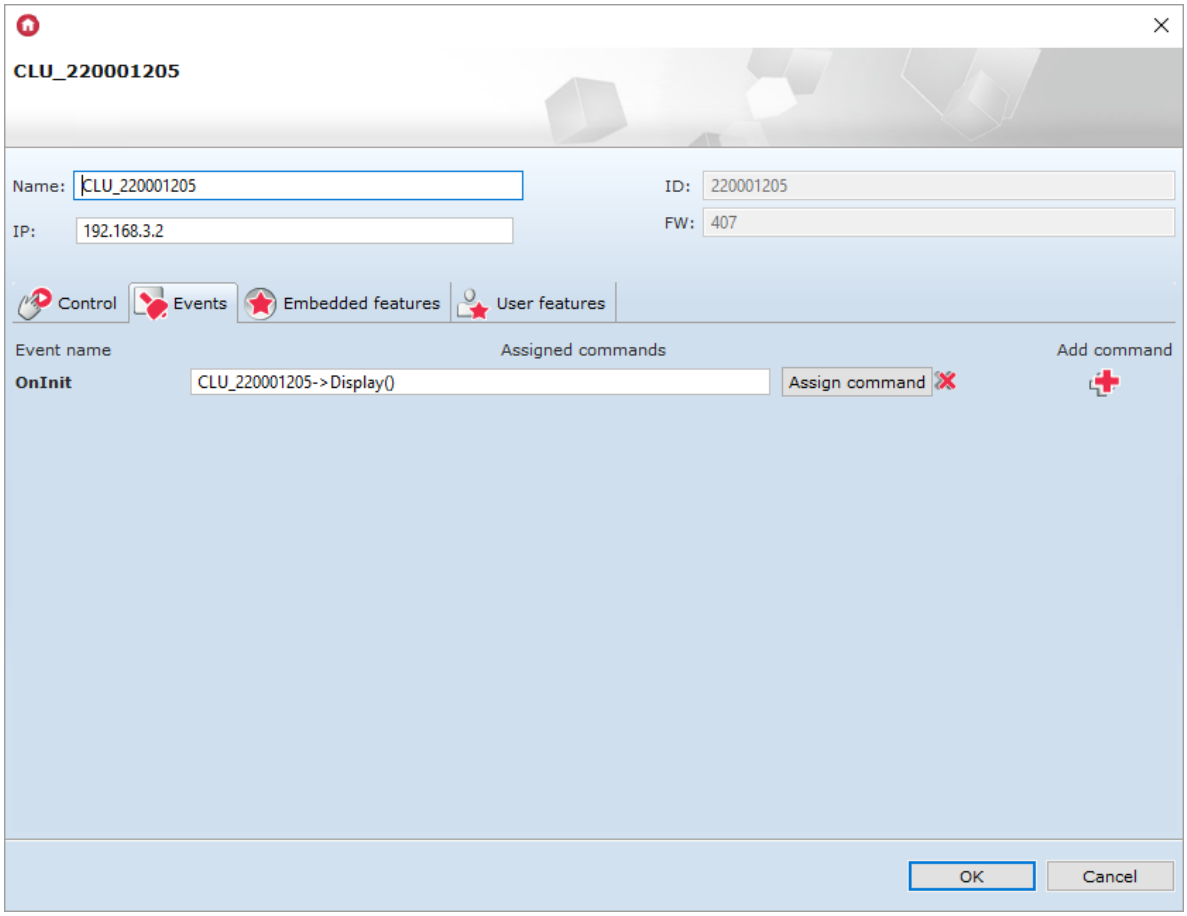
The above script is placed at the end of the document in the text version (point 3).

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:

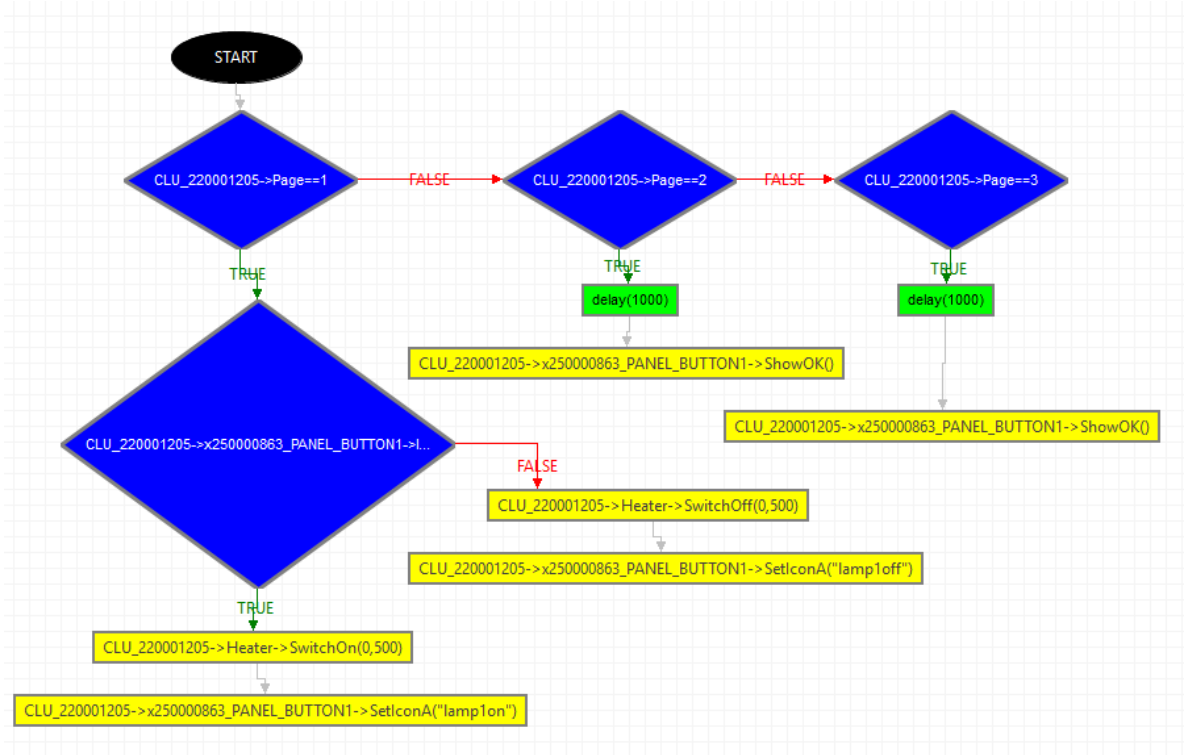


- Assign *CLU ->* `OnInit` to the script call *Display*:



- 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* -> `OnClick`

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
`GestureDisplayMode`	Display information about the currently executed gesture: `0 - Off`, `1 - On`

METHODS

Name	Description
`SwitchOnDisplay`	Wakes the display from sleep mode
`ShowButtons`	Changes the display mode to *buttons*. Clears the display and displays the icons (or text) again for all buttons
`ClearScreen`	Cleans the display content in *freedraw mode*
`PrintText`	Displays the text in *freedraw* mode using the parameters: `x`, `y`, `txt`, `font size`, where: `x` and `y` are the coordinates expressed in pixels, `txt` is a string, `font size` is the font size(1: 10p, 2: 14p, 3: 32p)
`PrintFloat`	Displays the number in *freedraw* mode using the parameters: `x`, `y`, `number`, `precision`, `font size`, where: `x` and `y` are coordinates expressed in pixels, `number` is the number, `precision` is the number of decimal places, `font size` is the font size (1:10p, 2:14p, 3:32p)
`DrawLine`	Draw lines in *freedraw* mode using the parameters: `x`, `y`, `xe`, `ye`, `color`, where: `x` and `y` are initial coordinates, `xe` and `ye` are final coordinates, `color` is the colour line (0 - black, 1 - white). The starting and ending coordinates are expressed in pixels
`DrawPoint`	Draws a point in the *freedraw* mode using the parameters: `x`, `y`, `color`, where: `x` and `y` are the coordinates expressed in pixels, `color` is the color of the point (0 - black, 1 - white)
`DrawIcon`	Draws the icon (bmp) in *freedraw* mode using the parameters: `x`, `y`, `Filename`, where: `x` and `y` are the coordinates expressed in pixels `Filename` is the name of the icon (without extension)
`DisplayContent`	Displays the contents of the graphical memory buffer. Changes the display mode to *freedraw*
`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 `holdDelay` time has elapsed, and then cyclically every `HoldInterval` value
`OnClick`	An event triggered after pressing the button for less than 500ms

C. Pages configuration (Panel_Page)

FEATURES

Name	Description
`PageType`	The type of page displayed on the Smart Panel: `0 - Inactive`, `1 - Buttons`, `2 - Thermostats`, `3 - FreeDraw`
`PageName`	Page name / name of the icon displayed on the Smart Panel (when switching between pages)
`Object_1_Id`	Thermostat object ID or button number depending on page type, e.g.: *Thermostats* page type: - for thermostat on local CLU: THE1325 - for remote thermostat CLU: CLU220000001-> THE4321 For the PageType feature set to *Buttons* / *FreeDraw* enter the number of the button (1..16)
`Object_1_Name`	The name of the thermostat displayed on the Smart Panel page. Applies only to page *Thermostats* (no name - inactive thermostat). In the case of the `PageType` feature set to *Buttons* / *FreeDraw*, the feature remains empty
`Object_2_Id`	Thermostat object ID or button number depending on page type, e.g.: *Thermostats* page type: - for thermostat on local CLU: THE1325 - for remote thermostat CLU: CLU220000001-> THE4321 For the PageType feature set to *Buttons* / *FreeDraw* enter the number of the button (1..16)
`Object_2_Name`	The name of the thermostat displayed on the Smart Panel page. Applies only to page *Thermostats* (no name - inactive thermostat). In the case of the `PageType` feature set to *Buttons* / *FreeDraw*, the feature remains empty
`Object_3_Id`	Thermostat object ID or button number depending on page type, e.g.: *Thermostats* page type: - for thermostat on local CLU: THE1325 - for remote thermostat CLU: CLU220000001-> THE4321 For the PageType feature set to *Buttons* / *FreeDraw* enter the number of the button (1..16)
`Object_3_Name`	The name of the thermostat displayed on the Smart Panel page. Applies only to page *Thermostats* (no name - inactive thermostat). In the case of the `PageType` feature set to *Buttons* / *FreeDraw*, the feature remains empty
`Object_4_Id`	Thermostat object ID or button number depending on page type, e.g.: *Thermostats* page type: - for thermostat on local CLU: THE1325 - for remote thermostat CLU: CLU220000001-> THE4321 For the PageType feature set to *Buttons* / *FreeDraw* enter the number of the button (1..16)
`Object_4_Name`	The name of the thermostat displayed on the Smart Panel page. Applies only to page *Thermostats* (no name - inactive thermostat). In the case of the `PageType` feature set to *Buttons* / *FreeDraw*, the feature remains empty

METHODS

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

EVENTS

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

D. Temperature and lighting sensors

FEATURES

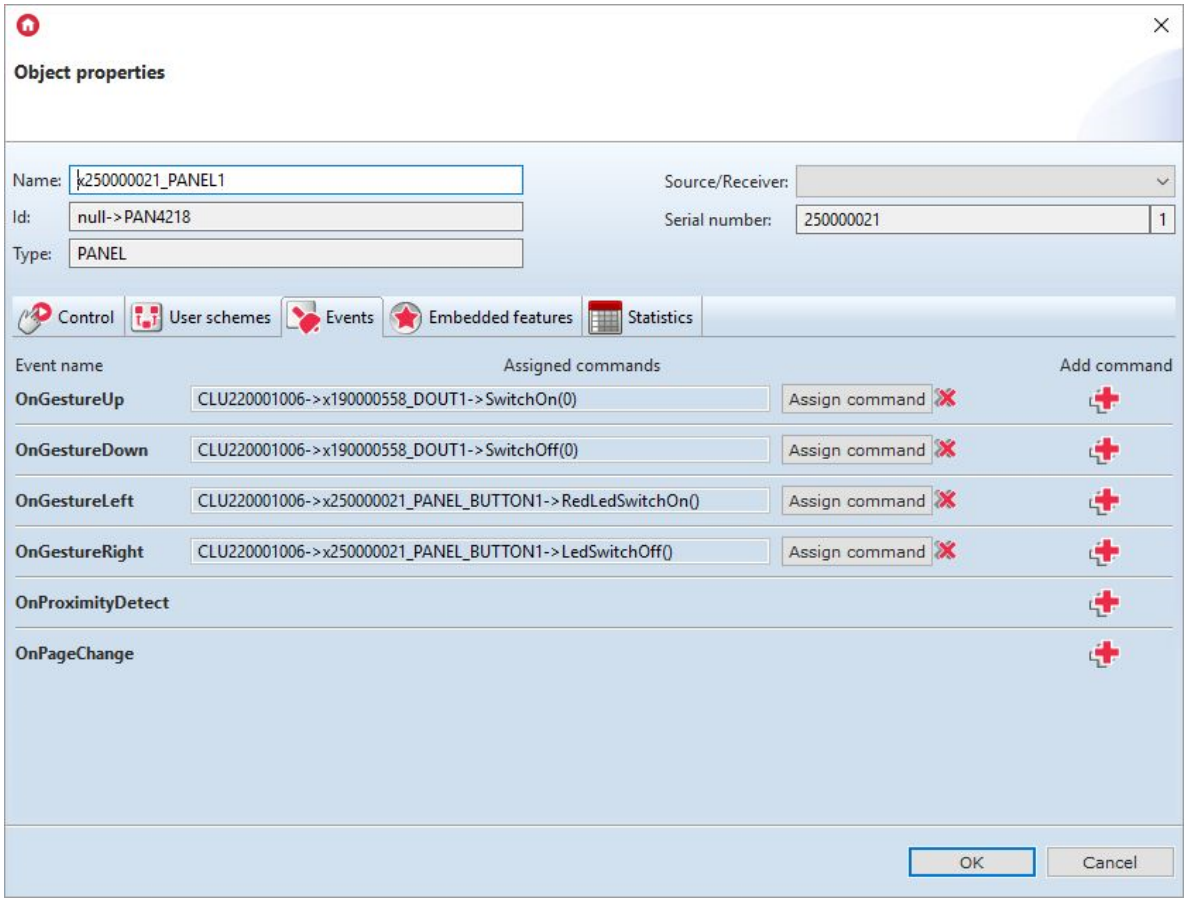
Name	Description
`Threshold`	Hysteresis size (accuracy 0.1 ° C / 0.1%) defining the sensitivity at which events are generated: `OnChange` , `OnLowerValue` , `OnRaiseValue`
`Sensitivity`	The period (in ms) at which the sampled values are averaged
`MinValue`	The minimum value of the `Value` feature, exceeded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

EVENTS

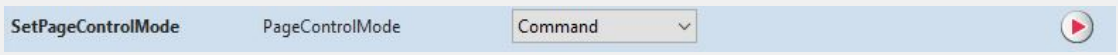
Name	Description
`OnChange`	An event triggered when the Value attribute changes
`OnRaiseValue`	An event triggered when the value changes to a higher one (rising edge)
`OnLowerValue`	An event triggered when the value changes to a lower one (falling edge)
`OnOutOfRange`	An event triggered when the input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

5.2. Creating a gesture sensor configuration

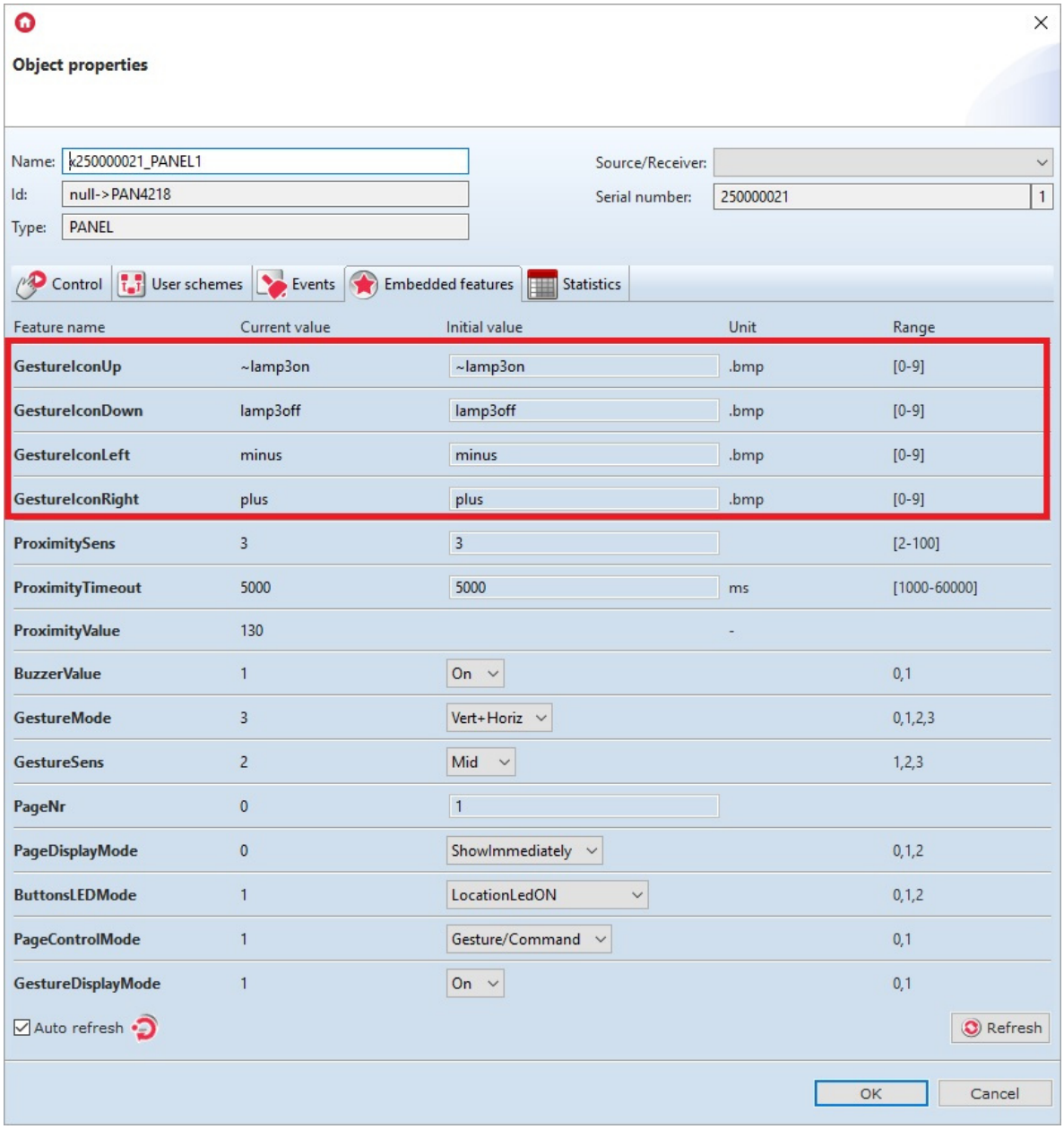
- To create a configuration for a gesture sensor:
  - Open - by double click - object *Panel*;
  - Go to the tab *Events*;
  - Assign methods to the events `OnGestureUp`, `OnGestureDown`, `OnGestureLeft`, `OnGestureRight` (clicking on the `+` on the right of each method):



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.

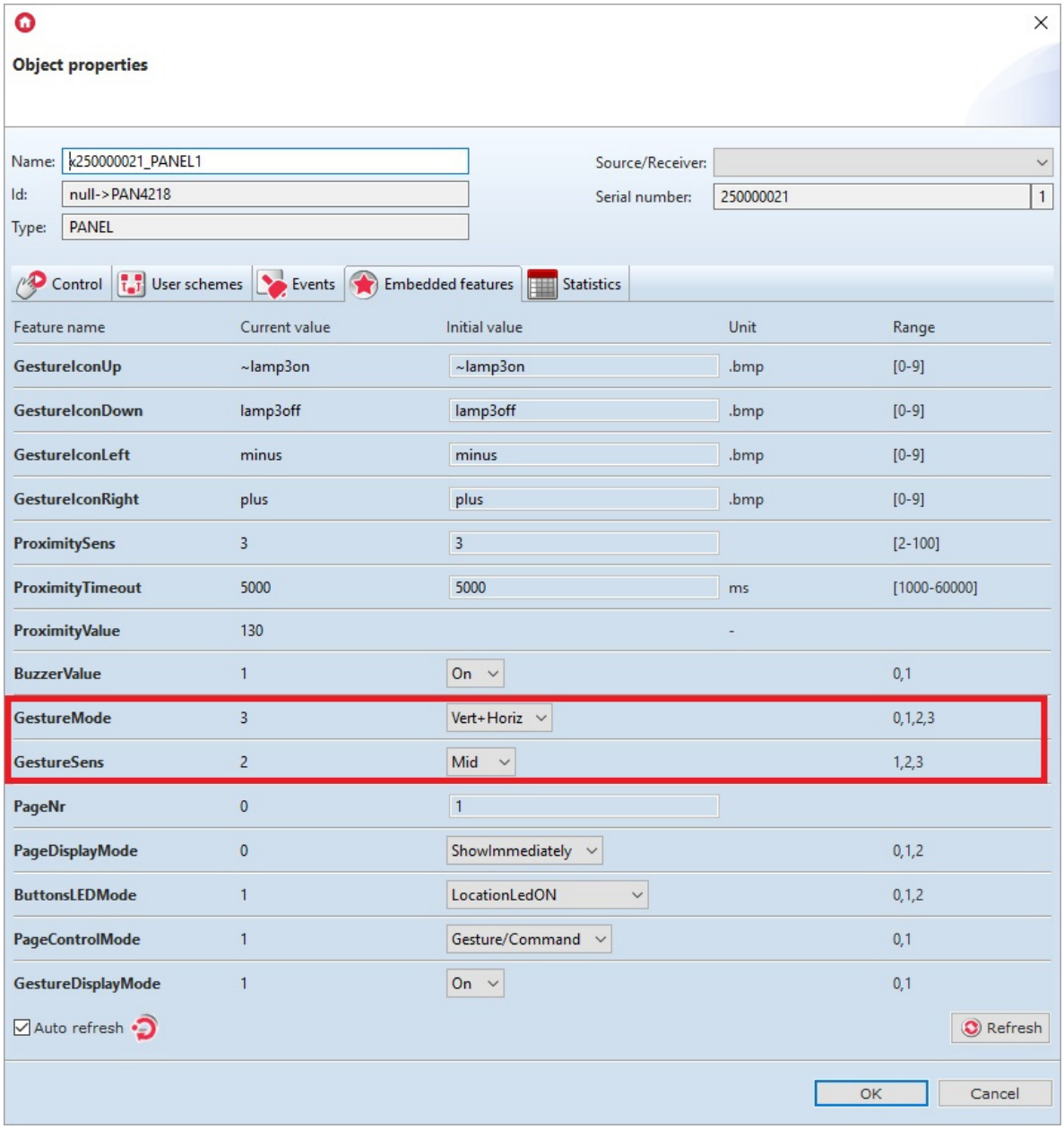


It is also possible to substitute the default icons displayed when gesturing - go to the *Built-in features* and enter the names of the icons you want without a *.bmp* extension:



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:



Embedded features, through which you can choose orientation and sensitivity:

- `GestureMode` - 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.
- `GestureSens` - 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.

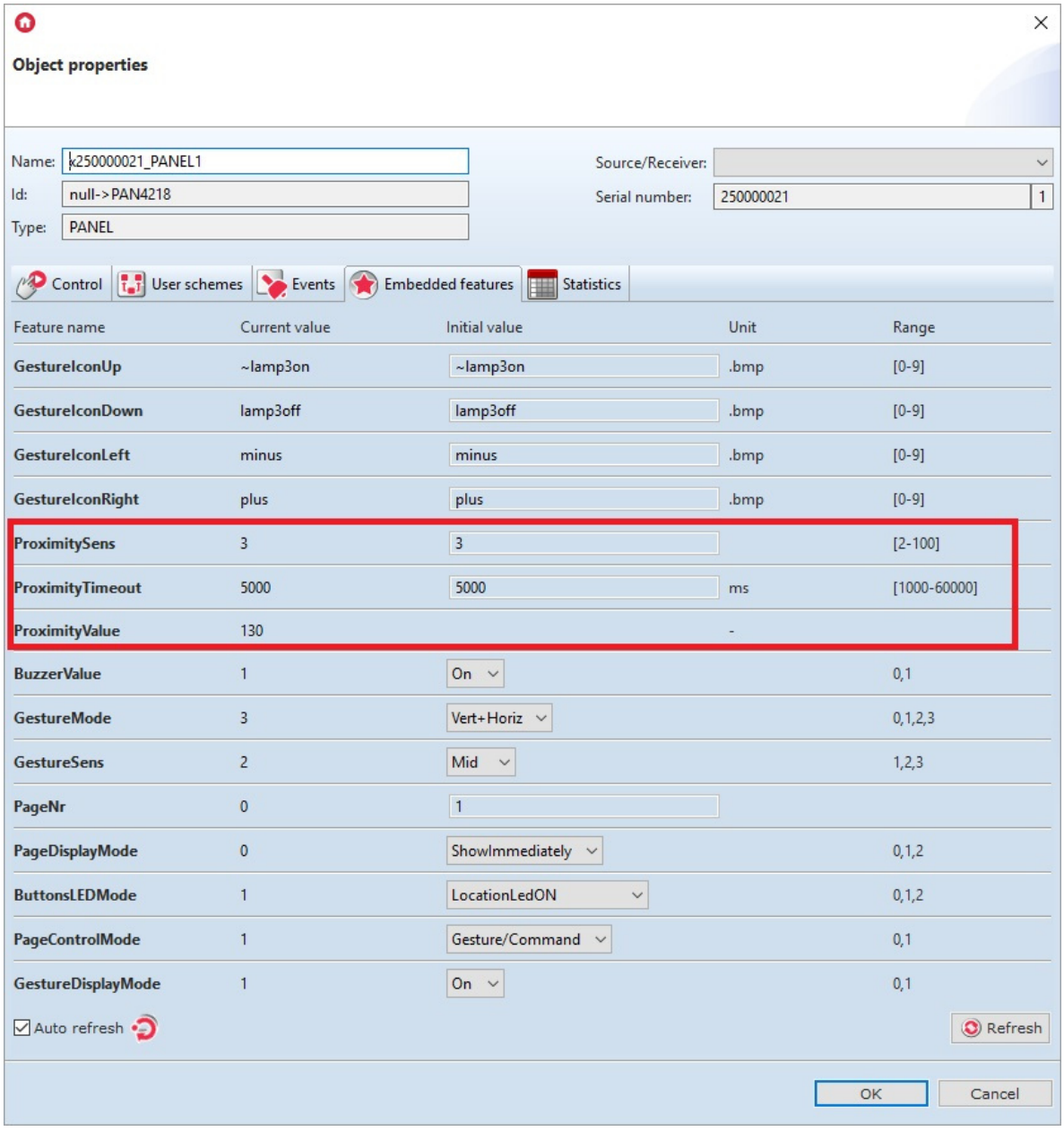
The above features can be set both in the tab *Built-in features*, as well as using methods: `SetGestureIconUp`, `SetGestureIconDown`, `SetGestureIconLeft`, `SetGestureIconRight`, `SetGestureMode`, `SetGestureSens` (in the methods of the Panel object).

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

5.3. Configuration of the proximity sensor

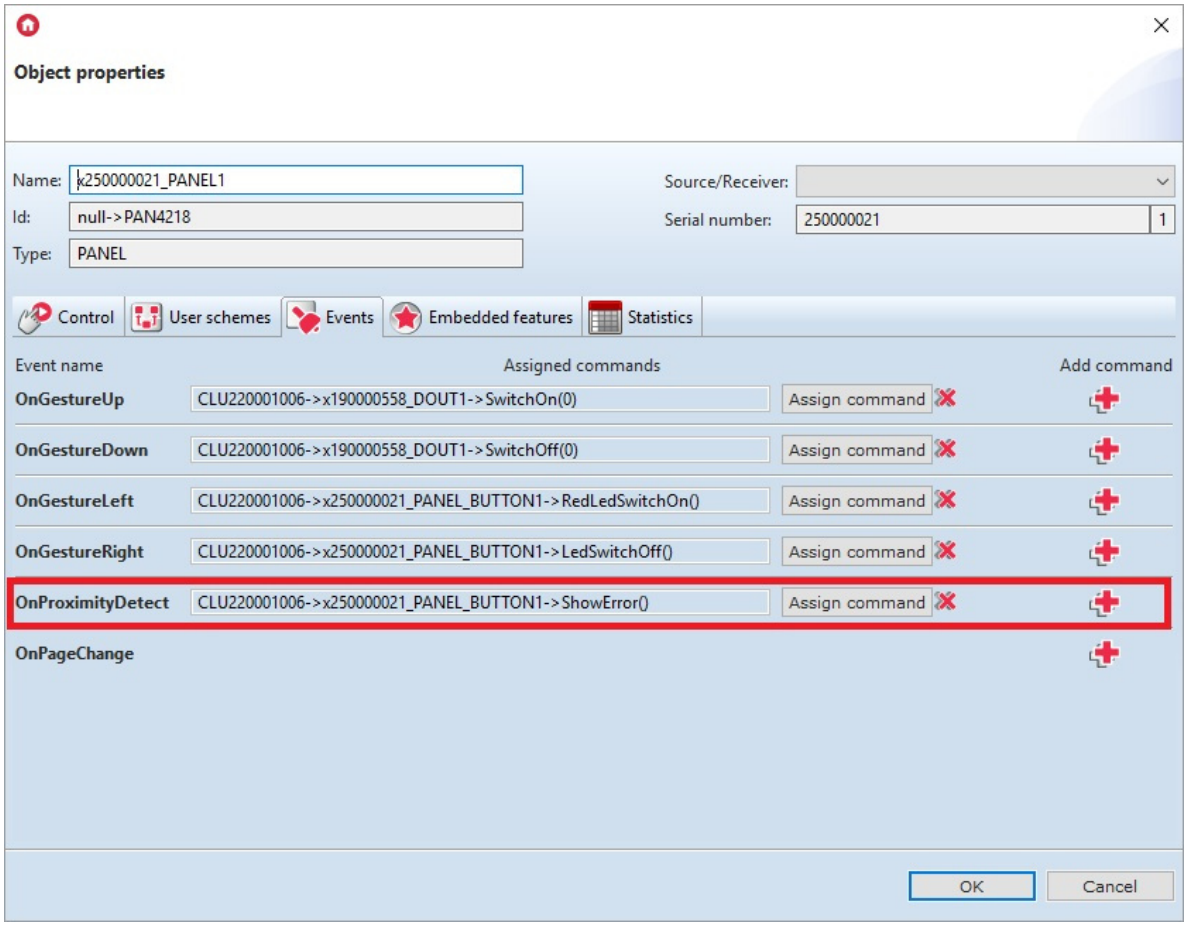
To set the proximity sensor parameters:

- 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 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:



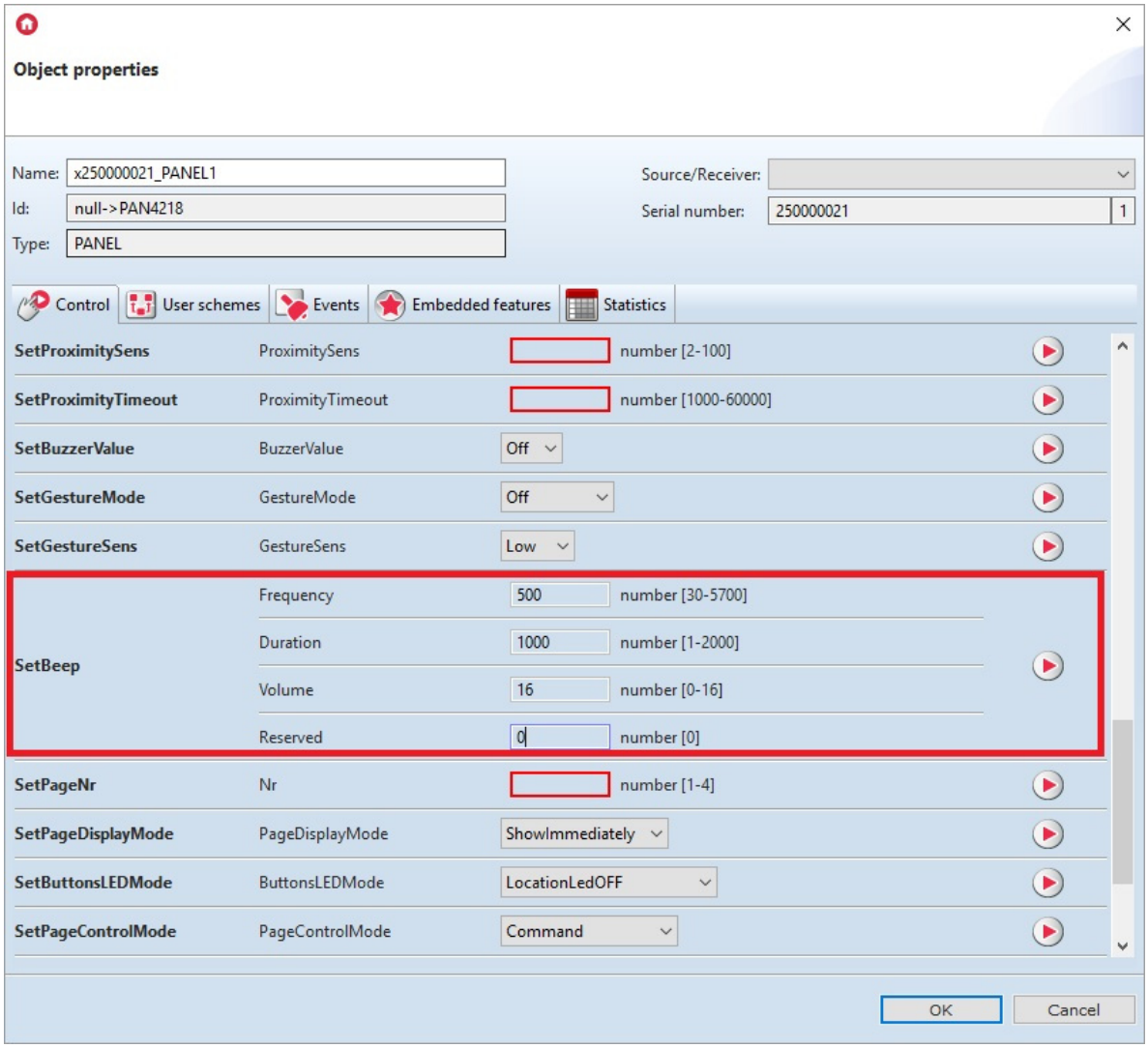
- Send the configuration to the CLU Z-Wave.

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:



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:

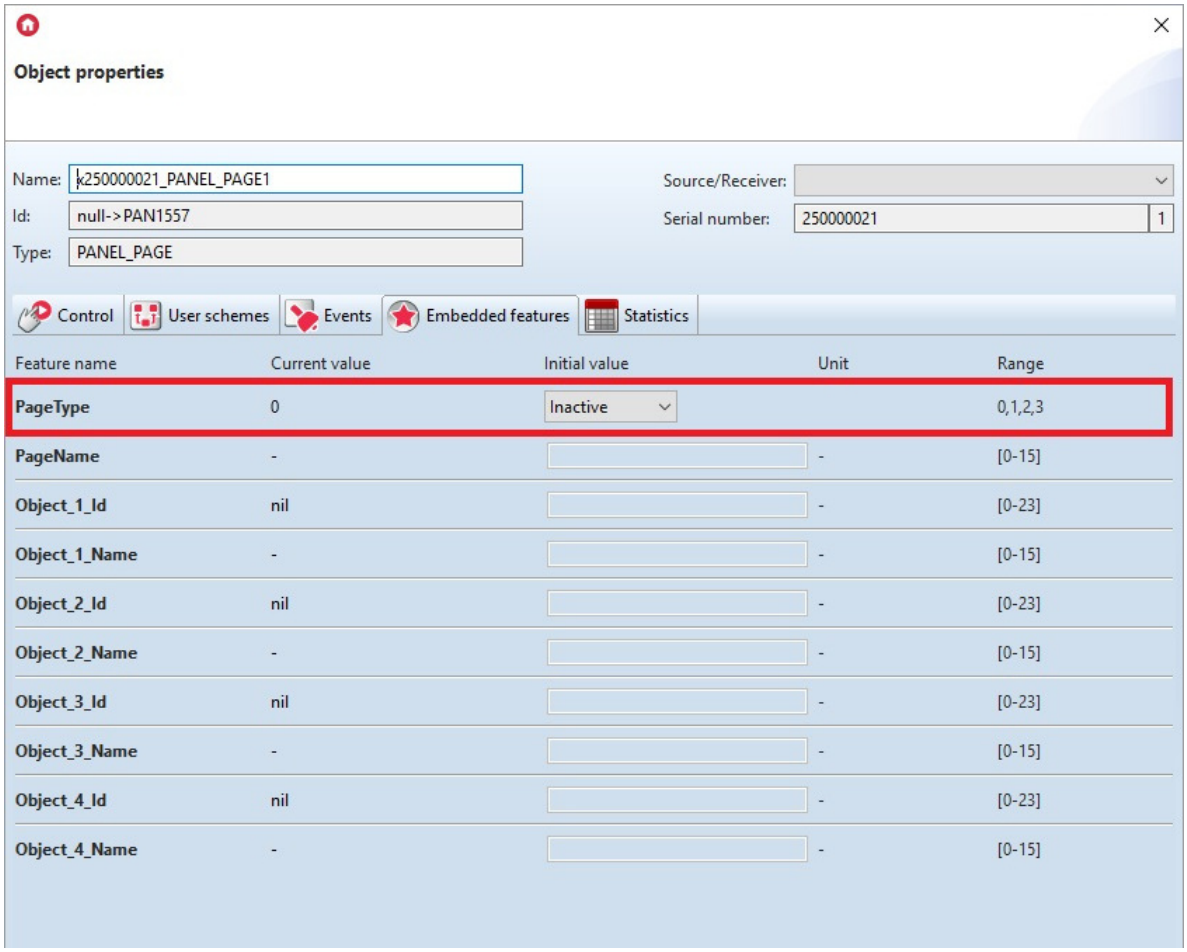
  Methods / Features:

- `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

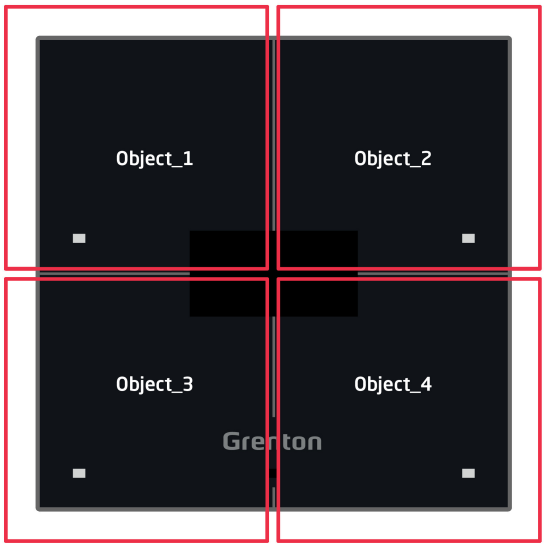
When starting work with the new version of the Smart Panel module, the device is in the default configuration, which is backward compatible. All four Panel_Page objects have the built-in feature `PageType` set to *Inactive*. This allows you to work with the panel in the same way as before (in version 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.



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).

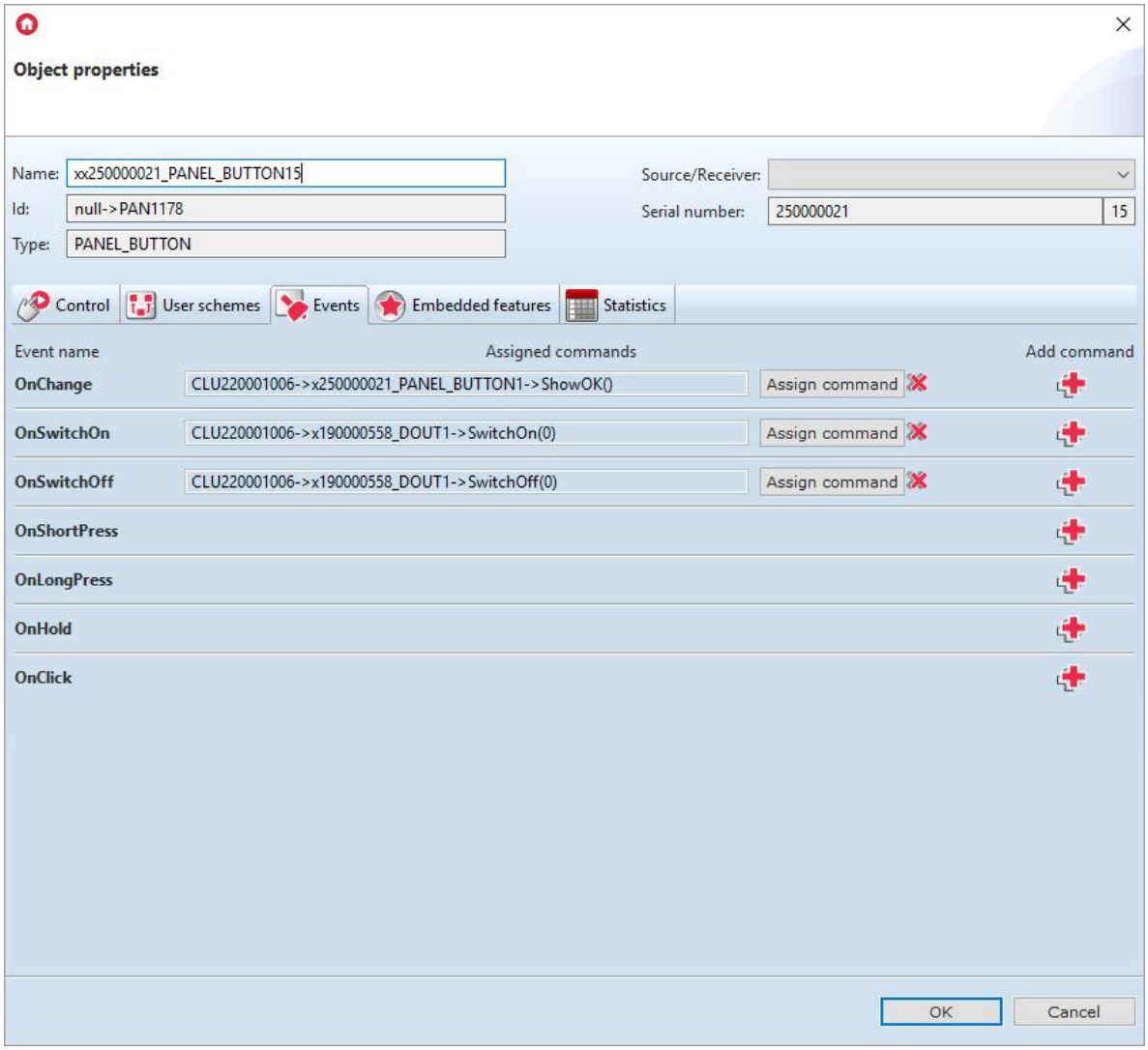
> **Note!**
> In the *Buttons* mode, drawing content on the display is blocked.



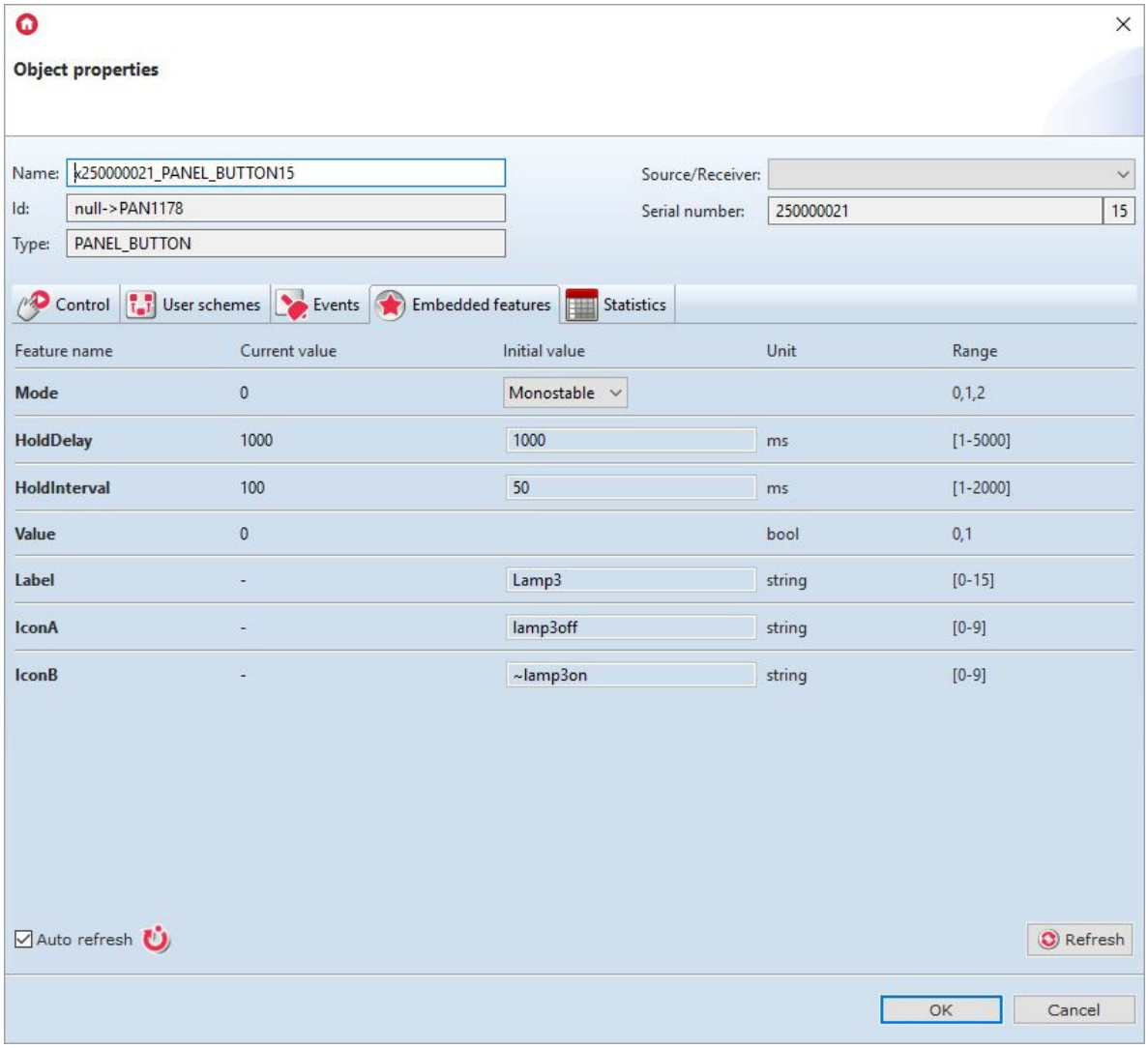
Page type „Buttons/FreeDraw"

- 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`):



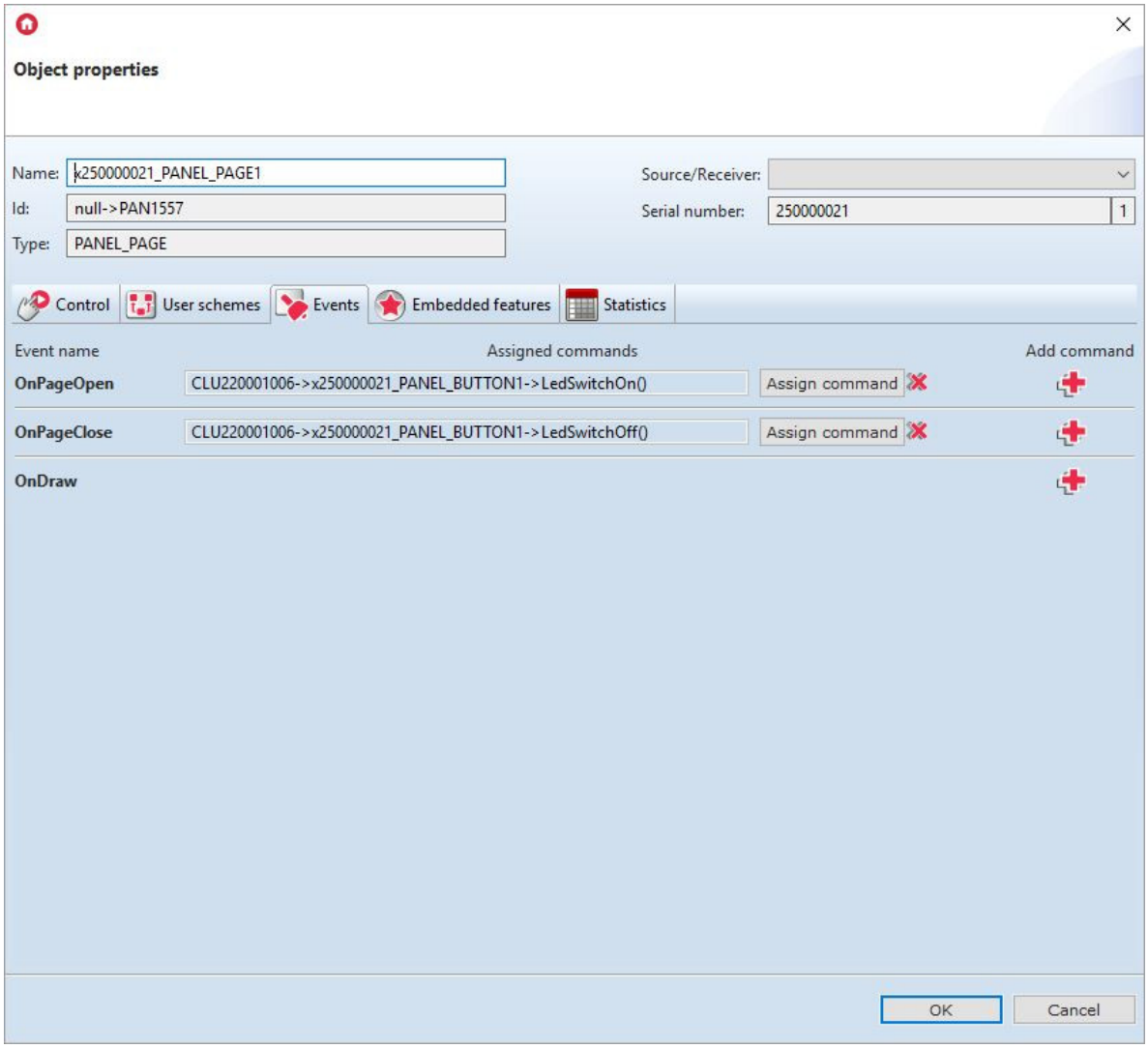
The above built-in features can be set both in the tab *Built-in features*, as well as via the methods: `SetLabel`, `SetIconA`, `SetIconB`.

> **Note!**
> The `SetIconA` method has a higher priority in the system than the `SetLabel` method!

- Send the configuration to the CLU Z-Wave.

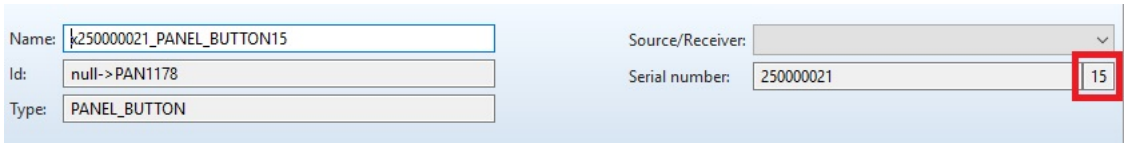
The next step in creating the configuration is configuring Panel_Page objects depending on the number of buttons. One Panel_Page object supports up to 4 buttons. To do this:

- 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):

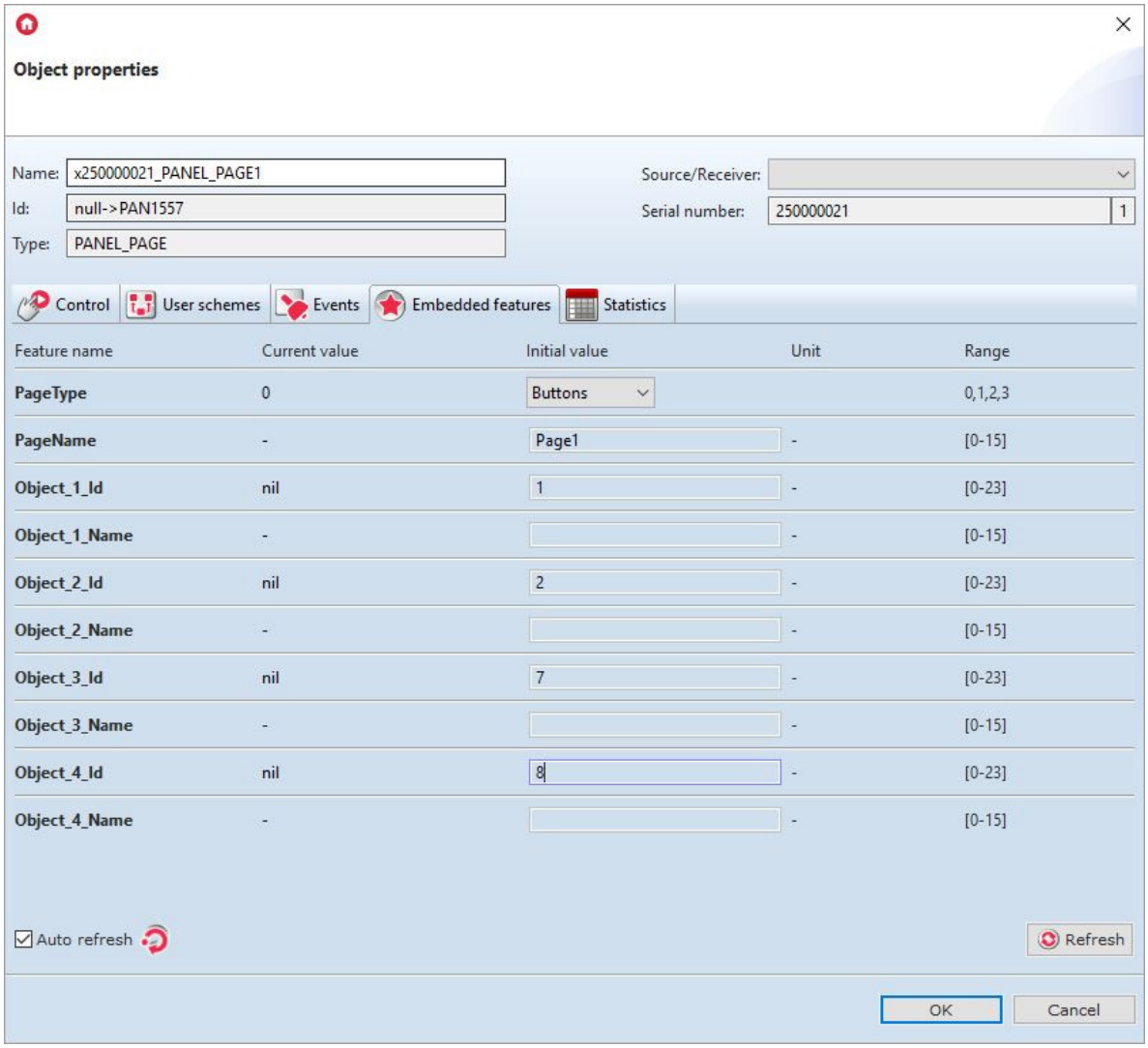


Note!
For page type *Buttons*, the `OnDraw` event is not generated.

- 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*



  - `Object_X_Name` - name of the thermostat. For the page type *Buttons*, the feature should be left blank;

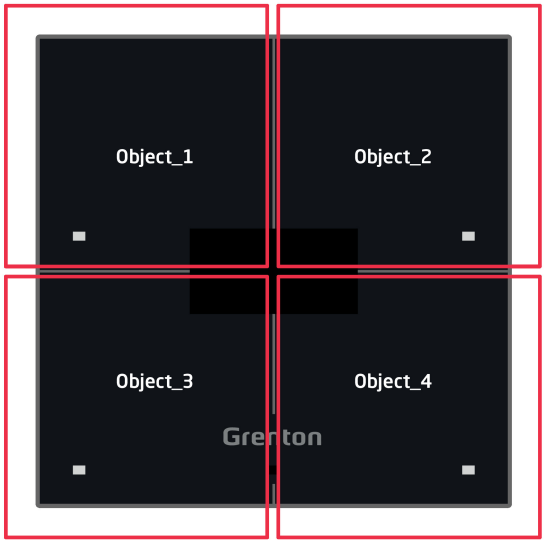


Note!
Sending the configuration only with the defined page type, without setting the binding of objects with the buttons is connected with starting the panel operation mode as *Buttons*. However, the buttons on the module will be inactive. This is related to the non-complementing of the Object_X_Id features.

- 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()
```

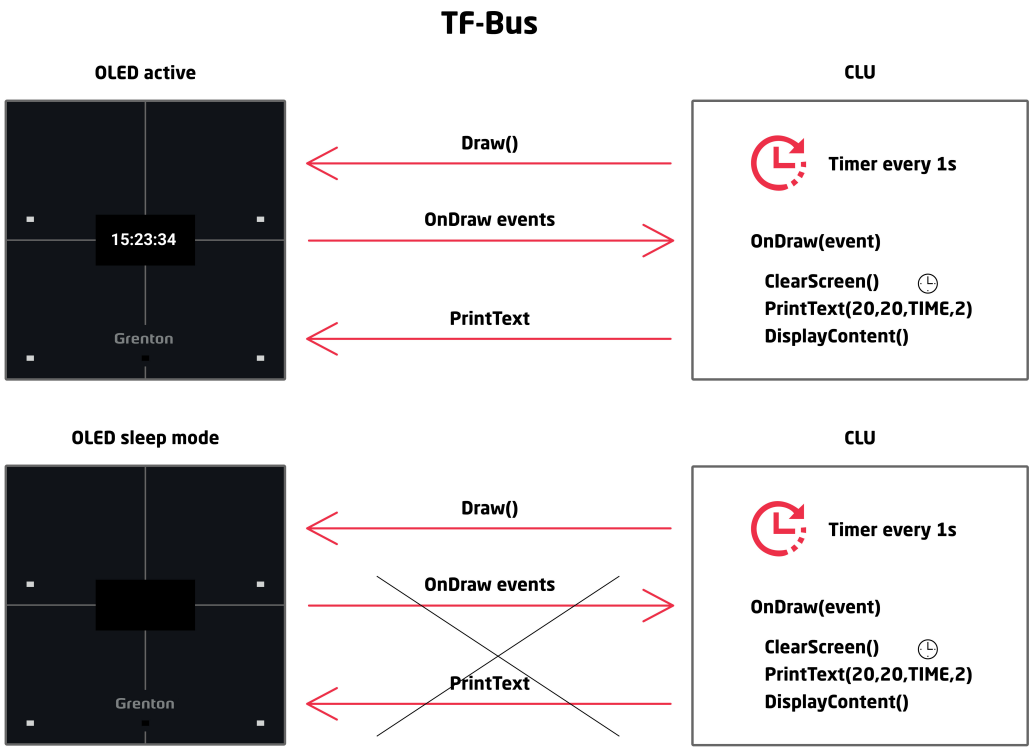
Note!
A restriction has been introduced in the drawing mechanism. CLU Z-Wave expects 2 seconds to finish drawing with the DisplayContent method. Otherwise, the following message will be displayed on the screen:

"page: `PageName`

free draw

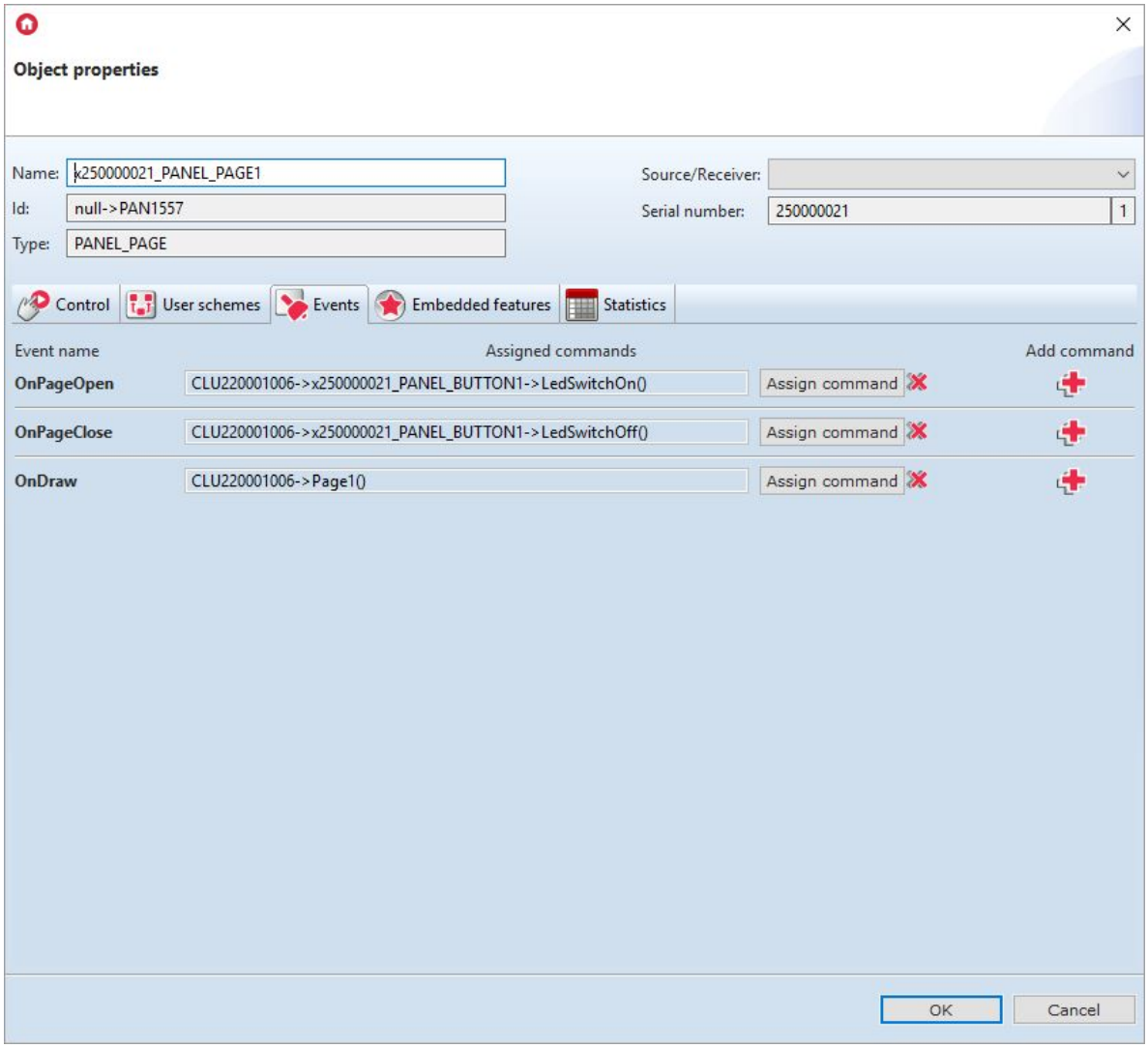
! TIMEOUT !"

The following figure shows the current drawing mechanism.



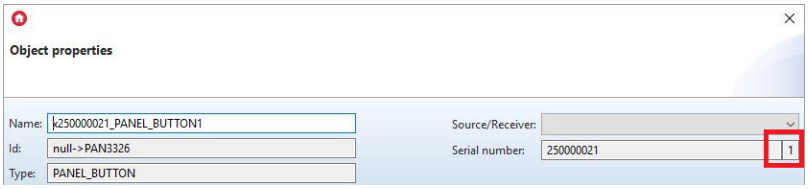
The next step in creating the configuration is configuring Panel_Page objects depending on the number of buttons. One Panel_Page object supports up to 4 buttons. To do this:

- 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):

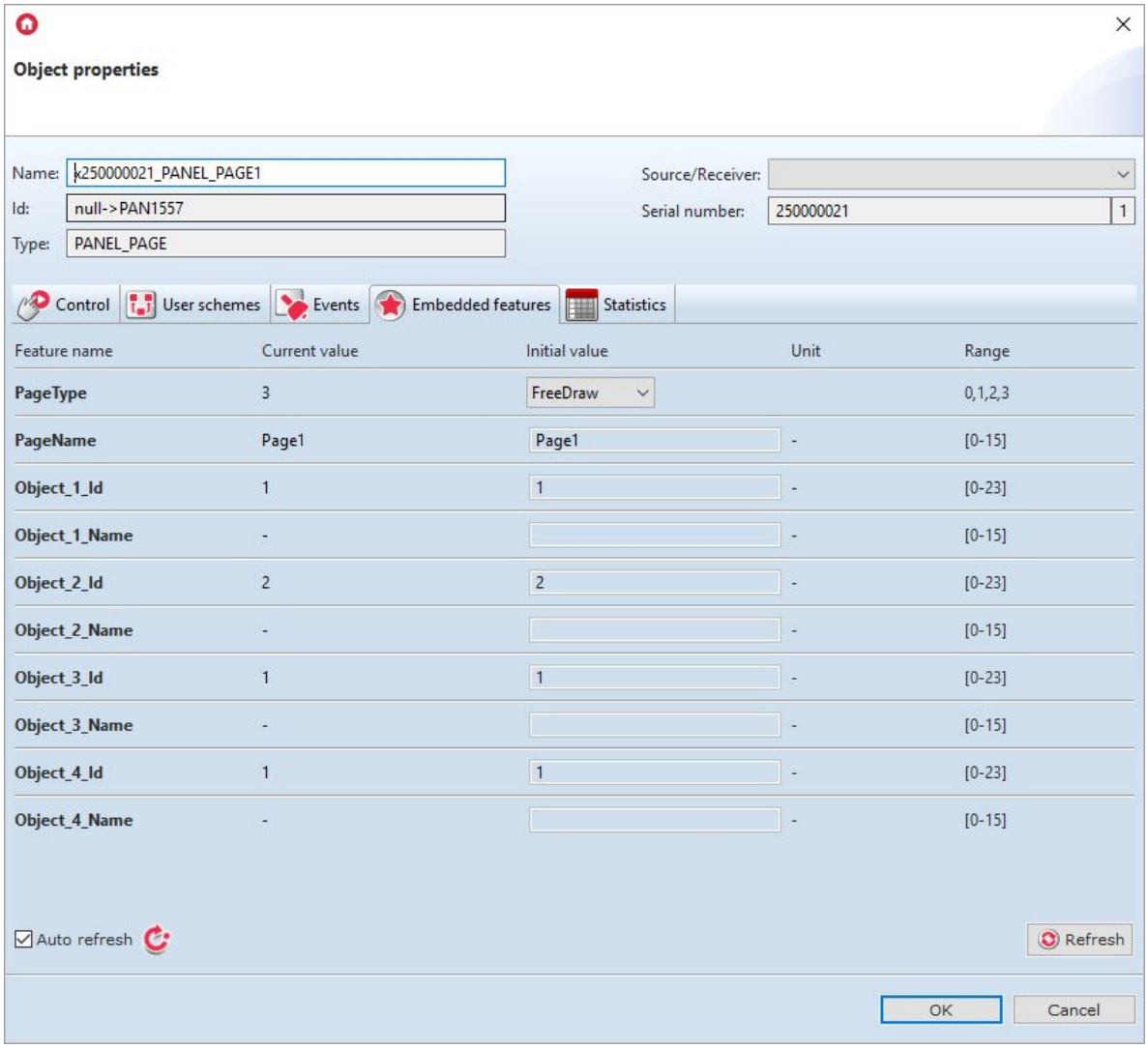


Note!
For the page type *FreeDraw*, complete the `OnDraw` event.

- 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;



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.

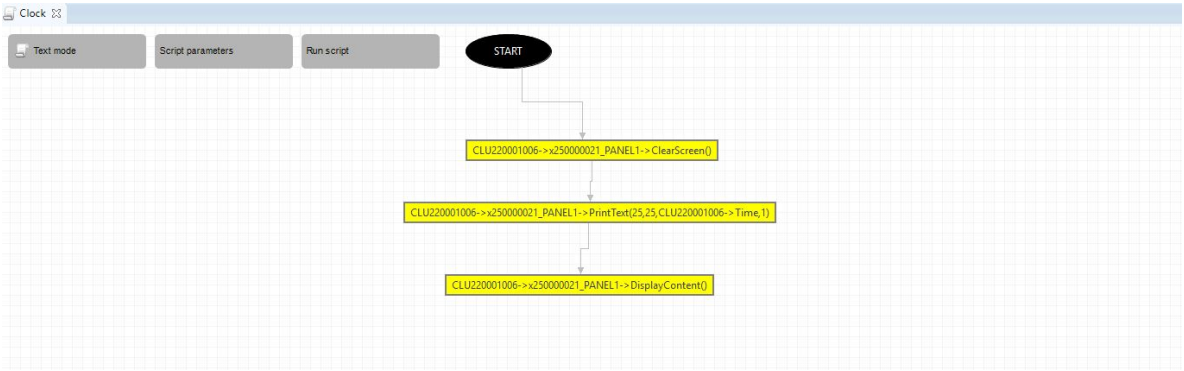
- Send the configuration to the CLU Z-Wave.

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.

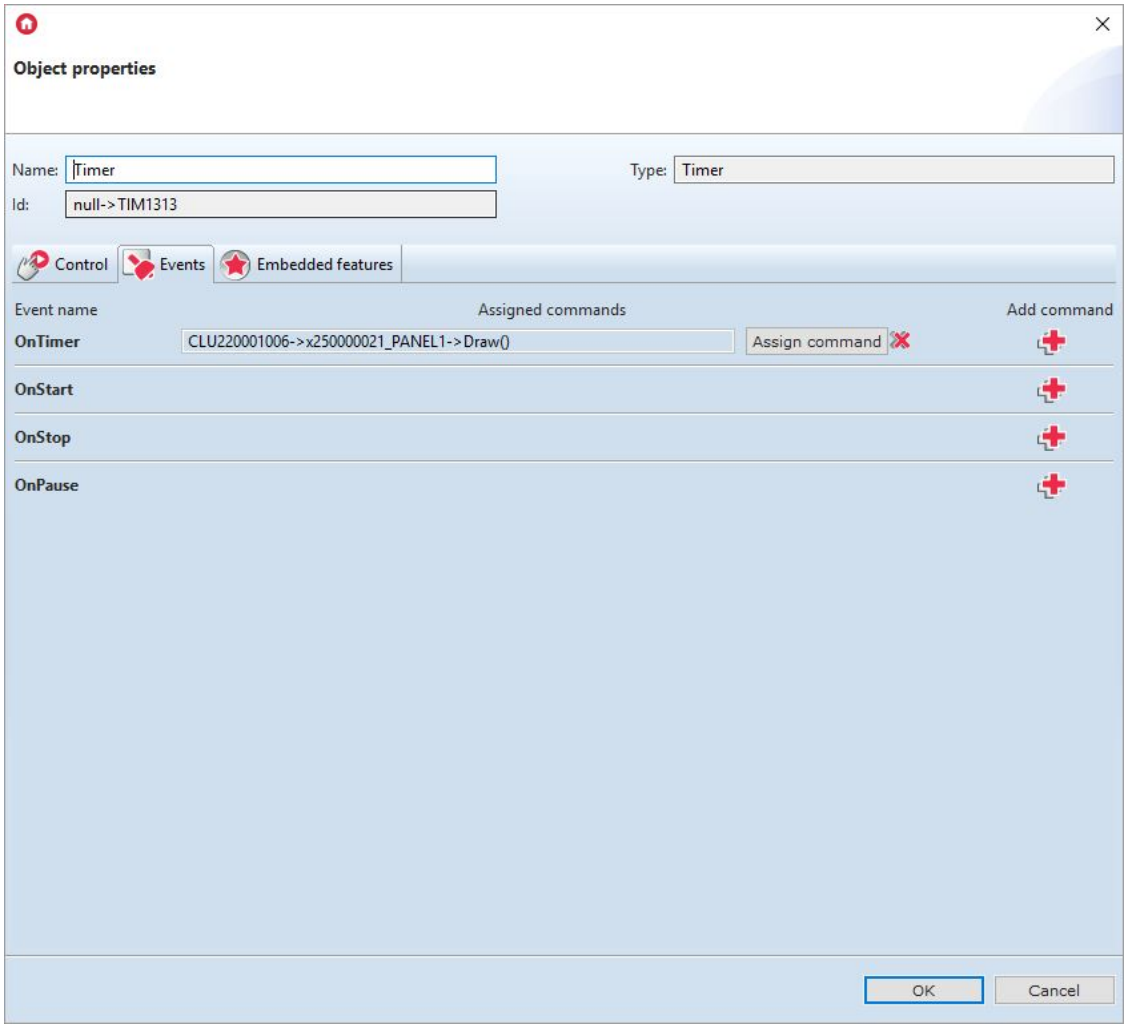
B. Set up the site as a clock

To configure the site as a clock:

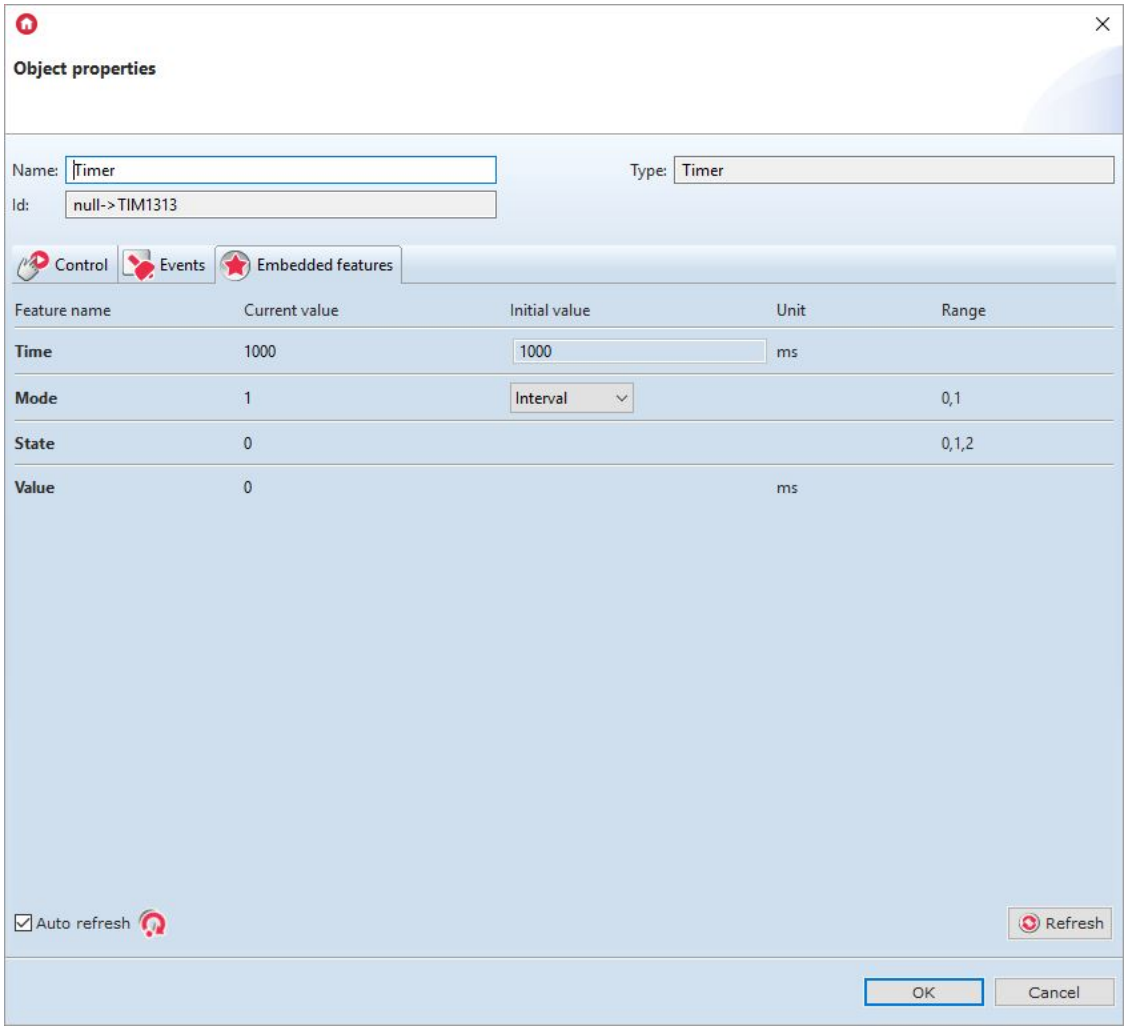
- Create a script displaying the current time (*Clock*);



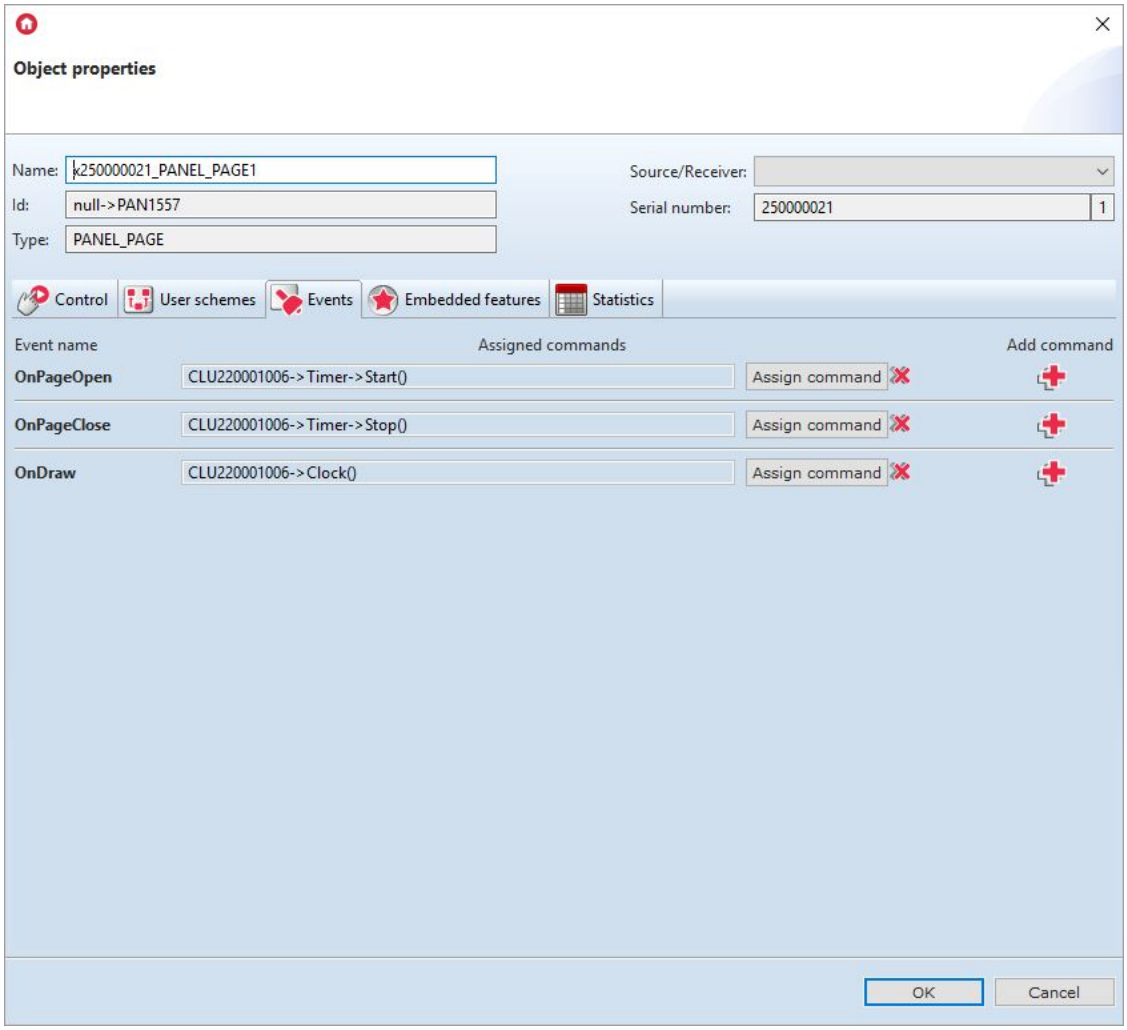
- Create a virtual object Timer:
  - Go to the tab *Events*;
  - Configure the operation of the virtual object by assigning the `Draw` method of the *Panel* object to the `OnTimer` event:



  - Select the Embedded features tab and define the configuration parameters of the object:



- 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.

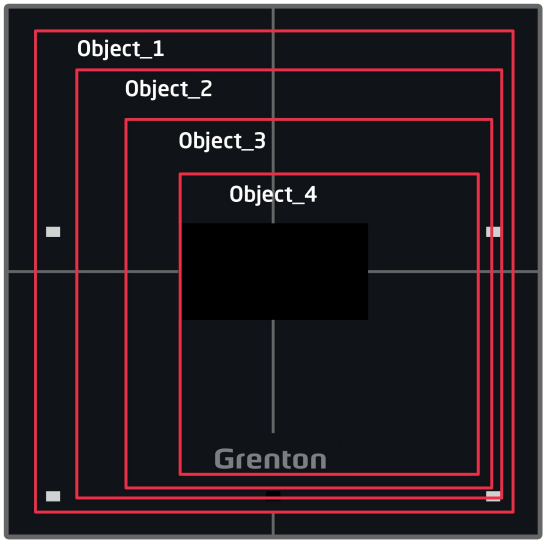
Script *Clock* in text version:

```
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)`).

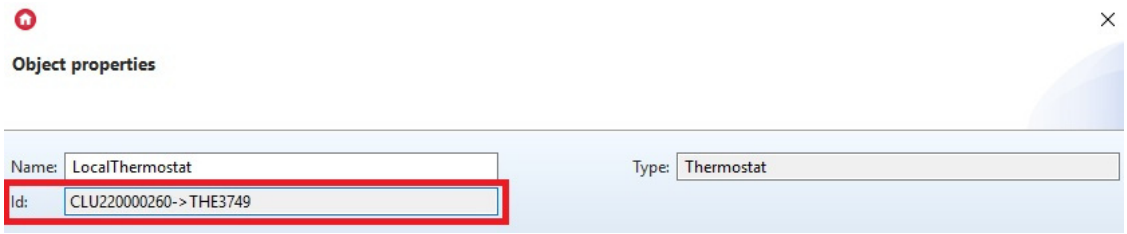
A. Creating a configuration with a local thermostat

To create a configuration using a local thermostat you should:

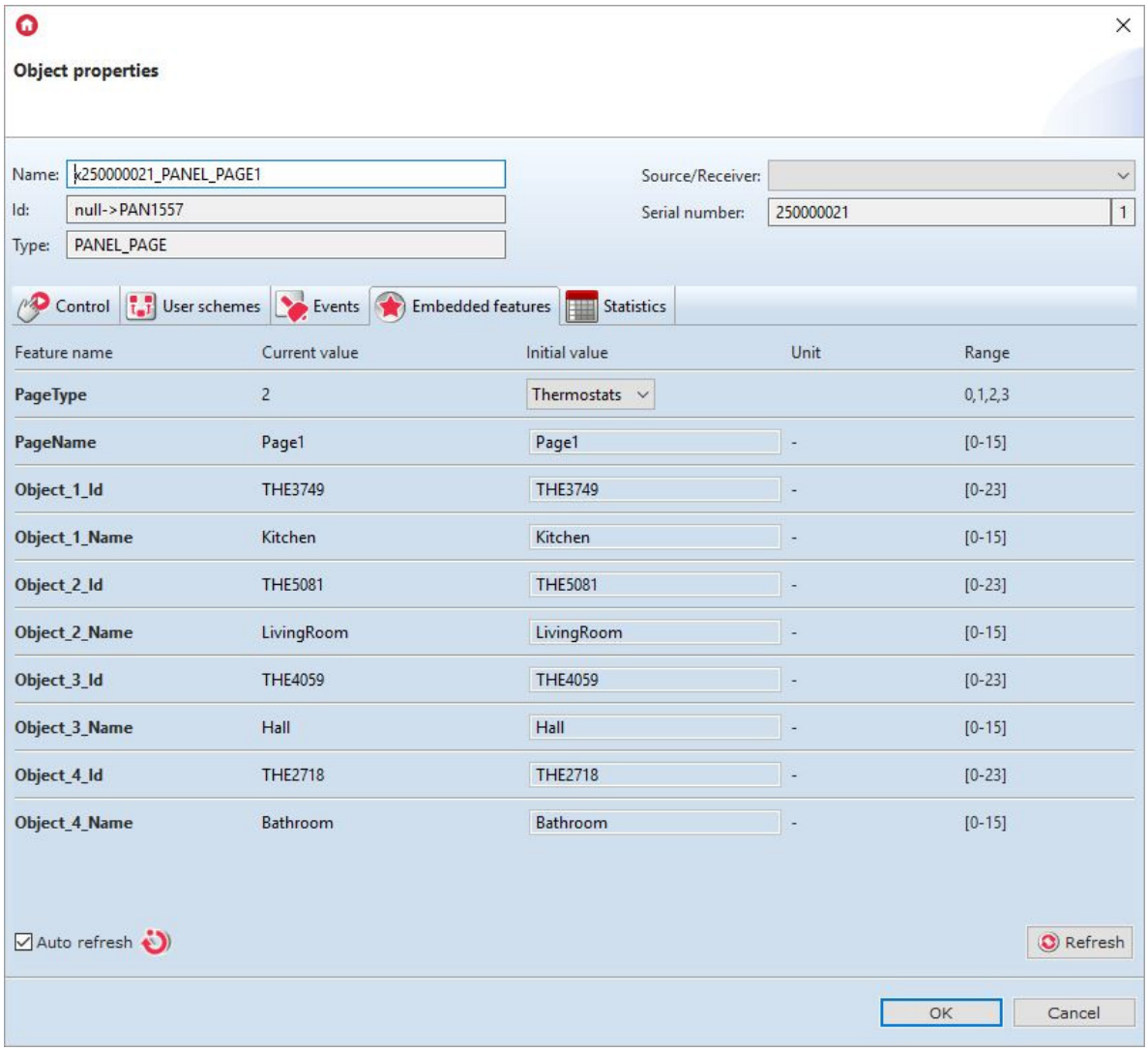
- Create a thermostat on the Z-Wave CLU, to which the Smart Panel module is connected;
- Configure the virtual object as intended;
- Open object *PANEL_PAGEX* (where X is the number of one of 4 pages) by double clicking on the list of objects
- Select the tab *Embedded features* and define the objects 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:



- `Object_X_Name` - name of the thermostat. The lack of a thermostat name in the parameter causes that the thermostat is not displayed;



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`.

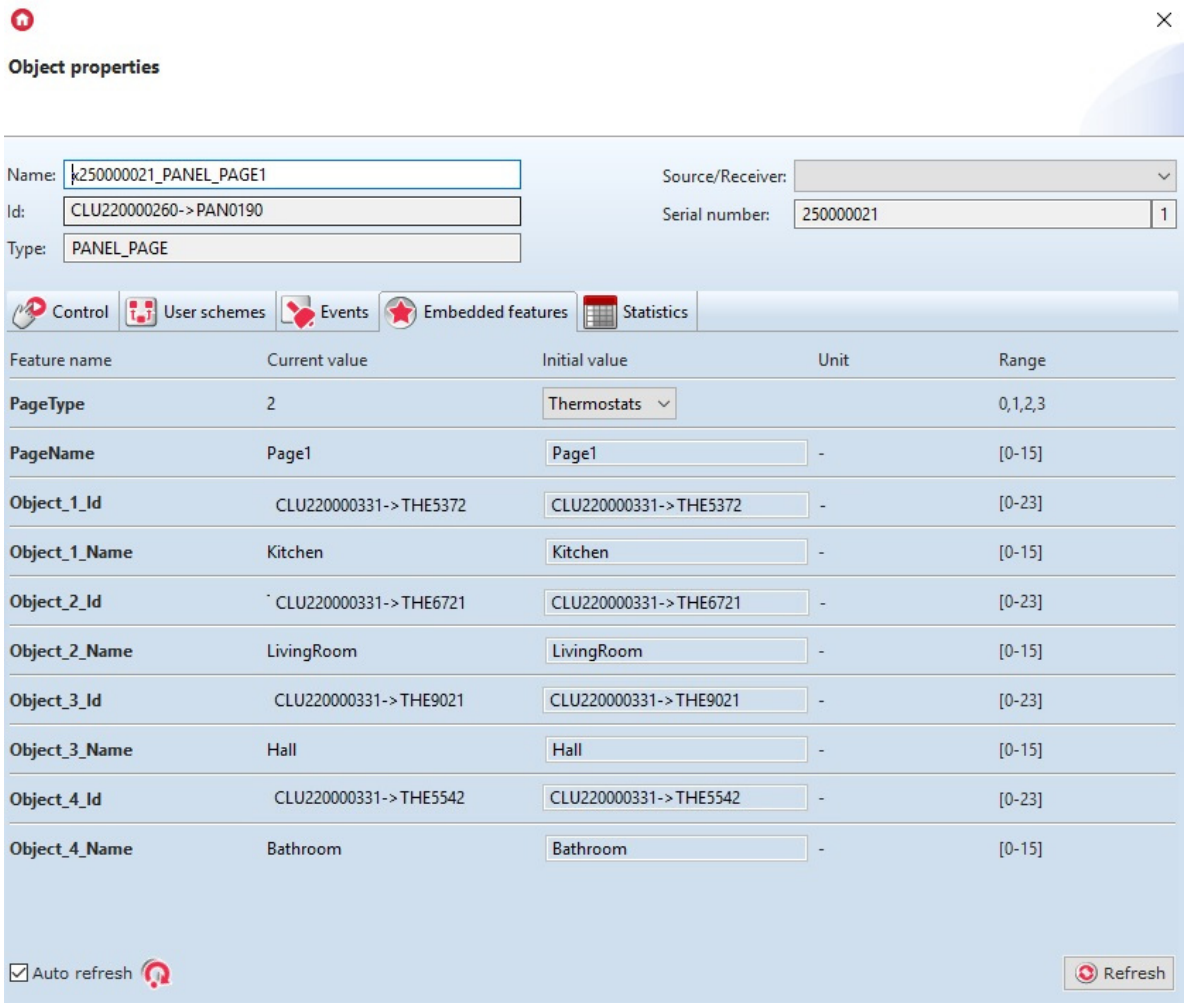
- Send the configuration to the CLU Z-Wave.

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:

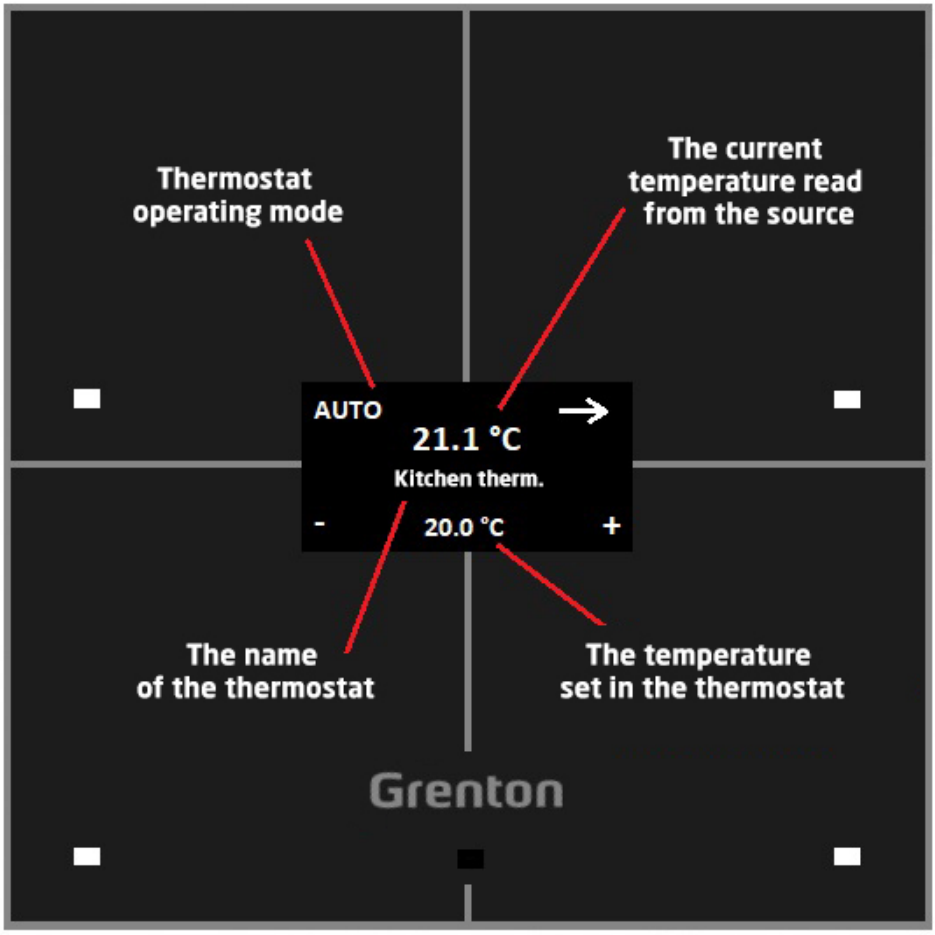
  - `Object_X_Name` - name of the thermostat. The lack of a thermostat name in the parameter causes that the thermostat is not displayed;



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 (`PointValue`) by 0.1 ° C
Lower left	Long press (hold)	Reduction of the set temperature (`PointValue`) - as long as the button is held down
Lower right	Short press (click)	Increasement of the set temperature (`PointValue`) by 0.1 ° C
Lower right	Long press (hold)	Increasement of the set temperature (`PointValue`) - 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_BUTTONX* 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

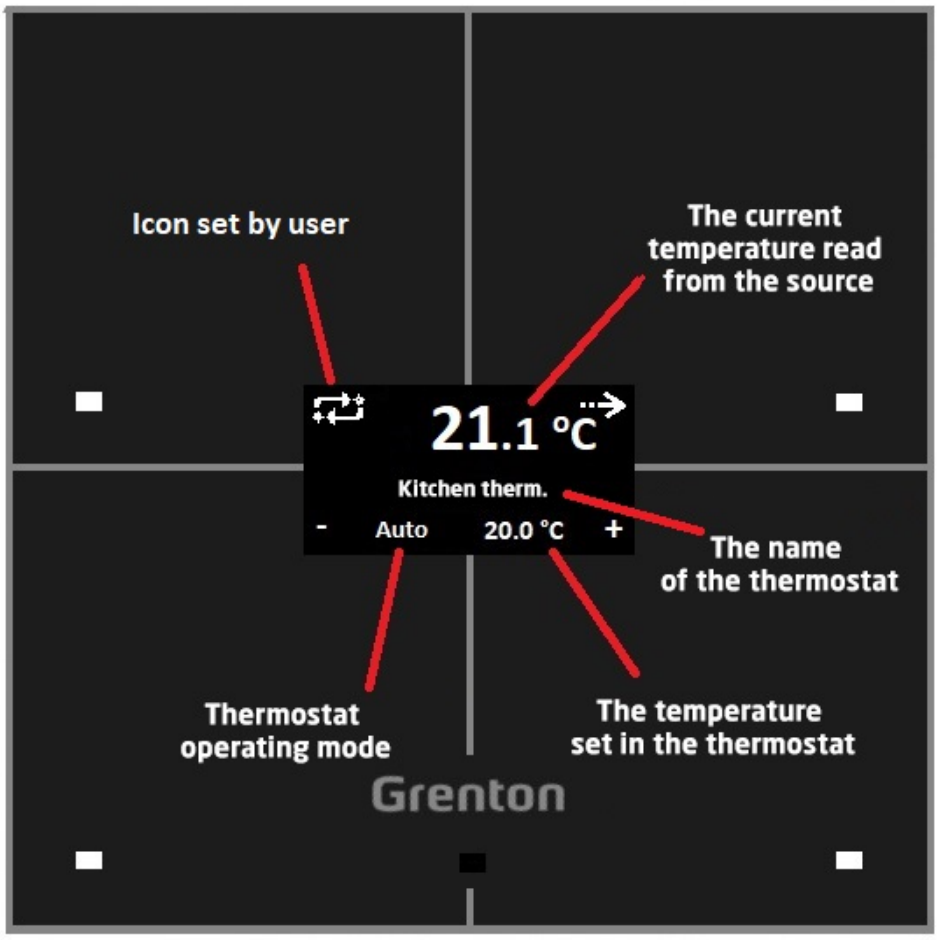
The Smart Panel module in version v6 offers a refreshed UI of Thermostats websites, as well as new functions. The method of creating and configuring a page with the use of Thermostats type objects is the same as for the previous versions - see XII.5.9.

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.

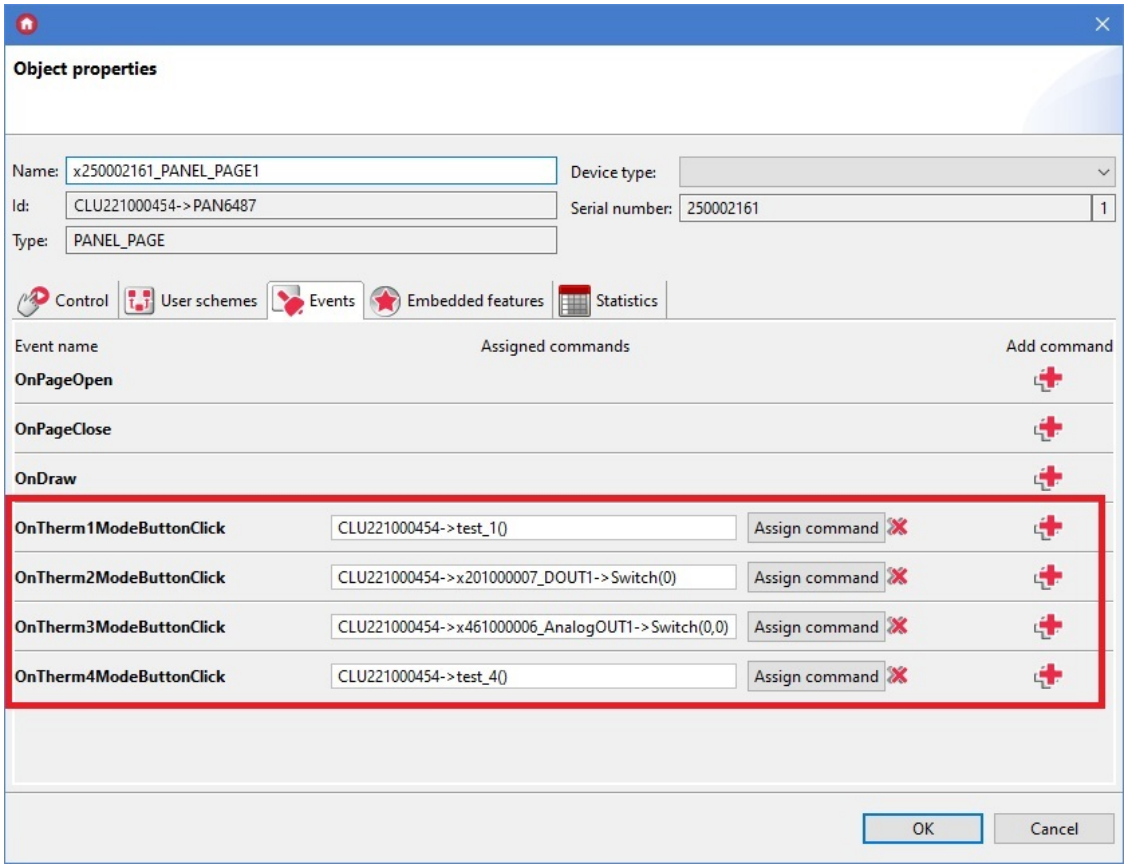
1. **Possibility to set your own icons**

   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.



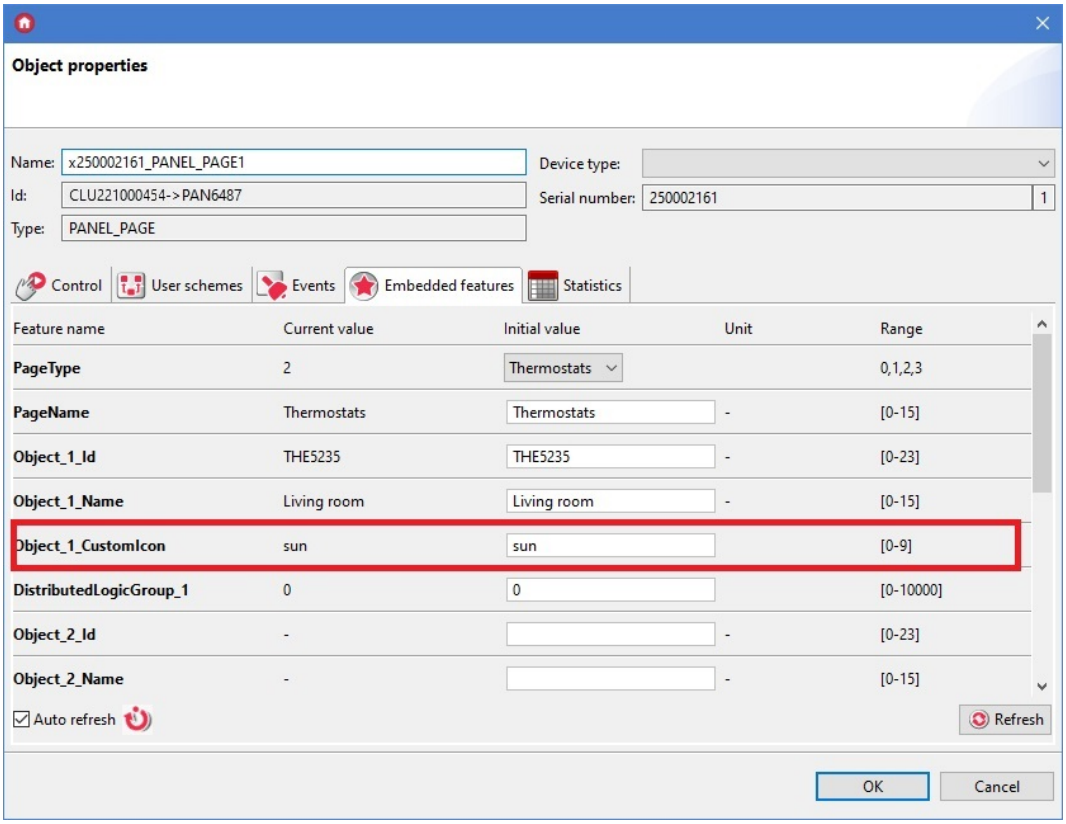
3. **An example of configuration of new functionalities**

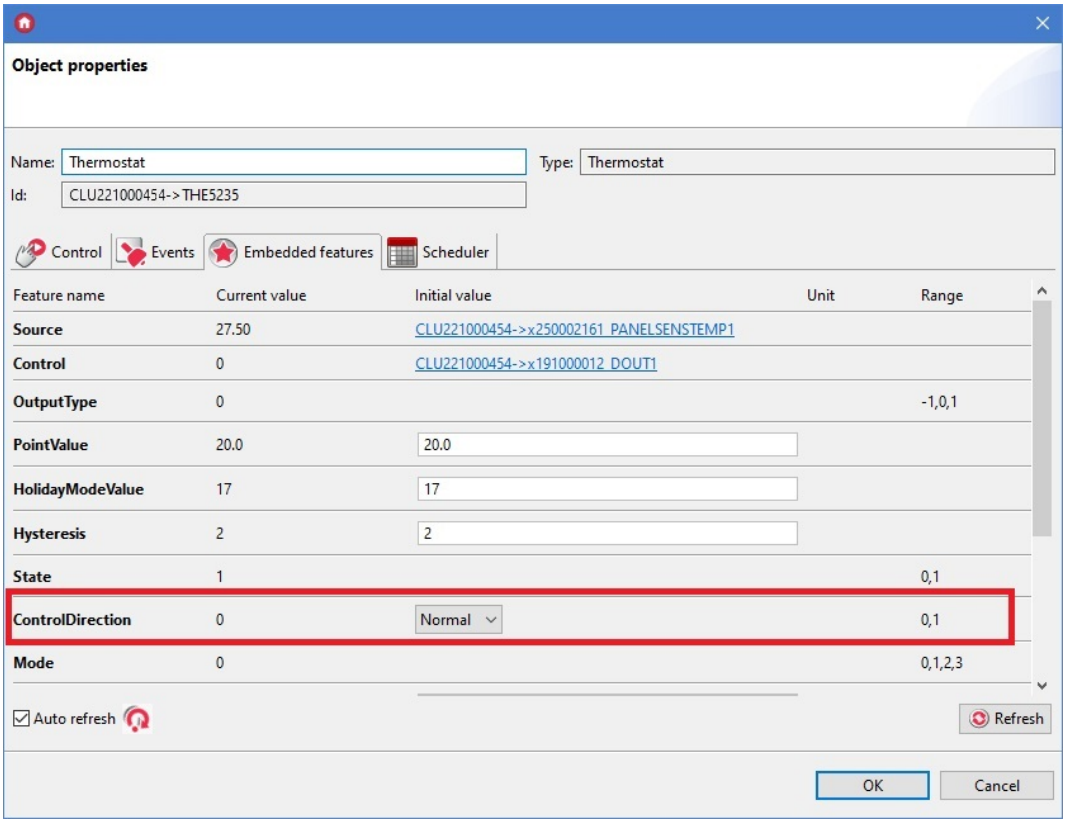
   1. Local thermostat configuration with operation change (heating / cooling)

      The following objects are used to create this configuration:

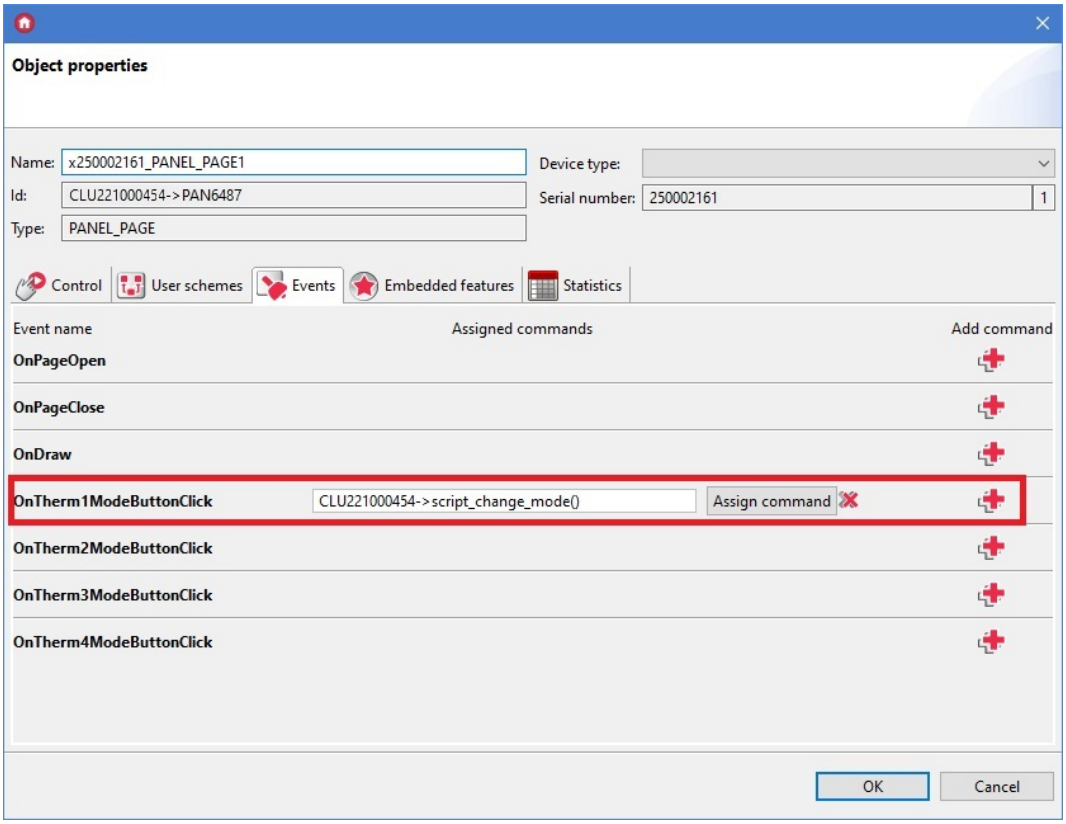
      - Thermostat virtual object,
      - DOUT1 object (e.g. Relay module) - responsible for heating / cooling activation - used in the Thermostat virtual object,
      - DOUT2 object (e.g. Relay module) - selection of what is to be switched on: heating or cooling - used in scripts.

      The following screenshot shows the configuration of the PANEL_PAGE object and the Thermostat virtual object in the default setting, which is heating.

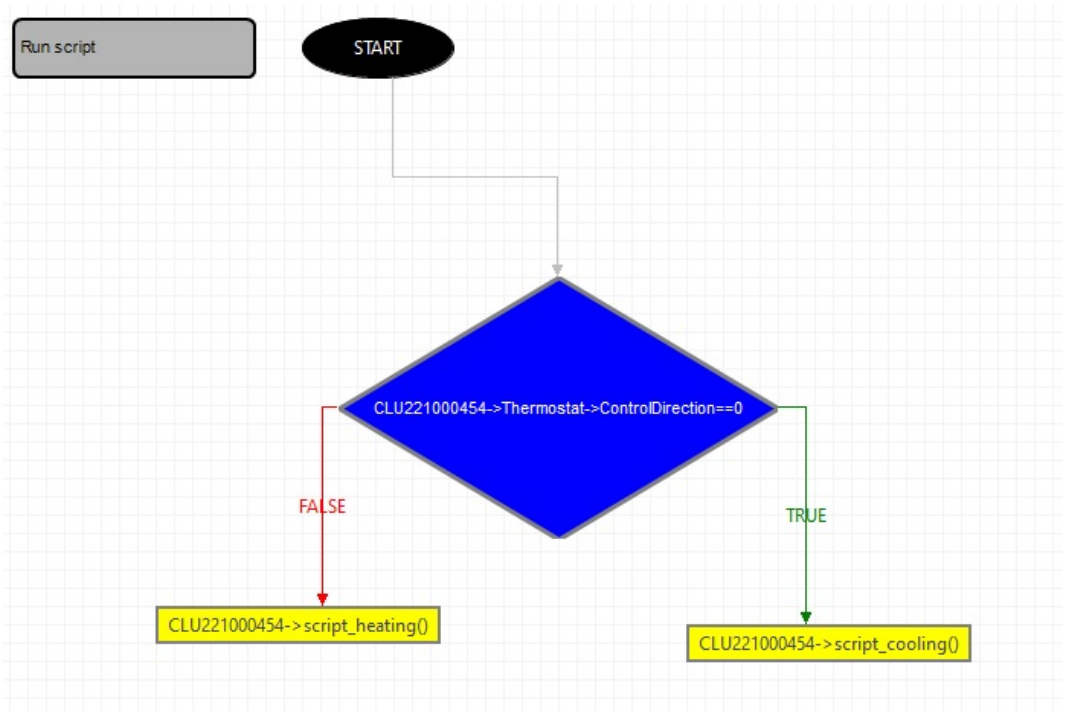




Configuration of the OnTherm1ModeButtonClick event:



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 (`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 (`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 (`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 (`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 1.4.2-2346 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 Alarm 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.

> **Note!**
> When the `UseCloud` feature is set to `true`, the `UseNTP` feature is automatically set to `true`.

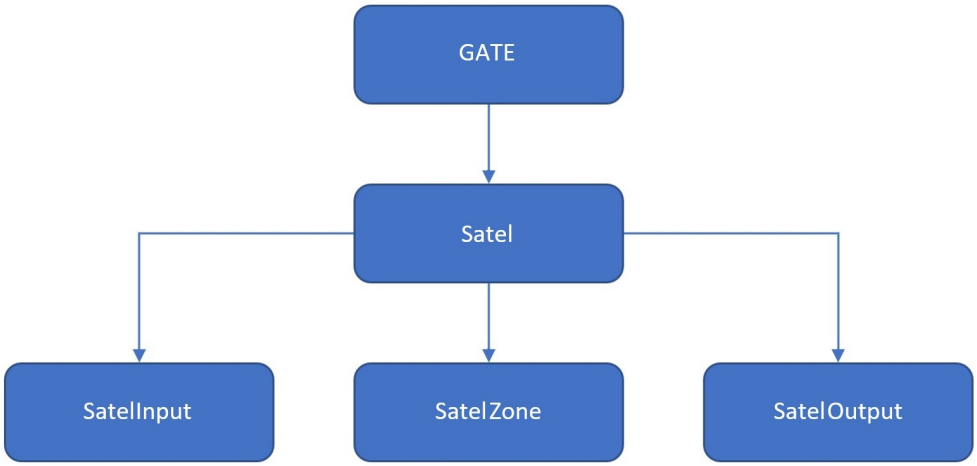
3. Integration with the Satel alarm control panel

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 virtual objects of the type: *SatelZone*, *SatelInput*, *SatelOutput*. It is also possible to use the integration coding offered by Satel.

> **Note!**
> There is no limit to the number of objects for the created virtual objects - the limitation is the device memory, which is influenced by e.g. level of logic expansion on the module. The **Satel** object is an exception - only one can be created.

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.

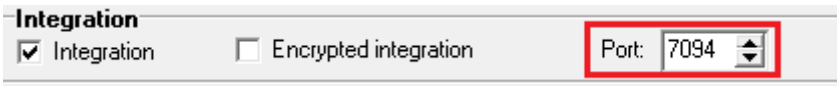
3.2. Configuration for the Satel system

> **Note!**
> An interface database update is required before any work with the GATE Alarm module!

> **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.

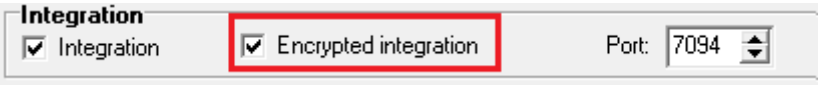
Before starting the configuration you should have information about the Satel central unit and the ETHM-1 module:

- 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)*;

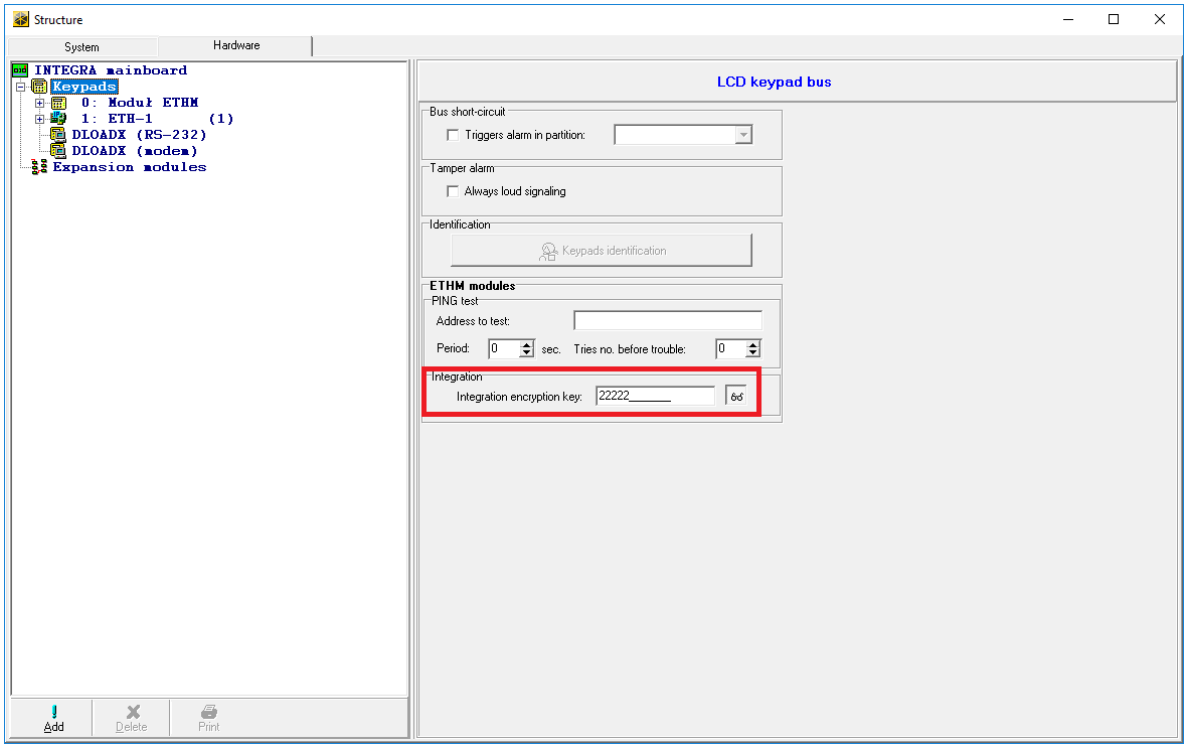


- Administrator / users password - the default password in the Satel configuration for the administrator is: 1111 *(DLOADX -> Users -> Users)*;
- Integration on the side of the ETHM **module must be enabled** *(DLOADX -> Data -> Structure and Hardware -> Equipment tab -> Keypads -> ETHM-1 -> Integration 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)*.

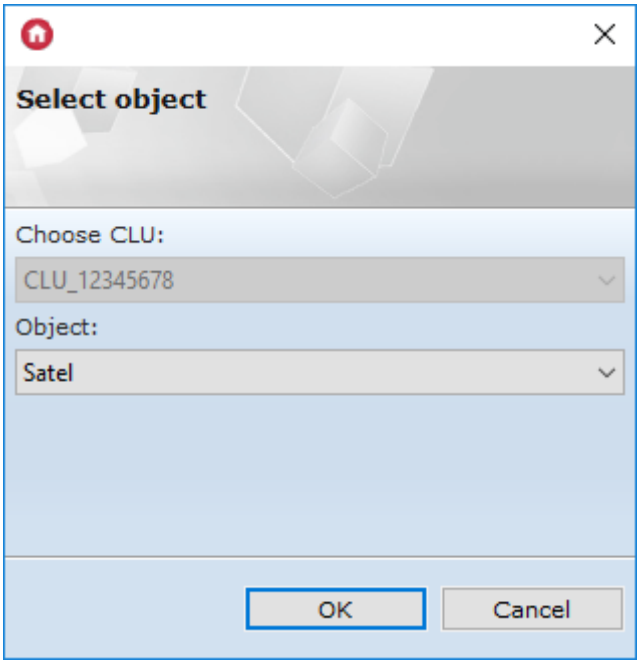


3.3. Virtual Objects

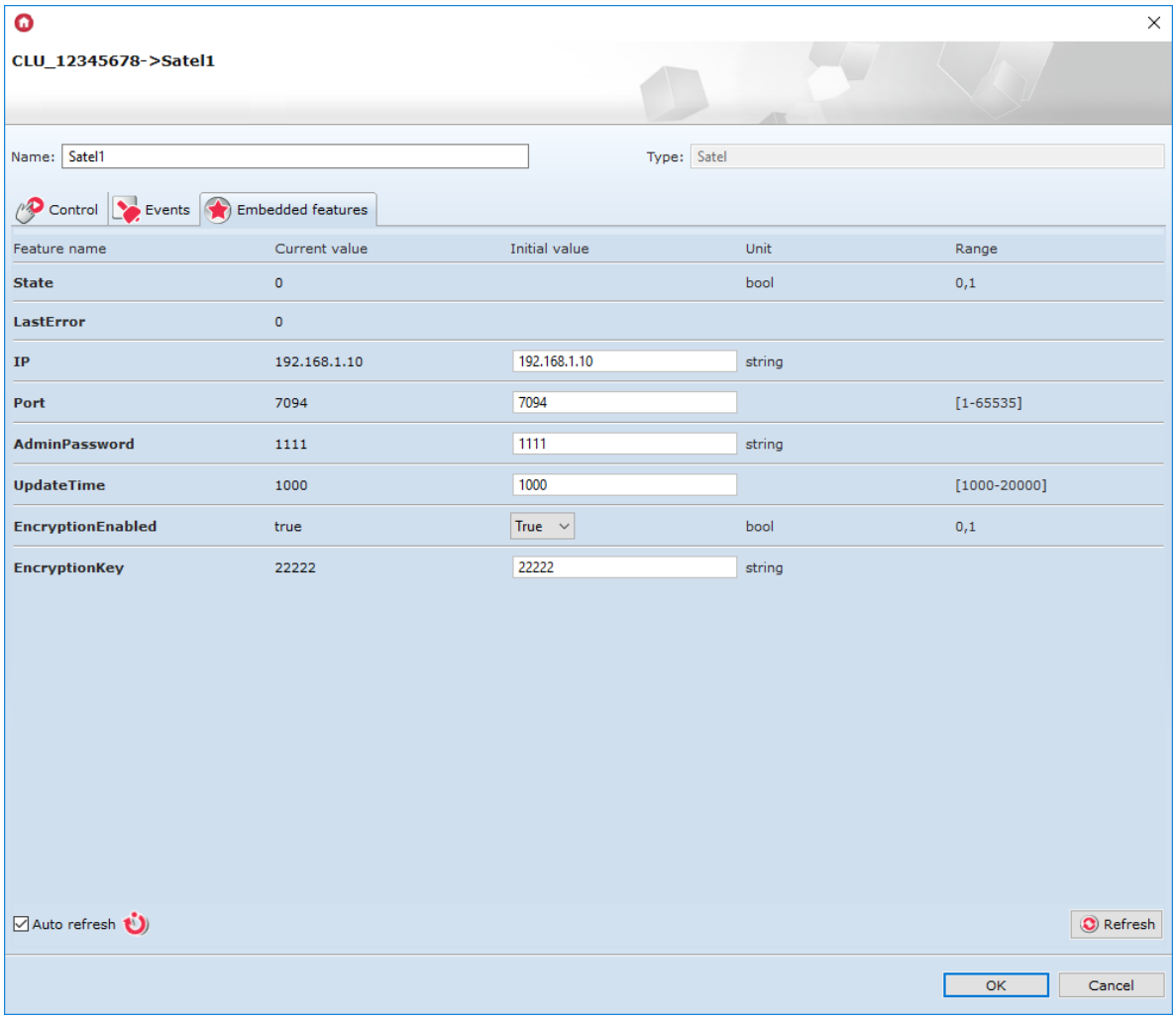
A. Satel

To perform the correct configuration of the GATE Alarm module it is necessary to:

- Create a virtual object *Satel*:

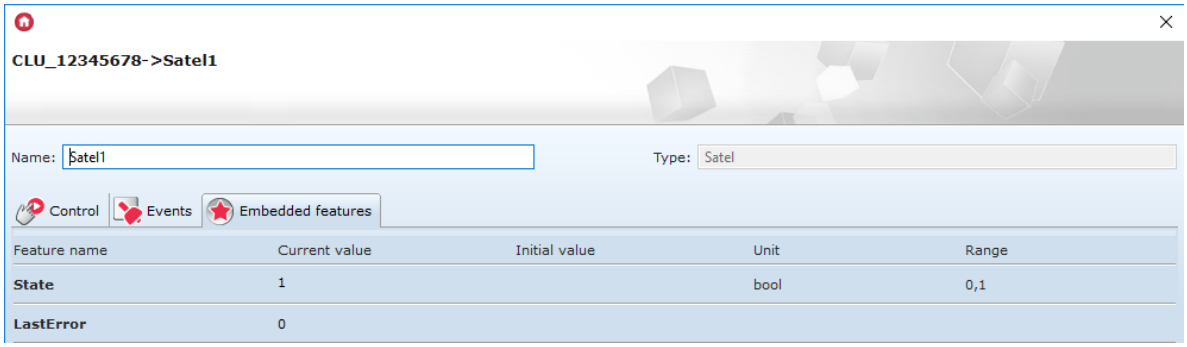


- Go to configuration - tab *Embedded features* and enter:
    - **IP** - IP address of the ETHM module (Satel);
    - **Port** - ETHM integration port;
    - **AdminPassword** - administrator password;
    - **EncryptionEnabled** - enable encoding - set if the integration on the ETHM module is marked with *Integration Encoding*;
    - **Encryption Key** - integration key (for attached encoding):



Information on where to find the information you need can be found in the second section - look up XIII.1.2..

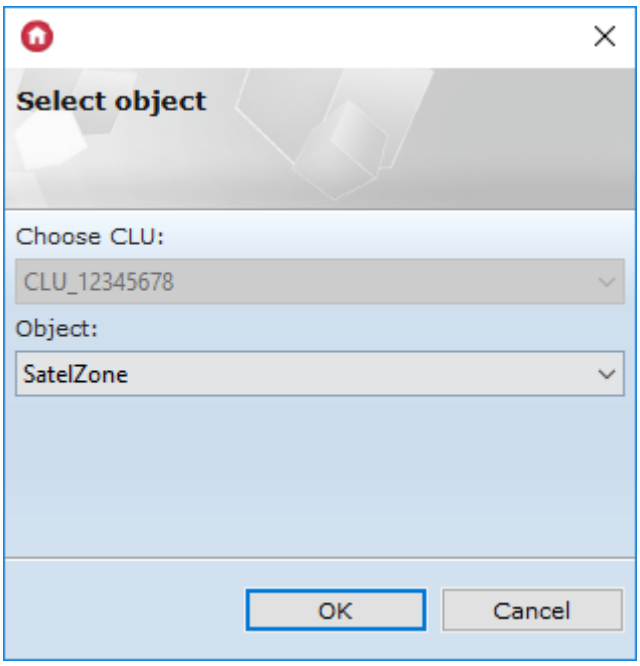
- Send configuration and verify connection - tab *Embedded features*, 'State' feature (1 - correctly connected to the control panel, 0 - no connection):



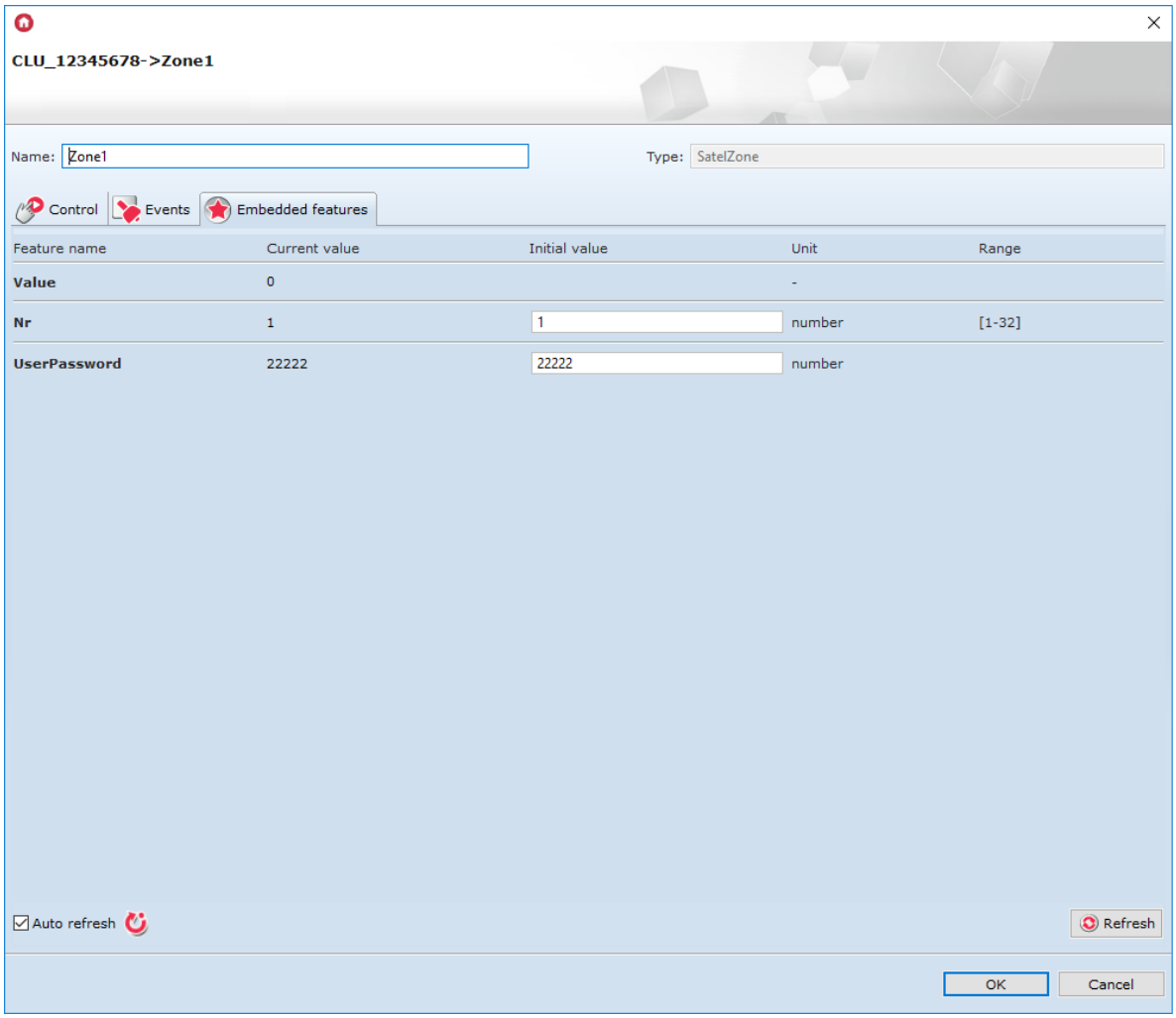
B. Zone

The GATE Alarm module allows you to add a virtual object *Zone*:

- Create an object *SatelZone*:



- Define the No. (number of the selected zone) and enter the user's password:

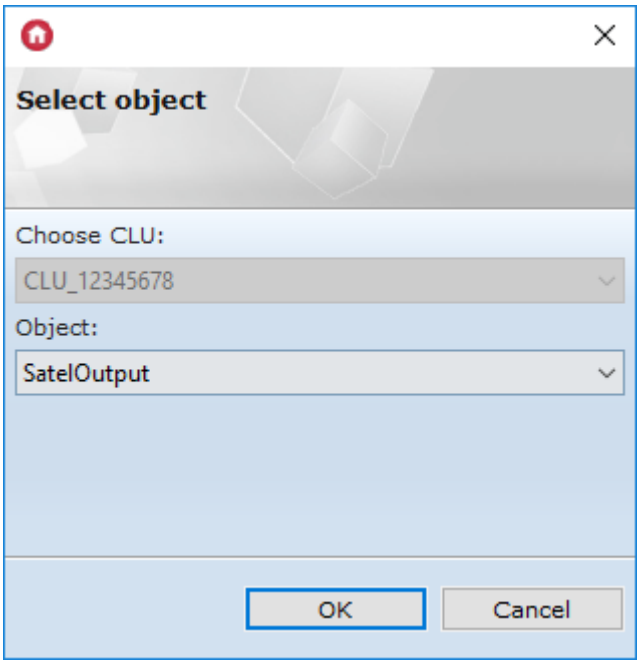


- Send configuration and verify the connection - tab *Built-in features*, the `Value` feature (-1 means no connection to the control panel, the others mean a correct connection and the state of the zone is returned: 0 or 1);
- Arm / disarm the zone - the `ArmZone` and `DisarmZone` methods.

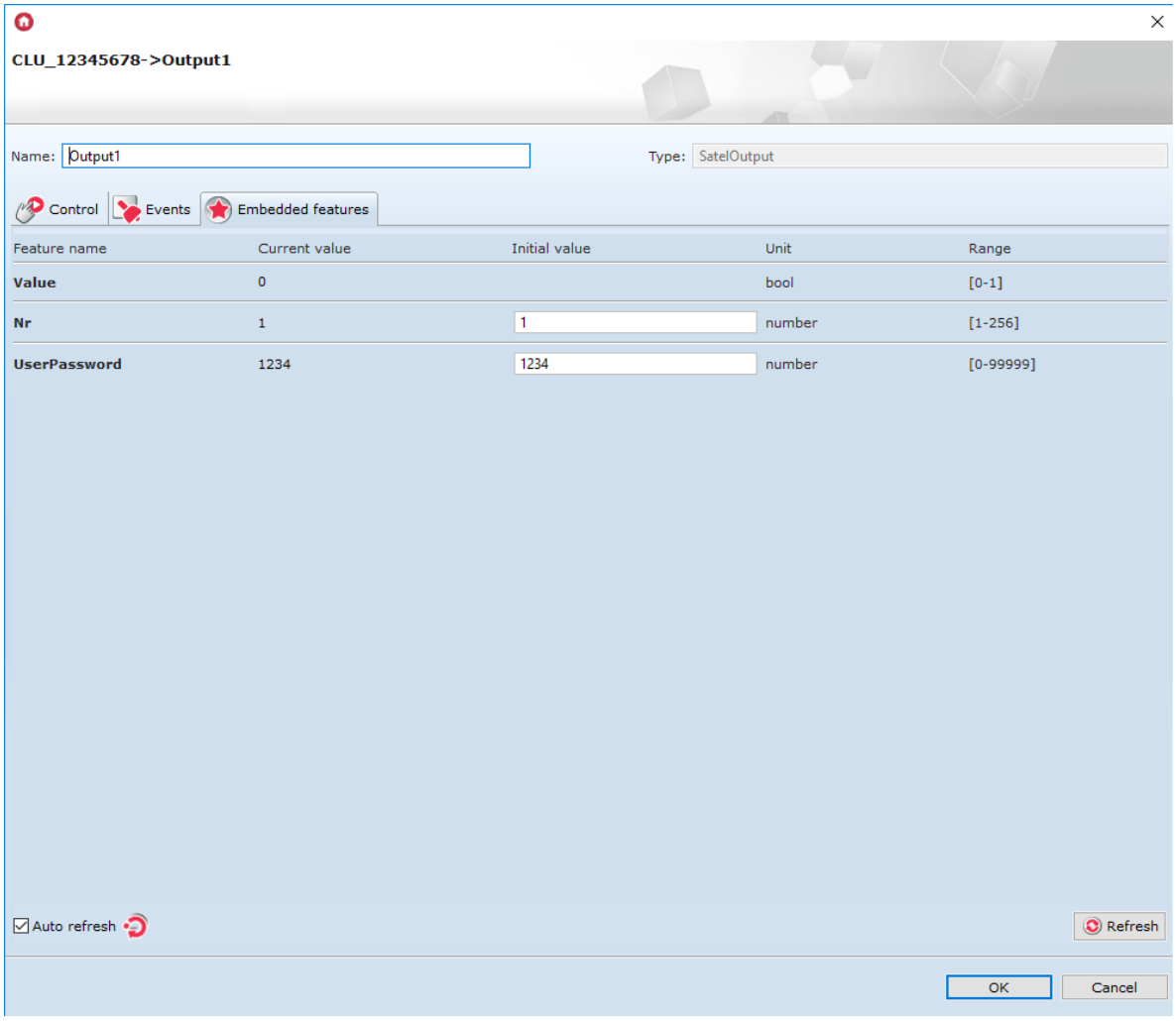
C. Output

GATE Alarm allows adding a virtual object *Output*:

- Create an *SatelOutput* object:



- Define the No. (number of the selected output on the Satel board) and enter the user's password:

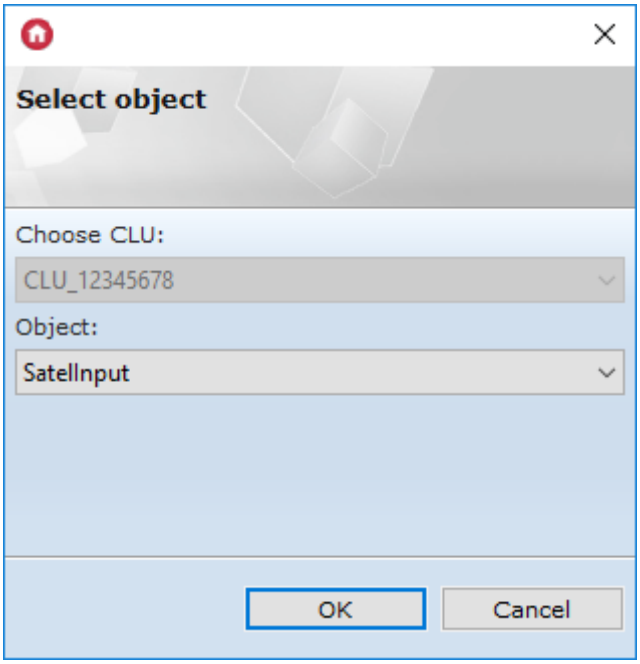


- Send configuration and verify the connection - tab *Embedded features*, the `Value` feature (-1 means no connection to the central unit, the others mean a correct connection and the status of the zone is returned: 0 or 1);
- Switch on / off the output - methods `SwitchOn` and `SwitchOff`.

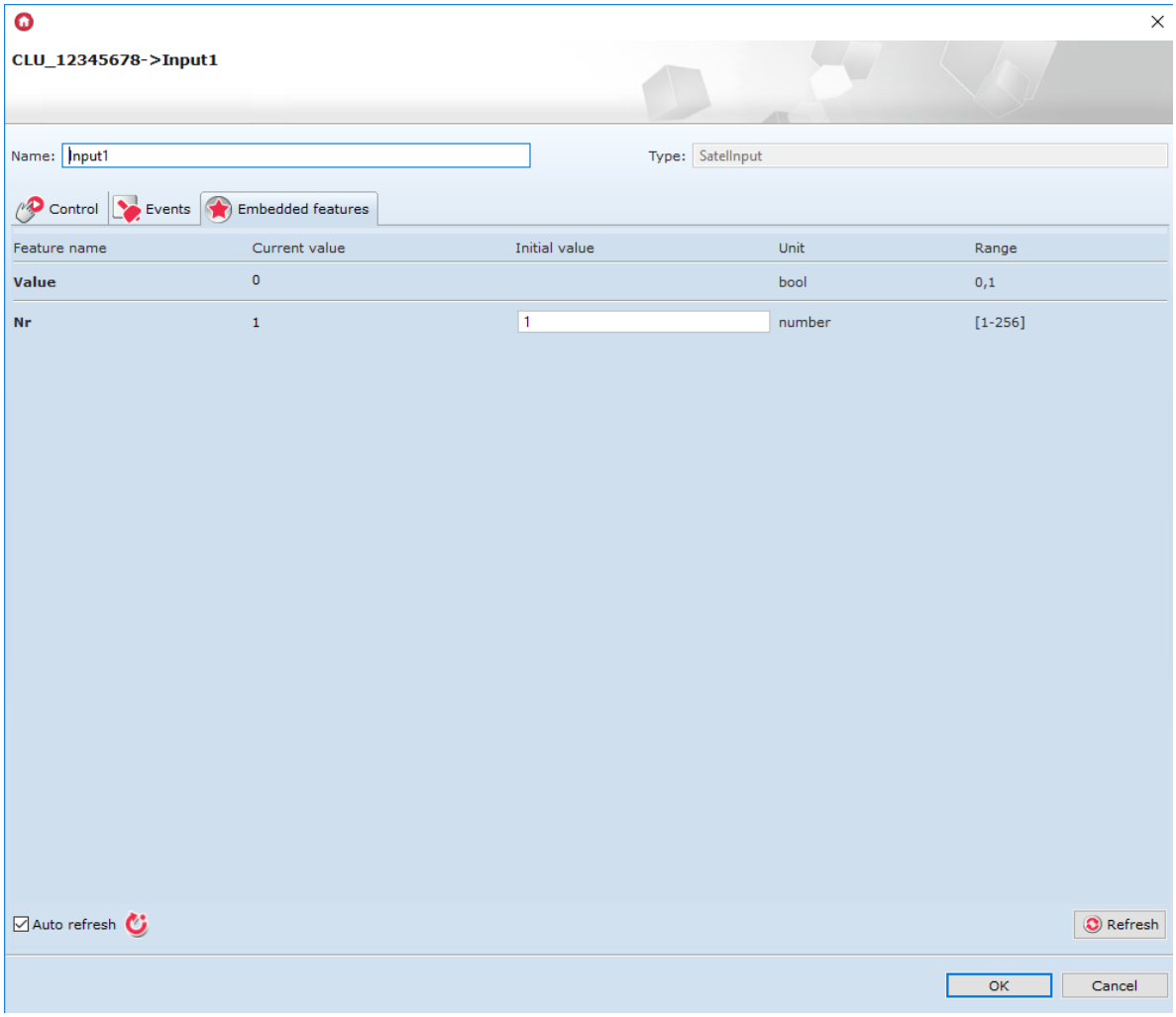
D. Input

GATE Alarm allows adding a virtual object *Input*:

- Create an *SatelInput* object:



- Define No. (number of the selected input on the Satel):



- 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

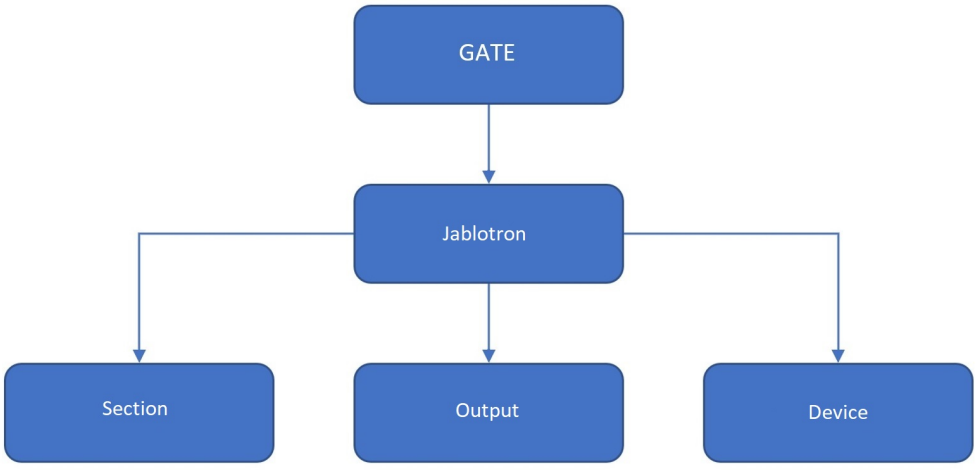
> **Note!**
> The described functionality and integration with the mentioned alarm control panels is available for **GRENTON GATE ALARM, DIN, Eth (INT-221-E-01)** with **firmware 1.4.2-2346 or higher!**

> **Note!**
> Integration of the Grenton system concerns the control panels:
>
> - **Jablotron JA-100K**,
> - **Jablotron JA-101K**,
> - **Jablotron JA-103K**,
> - **Jablotron JA-107K**.

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.*

> **Note!**
> There is no limit to the number of objects for the created virtual objects - the limitation is the device memory, which is influenced by e.g. level of logic expansion on the module. The **Jablotron** object is an exception - only one can be created.

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.

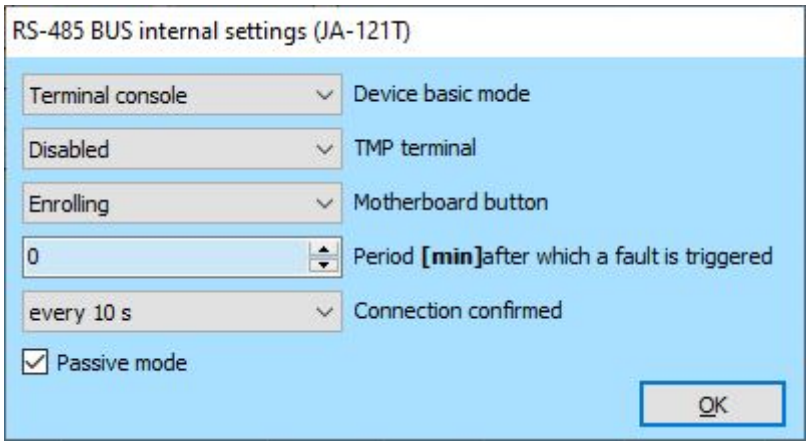
4.2. Configuration for the Jablotron system

> **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.



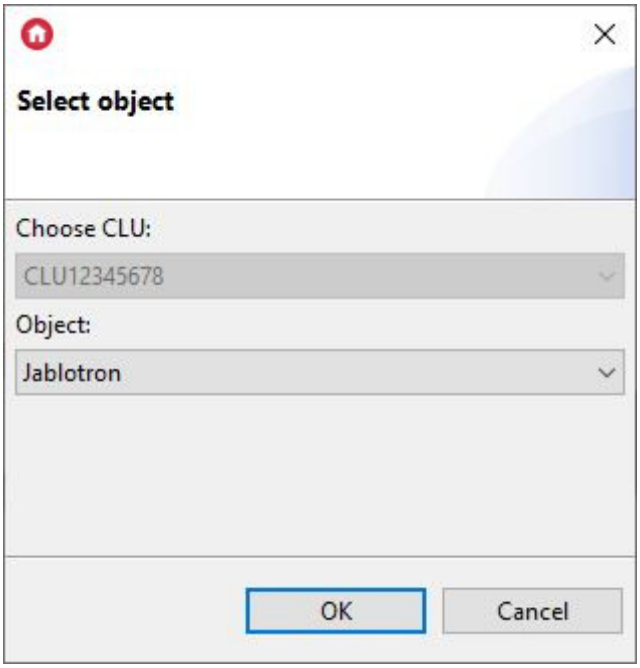
Note!
Information on the remaining settings and their application can be found in the documentation of the JA-121T module on the manufacturer's website.

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

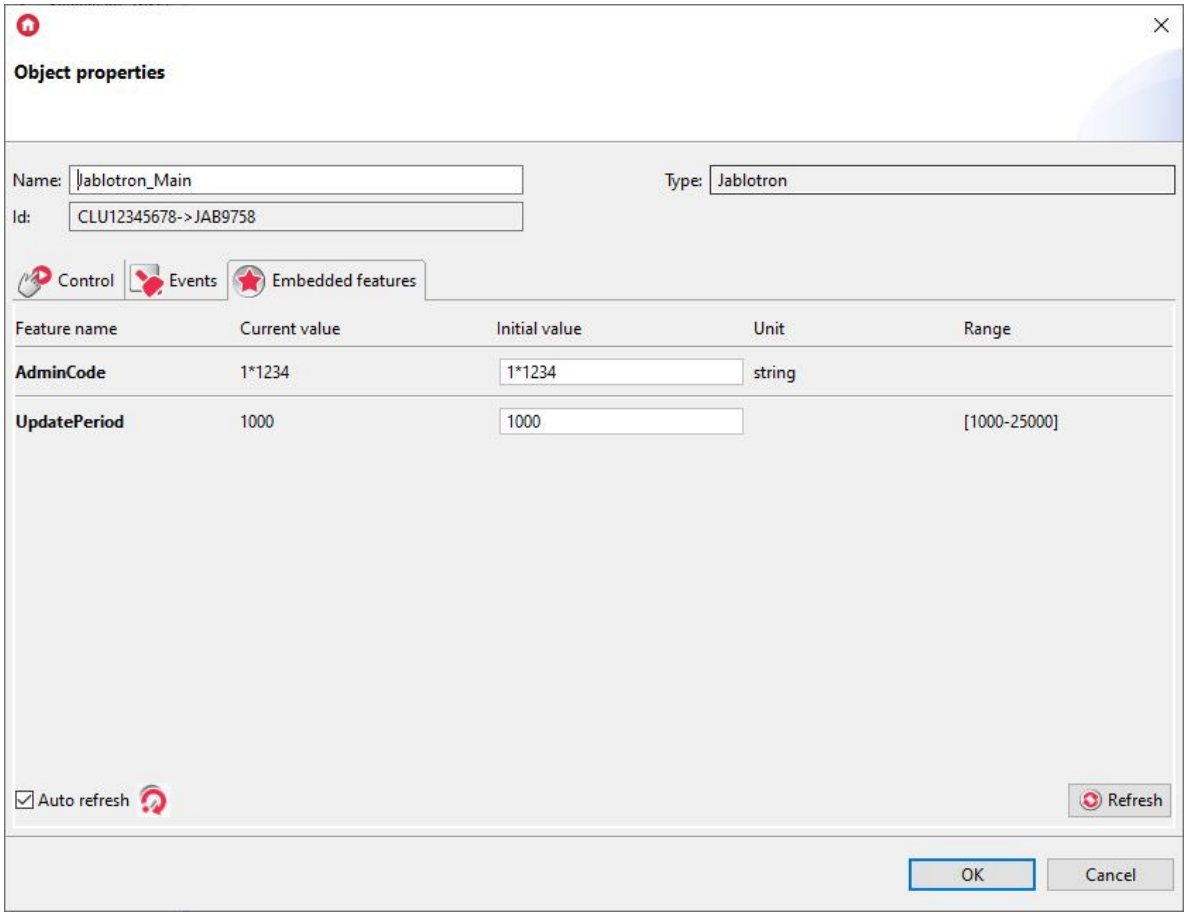
A. Jablotron

In order to properly configure the GATE Alarm module, you must create the virtual **Jablotron** object:



Then go to configuration - Built-in Features tab and enter

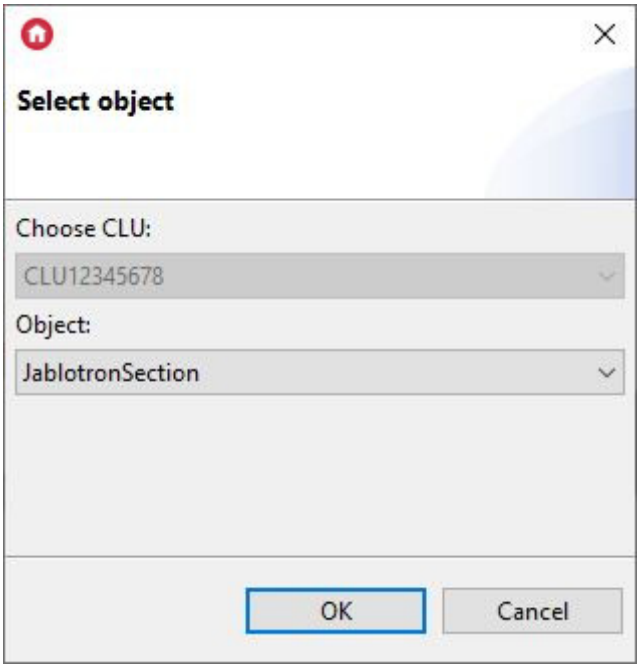
- **AdminCode** - administrator access code;
- **UpdatePeriod** - Control panel status update period;



B. Section / zone

GATE Alarm allows you to add a virtual **JablotronSection** object:

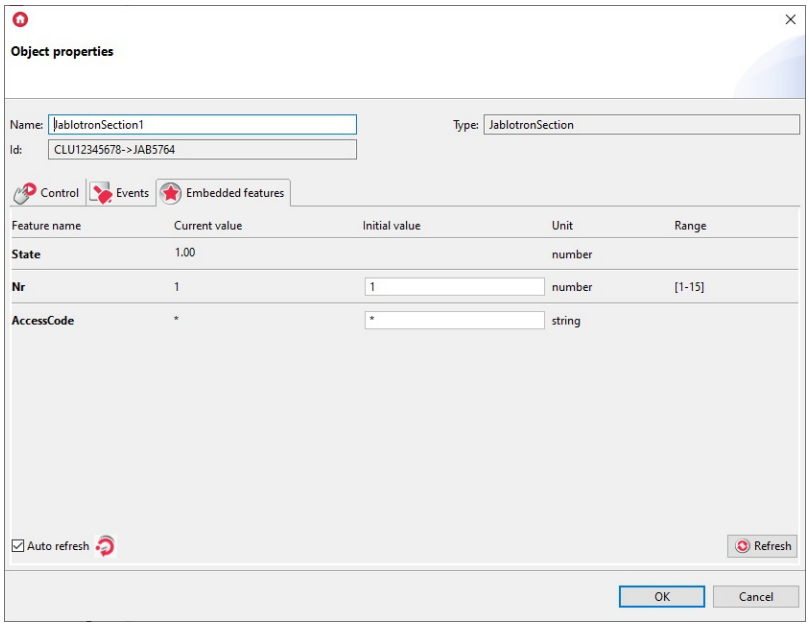
Creating the *JablotronSection* object:



Then go to configuration - Built-in Features tab and enter:

- **Nr** - parameter defining to which section the object refers;

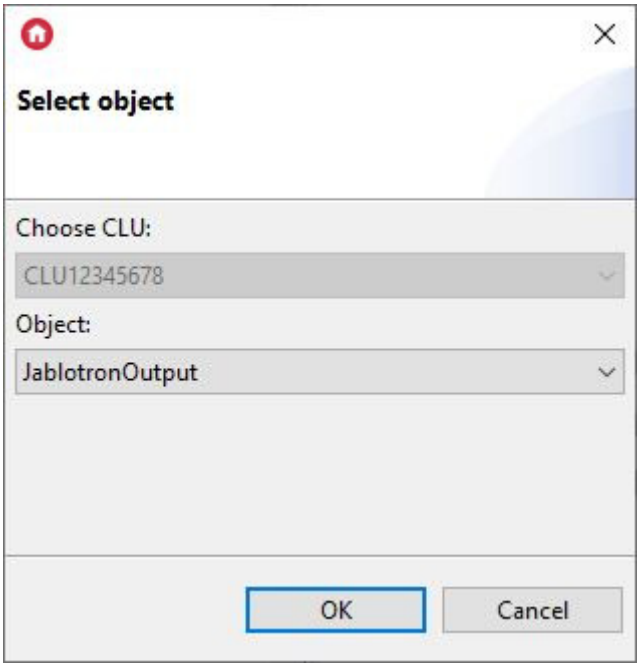
- **UpdatePeriod** - The access code will use the admin code for '*';



C. Output

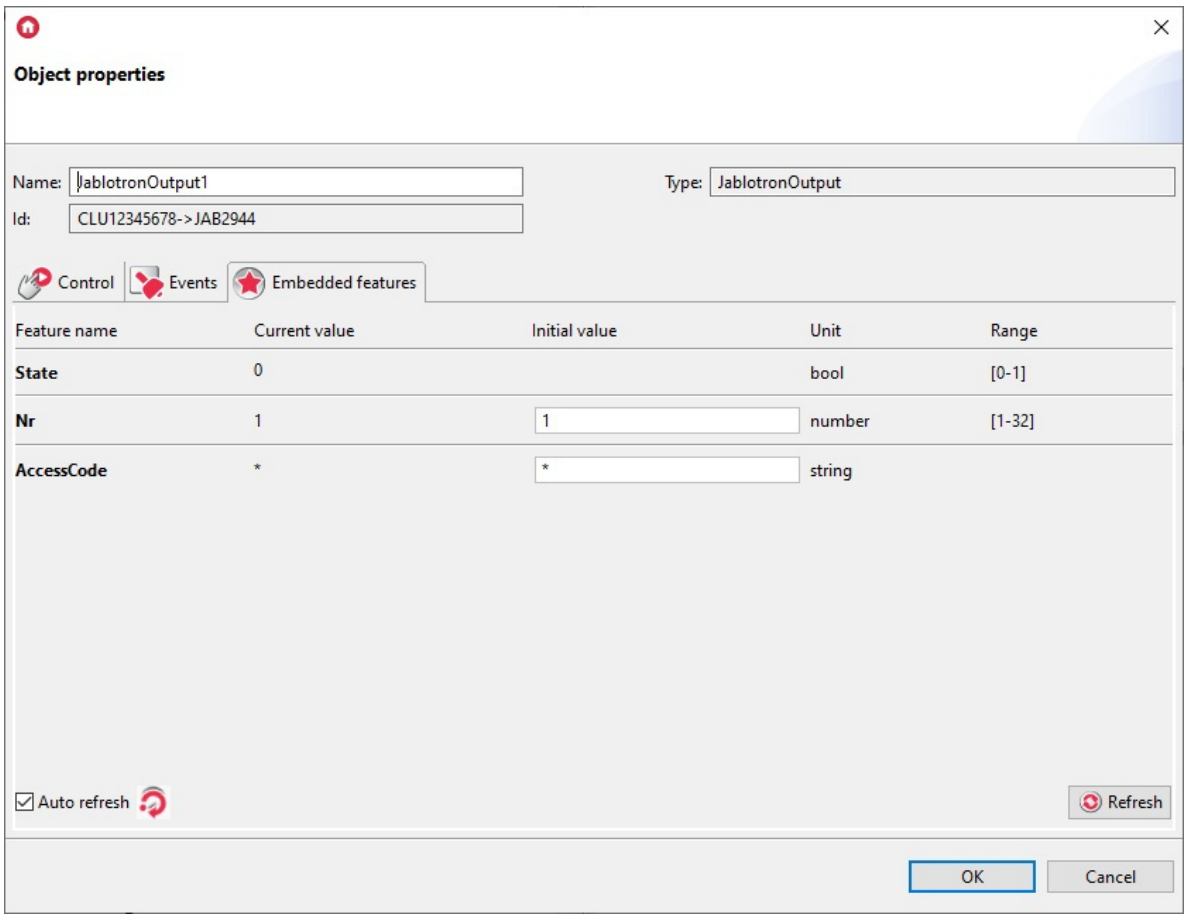
GATE Alarm allows you to add a virtual object **JablotronOutput**

Creating the *JablotronOutput* object:



Then go to configuration - Built-in Features tab and enter:

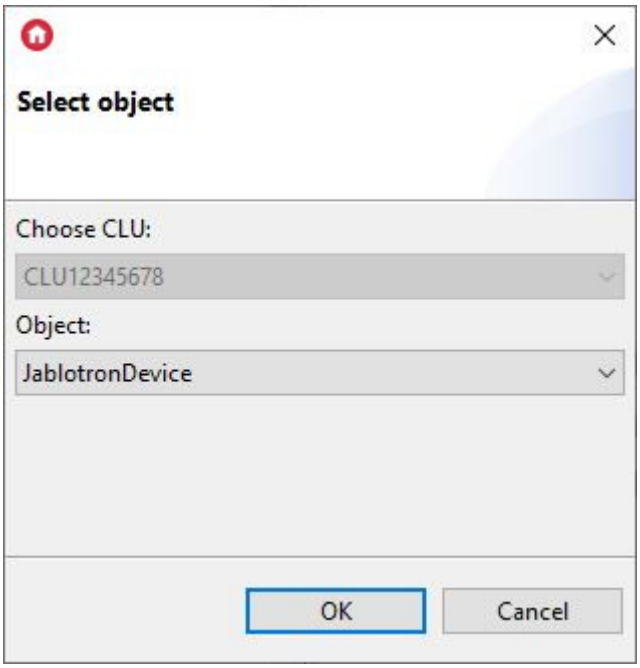
- **Nr** - parameter defining to which output the object relates;
- **AccessCode** - The access code will use the admin code for '*';



D. Input / Device

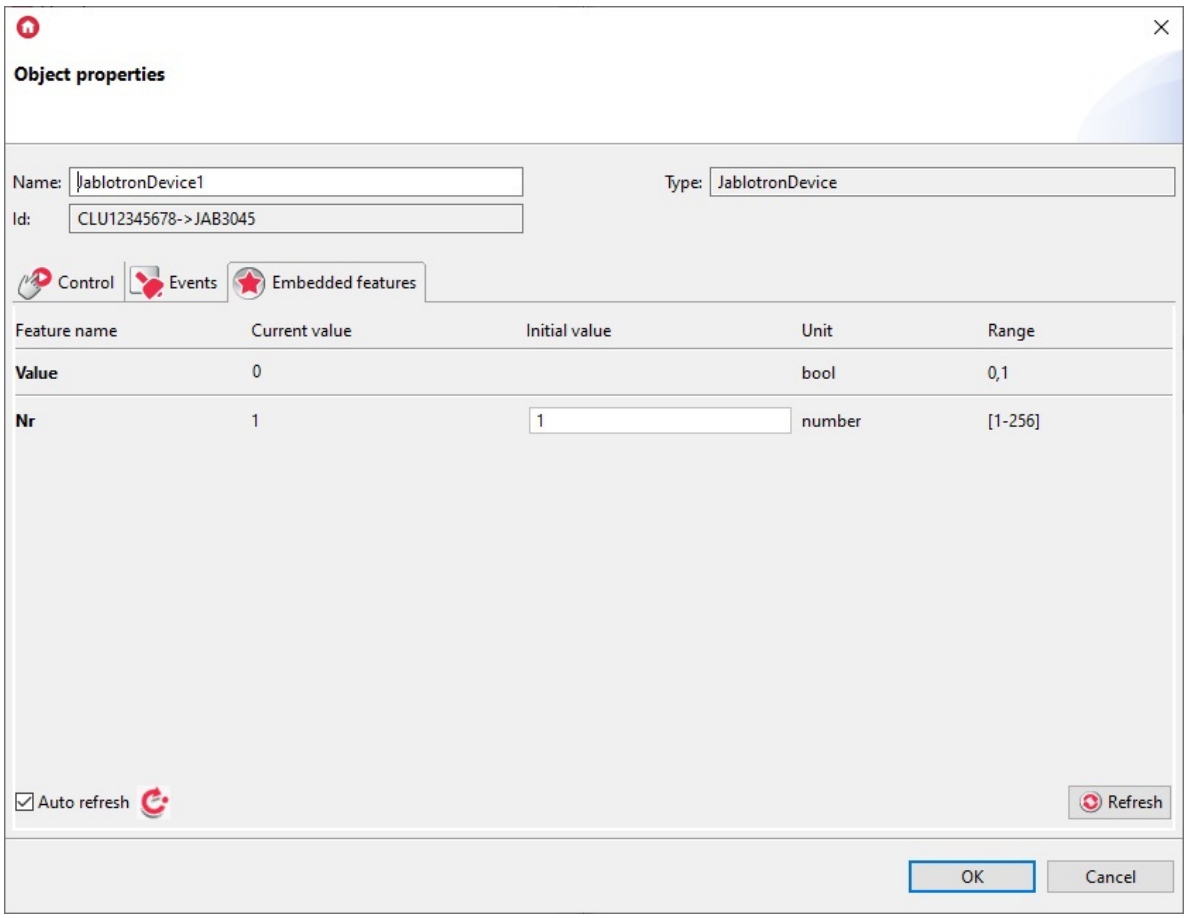
GATE Alarm allows you to add a virtual object **JablotronDevice**

Creating the *JablotronDevice* object:



Then go to configuration - Built-in Features tab and enter:

- **Nr** - parameter defining to which input the object refers;



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 1.4.2-2346 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
`TelnetLogLevel`	Specifies the logging level
`OverloadDetection`	Specifies whether Gate should report CPU overload using the red LED
`ResetReason`	Specifies the device reset reason: 0 - power-on; 2 - configuration reload; 3 - system exception

METHODS

Name	Description
`SetDateTime`	Sets the date and time
`SetClientReportInterval`	Sets the reporting period for feature changes
`SetPrimaryDNS`	Sets the PrimaryDNS feature
`SetSecondaryDNS`	Sets the SecondaryDNS feature
`SetTelnetLogLevel`	Specifies the logging level

EVENTS

Name	Description
`OnInit`	An event dispatched when the device initializes

B. Satel

FEATURES

Name	Description
`State`	Central status: `0` - no connection to the control panel, `1` - connected to the control panel
`LastError`	Last ETHM module error code: `0` - ok, `1` - incorrect password
`IP`	ETHM module IP address (Satel)
`Port`	ETHM module port (Satel)
`AdminPassword`	Satel administrator password
`UpdateTime`	Control panel status update period
`EncryptionEnabled`	Encryption status (*true* - enabled, *false* - disabled)
`EncryptionKey`	Satel encryption key
`Value`	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
`Nr`	Parameter defining the zone, input or output to which the object refers
`UserPassword`	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 (`LastError`)
`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`	*JablotronSection:* `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 *JablotronOutput*: `0` - Output switched on `1` - Output switched off
`Nr`	Parameter defining which input / output / section the object refers to
`AccessCode`	The access code will use the admin code for `*`
`Value`	Returns the current status: `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

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 1.4.1 - 2334 or higher!**

1. General information

The GATE MODBUS module enables the integration of the Grenton system with all devices supporting MODBUS RTU/TCP standard.

> **Note!**
> There is no limit to the number of objects for the created virtual objects - the limitation is the device memory, which is influenced by e.g. level of logic expansion on the module.

2. Module configuration

> **Note!**
> Before starting any work with the GATE Modbus module, it is necessary to update the interface database!

2.1. 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.

> **Note!**
> When the `UseCloud` feature is set to `true`, the `UseNTP` feature is automatically set to `true`.

3. Virtual objects

3.1. Modbus RTU protocol

> **Note!**

GATE MODBUS can only work as Master **or** Slave at the same time. After adding the `ModbusSlaveConfigRTU` object and sending configuration GATE MODBUS works only as a Slave, supporting `ModbusSlaveRTU` objects. Created `ModbusRTU` virtual objects are ignored.

Note!

Virtual objects of the GATE MODBUS module can operate with each other at the same time in the following configurations:

- `ModbusRTU`, `ModbusClient`, `ModbusServer`
- `ModbusSlave`, `ModbusClient`, `ModbusServer`

A. ModbusRTU

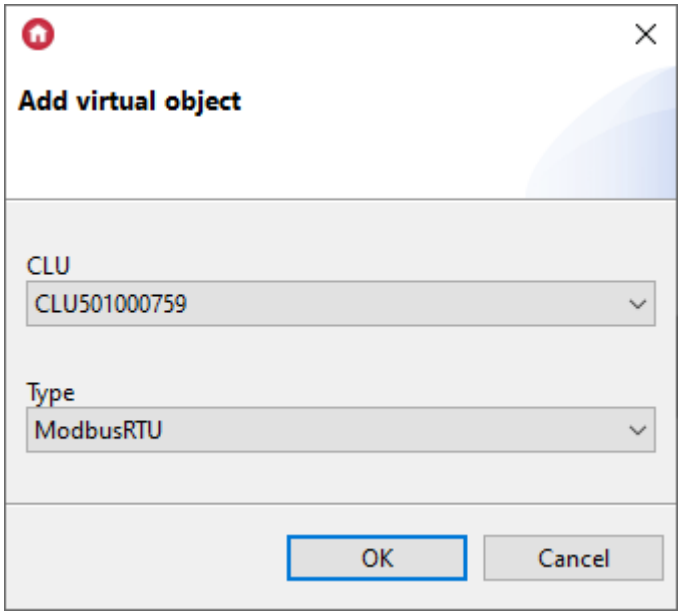
The `ModbusRTU` virtual object is used to read values from Slave devices using the RTU protocol.

Note!

The `ModbusRTU` virtual object replaces the deprecated `Modbus` and `ModbusValue` objects.

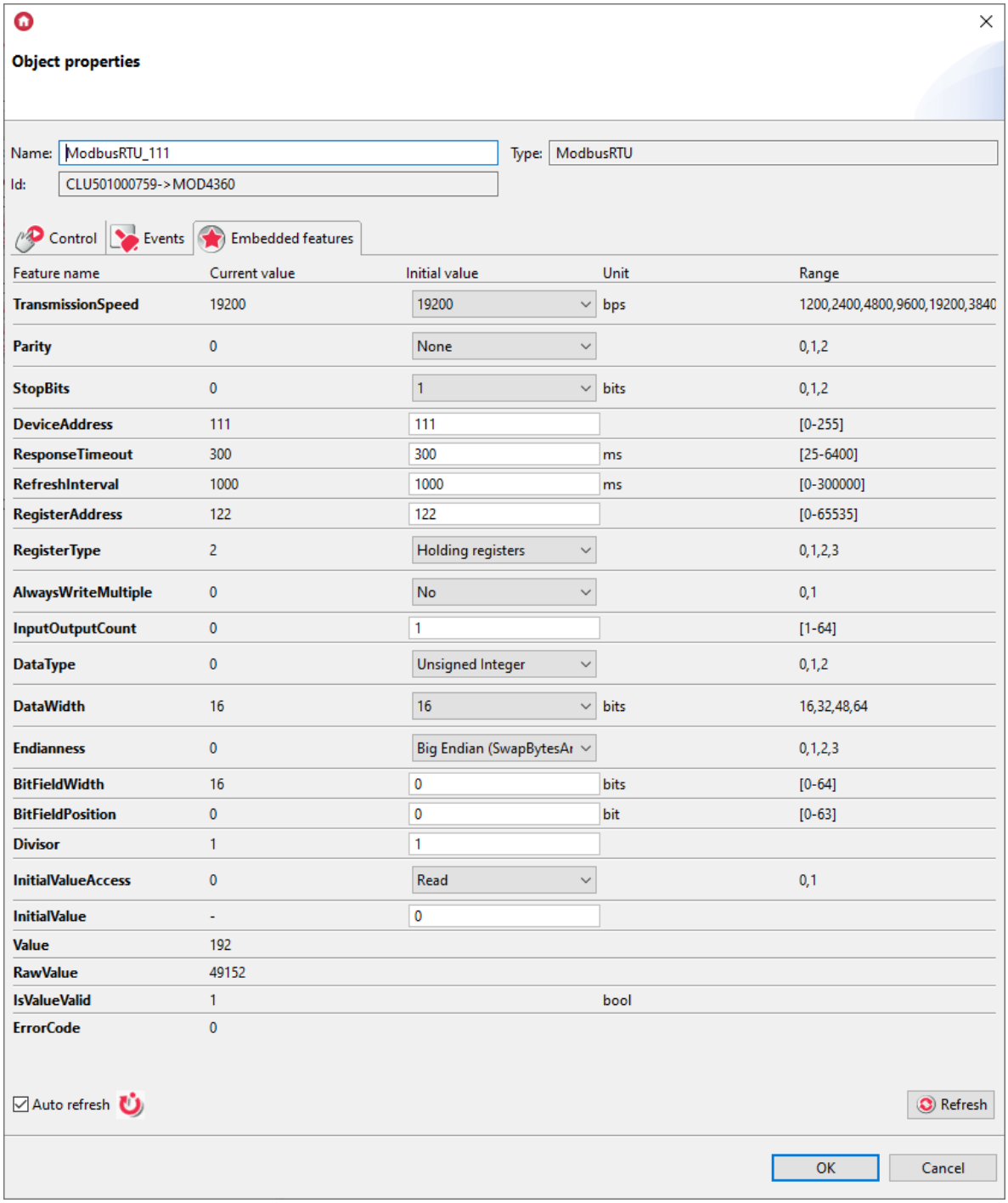
To read the value from an available Slave device, proceed as follows:

- Create a virtual object `ModbusRTU` and name it:



- Go to *Embedded features* tab and enter:
  - **TransmissionSpeed** - transmission speed;
  - **Parity** - sets the parity check:
    0 - None
    1 - Odd
    2 - Even
  - **StopBits** - sets the number of stop bits:
    0 - 1 stop bit
    1 - 1.5 stop bits
    2 - 2 stop bits
  - **DeviceAddress** - address of the slave device;
  - **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);
- **RegisterAddress** - address of the supported registry;
- parameters appropriate for the selected slave device type



Note!
In the Gate Modbus object from version firmware 1.4.1 - 2334 introduced `ModbusMasterFrameSpace` feature used to determine the gap between sent Modbus frames expressed in characters. In the case of projects with a large number of `ModbusRTU` objects and problems with polling fluency for a short `RefreshInterval` time, you can enter from 1 to 50 blank characters between sent frames.

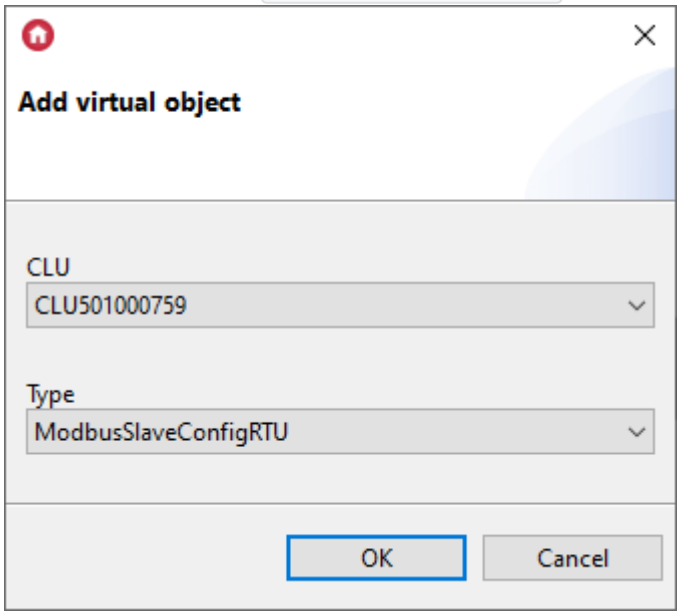
Note!

> Using `ModbusRTU` objects with different set communication parameters (`TransmissionSpeed`, `Parity`, `StopBits`) may cause incorrect operation of GATE MODBUS. It is recommended to set the same communication parameters for all created `ModbusRTU` objects.

B. ModbusSlaveConfigRTU

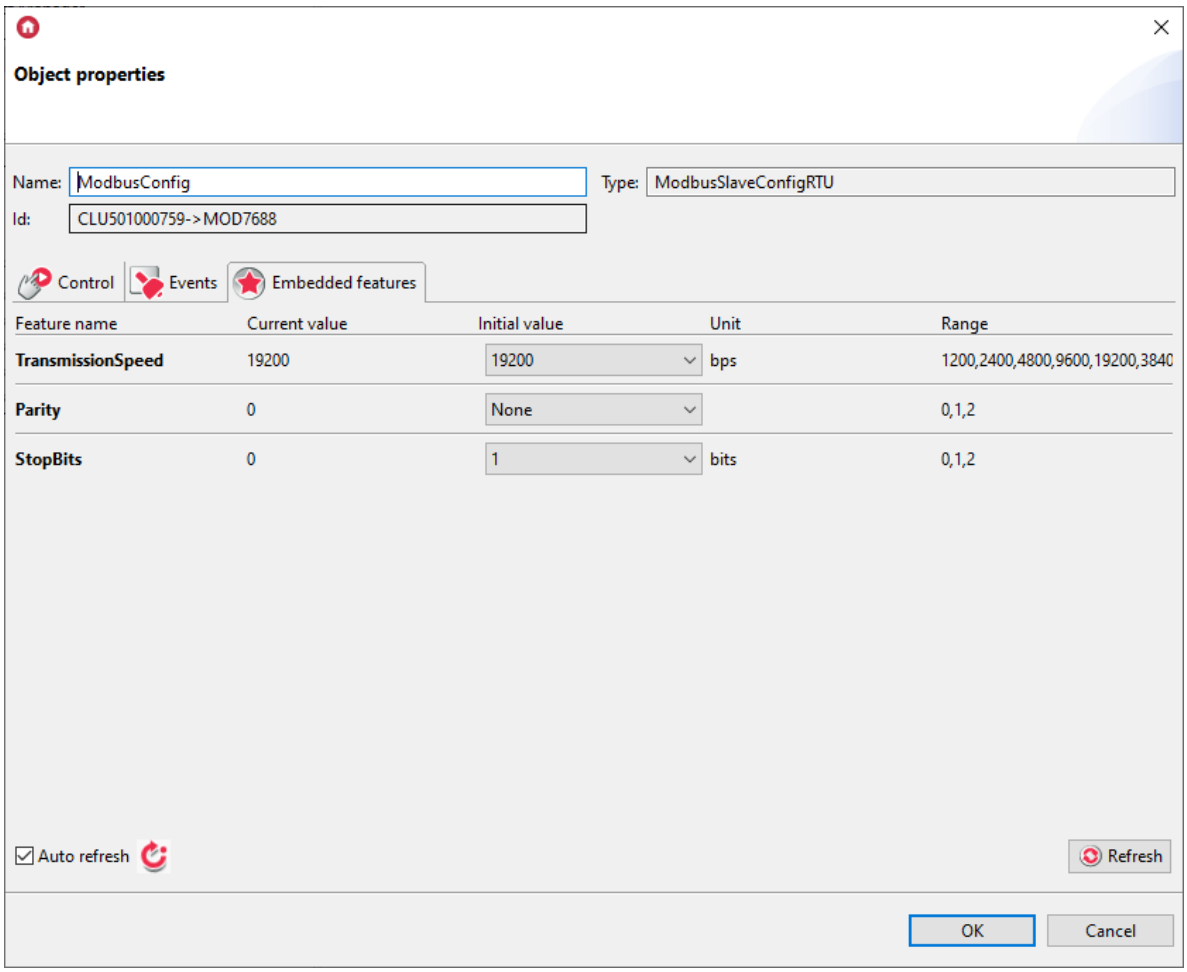
The `ModbusSlaveConfigRTU` virtual object is used to configure the Modbus Gate module working as a Slave device. In order to do this, need to:

- Create a virtual object `ModbusSlaveConfigRTU`:



- Go to Embedded Features tab and enter communication data with Master RTU device:
  - **TransmissionSpeed** - transmission speed;
  - **Parity** - sets the parity check:
    0 - None
    1 - Odd
    2 - Even
  - **StopBits** - sets the number of stop bits:
    0 - 1 stop bit
    1 - 1.5 stop bits
    2 - 2 stop bits

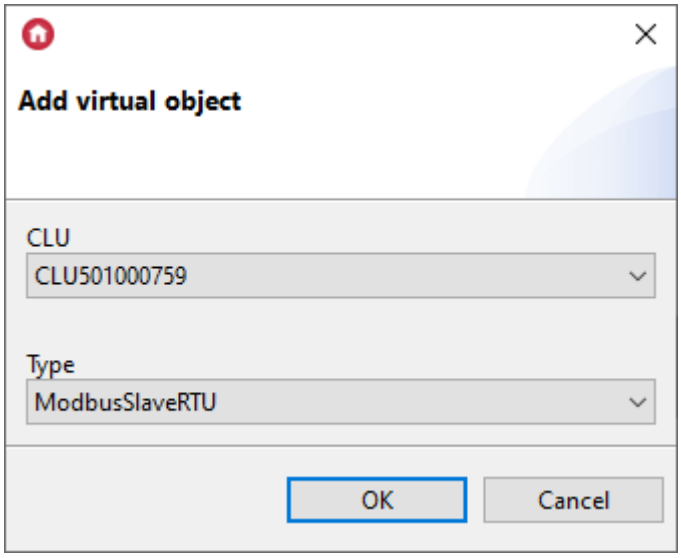
After sending the configuration, Gate Modbus works as a slave device.



C. ModbusSlaveRTU

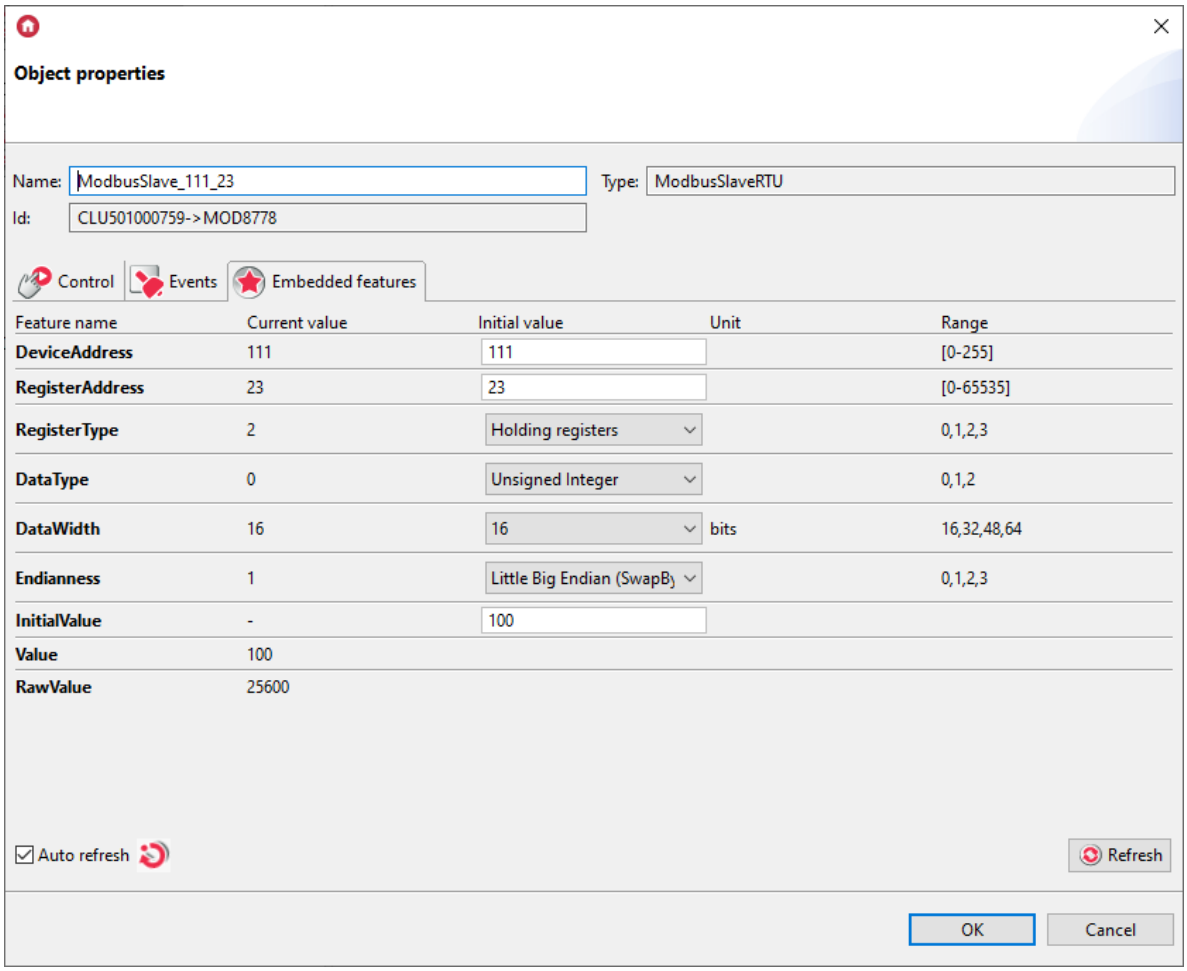
The `ModbusSlaveRTU` virtual object is used to define values for individual registers of the slave device. In order for the object to work properly first create the `ModbusSlaveConfigRTU` object (see B section). In order to use the functionality, need to:

- Create a virtual object `ModbusSlaveRTU`:



- Go to Embedded Features tab and enter communication data with Master RTU device:
  - **DeviceAddress** - address of the slave device;
  - **RegisterAddress** - address of the supported registry;
  - **RegisterType** - register type;
  - **DataType** - value type;
  - **DataWidth** - data width;

- **Endianness** - byte order;
- **InitialValue** - initial register value.

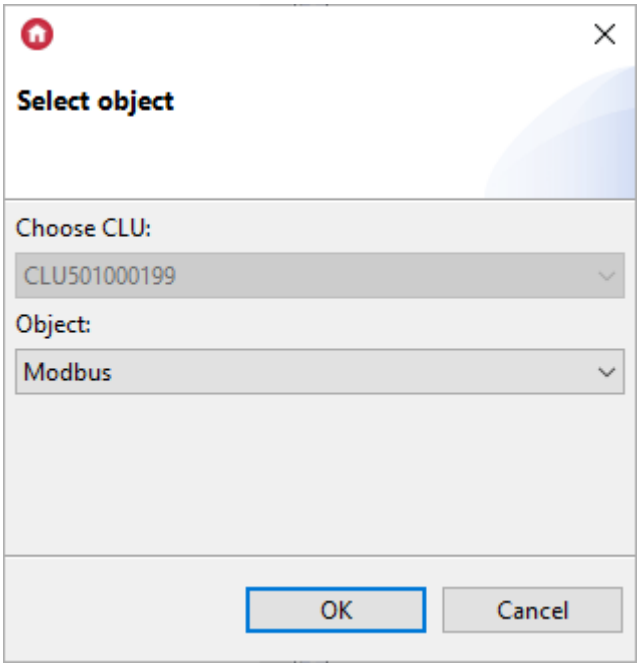


D. Modbus

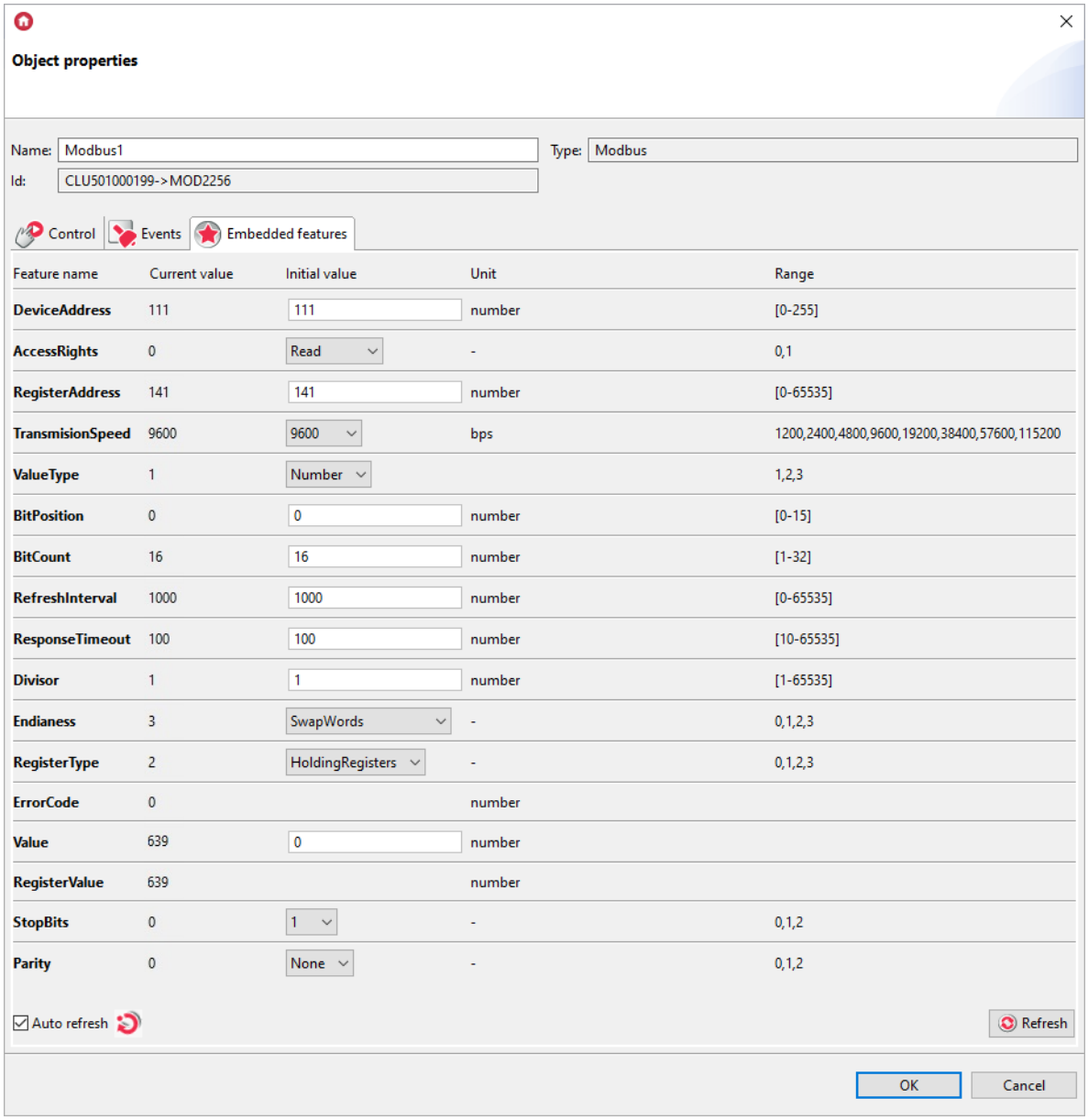
> **Note!**
> The `ModbusValue` virtual object was phased out in Gate Modbus version 1.4.1 - 2334 with the introduction of the new virtual object `ModbusRTU`, which is its equivalent. In order to ensure compatibility with existing projects, the object for use. It is not possible to create new objects of this type.

To perform the correct configuration of the GATE Modbus module it is necessary to:

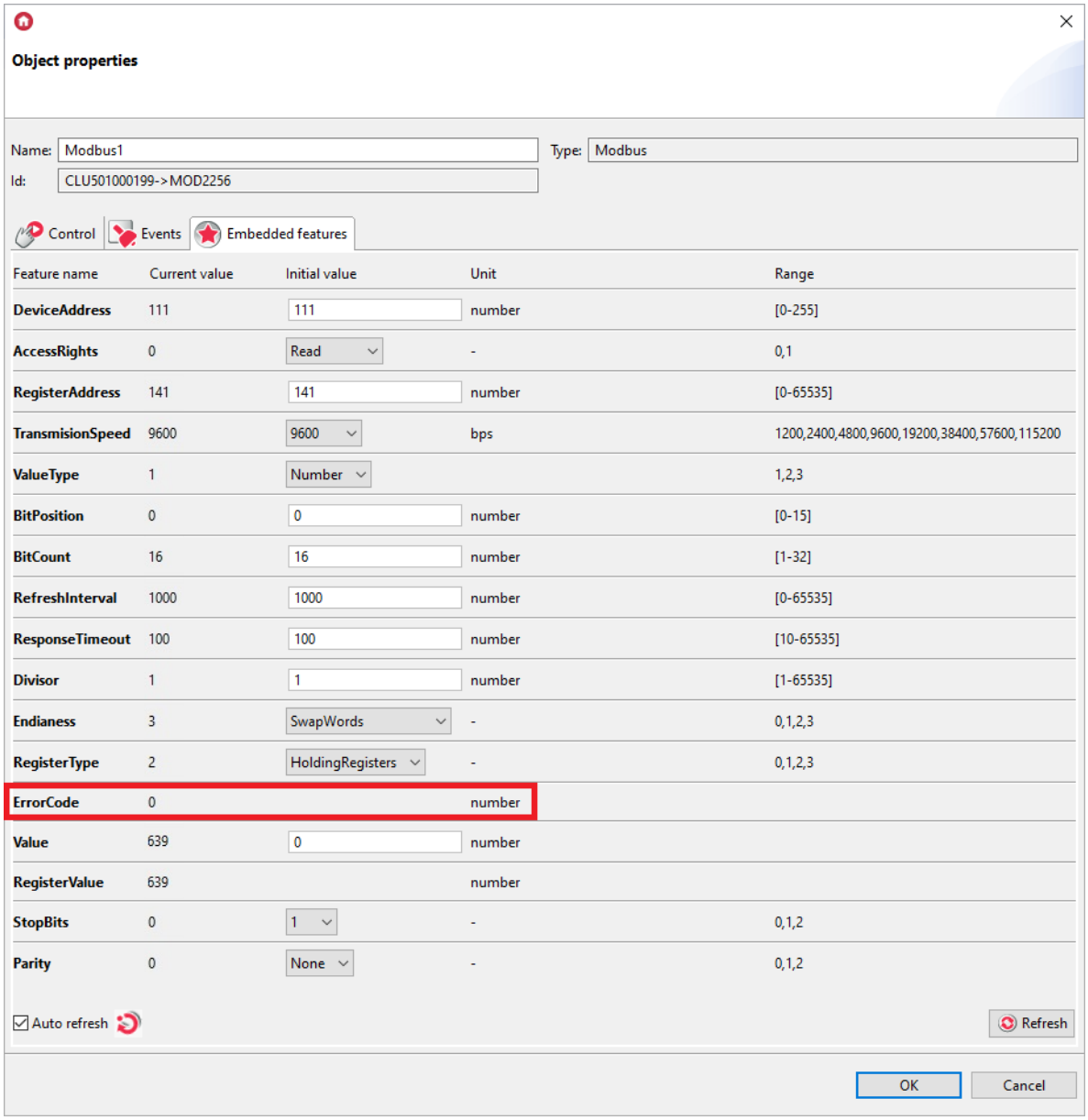
- Create a virtual object *Modbus* and name it:



- 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;
  - **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



- Send configuration and verify connection - tab *Embedded features*, feature `ErrorCode` = 0 (correct read / write):



Note!

In the Gate Modbus object from version firmware 1.4.1 - 2334 introduced `ModbusMasterFrameSpace` feature used to determine the gap between sent Modbus frames expressed in characters. In the case of projects with a large number of `ModbusRTU` objects and problems with polling fluency for a short `RefreshInterval` time, you can enter from 1 to 50 blank characters between sent frames.

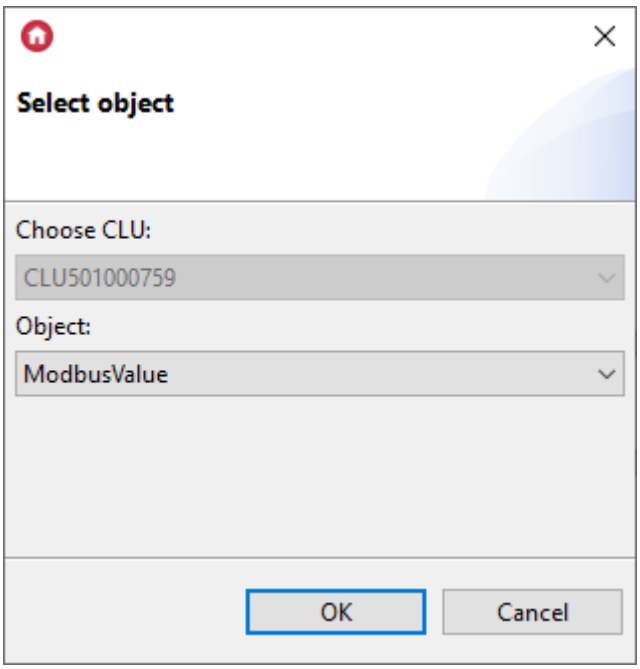
E. ModbusValue

Note!

The `ModbusValue` virtual object was phased out in Gate Modbus version 1.4.1 - 2334 with the introduction of the new virtual object `ModbusRTU`, which is its equivalent. In order to ensure compatibility with existing projects, the object for use. It is not possible to create new objects of this type.

To use the `ModbusValue` virtual object:

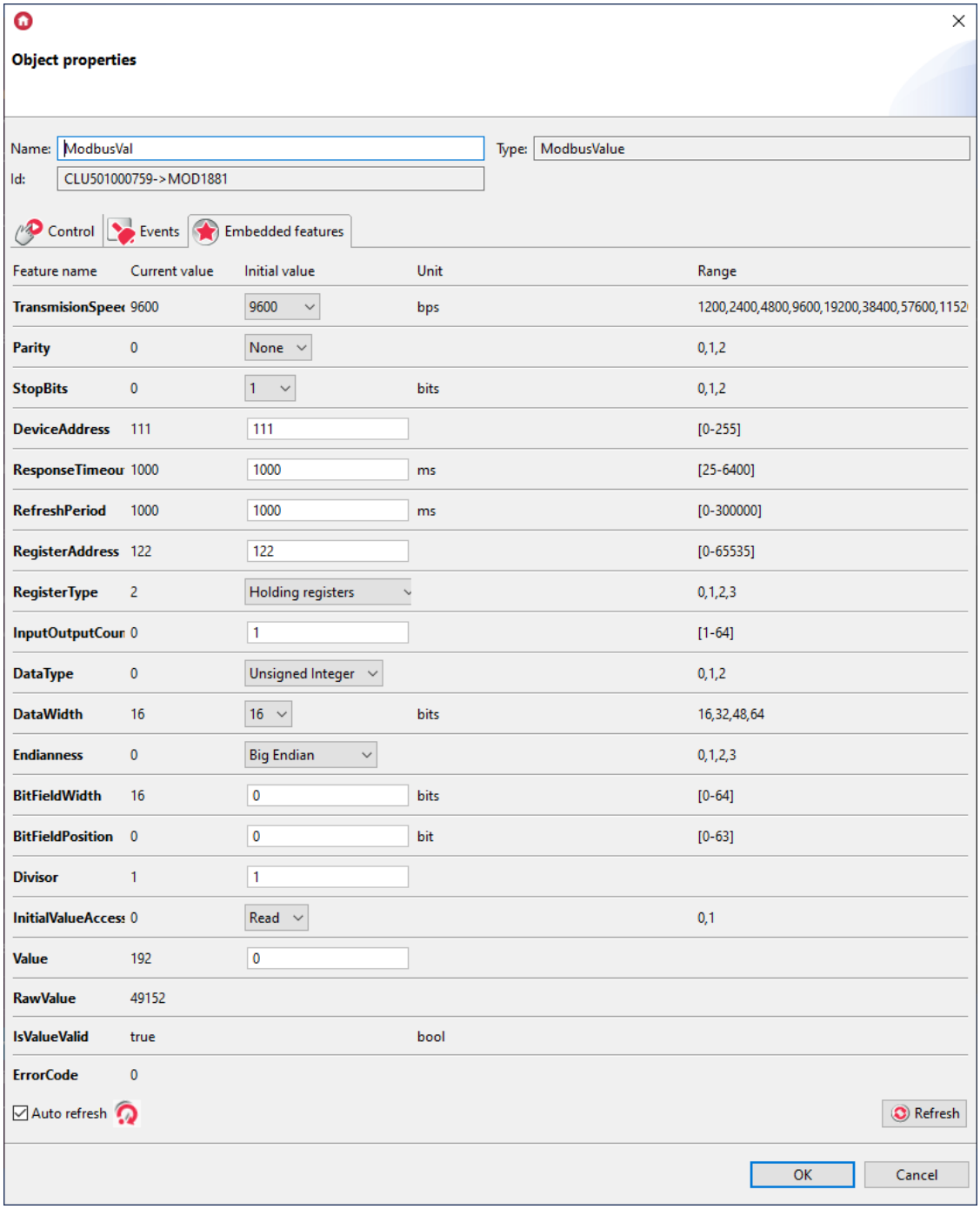
- Create a virtual object `ModbusValue` and name it:



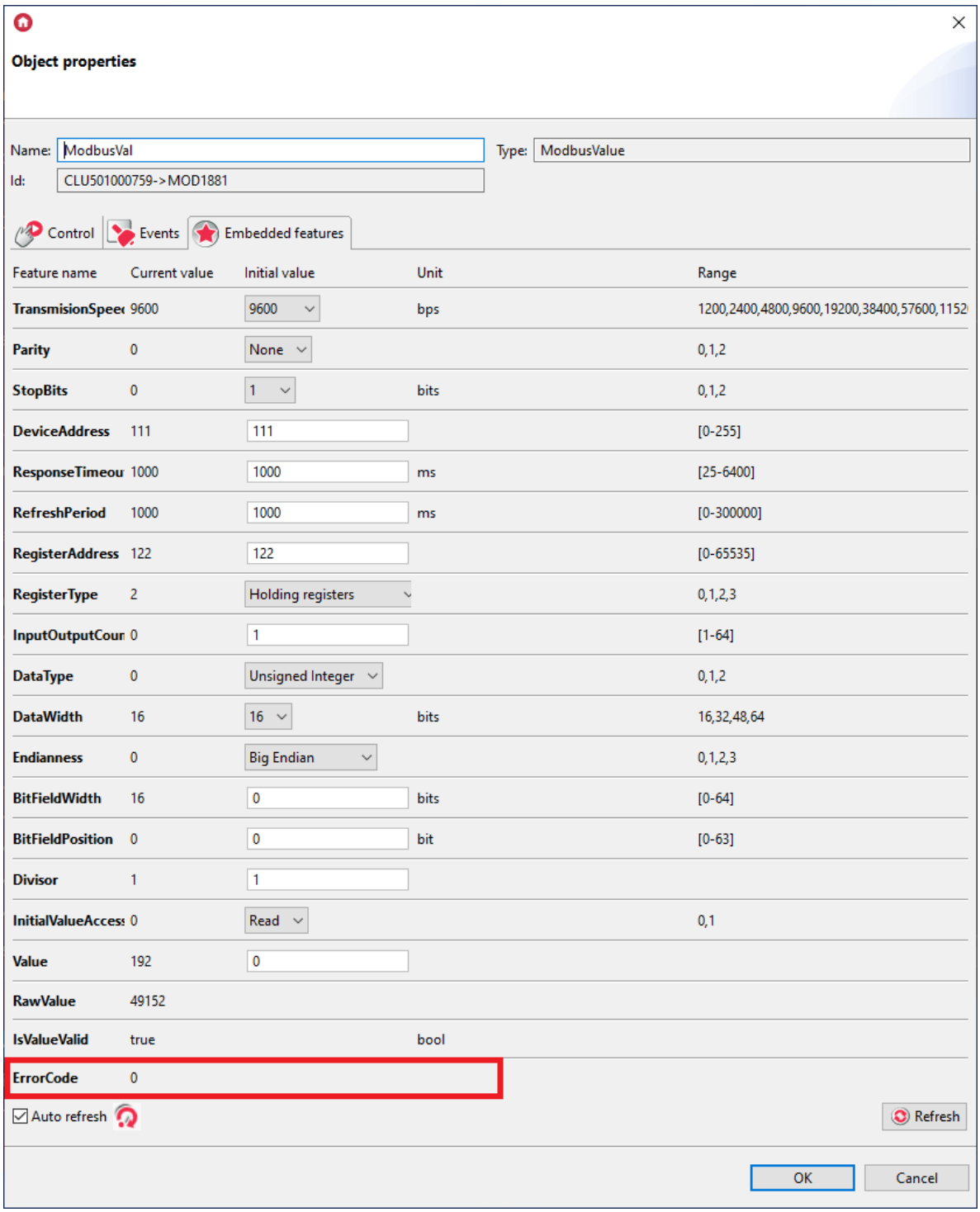
- Go to *Embedded features* tab and enter:
  - **TransmissionSpeed** - transmission speed;
  - **Parity** - sets the parity check:
    0 - None
    1 - Odd
    2 - Even
  - **StopBits** - sets the number of stop bits:
    0 - 1 stop bit
    1 - 1.5 stop bits
    2 - 2 stop bits
  - **DeviceAddress** - address of the slave device;
  - **ResponseTimeout** - response timeout in 25ms steps;
  - **RefreshPeriod** - minimum refresh period in 5ms steps - 0 means automatic refresh is disabled;
  - **RegisterAddress** - address of the supported registry;
  - **RegisterType** - type of the register set:
    0 - Discrete outputs / coils
    1 - Discrete inputs
    2 - Holding registers
    3 - Input registers
  - **Divisor** - value divisor (scale);
  - **InitialValueAccess** - initial Value access method:
    0 - The initial Value and SetValue parameter are sent to the device;
    1 - The initial Value and SetValue parameter are ignored the method of first accessing to the value;

    > **Note!**
    >
    > Entering a value in `Value` field when setting the `InitialValueAccess` = 1 feature causes saving the value (sending the appropriate frame) after sending the configuration to the CLU. If the value of the `InitialValueAccess` = 0 feature, the entered value in `Value` field is ignored.

- parameters appropriate for the selected slave device type;



- Send configuration and verify connection - tab *Embedded features*, feature `ErrorCode` = 0 (correct read / write):



Note!

In the Gate Modbus object from version firmware 1.4.1 - 2334 introduced `ModbusMasterFrameSpace` feature used to determine the gap between sent Modbus frames expressed in characters. In the case of projects with a large number of `ModbusRTU` objects and problems with polling fluency for a short `RefreshInterval` time, you can enter from 1 to 50 blank characters between sent frames.

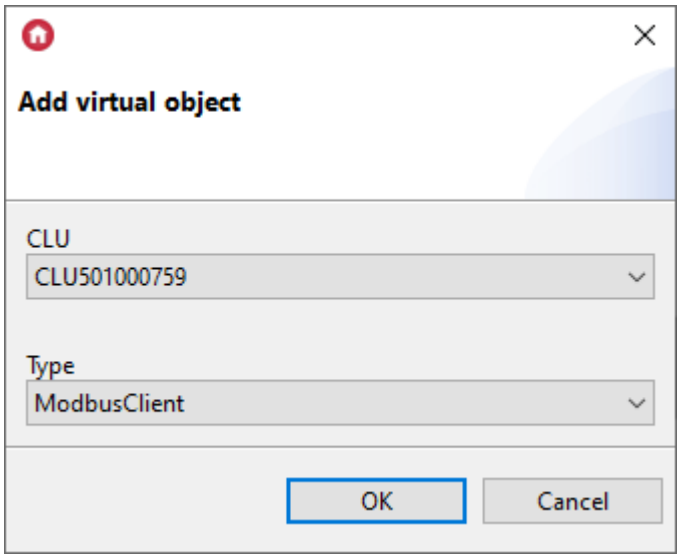
3.2. Modbus TCP protocol

A. ModbusClient

The `ModbusClient` virtual object is used to communicate with Server devices via LAN.

To use the `ModbusClient` virtual object:

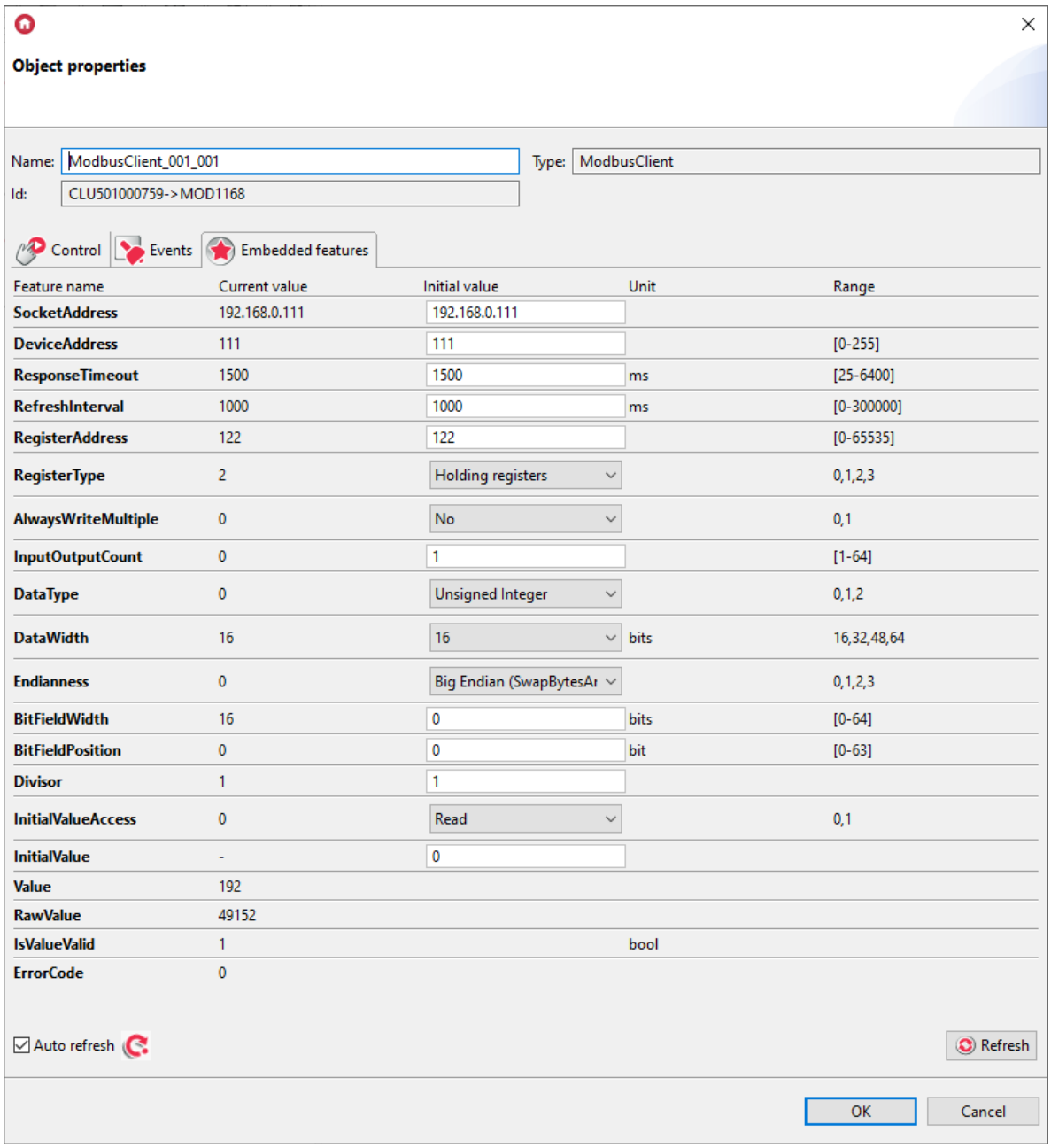
- Create a `ModbusClient` virtual object and give it a name:



- Go to *Embedded features* tab and enter:
  - **SocketAddress** - device IP address;
  - **DeviceAddress** - address of the Modbus device;
  - **ResponseTimeout** - response timeout in 25ms steps;
  - **RefreshInterval** - minimum refresh period in 5ms steps - 0 means automatic refresh is disabled;
  - **RegisterAddress** - address of the supported registry;
  - **RegisterType** - type of the register set:
    0 - Discrete outputs / coils
    1 - Discrete inputs
    2 - Holding registers
    3 - Input registers
  - **Divisor** - value divisor (scale);
  - **InitialValueAccess** - initial Value access method:
    0 - The InitialValue is read from the device
    1 - The initial Value is written to the device
  - parameters appropriate for the selected registry type of the Server device;

> **Note!**
>
> Entering the selected value in the `InitialValue` field with the `InitialValueAccess` = 1 feature saves the value (sending the appropriate frame) after sending the configuration to the CLU. If the value of the `InitialValueAccess` feature = 0, the value entered in the `InitialValue` field is ignored.

- Send the configuration and verify the connection - ErrorCode =0 (correct reading/writing);



Note!

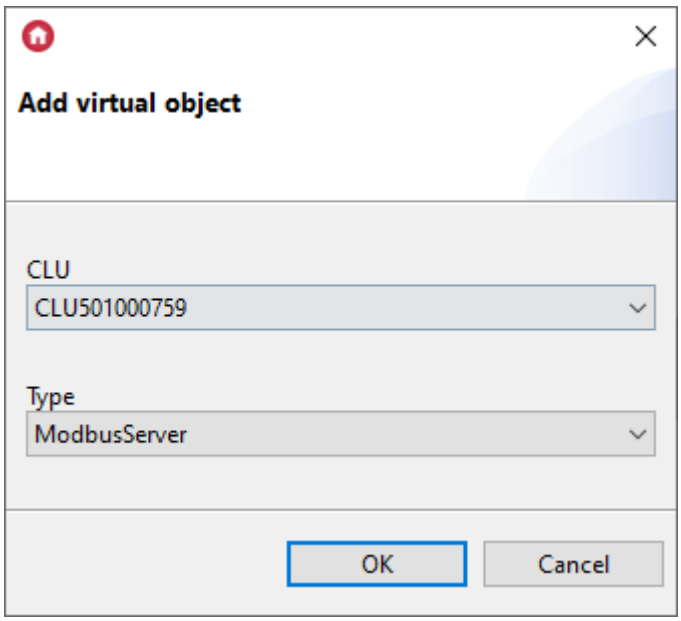
The default operating port of the Server device is 502. It is possible to work on a different, defined port. In this case, after entering the IP address of the device, add the target communication port - e.g: `192.168.0.103:8888` .

B. ModbusServer

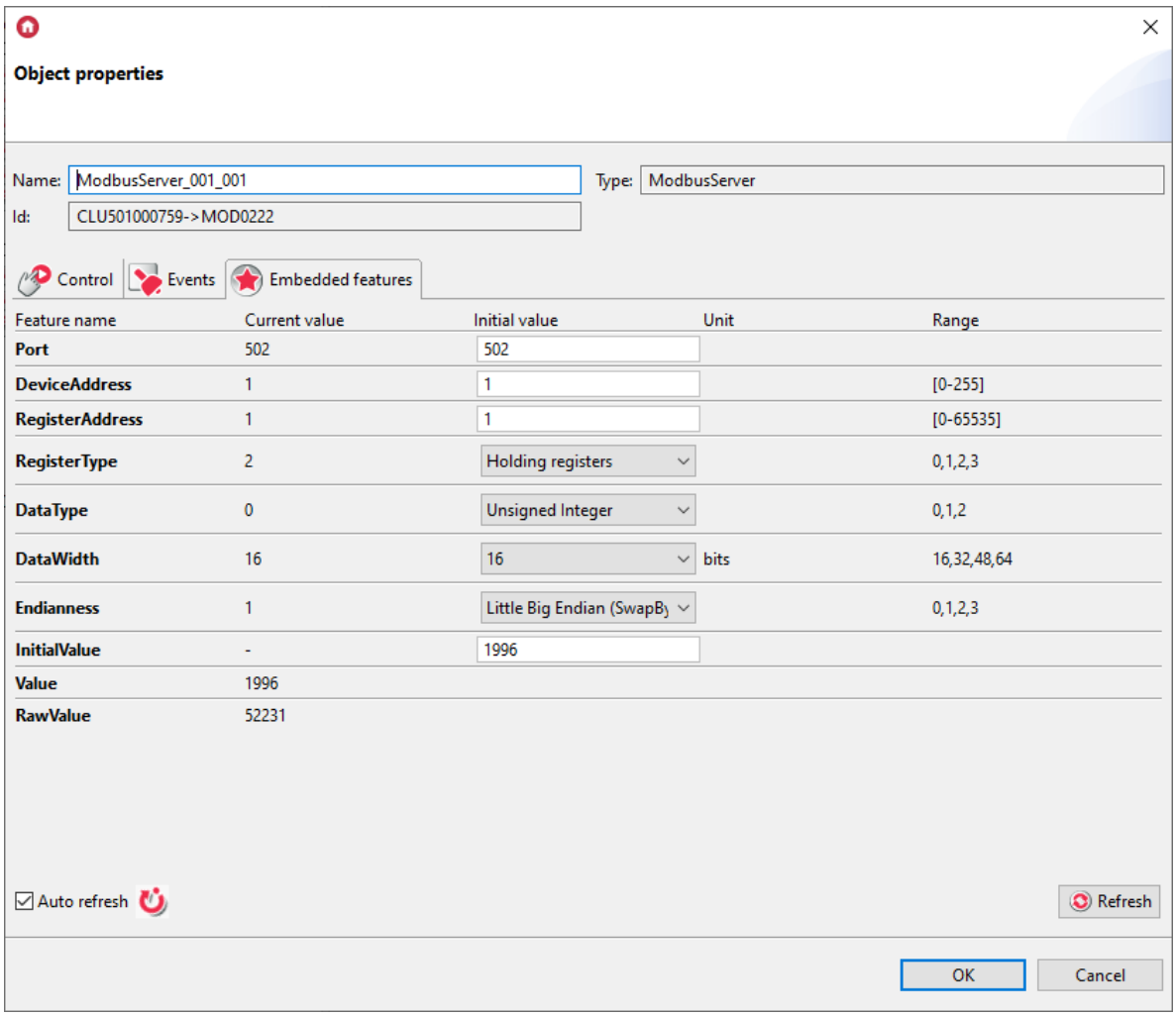
The `ModbusServer` virtual object is used to communicate with Client devices via LAN.

To use the `ModbusServer` virtual object:

- Create a `ModbusServer` virtual object and give it a name:



- Go to *Embedded features* tab and enter:
  - **Port** - server listening port;
  - **DeviceAddress** - address of the Modbus device;
  - **RegisterAddress** - address of the supported registry;
  - **RegisterType** - type of the register set:
    0 - Discrete outputs / coils;
    1 - Discrete inputs;
    2 - Holding registers;
    3 - Input registers;
  - **DataType** - value type;
  - **DataWidth** - data width (1 to 4 16 bit registers);
  - **Endianness** - byte order;
  - **InitialValue** - initial register value;
  - parameters appropriate for the selected registry type of the Client device;
- Send the configuration and verify the connection using the Client device



4. Register parameters

> **Note!**
> The described functionality and integration is available for **GRENTON GATE MODBUS MASTER, DIN, Eth (INT-201 - E-01)** with **firmware 1.4.1 - 2334 or higher!**

4.1. ModbusRTU and ModbusClient object

The register parameters for `ModbusRTU` and `ModbusClient` objects do not differ in the configuration area - the only difference is in the method of communication with the target device. Below are the ways to read values from the registers of Slave/Server devices.

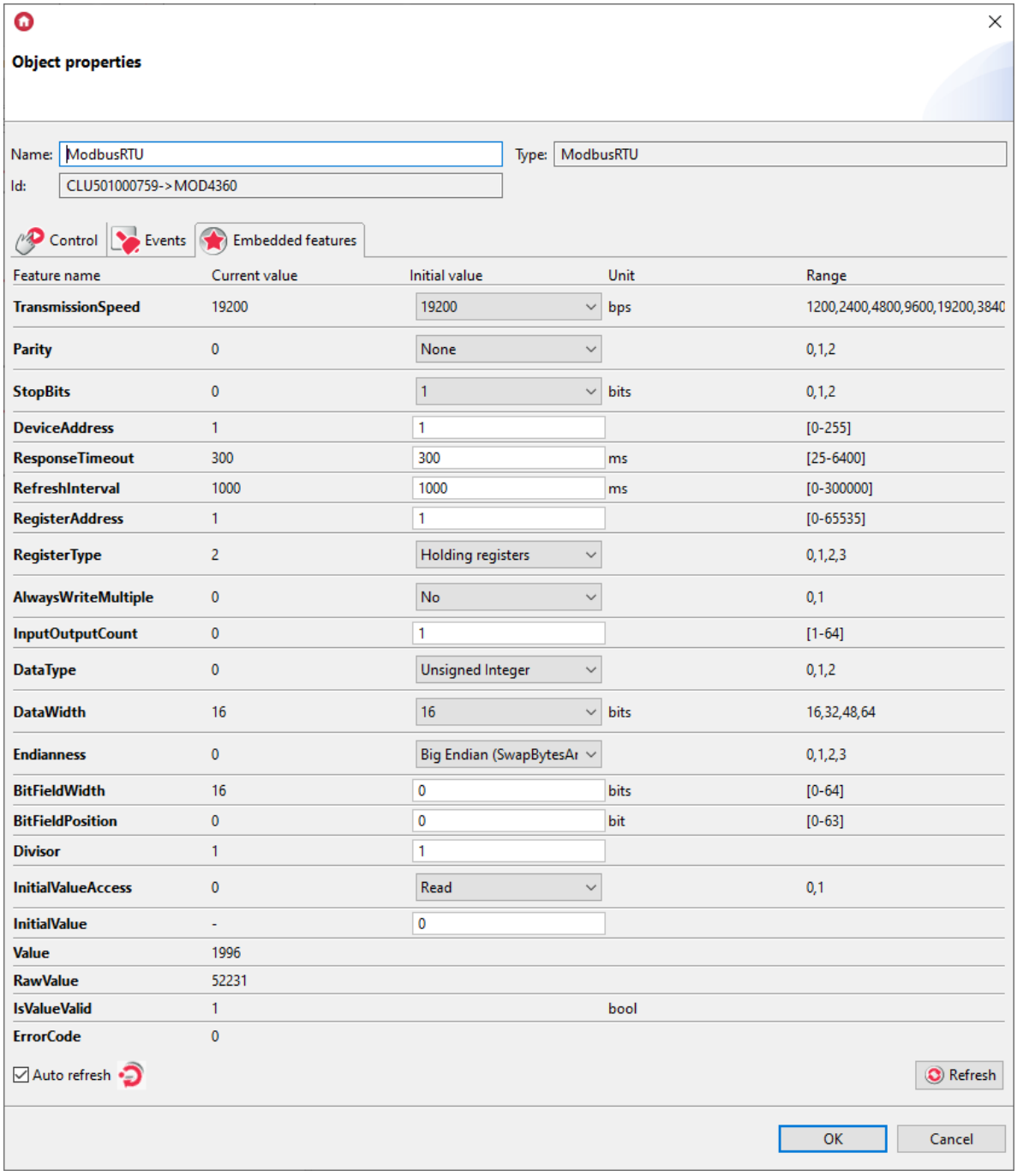
> **Note!**
>
> For the use of synchronous read and write operations, i.e., performing the `WriteValue` and reading values or error codes directly within one script, keep the `OnValueChange` and `OnValueRead` events empty. Also, it is recommended, in such a case, to set `RefreshInterval` = 0.

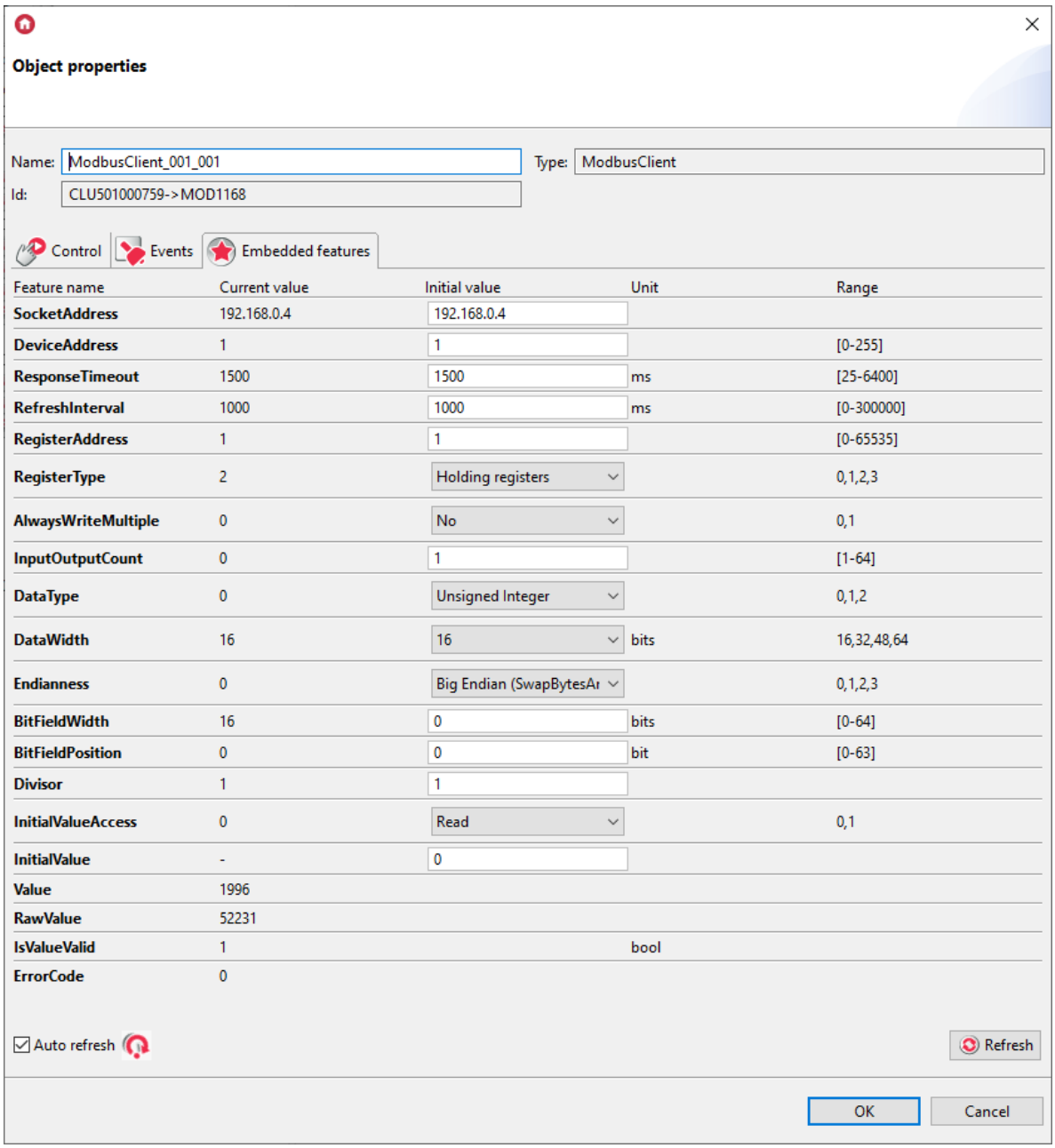
A. 16-bit registers

Reading 16-bit holding registers (`Read Holding Registers`, *FunctionCode = 03*):

- `RegisterType`: Holding Register;
- `DataType`: Unsigned Integer;

- `DataWidth` :16;
- `Endianness` : default value;
- `BitFieldWidth` : default value;
- `BitFieldPosition` : default value.





Reading 16-bit input registers (`Read Input Registers`, *FunctionCode = 04*):

- `RegisterType`: Input Register;
- `DataType`: Unsigned Integer;
- `DataWidth` :16;
- `Endianness`: default value;
- `BitFieldWidth`: default value;
- `BitFieldPosition`: default value;

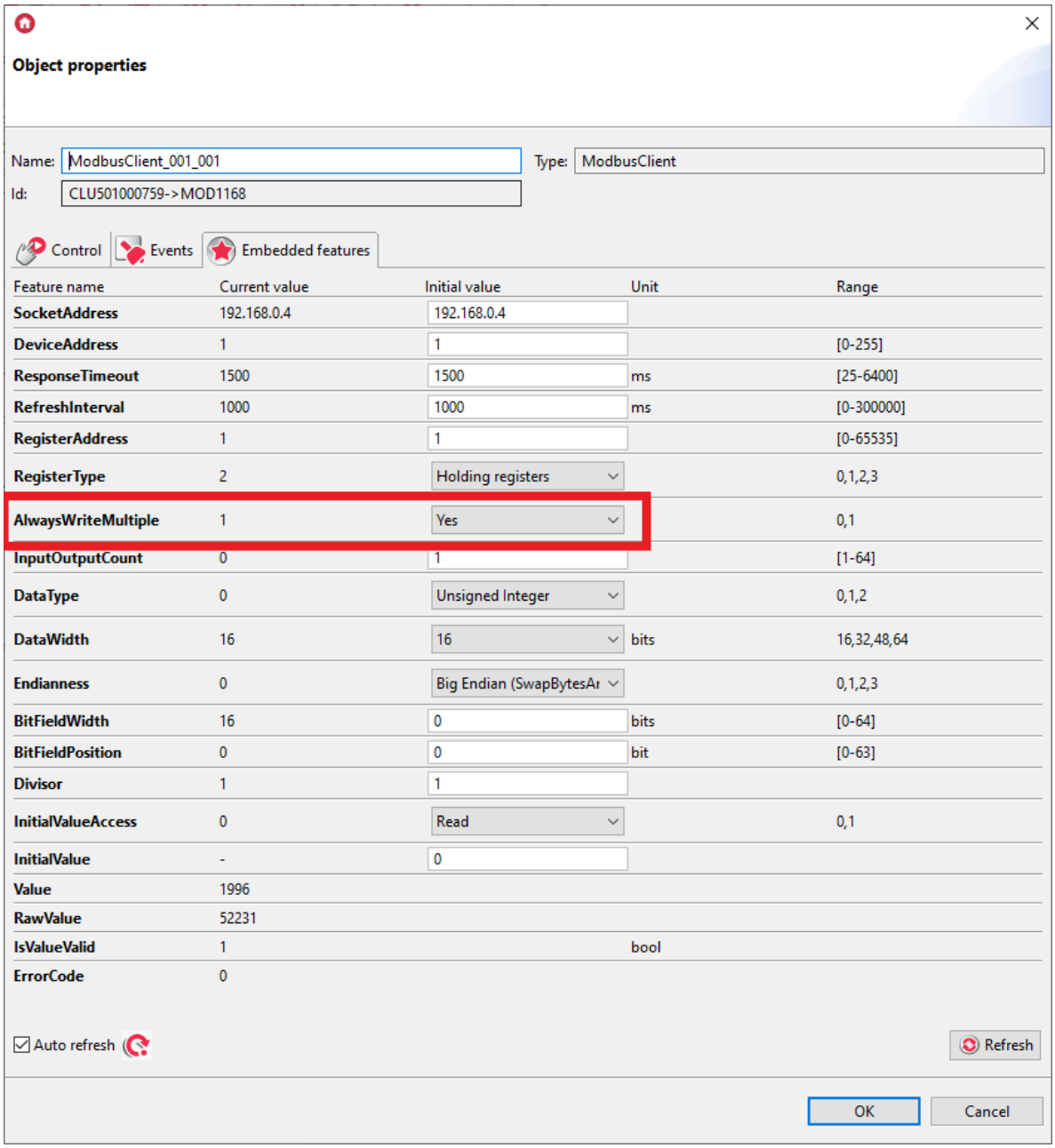
Records of 16-bit holding registers (`Preset / Write Single Holding Register`, *FunctionCode = 06*):

- `RegisterType`: Holding Register;
- `AlwaysWriteMultiple`: No;
- `DataType`: Unsigned Integer;
- `DataWidth` :16;
- `Endianness`: default value;
- `BitFieldWidth`: default value;
- `BitFieldPosition`: default value.

Note!
When writing to 16-bit registers, it is possible to use function 16 (`Write Multiple Holding Register`) instead of function 6 (`Write Single Holding Register`) when the Slave device only accepts writing of this type. To do this, use the `AlwaysWriteMultiple` feature and set it to 1.

Records of 16-bit holding registers (`Preset / Write Single Holding Register`, *FunctionCode = 06*):

- `RegisterType`: Holding Register;
- `AlwaysWriteMultiple`: Yes;
- `DataType`: Unsigned Integer;
- `DataWidth` :16;
- `Endianness`: default value;
- `BitFieldWidth`: default value;
- `BitFieldPosition`: default value;



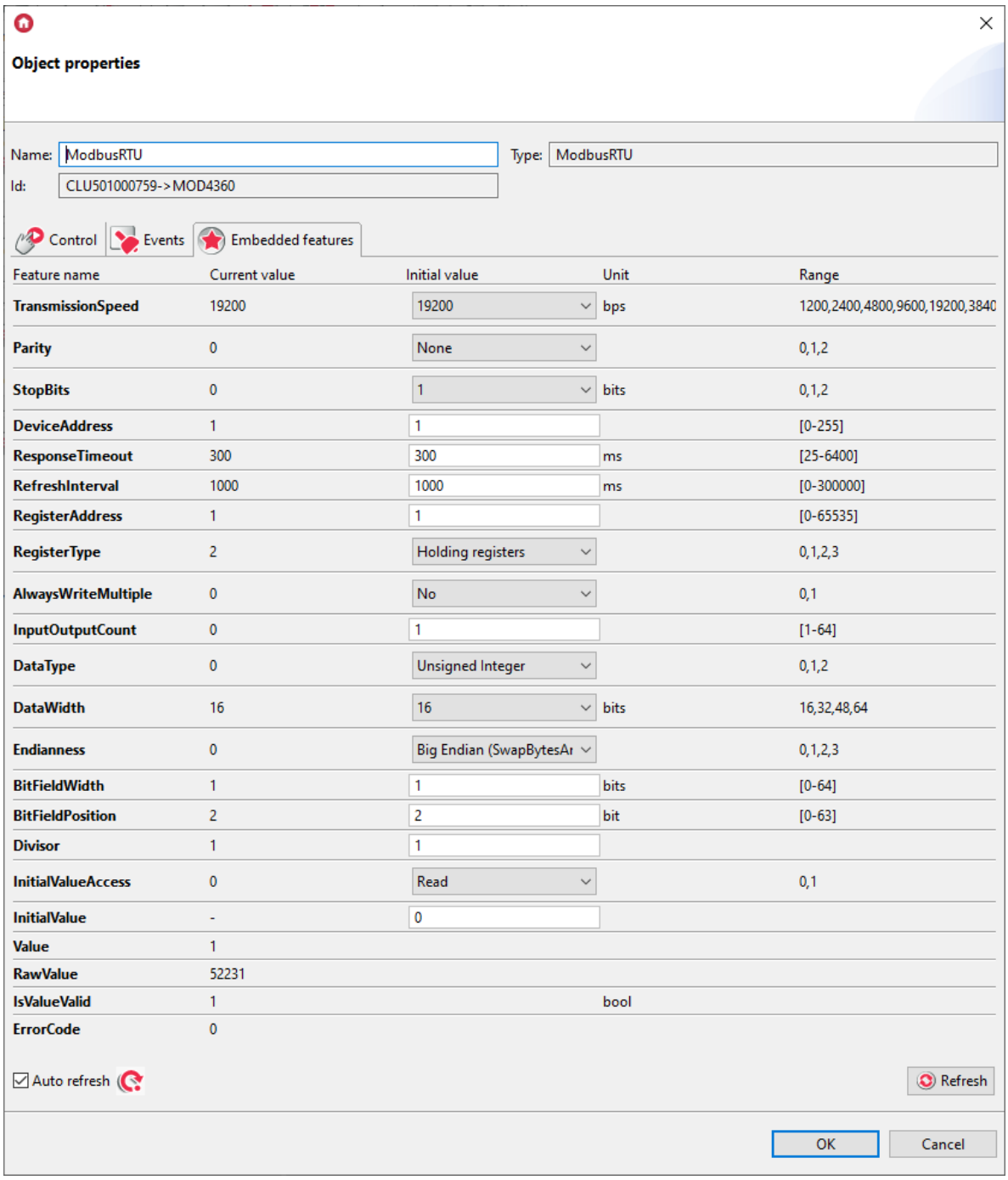
B. Bit fields in 16-bit registers

- Reading of bit fields in a 16-bit remembering register (`Read Holding Registers`, *FunctionCode = 03*):
  - `RegisterType`: Holding Registers;
  - `DataType`: Unsigned Integer;
  - `DataWidth`: 16;
  - `Endianness`: default value;
  - `BitFieldWidth`: 0 - 16 (number of bits read sequentially; for the value 0, maximal width = `DataWidth` is automatically assumed);
  - `BitFieldPosition`: 0 - 15 (position of the first interesting bit).

> **Note!**
> The range of the `BitFieldWidth` feature depends on the setting of the value of the `DataWidth` feature.
>
> For `DataWidth` = 16, the range of the `BitFieldWidth` feature is [0 - 16].
>
> At the moment of setting the `BitFieldWidth` feature to 0 and sending the configuration, the feature takes the maximum value for the currently set value of the `DataWidth` feature.

> **Note!**
> The range of the `BitFieldPosition` feature depends on the setting of the value of the `DataWidth` feature.
>
> For `DataWidth` = 16, the range of the `BitFieldWidth` feature is [0 - 15].

> **Note!**
> The `BitFieldWidth` and `BitFieldPosition` features are dependent on the `DataWidth` feature according to the condition: `BitFieldWidth` - `BitFieldPosition` <= `DataWidth`
>
> For example:
>
> When setting `DataWidth` and `BitFieldWidth` = 16 and `BitFieldPosition` = 15, the `BitFieldWidth` will automatically be set to = 1.
>
> For `BitFieldWidth` = 0, the `BitFieldPosition` attribute is always 0.



- Reading of bit fields in a 16-bit input register (`Read Input Registers`, *FunctionCode = 04*):
  - `RegisterType`: Input Registers;
  - `DataType`: Unsigned Integer;
  - `DataWidth`: 16;
  - `Endianness`: default value;
  - `BitFieldWidth`: 0 - 16 (number of bits read sequentially; for the value 0, maximal width = `DataWidth` is automatically assumed);
  - `BitFieldPosition`: 0 - 15 (position of the first interesting bit).

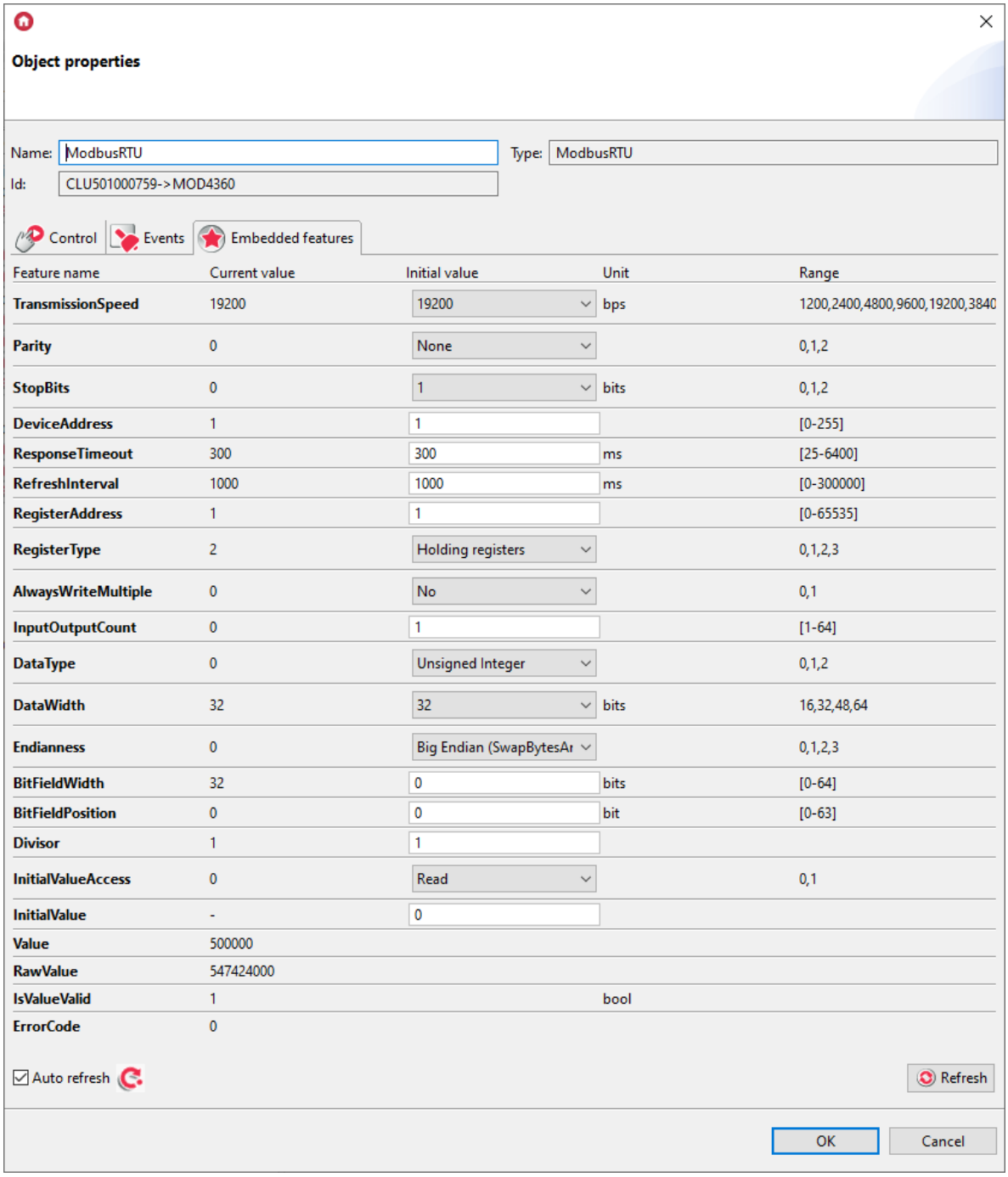
  Writing bit fields in a 16-bit reminder register (`Preset / Write Single Holding Register`, *FunctionCode = 06*):
  - `RegisterType`: Holding Registers;
  - `DataType`: Unsigned Integer;
  - `DataWidth`: 16;
  - `Endianness`: default value;

- `BitFieldWidth`: 0 - 16 (number of bits read sequentially; for the value 0, maximal width = `DataWidth` is automatically assumed);
- `BitFieldPosition`: 0 - 15 (position of the first interesting bit).

C. 32 - bit integer values of registers

Reading 32 - bit integer values of the retaining register (`Read Holding Registers`, *FunctionCode = 03*):

- `RegisterType`: Holding Registers;
- `DataType`: Unsigned Integer;
- `DataWidth`: 32;
- `Endianness`: Big Endian;
- `BitFieldWidth`: default value;
- `BitFieldPosition`: default value.



Reading 32 - bit integer values of the input register (`Read Input Registers`, *FunctionCode = 04*):

- `RegisterType`: Input Registers;
- `DataType`: Unsigned Integer;
- `DataWidth`: 32;
- `Endianness`: Big Endian;
- `BitFieldWidth`: default value;
- `BitFieldPosition`: default value.

Writing of 32 - bit integers in the remembering register (`Preset / Write Multiple Holding Registers`, *FunctionCode = 16*):

- `RegisterType`: Holding Registers;
- `DataType`: Unsigned Integer;
- `DataWidth`: 32;
- `Endianness`: Big Endian;
- `BitFieldWidth`: default value;
- `BitFieldPosition`: default value.

D. Bit fields in 32 - bit registers

Reading of bit fields in a 32 - bit remembering register (`Read Holding Registers`, *FunctionCode = 03*):

- `RegisterType`: Holding Registers;
- `DataType`: Unsigned Integer;
- `DataWidth`: 32;
- `Endianness`: default value;
- `BitFieldWidth`: 0 - 32 (number of bits read sequentially; for the value 0, maximal width = `DataWidth` is automatically assumed);
- `BitFieldPosition`: 0 - 31 (position of the first interesting bit).

Note!
The range of the `BitFieldWidth` feature depends on the setting of the value of the `DataWidth` feature.

For `DataWidth` = 32, the range of the `BitFieldWidth` feature is [0 - 32].

At the moment of setting the `BitFieldWidth` feature to 0 and sending the configuration, the feature takes the maximum value for the currently set value of the `DataWidth` feature.

Note!
The range of the `BitFieldPosition` feature depends on the setting of the value of the `DataWidth` feature.

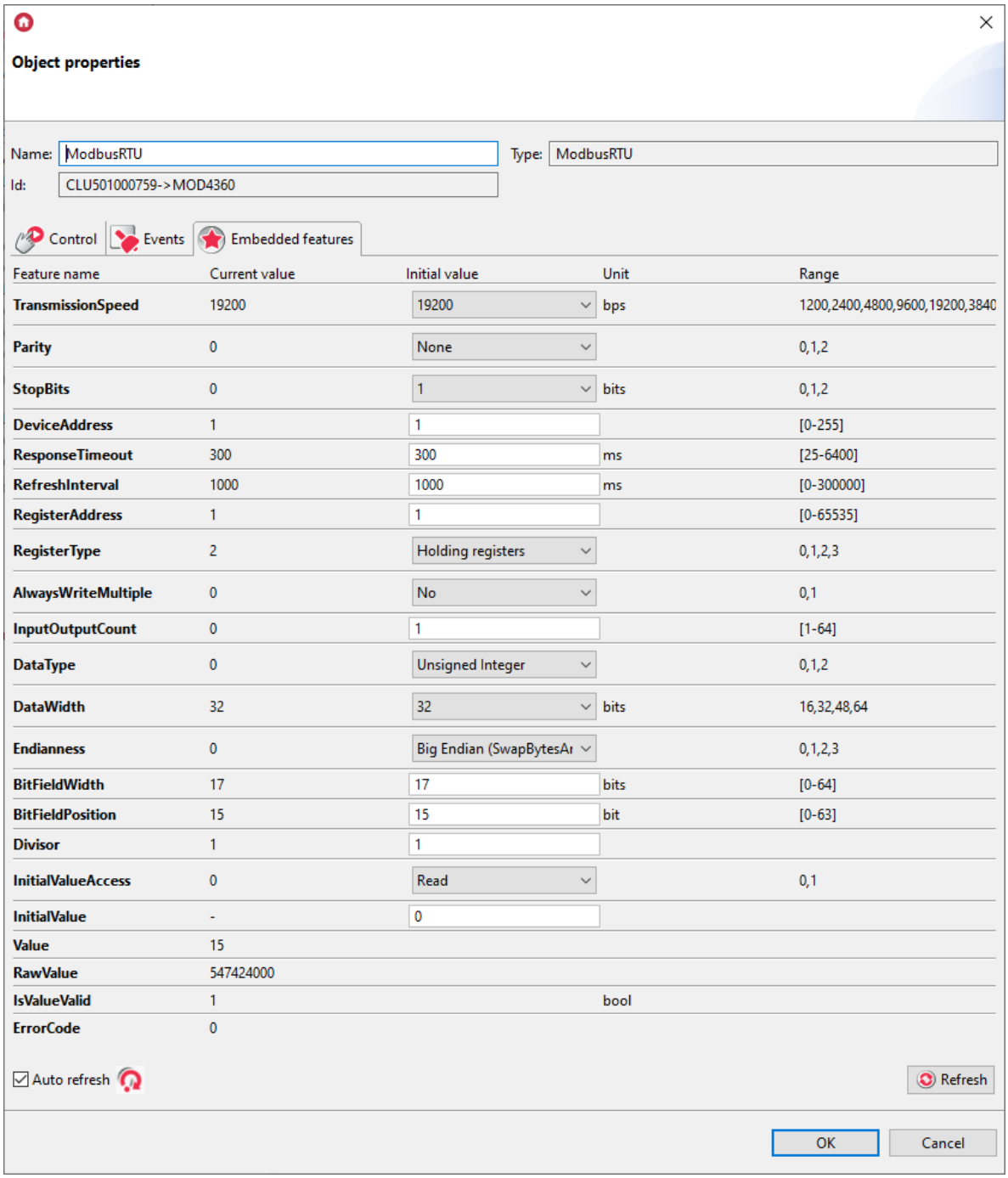
For `DataWidth` = 32, the range of the `BitFieldPosition` feature is [0 - 31].

Note!
The `BitFieldWidth` and `BitFieldPosition` features are dependent on the `DataWidth` feature according to the condition: `BitFieldWidth` - `BitFieldPosition` <= `DataWidth`

For example:

When setting `DataWidth` and `BitFieldWidth` = 32 and `BitFieldPosition` = 15, the `BitFieldWidth` will automatically be set to = 17.

> For `BitFieldWidth` = 0, the `BitFieldPosition` attribute is always 0.



Reading of bit fields in a 32 - bit input register (`Read Input Registers`, *FunctionCode = 04*):

- `RegisterType`: Input Registers;
- `DataType`: Unsigned Integer;
- `DataWidth`: 32;
- `Endianness`: default value;
- `BitFieldWidth`: 0 - 32 (number of bits read sequentially; for the value 0, maximal width = `DataWidth` is automatically assumed);
- `BitFieldPosition`: 0 - 31 (position of the first interesting bit).

Writing bit fields in a 32 - bit reminder register (`Preset / Write Single Holding Register`, *FunctionCode = 06*):

- `RegisterType`: Holding Registers;
- `DataType`: Unsigned Integer;
- `DataWidth`: 32;

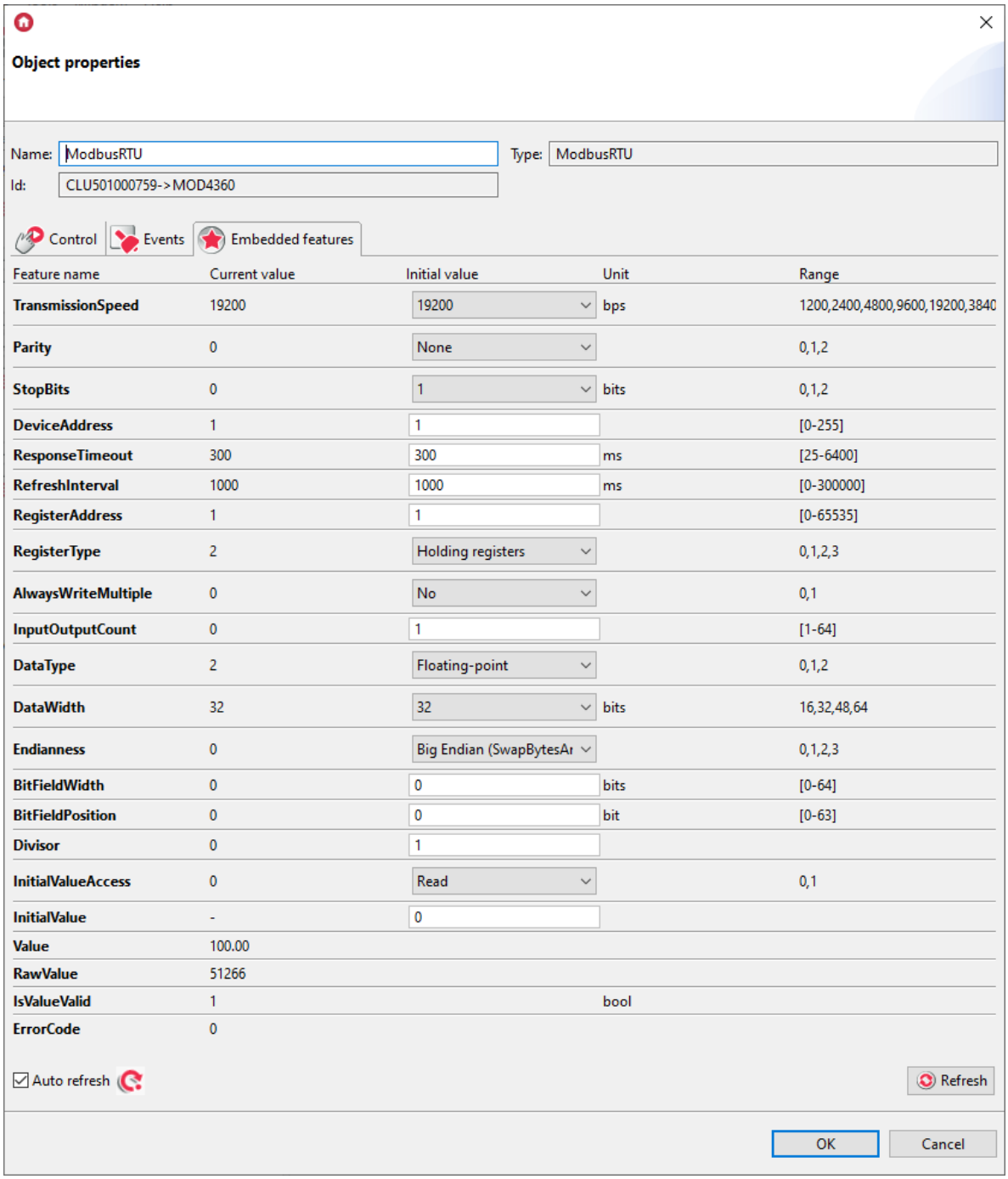
- `Endianness`: default value;
- `BitFieldWidth`: 0 - 32 (number of bits read sequentially; for the value 0, maximal width = `DataWidth` is automatically assumed);
- `BitFieldPosition`: 0 - 31 (position of the first interesting bit).

E. 32 - bit floating point values of registers

Reading of the 32 - bit floating point values of the remembering register (`Read Holding Registers`, *FunctionCode = 03*):

- `RegisterType`: Holding Registers;
- `DataType`: Floating-point;
- `DataWidth`: 32;
- `Endianness`: Big Endian.

> For `DataType` = Floating-point, the `BitFieldWidth`, `BitFieldPosition` and `Divisor` features are inactive and are always 0!



Reading of 32 - bit floating point register values (`Read Input Registers`, *FunctionCode = 04*):

- `RegisterType`: Input Registers;
- `DataType`: Floating-point;
- `DataWidth`: 32;
- `Endianness`: Big Endian.

Record of 32 - bit floating point values in the reminder register (`Preset / Write Multiple Holding Registers`, *FunctionCode = 16*):

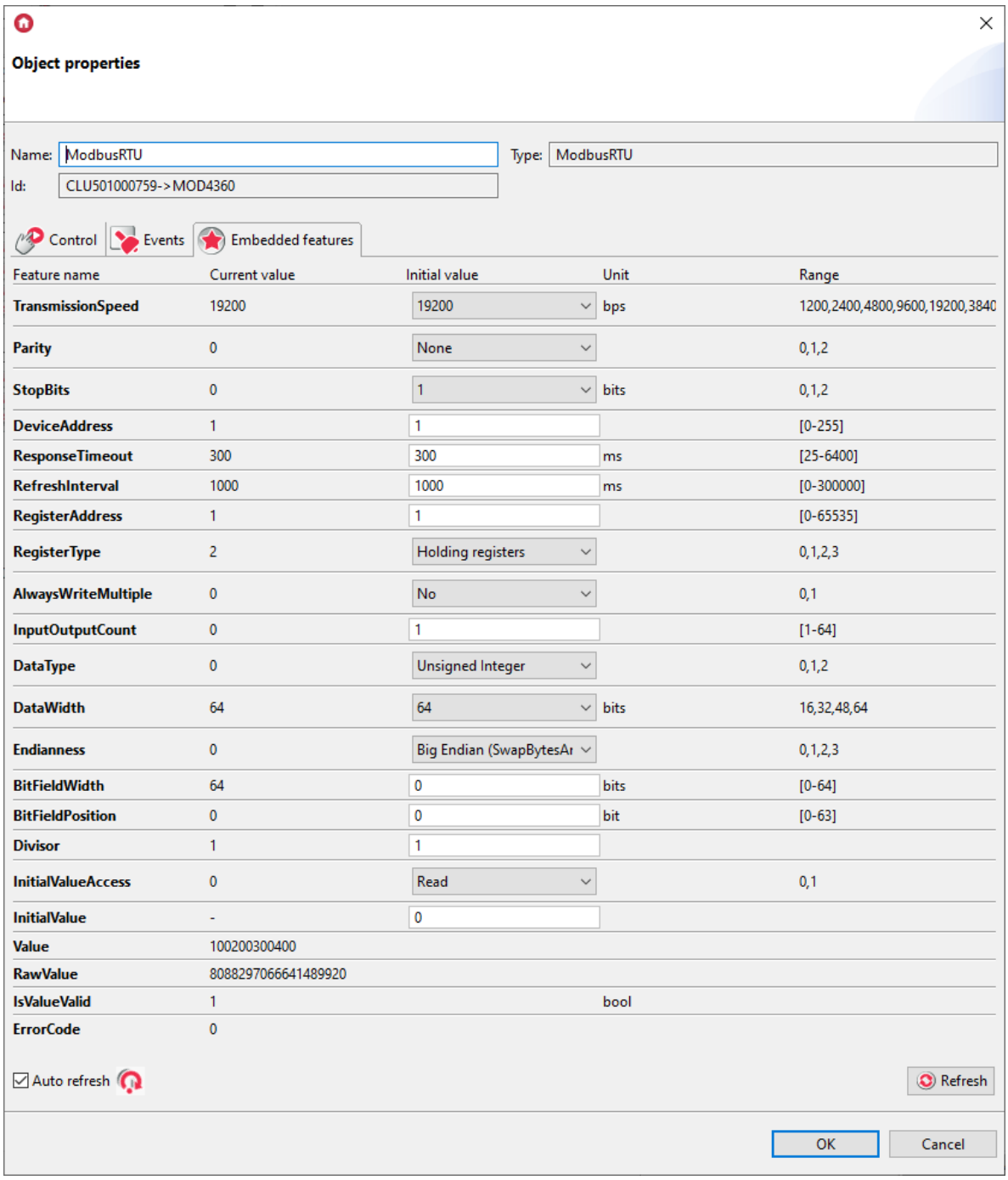
- `RegisterType`: Holding Registers;
- `DataType`: Floating-point;
- `DataWidth`: 32;
- `Endianness`: Big Endian.

> **Note!**
>
> Incorrect setting of `DataWidth` = 16 or 48 for the `Floating-point` type will cause the GATE module to go into emergency mode.

F. 64-bit integer values of registers

Reading 64-bit holding registers (`Read Holding Registers`, *FunctionCode = 03*):

- `RegisterType`: Holding Registers;
- `DataType`: Unsigned Integer;
- `DataWidth`: 64;
- `Endianness`: default value;
- `BitFieldWidth`: default value;
- `BitFieldPosition`: default value.



Reading 64-bit input registers (`Read Input Registers` , *FunctionCode = 04*):

- `RegisterType` : Input Registers;
- `DataType` : Unsigned Integer;
- `DataWidth` : 64;
- `Endianness` : default value;
- `BitFieldWidth` : default value;
- `BitFieldPosition` : default value.

Records of 64-bit holding registers (`Preset / Write Single Holding Register` , *FunctionCode = 06*):

- `RegisterType` : Holding Registers;
- `DataType` : Unsigned Integer;
- `DataWidth` : 64;
- `Endianness` : default value;
- `BitFieldWidth` : default value;
- `BitFieldPosition` : default value.

G. Bit fields in 64-bit registers

Reading of bit fields in a 64-bit remembering register (`Read Holding Registers`, *FunctionCode = 03*):

- `RegisterType`: Holding Registers;
- `DataType`: Unsigned Integer;
- `DataWidth`: 64;
- `Endianness`: default value;
- `BitFieldWidth`: 0 - 64 (number of bits read sequentially);
- `BitFieldPosition`: 0 - 63 (position of the first interesting bit).

Note!
The range of the `BitFieldWidth` feature depends on the setting of the value of the `DataWidth` feature.

For `DataWidth` = 64, the range of the `BitFieldWidth` feature is [0 - 64].

At the moment of setting the `BitFieldWidth` feature to 0 and sending the configuration, the feature takes the maximum value for the currently set value of the `DataWidth` feature.

Note!
The range of the `BitFieldPosition` feature depends on the setting of the value of the `DataWidth` feature.

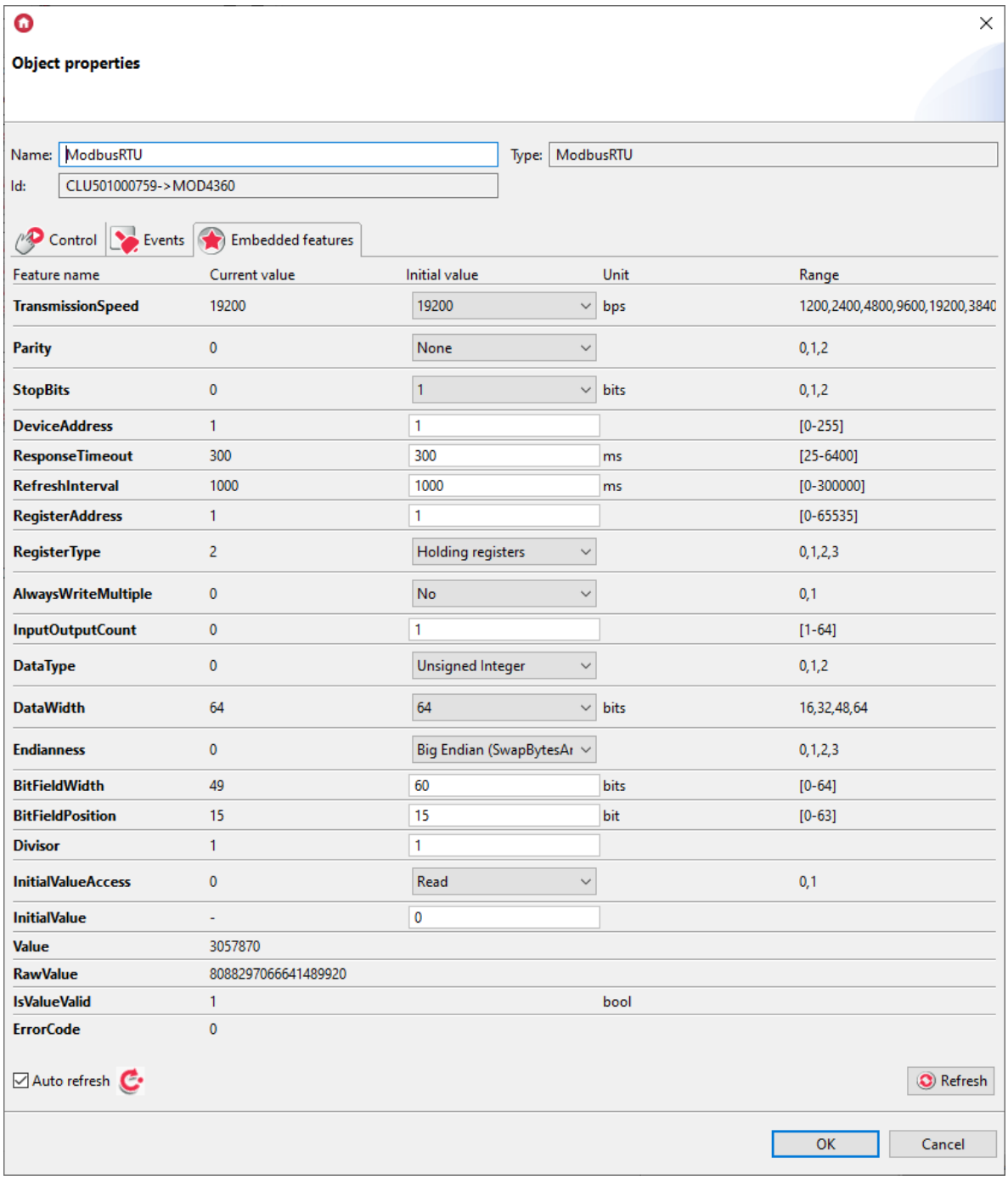
For `DataWidth` = 64, the range of the `BitFieldPosition` feature is [0 - 63].

Note!
The `BitFieldWidth` and `BitFieldPosition` features are dependent on the `DataWidth` feature according to the condition: `BitFieldWidth` - `BitFieldPosition` <= `DataWidth`

For example:

When setting `DataWidth` and `BitFieldWidth` = 60 and `BitFieldPosition` = 15, the `BitFieldWidth` will automatically be set to = 49.

For `BitFieldWidth` = 0, the `BitFieldPosition` attribute is always 0.



Reading of bit fields in a 64-bit input register (`Read Input Registers`, *FunctionCode = 04*):

- `RegisterType`: Input Registers;
- `DataType`: Unsigned Integer;
- `DataWidth`: 64;
- `Endianness`: default value;
- `BitFieldWidth`: 0 - 64 (number of bits read sequentially; for the value 0, maximal width = `DataWidth` is automatically assumed);
- `BitFieldPosition`: 0 - 63 (position of the first interesting bit).

Writing bit fields in a 64-bit reminder register (`Preset / Write Single Holding Register`, *FunctionCode = 06*):

- `RegisterType`: Holding Registers;
- `DataType`: Unsigned Integer;
- `DataWidth`: 64;
- `Endianness`: default value;

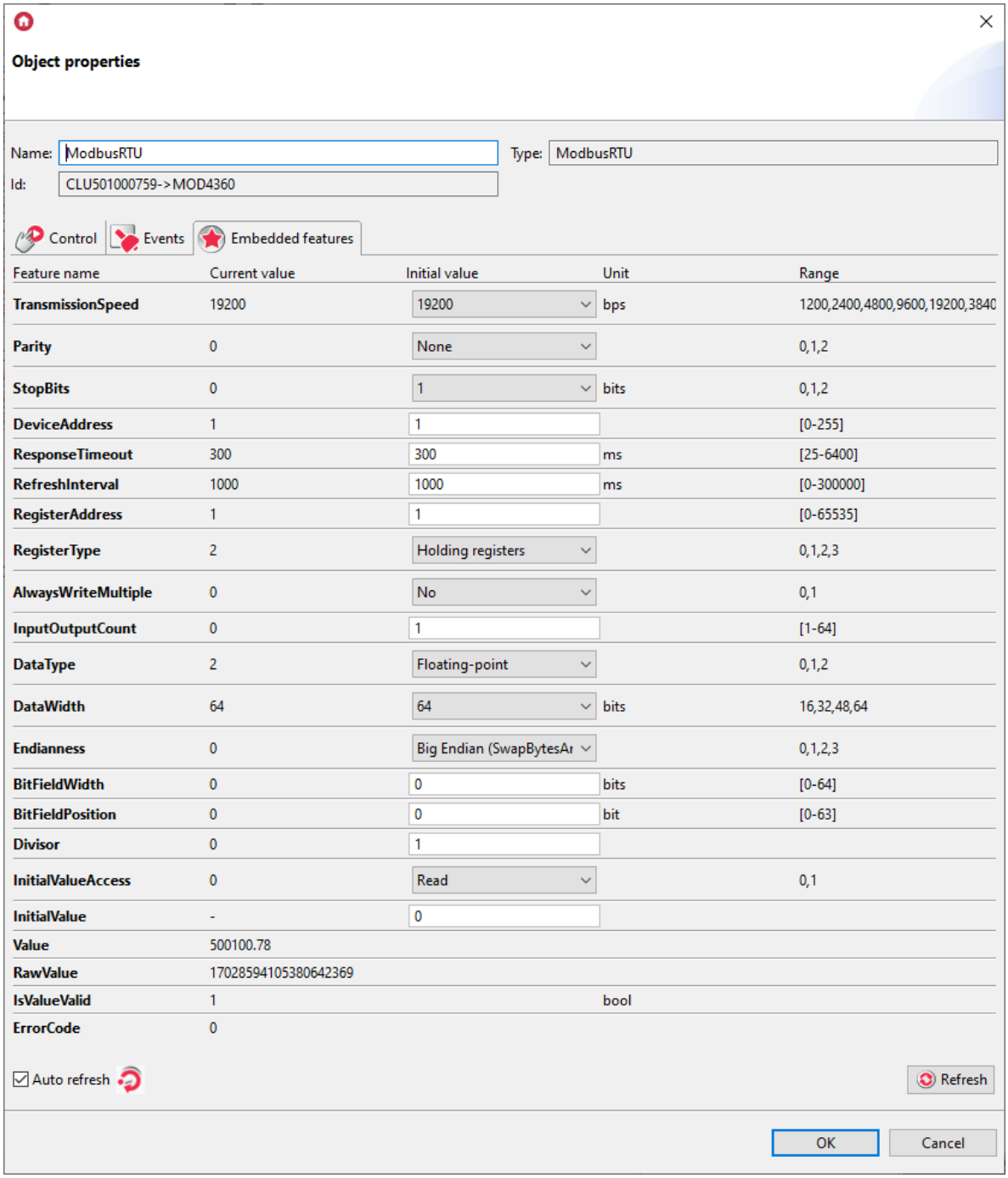
- `BitFieldWidth`: 0 - 64 (number of bits read sequentially; for the value 0, maximal width = `DataWidth` is automatically assumed);
- `BitFieldPosition`: 0 - 63 (position of the first interesting bit).

H. 64-bit floating point values of registers

Reading of the 64-bit floating point values of the remembering register (`Read Holding Registers`, *FunctionCode = 03*):

- `RegisterType`: Holding Registers;
- `DataType`: Floating-point;
- `DataWidth`: 64;
- `Endianness`: Big Endian.

> For `DataType` = Floating-point, the `BitFieldWidth`, `BitFieldPosition` and `Divisor` features are inactive and are always 0!



Reading of 64-bit floating point register values (`Read Input Registers`, *FunctionCode = 04*):

- `RegisterType`: Input Registers;
- `DataType`: Floating-point;
- `DataWidth`: 64;
- `Endianness`: Big Endian.

Record of 64-bit floating point values in the reminder register (`Preset / Write Multiple Holding Registers`, *FunctionCode = 16*):

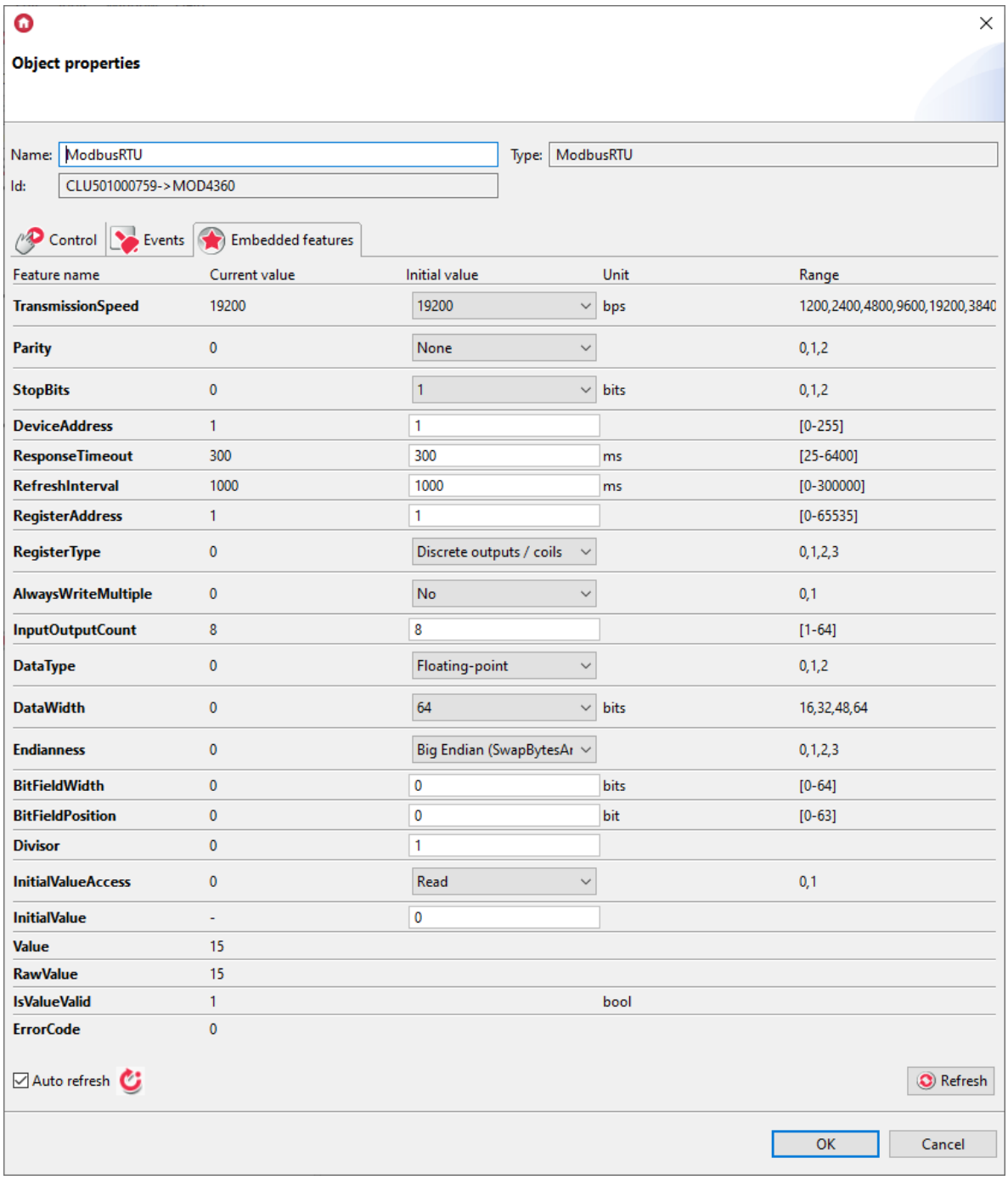
- `RegisterType`: Holding Registers;
- `DataType`: Floating-point;
- `DataWidth`: 64;
- `Endianness`: Big Endian.

J. Discrete inputs / outputs

Readout of discrete outputs / bit inputs (`Read Coils`, *FunctionCode = 01*):

- `RegisterType`: Discrete outputs / coils;
- `InputOutputCount`: 1 - 64 (number of discrete inputs / outputs subject to read / write operations);

> For the `Discrete outputs / coils` and `Discrete Inputs` register types the features `DataType`, `DataWidth`, `Endianness`, `BitFieldWidth`, `BitFieldPosition`, `Divisor` are inactive and always have the value 0!



Readout of discrete binary inputs (`Read Discrete Inputs` , *FunctionCode = 02*):

- `RegisterType` : Discrete Inputs;
- `InputOutputCount` : 1 - 64 (number of discrete inputs / outputs subject to read / write operations);

Writing of discrete outputs / bit inputs (`Write Single Coil` , *FunctionCode = 05*):

- `RegisterType` : Discrete outputs / coils;
- `AlwaysWriteMultiple` : No
- `InputOutputCount` : 1 - 64 (number of discrete inputs / outputs subject to read / write operations);

Writing of discrete outputs / bit inputs (`Write Multiple Coils` , *FunctionCode = 15*):

- `RegisterType` : Discrete outputs / coils;
- `AlwaysWriteMultiple` : No
- `InputOutputCount` : 2 - 64 (number of discrete inputs / outputs subject to read / write operations);

> **Note!**
> When writing values to a single binary input `InputOutputCount` = 1, it is possible to use function 15 (Force/Write Single Coil) instead of function 5 (`Force/Write Multiple Coils`) if the Slave device only accepts this type of writing. To do this, use the `AlwaysWriteMultiple` feature and set it to 1.

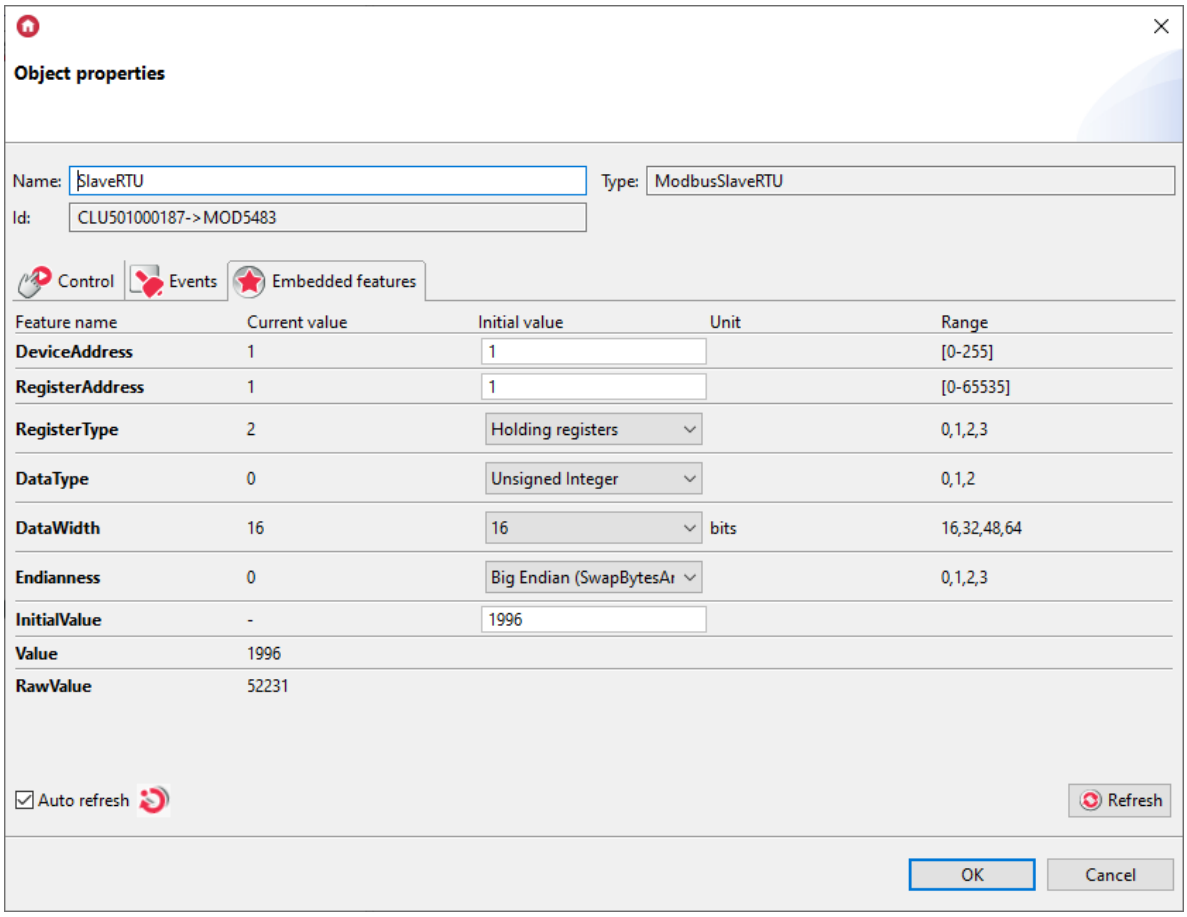
Writing of discrete outputs / bit inputs (`Write Multiple Coils` , *FunctionCode = 15*):

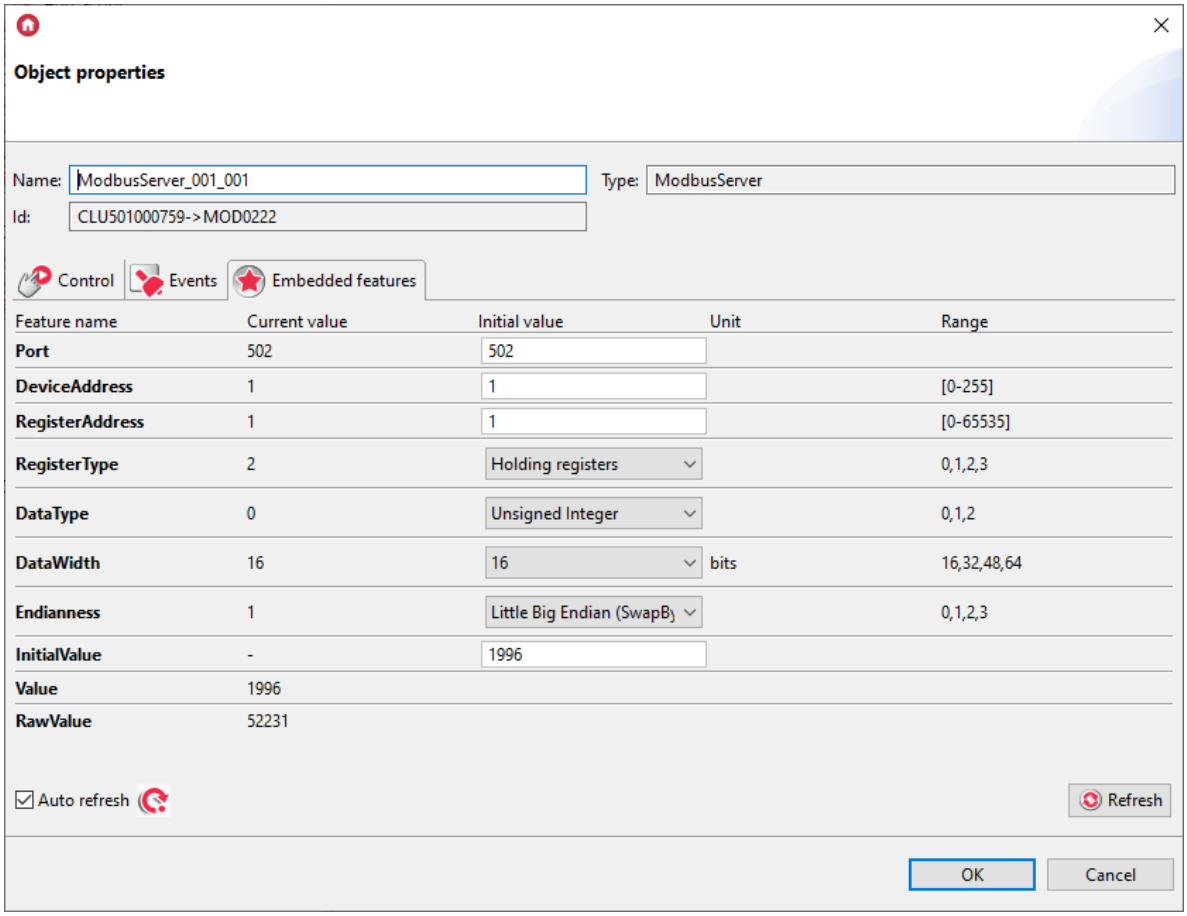
- `RegisterType` : Discrete outputs / coils;
- `AlwaysWriteMultiple` : Yes
- `InputOutputCount` : 1 (number of discrete inputs / outputs subject to read / write operations);

4.2. ModbusSlaveRTU and ModbusServer object

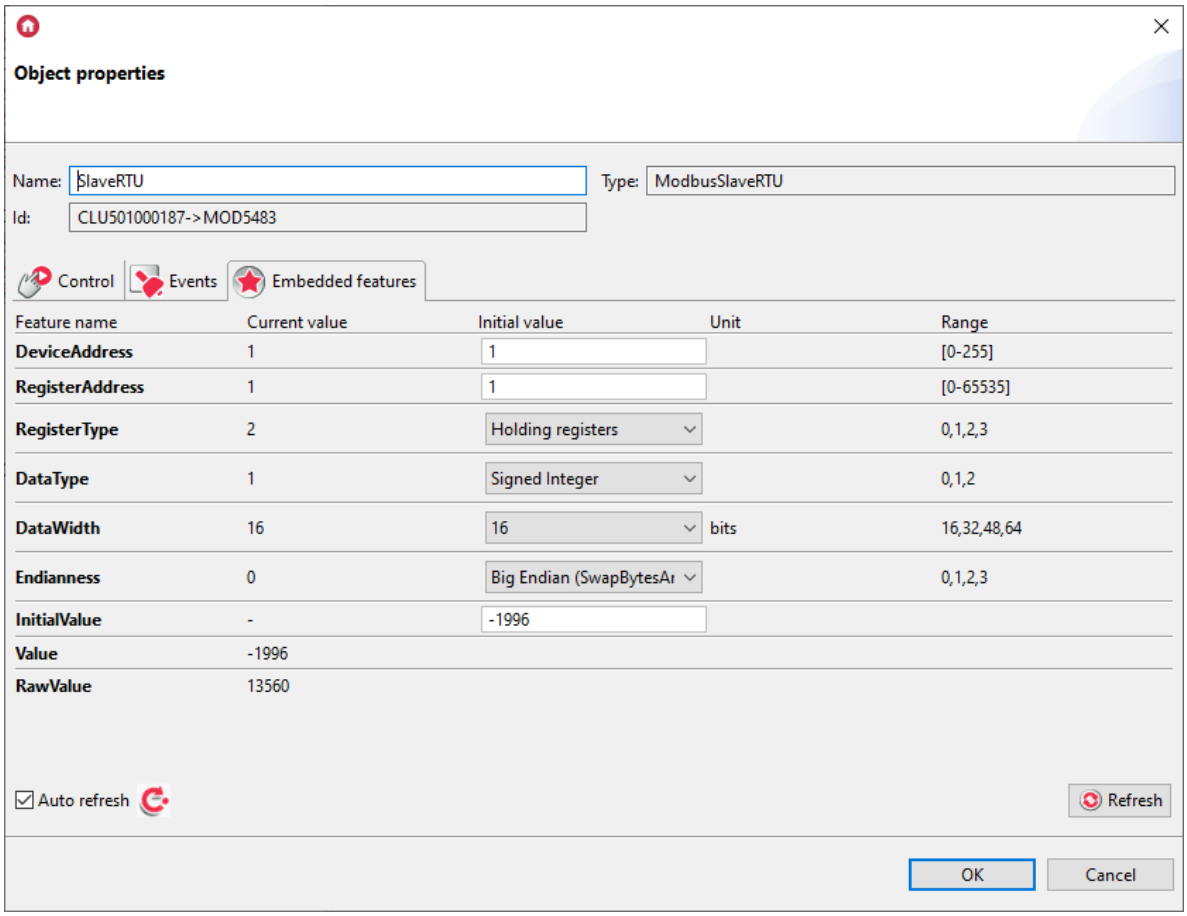
`ModbusSlaveRTU` / `ModbusServer` objects are used to store data using registers available in the MODBUS protocol. The register parameters for objects do not differ in the configuration area - the only difference is in the method of communication with the target device. Below are examples of ways to store values in registers.

A. 16-bit register integer values

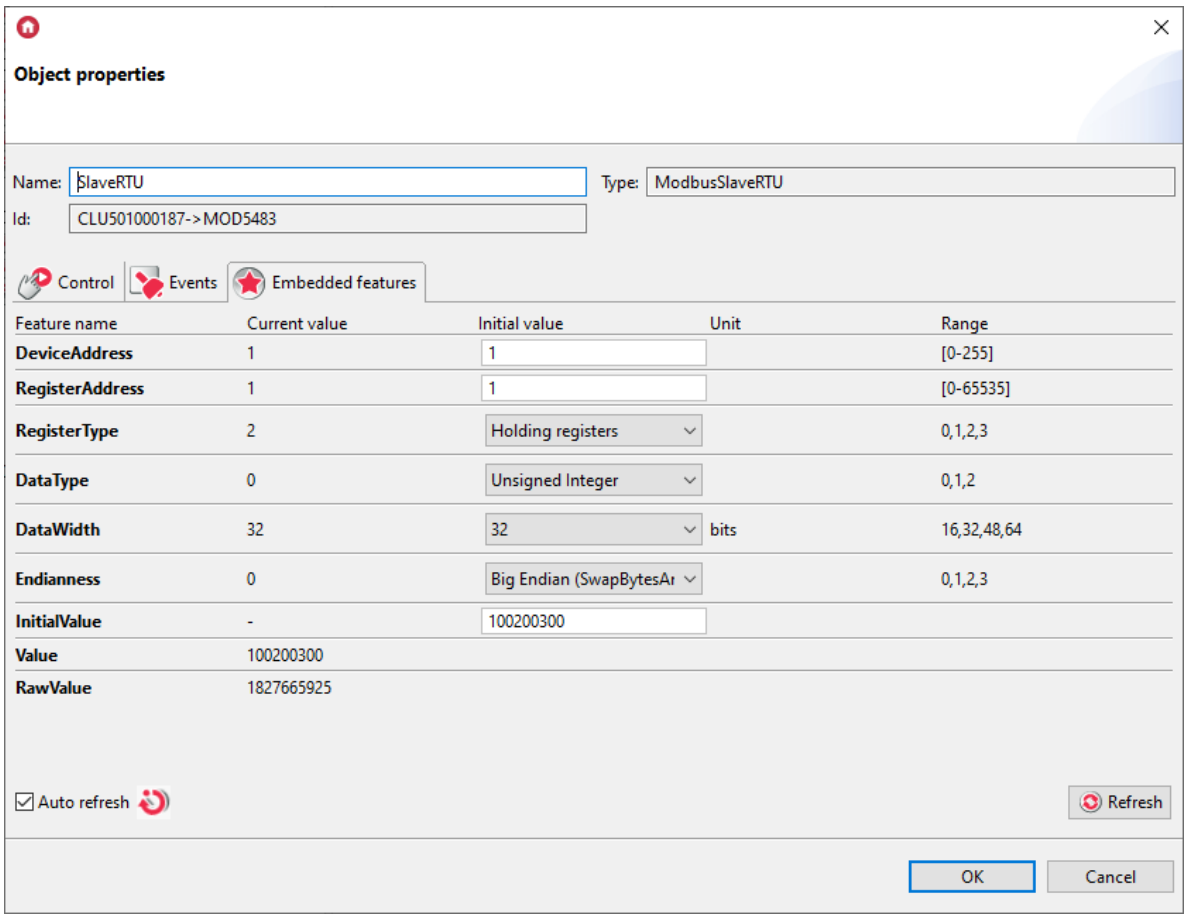




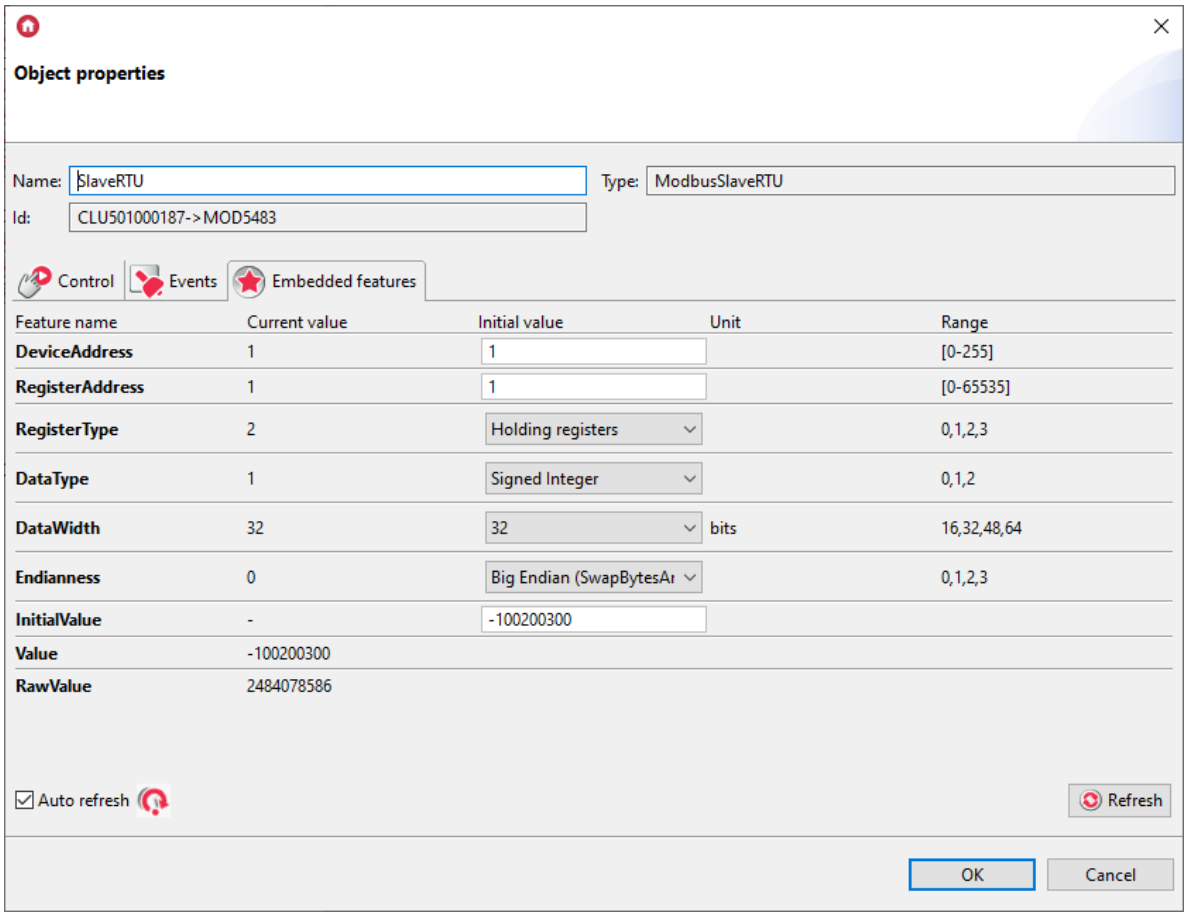
B. 16-bit negative integer register values



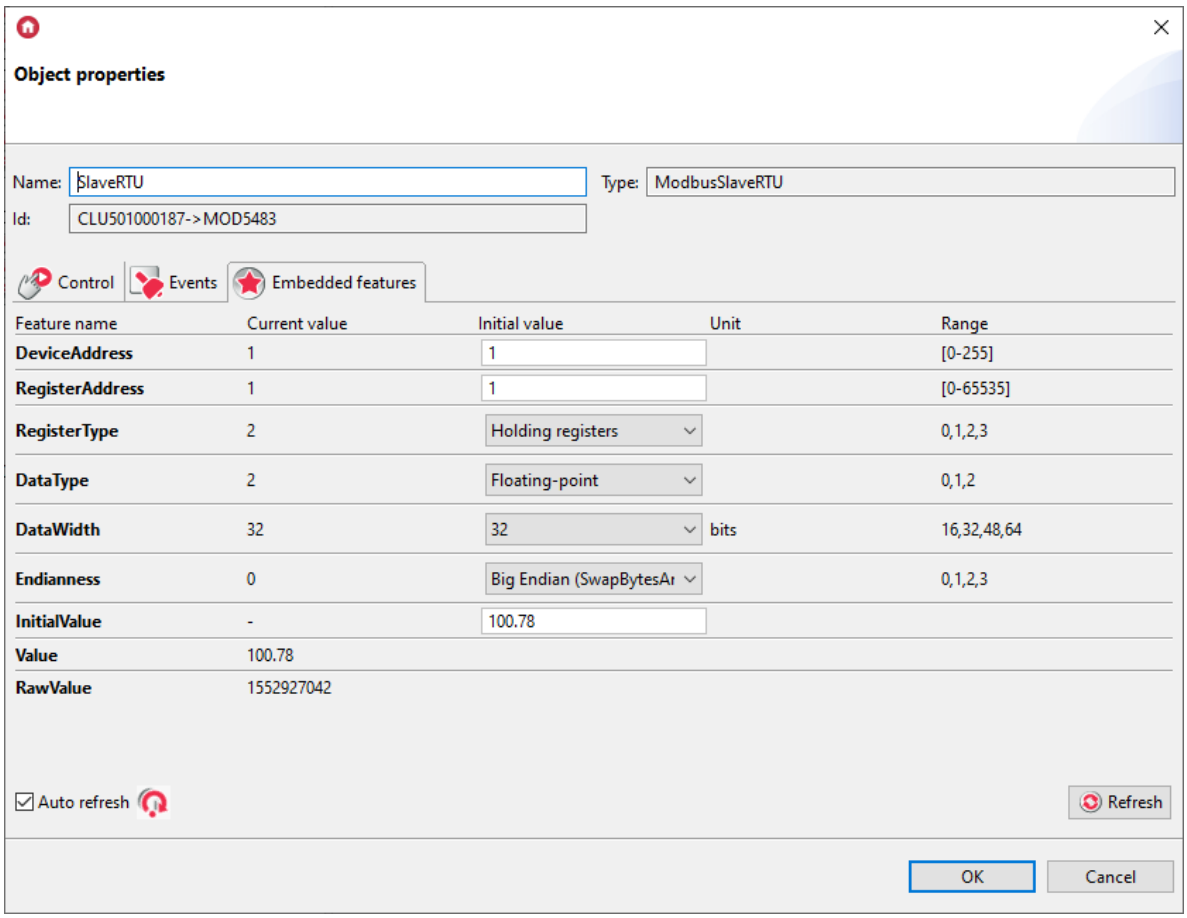
C. 32 - bit register integer values



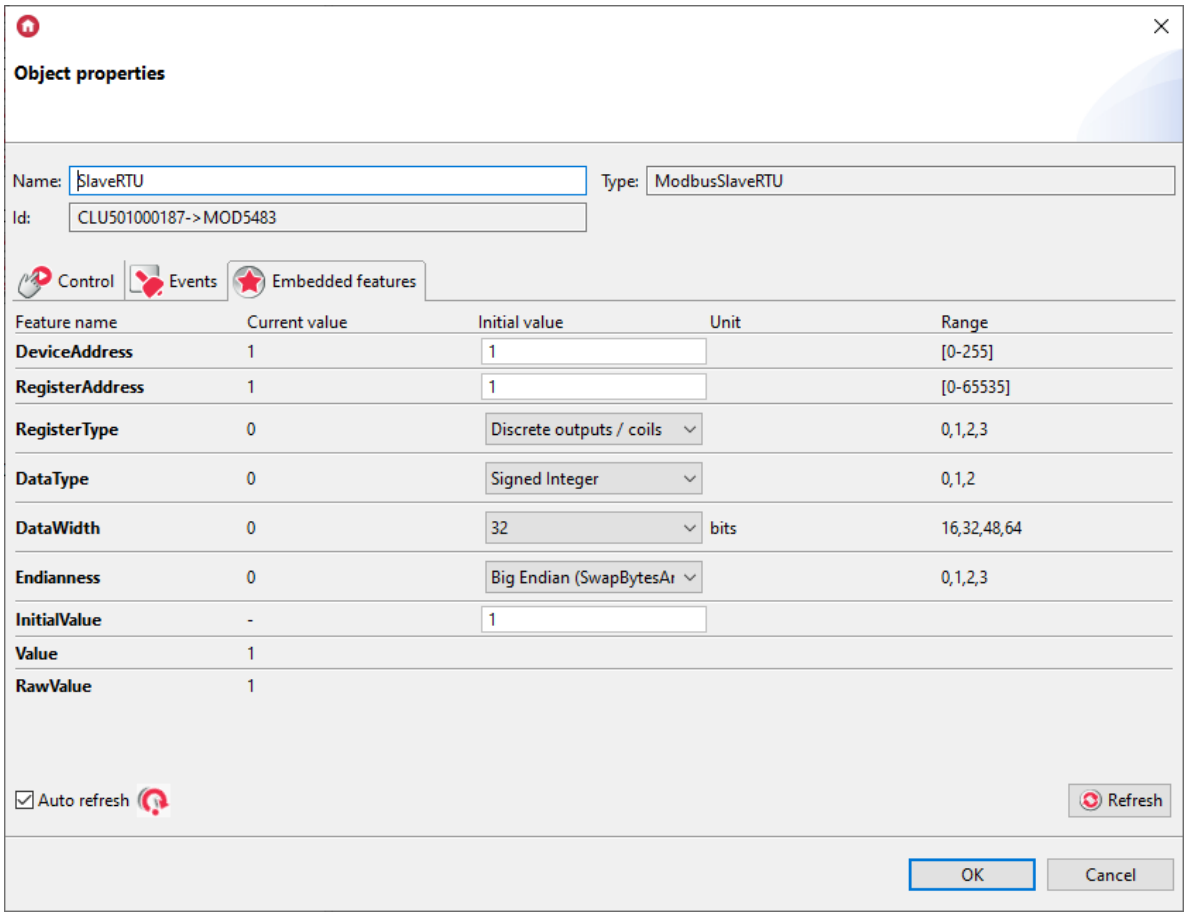
D. 32 - bit negative integer register values

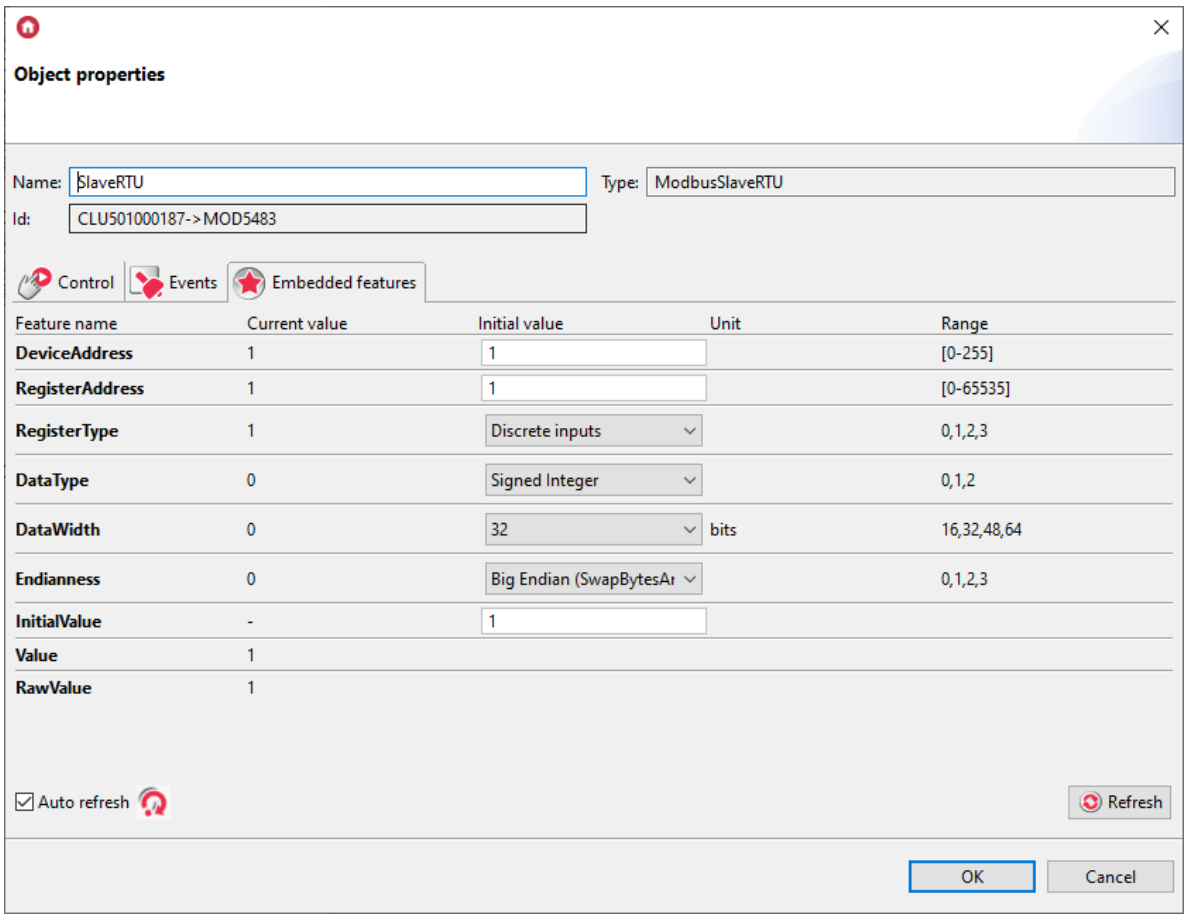


E. 32 - bit floating point values



F. Discrete outputs (coils) and Discrete inputs





Note!

`ModbusSlaveRTU` object for the `Discrete Inputs` and `Discrete Outputs` register types corresponds to one data bit. Each subsequent object created is another bit.

5. Restoring factory settings - *Hard Reset*

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.

6. Configuration parameters

> **Note!**
> The described functionality and integration is available for **GRENTON GATE MODBUS MASTER, DIN, Eth (INT-201 - E-01)** with **firmware 1.4.1 - 2334 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
`TelnetLogLevel`	Specifies the logging level
`TelnetBusLogLevel`	Specifies the Modbus logging level
`ModbusMasterFrameSpace`	Specifies extra Modbus silent interval in character times
`OverloadDetection`	Specifies whether Gate should report CPU overload using the red LED
`ResetReason`	Specifies the device reset reason: 0 - power-on; 2 - configuration reload; 3 - system exception;

METHODS

Name	Description
`SetDateTime`	Sets date and time
`SetClientReportInterval`	Sets the reporting period for feature changes
`SetPrimaryDNS`	Sets the PrimaryDNS feature
`SetSecondaryDNS`	Sets the SecondaryDNS feature
`SetTelnetLogLevel`	Specifies the logging level
`SetTelnetBusLogLevel`	Specifies the Modbus logging level

EVENTS

Name	Description
`OnInit`	An event dispatched when the device initializes

B. Object ModbusRTU

> **Note!**
> The described functionality and integration is available for **GRENTON GATE MODBUS MASTER, DIN, Eth (INT-201 - E-01)** with **firmware 1.4.1 - 2334 or higher!**

FEATURES

Name	Description
`TransmissionSpeed`	Transmission Speed
`Parity`	Parity bit: 0 - None 1 - Odd 2 - Even
`StopBits`	Stop bits: 0 - 1 stop bit 1 - 1.5 stop bits 2 - 2 stop bits
`DeviceAddress`	Modbus device address
`ResponseTimeout`	Response timeout in 25ms steps
`RefreshPeriod`	Minimum refresh period in 5ms steps. 0 means automatic refresh is disabled
`RegisterAddress`	Supported register address
`RegisterType`	Modbus register type: 0 - discrete outputs (coils) - Modbus function: 5 (single output write), 15 (multiple output write) or 1 (read output state) 1 - binary inputs - Modbus function: 2 2 - holding registers - Modbus function: 6 (single register write), 16 (multiple register write) or 3 (register read) 3 - input registers - Modbus function: 4
`AlwaysWriteMultiple`	Always use function 15 or 16 when writing the value
`InputOutputCount`	Specifies the number of discrete IOs to be read / written
`DataType`	Value type: 0 - An integer, a fixed-point number or a bit field without a sign bit 1 - An integer, a fixed-point number or a bit field with a sign bit 2 - A floating-point number
`DataWidth`	Data width (from 1 to 4 16-bit registers)
`Endianness`	Endianness: 0 - Big Endian order of words; Big Endian order of bytes in a word (SwapBytesAndWords) 1 - Little Endian order of words; Big Endian order of bytes in a word (SwapBytes) 2 - Big Endian order of words; Little Endian order of bytes in a word (SwapWords) 3 - Little Endian order of words; Little Endian order of bytes in a word (NoSwap)
`BitFieldWidth`	The number of bits in the bit field. The sum of `BitFieldWidth` and `BitFieldPosition` should be <= `DataWidth`; 0 means no bit field (full data width = `DataWidth`)

Name	Description
`BitFieldPosition`	Starting position of the bit field. Sum of `BitFieldWidth` and `BitFieldPosition` should be <= `DataWidth`
`Divisor`	Value divisor (scale)
`InitialValueAccess`	Initial Value access method: 0 - The initial Value is read from the device 1 - The initial Value is written to the device
`InitialValue`	Defines the initial value
`Value`	Returns the last read value
`RawValue`	Raw value
`IsValueValid`	Determines whether the value is valid
`ErrorCode`	Error code: 1 - illegal function 2 - illegal register number 3 - illegal data value 4 - connected device damaged 5 - positive confirmation 6 - no readiness, message removed 7 - negative confirmation 8 - memory parity error 10 - gateway path unavailable 11 - target device failed to respond 0 - correct read/write register -2 - exceeding the response timeout -3 - frame error (error decoding the frame) -4 - unexpected reply size -5 - unexpected reply code -6 - invalid object state -7 - connection error

Note!

Positive `ErrorCode` are values passed from the Slave device. Their detailed description should be included in the manufacturer's documentation.

METHODS

Name	Description
`SetTransmissionSpeed`	Sets transmission speed
`SetParity`	Sets parity check type
`SetStopBits`	Sets stop bit count
`SetDeviceAddress`	Sets Modbus device address
`SetResponseTimeout`	Sets the response timeout in 25ms steps
`SetRefreshPeriod`	Sets the refresh period in 5ms steps. 0 means automatic refresh is disabled
`SetRegisterAddress`	Sets the supported register address
`SetRegisterType`	Sets the Modbus register type
`SetAlwaysWriteMultiple`	Always use function 15 or 16 when writing the value
`SetInputOutputCount`	Sets the number of discrete IOs to be read / written
`SetDataType`	Sets the variable type
`SetDataWidth`	Sets the data width
`SetEndianness`	Sets byte order
`SetBitFieldWidth`	Sets the bit field width. 0 means no bit field (full `DataWidth`)
`SetBitFieldPosition`	Sets the starting position of the bit field
`SetDivisor`	Sets the divisor
`ReadValue`	Starts reading from the device. Waits for completion in case of no `OnValueRead`
`WriteValue`	Writes a new value to the device

EVENTS

Name	Description
`OnValueChange`	Event occurring after the feature value has been changed over Modbus
`OnValueRead`	Event occurring after the read started by the `ReadValue` method completes
`OnValueWritten`	Event occurring after the write started by the `WriteValue` method completes
`OnError`	Event occurring when the device reports an error

C. Object ModbusSlaveConfigRTU

> **Note!**
> The described functionality and integration is available for **GRENTON GATE MODBUS MASTER, DIN, Eth (INT-201 - E-01)** with **firmware 1.4.1 - 2334 or higher!**

FEATURES

Name	Description
`TransmissionSpeed`	Transmission Speed
`Parity`	Parity bit: 0 - None 1 - Odd 2 - Even
`StopBits`	Stop bits: 0 - 1 stop bit 1 - 1.5 stop bits 2 - 2 stop bits

D. Object ModbusSlaveRTU

> **Note!**
> The described functionality and integration is available for **GRENTON GATE MODBUS MASTER, DIN, Eth (INT-201 - E-01)** with **firmware 1.4.1 - 2334 or higher!**

FEATURES

Name	Description
`DeviceAddress`	Modbus device address
`RegisterAddress`	Supported device address
`RegisterType`	Modbus register type: 0 - discrete outputs (coils) - Modbus function: 5 (single output write), 15 (multiple output write) or 1 (read output state) 1 - binary inputs - Modbus function: 2 2 - holding registers - Modbus function: 6 (single register write), 16 (multiple register write) or 3 (register read) 3 - input registers - Modbus function: 4 "
`DataType`	Value type: 0 - An integer, a fixed-point number or a bit field without a sign bit 1 - An integer, a fixed-point number or a bit field with a sign bit 2 - A floating-point number
`DataWidth`	Data width (from 1 to 4 16-bit registers)
`Endianness`	Endianness: 0 - Big Endian order of words; Big Endian order of bytes in a word (SwapBytesAndWords) 1 - Little Endian order of words; Big Endian order of bytes in a word (SwapBytes) 2 - Big Endian order of words; Little Endian order of bytes in a word (SwapWords) 3 - Little Endian order of words; Little Endian order of bytes in a word (NoSwap)
`InitialValue`	Defines the initial value
`Value`	Register value
`RawValue`	Raw value

METHODS

Name	Description
`SetValue`	Sets the register value

EVENTS

Name	Description
`OnValueChange`	Event occurring after the feature value has been changed over Modbus

E. Object ModbusClient

> **Note!**
> The described functionality and integration is available for **GRENTON GATE MODBUS MASTER, DIN, Eth (INT-201 - E-01)** with **firmware 1.4.1 - 2334 or higher!**

Name	Description
`SocketAddress`	Device IP address
`DeviceAddress`	Modbus device address
`ResponseTimeout`	Time to respond to a frame in steps of 25ms
`RefreshPeriod`	Minimum refresh period in 5ms steps. 0 means automatic refresh is disabled
`RegisterAddress`	Supported register address
`RegisterType`	Modbus register type: 0 - discrete outputs (coils) - Modbus function: 5 (single output write), 15 (multiple output write) or 1 (read output state) 1 - binary inputs - Modbus function: 2 2 - holding registers - Modbus function: 6 (single register write), 16 (multiple register write) or 3 (register read) 3 - input registers - Modbus function: 4
`AlwaysWriteMultiple`	Always use function 15 or 16 when writing the value
`InputOutputCount`	Specifies the number of discrete IOs to be read / written
`DataType`	Value type: 0 - An integer, a fixed-point number or a bit field without a sign bit 1 - An integer, a fixed-point number or a bit field with a sign bit 2 - A floating-point number
`DataWidth`	Data width (from 1 to 4 16-bit registers)
`Endianness`	Endianness: 0 - Big Endian order of words; Big Endian order of bytes in a word (SwapBytesAndWords) 1 - Little Endian order of words; Big Endian order of bytes in a word (SwapBytes) 2 - Big Endian order of words; Little Endian order of bytes in a word (SwapWords) 3 - Little Endian order of words; Little Endian order of bytes in a word (NoSwap)
`BitFieldWidth`	Width of the bit field. Sum of `BitFieldWidth` and `BitFieldPosition` should be <= `DataWidth`; 0 means no bit field (full `DataWidth`)
`BitFieldPosition`	Starting position of the bit field. Sum of `BitFieldWidth` and `BitFieldPosition` should be <= `DataWidth`
`Divisor`	Value divisor (scale)
`InitialValueAccess`	Initial Value access method: 0 - The initial Value is read from the device 1 - The initial Value is written to the device
`InitialValue`	Defines the initial value

Name	Description
`Value`	Returns the last read value
`RawValue`	Raw value
`IsValueValid`	Determines whether the value is valid
`ErrorCode`	Error code: 1 - illegal function 2 - illegal register number 3 - illegal data value 4 - connected device damaged 5 - positive confirmation 6 - no readiness, message removed 7 - negative confirmation 8 - memory parity error 10 - gateway path unavailable 11 - target device not responding 0 - correct read/write register -2 - exceeding the response timeout -3 - frame error (error decoding the frame) -4 - unexpected reply size -5 - unexpected reply code -6 - invalid object state -7 - connection error

Note!

Positive `ErrorCode` are values passed from the Client device. Their detailed description should be included in the manufacturer's documentation.

METHODS

Name	Description
`SetDeviceAddress`	Sets Modbus device address
`SetResponseTimeout`	Sets the response timeout in 25ms steps
`SetRefreshPeriod`	Sets the refresh period in 5ms steps. 0 means automatic refresh is disabled
`SetRegisterAddress`	Sets the supported register address
`SetRegisterType`	Sets the Modbus register type
`SetAlwaysWriteMultiple`	Always use function 15 or 16 when writing the value
`SetInputOutputCount`	Sets the number of discrete IOs to be read / written
`SetDataType`	Sets the variable type
`SetDataWidth`	Sets the data width
`SetEndianness`	Sets byte order
`SetBitFieldWidth`	Sets the bit field width. 0 means no bit field (full `DataWidth`)
`SetBitFieldPosition`	Sets the starting position of the bit field
`SetDivisor`	Sets the divisor
`ReadValue`	Starts reading from the device. Waits for completion in case of no `OnValueRead`
`WriteValue`	Writes a new value to the device

EVENTS

Name	Description
`OnValueChange`	Event occurring after the feature value has been changed over Modbus
`OnValueRead`	Event occurring after the read started by the `ReadValue` method completes
`OnValueWritten`	Event occurring after the write started by the `WriteValue` method completes
`OnError`	Event occurring when the server reports an error

F. Object ModbusServer

> **Note!**
> The described functionality and integration is available for **GRENTON GATE MODBUS MASTER, DIN, Eth (INT-201 - E-01)** with **firmware 1.4.1 - 2334 or higher!**

FEATURES

Name	Description
`Port`	Server listening port
`DeviceAddress`	Modbus device address
`RegisterAddress`	Supported device address
`RegisterType`	Modbus register type: 0 - discrete outputs (coils) - Modbus function: 5 (single output write), 15 (multiple output write) or 1 (read output state) 1 - binary inputs - Modbus function: 2 2 - holding registers - Modbus function: 6 (single register write), 16 (multiple register write) or 3 (register read) 3 - input registers - Modbus function: 4 "
`DataType`	Value type: 0 - An integer, a fixed-point number or a bit field without a sign bit 1 - An integer, a fixed-point number or a bit field with a sign bit 2 - A floating-point number
`DataWidth`	Data width (from 1 to 4 16-bit registers)
`Endianness`	Endianness: 0 - Big Endian order of words; Big Endian order of bytes in a word (SwapBytesAndWords) 1 - Little Endian order of words; Big Endian order of bytes in a word (SwapBytes) 2 - Big Endian order of words; Little Endian order of bytes in a word (SwapWords) 3 - Little Endian order of words; Little Endian order of bytes in a word (NoSwap)
`InitialValue`	Defines the initial value
`Value`	Register value
`RawValue`	Raw value

METHODS

Name	Description
`SetValue`	Sets the register value

EVENTS

Name	Description
`OnValueChange`	Event occurring after the feature `Value` has been changed over Modbus

G. Object Modbus

> **Note!**

> The `Modbus` virtual object has been deprecated since **firmware version 1.4.1 - 2334**. To maintain compatibility, it is possible to use the object only in previously created projects.

FEATURES

Name	Description
`DeviceAddress`	Address of the Slave Modbus device
`AccessRights`	Operating mode: *read* (0 - reading); *read / write* (1 - read / write)
`RegisterAddress`	Address of the supported registry
`TransmissionSpeed`	Transmission speed
`ValueType`	Variable type (1 - *number*; 2 - *float* ; 3 - *bit*)
`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]: `No swap` (0) - no exchange; `Swap bytes and words` (1) - change the order of bytes and words); `Swap bytes` (2) - changing the order of bytes within each word); `Swap words` (3) - exchange of words
`RegisterType`	Modbus register type: `0` - bit inputs / outputs, `1` - binary inputs, `2` - holding registers, `3` - input registers
`ErrorCode`	Error code: `-3` - frame error; `-2` - exceeding the response time; `-1` - out of date value of the last read out register; `0` - correct reading / writing of the register; `1` - not allowed function; `2` - not allowed register number; `3` - unauthorized data value; `4` - damage to the connected device; **5** - positive confirmation; `6` - no readiness / message removed; `7` - negative confirmation; `8` - 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

Name	Description
`Parity`	Parity bit: `0` - None `1` - Odd `2` - Even

METHODS

Name	Description
`SetDeviceAddress`	Sets the address of the Slave Modbus device
`SetAccessRights`	Sets the operating mode: reading or reading / writing
`SetRegisterAddress`	Sets the address of the supported registry
`SetTransmissionSpeed`	Sets the transmission speed
`SetValueType`	Sets the type of variable
`SetBitPosition`	Sets the position of the bit
`SetBitCount`	Sets the number of registry bits to read
`SetRefreshInterval`	Sets the refresh time
`SetReadWriteTimeout`	Sets the response timeout
`SetDivisor`	Sets the divisor
`SetEndianness`	Sets the byte order type
`SetRegisterType`	Sets the Modbus register type
`SetValue`	Sets the read / write value
`SetStopBits`	Sets the number of stop bits
`SetParity`	Sets the parity check

EVENTS

Name	Description
`OnChange`	Event occurring when the state changes (regardless of the value)
`OnError`	Event occurring when the slave device reports an error

H. Object ModbusValue

> **Note!**
>
> The `Modbus` virtual object has been deprecated since **firmware version 1.4.1 - 2334**. To maintain compatibility, it is possible to use the object only in previously created projects.

Name	Description
`TransmissionSpeed`	Transmission Speed
`Parity`	Parity bit: `0` - None `1` - Odd `2` - Even
`StopBits`	Stop bits: `0` - 1 stop bit `1` - 1.5 stop bits `2` - 2 stop bits
`DeviceAddress`	Address of the Slave Modbus device
`ResponseTimeout`	Response timeout in 25ms steps
`RefreshPeriod`	Minimum refresh period in 5ms steps. 0 means automatic refresh is disabled
`RegisterAddress`	Address of the supported registry
`RegisterType`	Modbus register type (0 - Discrete outputs / coils, 1 - Discrete inputs, 2 - Holding registers, 3 - Input registers)
`InputOutputCount`	Specifies the number of discrete I / O that can be read / written
`DataType`	Value type: `0` - An integer, a fixed-point number or a bit field without a sign bit `1` - An integer, a fixed-point number or a bit field with a sign bit `2` - A floating-point number
`DataWidth`	Data width (from 1 to 4 16-bit registers)
`Endianness`	Endianness: `0` - Big Endian order of words; Big Endian order of bytes in a word `1` - Little Endian order of words; Big Endian order of bytes in a word `2` - Big Endian order of words; Little Endian order of bytes in a word `3` - Little Endian order of words; Little Endian order of bytes in a word
`BitFieldWidth`	The number of bits in the bit field. The sum of `BitFieldWidth` and `BitFieldPosition` should be <= `DataWidth`; 0 means no bit field (full data width = `DataWidth`)
`BitFieldPosition`	The position of the youngest bit of the bit field. The sum of `BitFieldWidth` and `BitFieldPosition` should be < = `DataWidth`.
`Divisor`	Divisor
`InitialValueAccess`	Initial Value access method: `0` - The initial Value is read from the device `1` - The initial Value is written to the device

Name	Description
`Value`	Returns the last value read and specifies the initial value
`RawValue`	Unscaled register value
`IsValueValid`	Determines whether a value matches the state of an object
`ErrorCode`	Error code: `1` - illegal function `2` - illegal register number `3` - illegal data value `4` - connected device damaged `5` - positive confirmation `6` - no readiness, message removed `7` - negative confirmation `8` - memory parity error `0` - correct read/write register `-2` - exceeding the response timeout `-3` - frame error (error decoding the frame) `-4` - unexpected reply size `-5` - unexpected reply code

METHODS

Name	Description
`SetTransmissionSpeed`	Sets transmission speed
`SetParity`	Sets parity check type
`SetStopBits`	Sets stop bit count
`SetDeviceAddress`	Sets Modbus slave device address
`SetResponseTimeout`	Sets the response timeout in 25ms steps
`SetRefreshPeriod`	Sets the refresh period in 5ms steps. 0 means automatic refresh is disabled
`SetRegisterAddress`	Sets the supported register address
`SetRegisterType`	Sets the Modbus register type
`SetInputOutputCount`	Sets the number of discrete IOs to be read / written
`SetDataType`	Sets the variable type
`SetDataWidth`	Sets the data width
`SetEndianness`	Sets byte order
`SetBitFieldWidth`	Sets the bit field width. 0 means no bit field (full `DataWidth`)
`SetBitFieldPosition`	Sets the starting position of the bit field
`SetDivisor`	Sets the divisor
`ReadValue`	Starts reading from the device. Waits for completion in case of no OnValueRead
`WriteValue`	Writes a new value to the device

EVENTS

Name	Description
`OnValueChange`	Event occurring after the feature value has been changed or the object parameters have changed
`OnValueRead`	Event occurring after the read started by the ReadValue method completes
`OnError`	Event occurring 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 1.4.2-2346 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.

> **Note!**
> There is no limit to the number of objects for the created virtual objects - the limitation is the device memory, which is influenced by e.g. level of logic expansion on the module.

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 HTTP 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.

> **Note!**
> When the `UseCloud` feature is set to `true`, the `UseNTP` feature is automatically set to `true`.

2.1. Virtual objects

2.1.1. HTTP Request

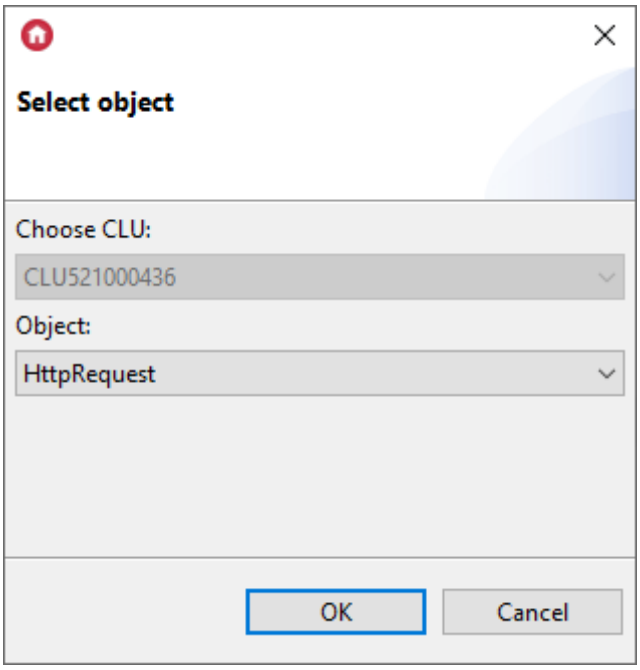
For the HttpRequest, for example, the weather service http://api.openweathermap.org is used

According to the example on the openweathermap.org site, the API query looks like this:

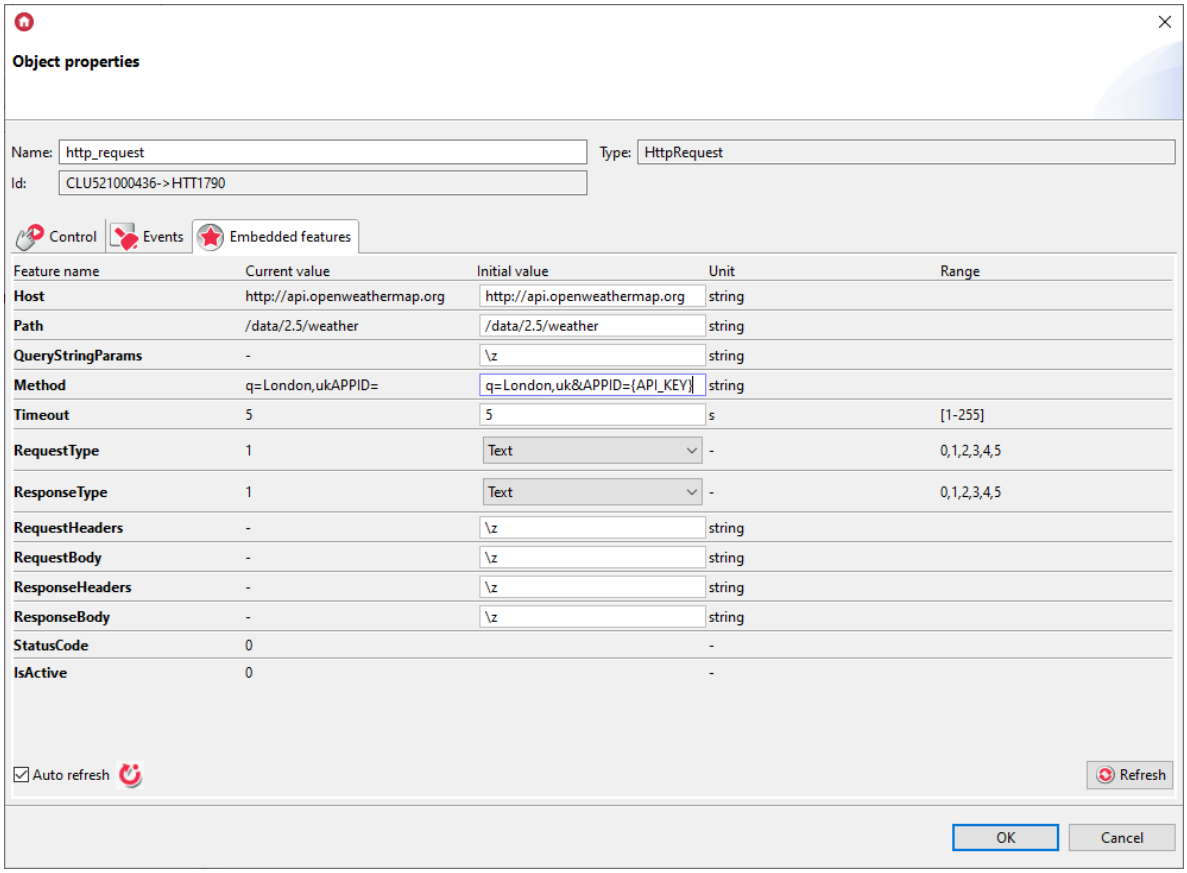
API call: http://api.openweathermap.org/data/2.5/weather?q=London&APPID={APIKEY}

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-> httpr_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).

2.1.3. Setting Request Headers / Reading Response Headers

To set request headers, you need to create a script and define a local script variable.

Examples of setting headers:

- An array of name/value pairs, with each index containing a pair consisting of the header name and value:

```
local header = {
 { name = 'Month', value = 'April' },
 { name = 'Year', value = '2023' },
    { name = 'Name', value = 'Home' },
}

GATE->HttpRequest->RequestHeaders = header
```

- A string (headers separated by '\r\n'):

```
local header =
 'Month: April\r\n' ..
 'Year: 2023\r\n' ..
 'Name: Home\r\n'

GATE->HttpRequest->RequestHeaders = header
```

The `ResponseHeaders` are returned as a Lua array. Below is a sample script for reading `ResponseHeaders`:

```
local header = GATE->HttpRequest->ResponseHeaders

for i,v in ipairs(header) do
 print(v.name .. ': ' .. v.value)
end
```

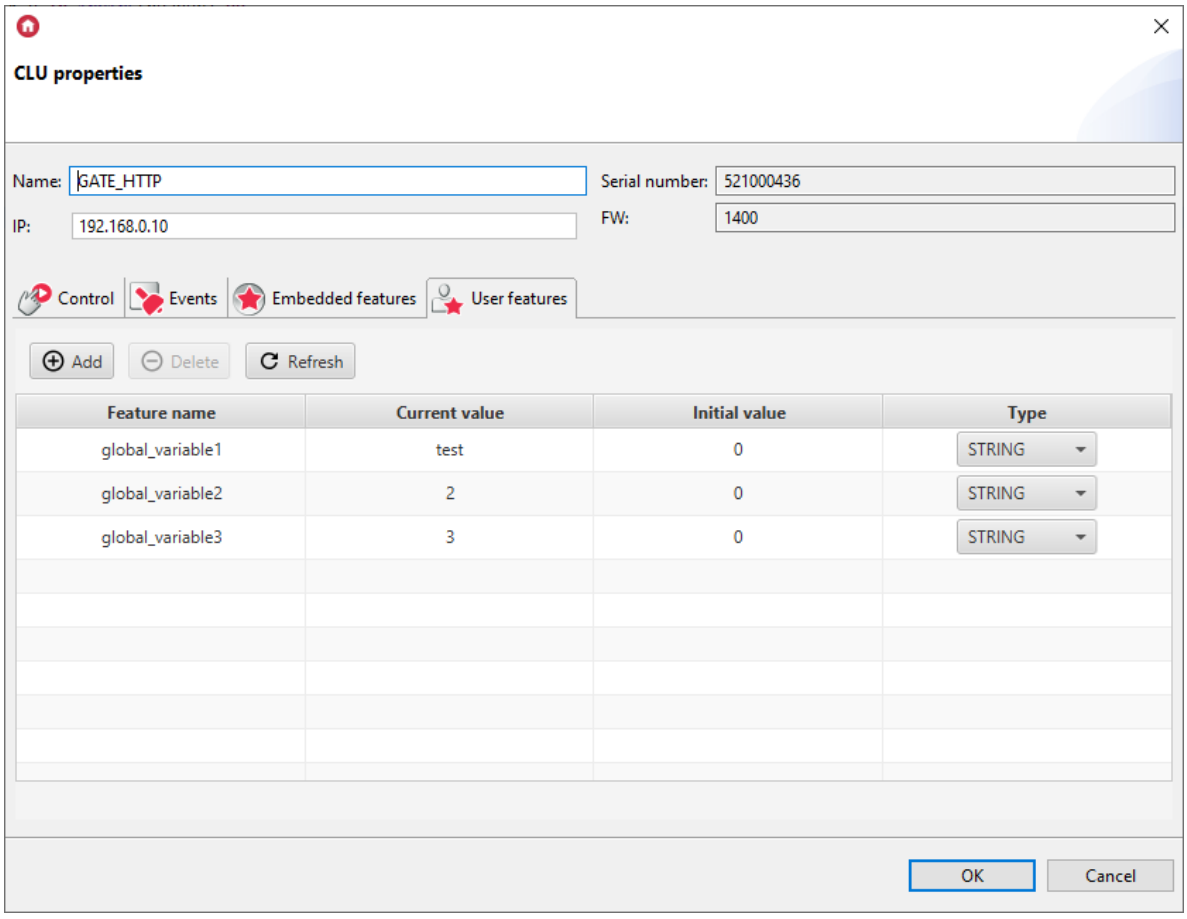
The values are stored in local script variables. To use the values in the system, you need to extract them into global variables. For example:

- Create three global variables: `global_variable1`, `global_variable2`, `global_variable3`
- Create a script where `test1`, `test2`, `test3` are the names of headers received in the response:

```
local header = GATE_HTTP->Request->ResponseHeaders

for i,v in ipairs(header) do
    if v.name == 'test1' then
        zmienna_globalna1 = v.value
    end
    if v.name == 'test2' then
        zmienna_globalna2 = v.value
    end
    if v.name == 'test3' then
        zmienna_globalna3 = v.value
    end
end
```

After running the script, the values of the respective headers are returned:



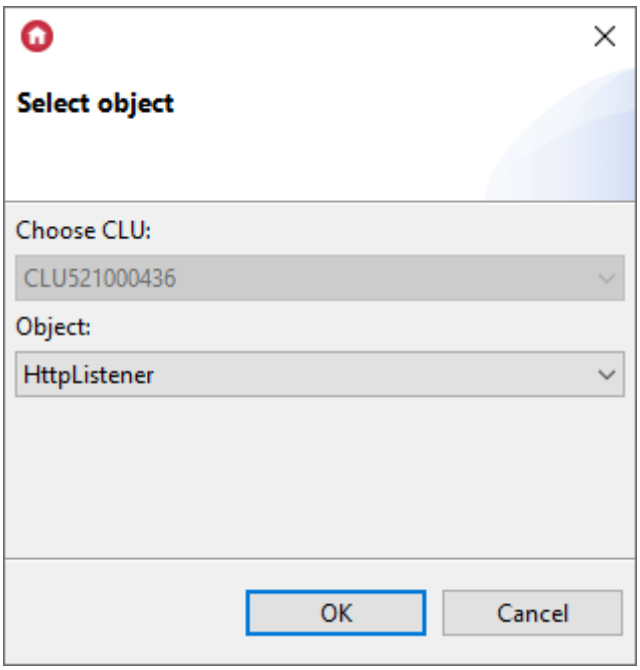
2.2.1. HttpListener

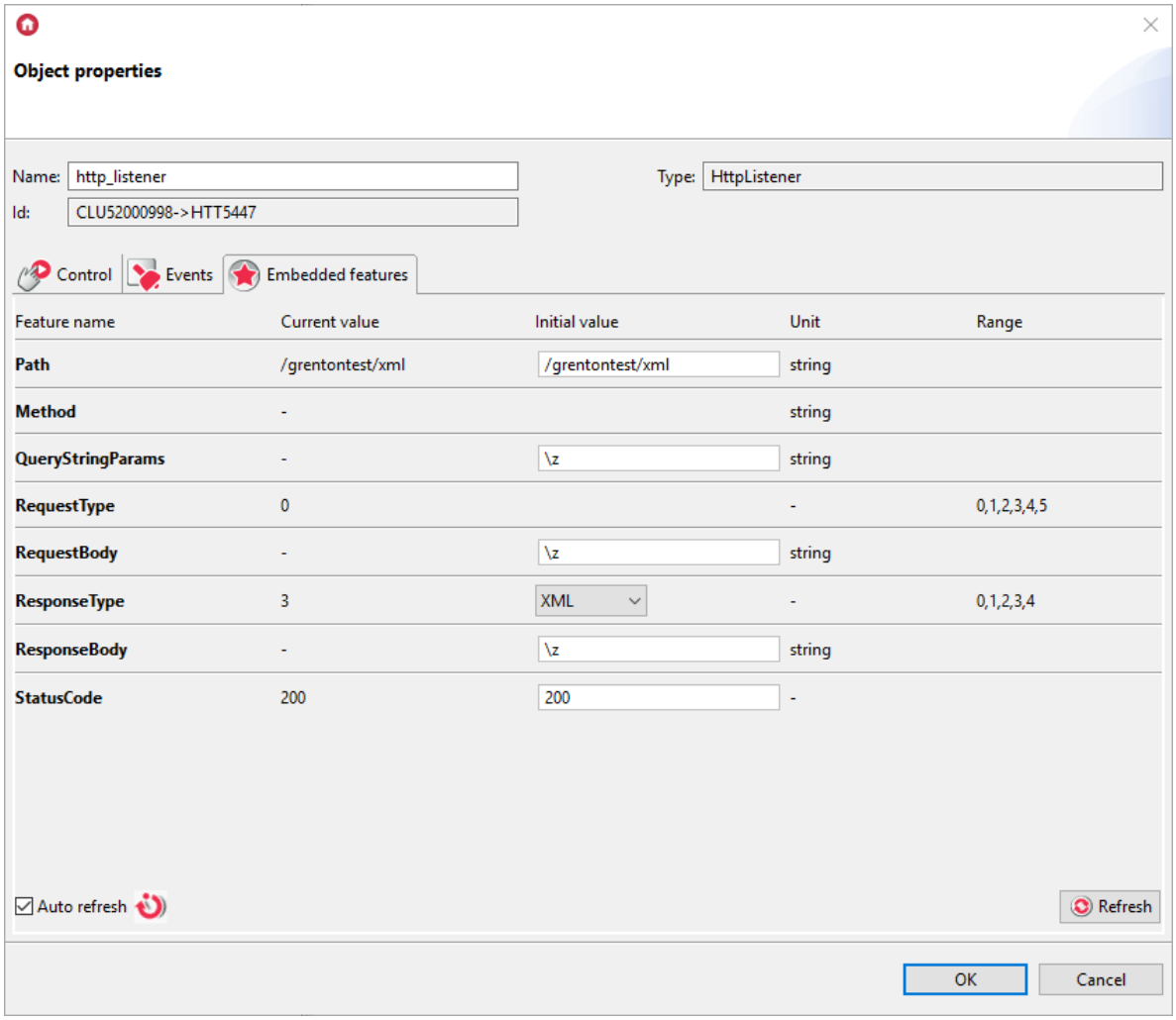
The HttpListener object is used for receiving HTTP (GET, POST) requests. The returned response can be serialized to one of the standard types including JSON, XML. In the HttpListener object, it is important to return the response to every incoming Request.

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).

2.2.4. Setting Response Headers / Reading Request Headers

To set response headers `ResponseHeaders`, you need to create a script and define a local script variable.

Examples of setting headers:

- An array of name/value pairs, with each index containing a pair consisting of the header name and value:

```
local header = {
 { name = 'Month', value = 'April' },
 { name = 'Year', value = '2023' },
    { name = 'Name', value = 'Home' },
}

GATE->HttpListener->ResponseHeaders = header
```

- A string (headers separated by '\r\n'):

```
local header =
 'Month: April\r\n' ..
 'Year: 2023\r\n' ..
 'Name: Home\r\n'

GATE->HttpListener->ResponseHeaders = header
```

`RequestHeaders` are returned as a Lua array. Below is a sample script for reading `RequestHeaders`:

```
local header = GATE->HttpRequest->ResponseHeaders

for i,v in ipairs(header) do
 print(v.name .. ': ' .. v.value)
end
```

The values are stored in local script variables. To use the values in the system, you need to extract them into global variables. An example of execution is described in section 2.1.3.

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:

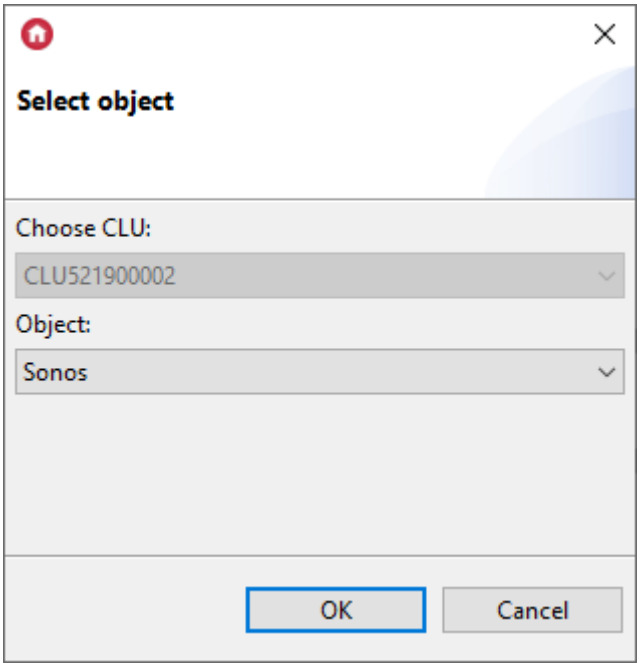
- `Countdown`
  After starting, it counts down the set time. At the end of the countdown, the method associated with the `OnTimer` event is started, and the timer stops and does not count down until the next start using the `Start` method.
- `Interval`
  Cyclic timer - after the start, it starts counting 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.

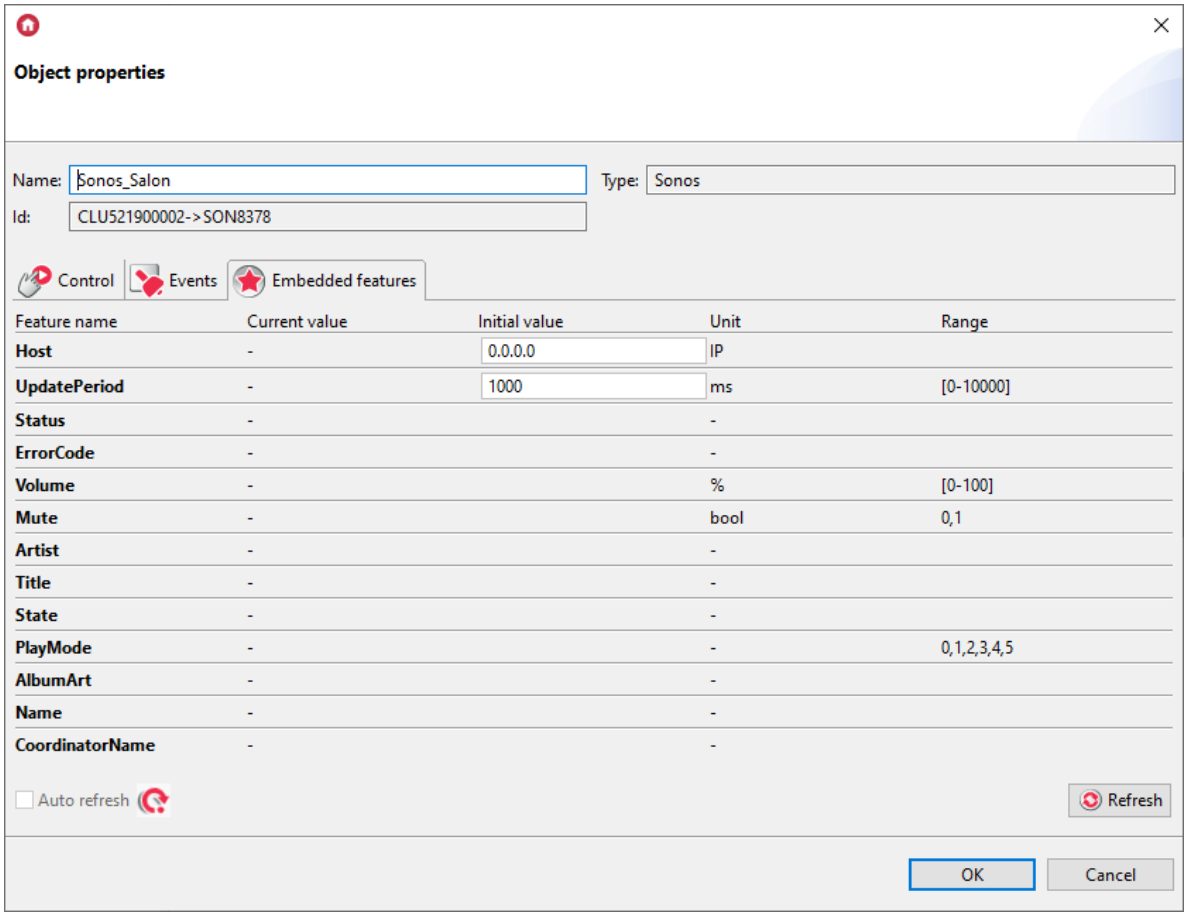
2.4.1. Sonos

The Sonos object is used to integrate Sonos speaker with the Grenton system using the GATE module.

Before integrating the speaker with the system, the speaker should be configured in the local network using dedicated application and read its assigned IP address.

To connect the speaker to system, create a new Sonos virtual object:

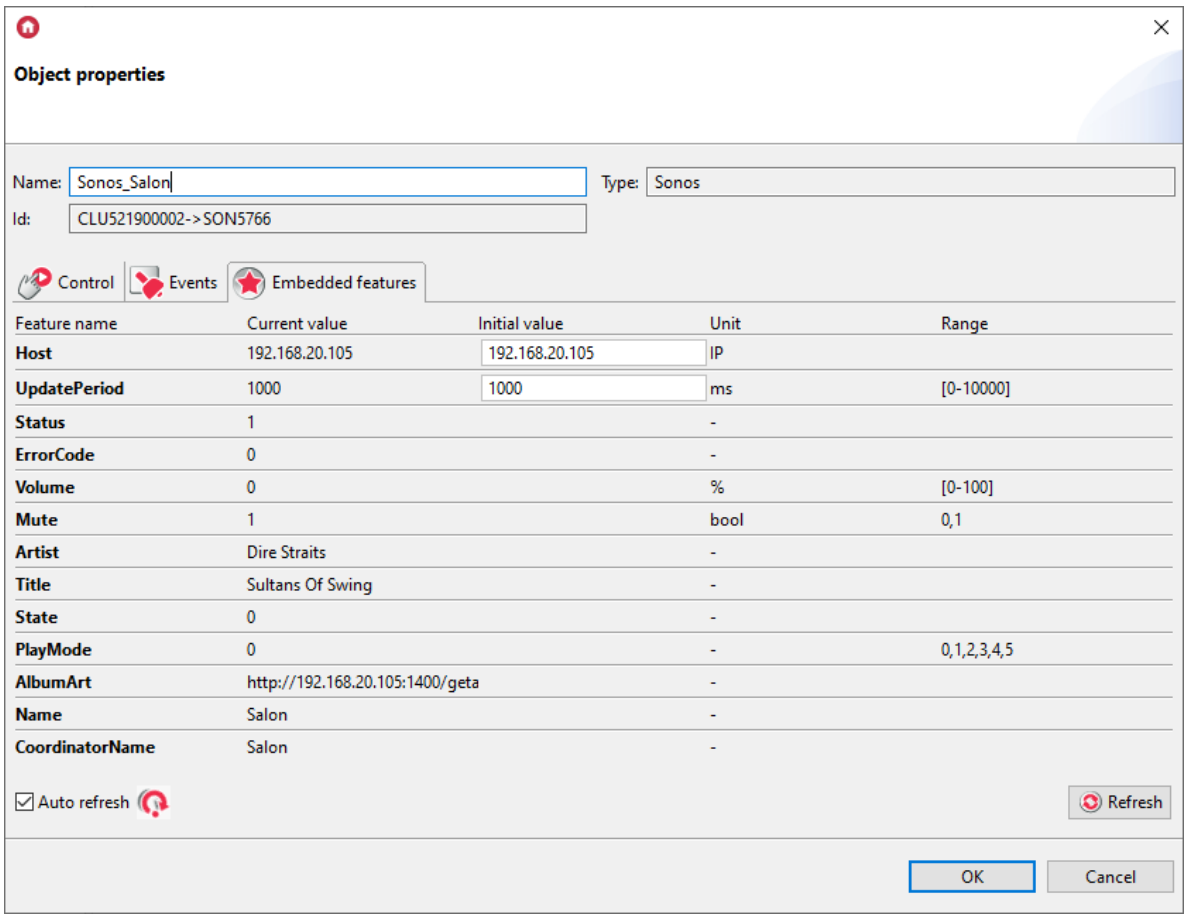




Set the parameters in the Embedded features tab:

- **Host:** e.g. - 192.168.20.105 (speaker IP address)
- **UpdatePeriod:** 1000

After sending the configuration, the `Status` feature should take the value `1`. This is information about the correct connection of the speaker to the system.



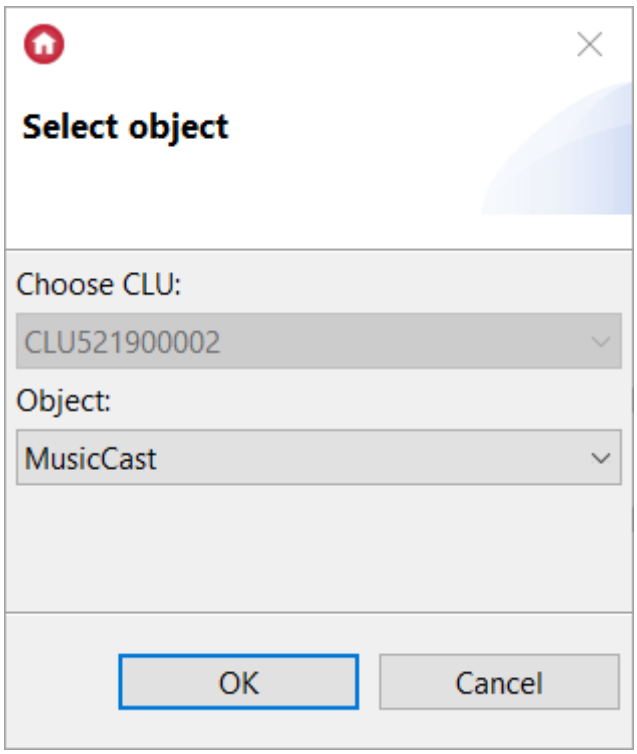
Note!
If more objects are used, there may be issues due to limited network and/or device bandwidth. In such a situation, as the number of created objects continues to increase, it is recommended to increase the value of the UpdatePeriod feature.

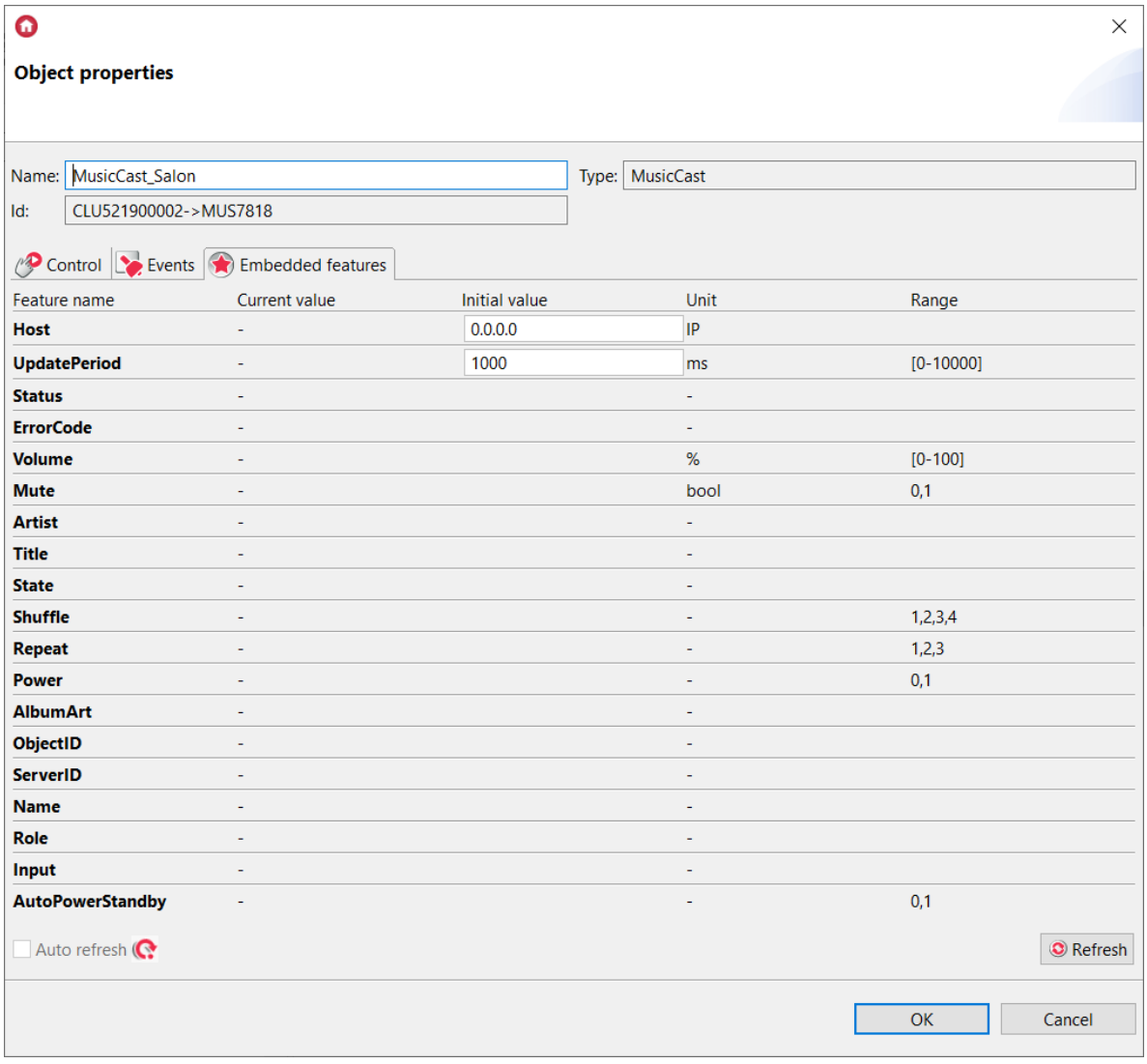
2.5.1. MusicCast

The MusicCast object is used to integrate Yamaha speaker with the Grenton system using the GATE module.

Before integrating the speaker with the system, the speaker should be configured in the local network using dedicated application and read its assigned IP address.

To connect the speaker to system, create a new MusicCast virtual object:

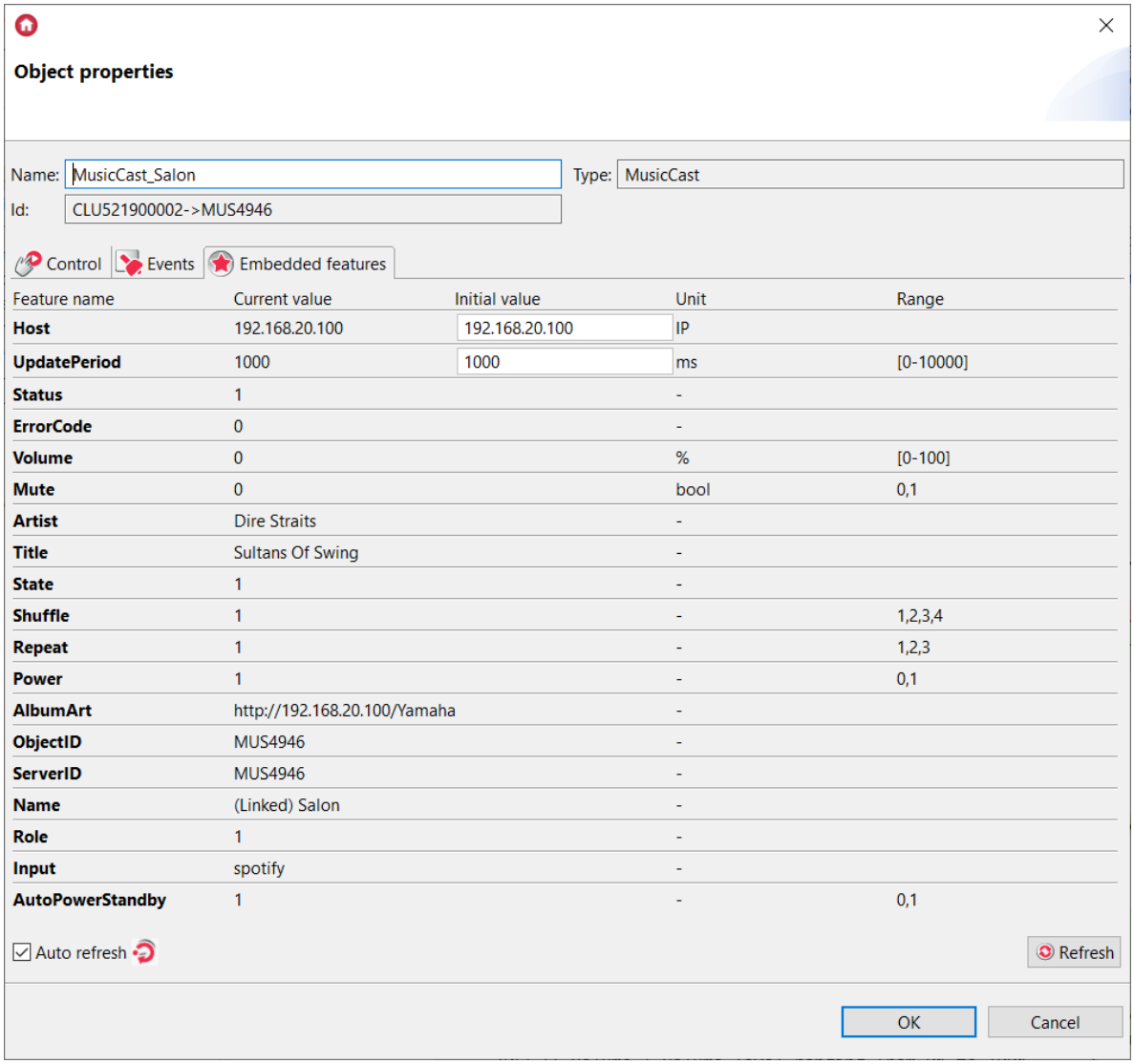




Set the parameters in the Embedded features tab:

- **Host:** e.g. - 192.168.20.105 (speaker IP address)
- **UpdatePeriod:** 1000

After sending the configuration, the `Status` feature should take the value `1`. This is information about the correct connection of the speaker to the system.

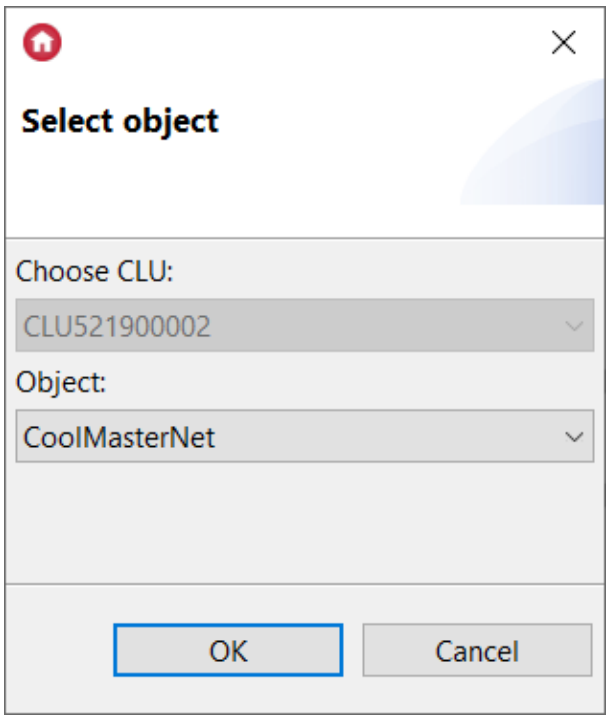


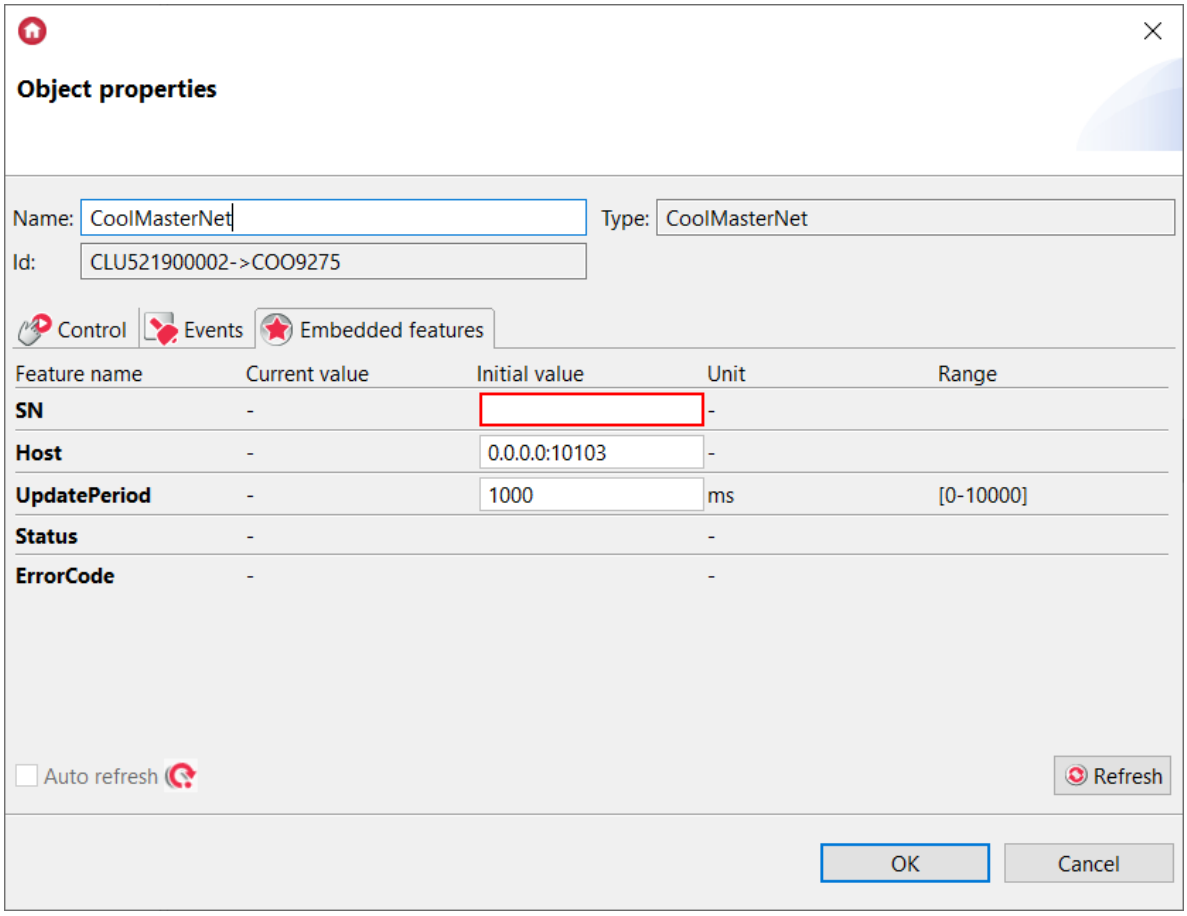
Note!
If more objects are used, there may be issues due to limited network and/or device bandwidth. In such a situation, as the number of created objects continues to increase, it is recommended to increase the value of the UpdatePeriod feature.

2.6.1. CoolMasterNet

The CoolMasterNet virtual object is used to integrate the Grenton system with CoolAutomation units (CoolMasterNet, CoolLinkHub) in order to control an air conditioner or a group of air conditioners.

To connect the CoolAutomation unit to the system, create a new CoolMasterNet virtual object:

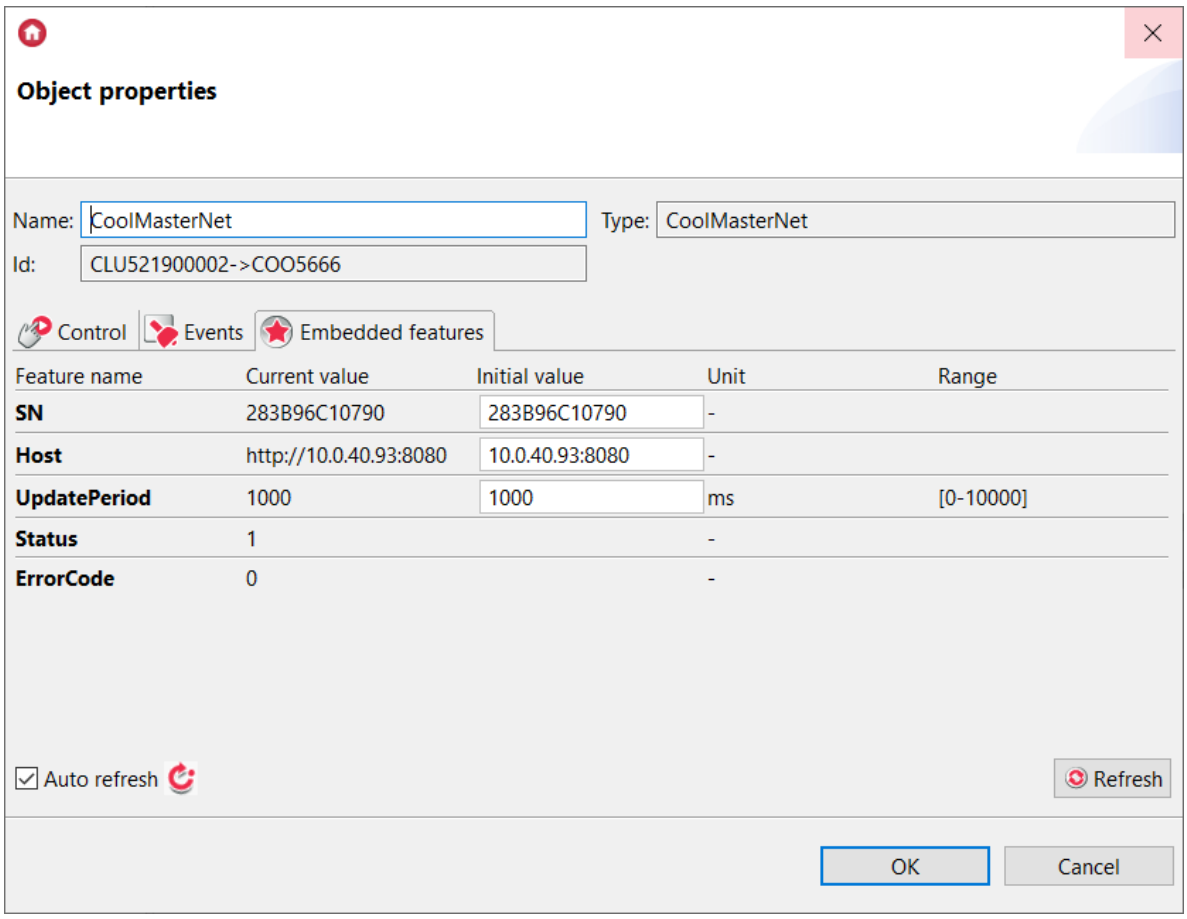




Set the parameters in the Embedded features tab:

- **SN:** e.g. - 283B96C10790 (unit serial number)
- **Host:** e.g. - 192.168.0.213:10103 (unit IP address)

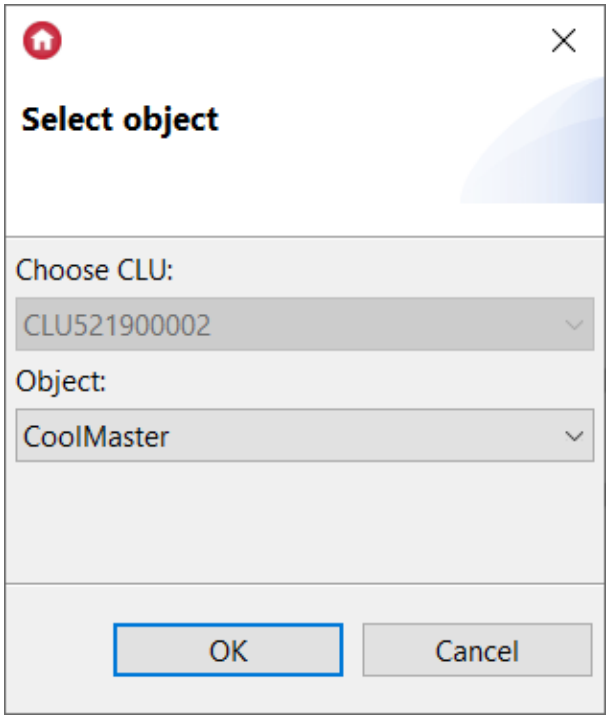
After sending the configuration, the `Status` feature should take the value `1`. This is information about the correct connection of the unit to the system.

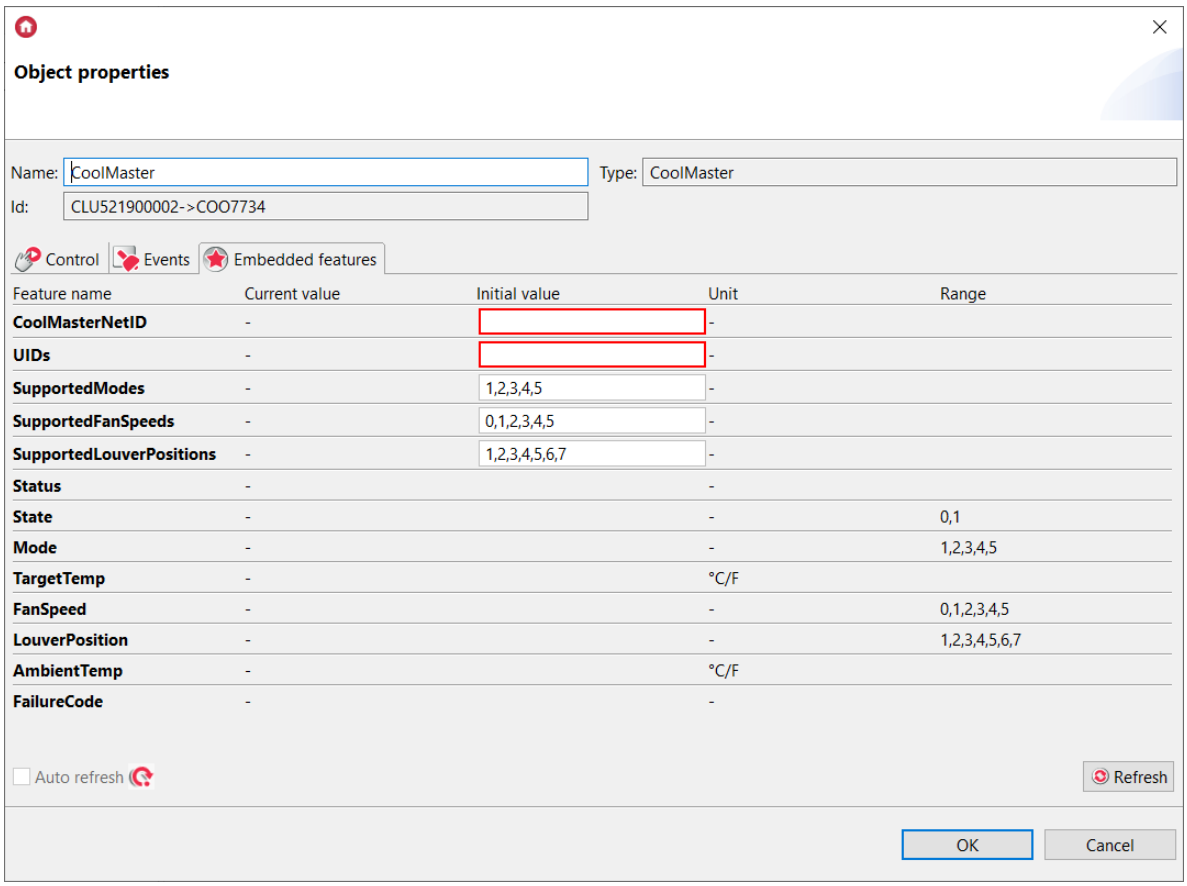


2.7.1. CoolMaster

The CoolMaster virtual object is used to control an air conditioner or a group of air conditioners. To use the object properly, you must first configure the CoolMasterNet virtual object (see point 2.6.1.).

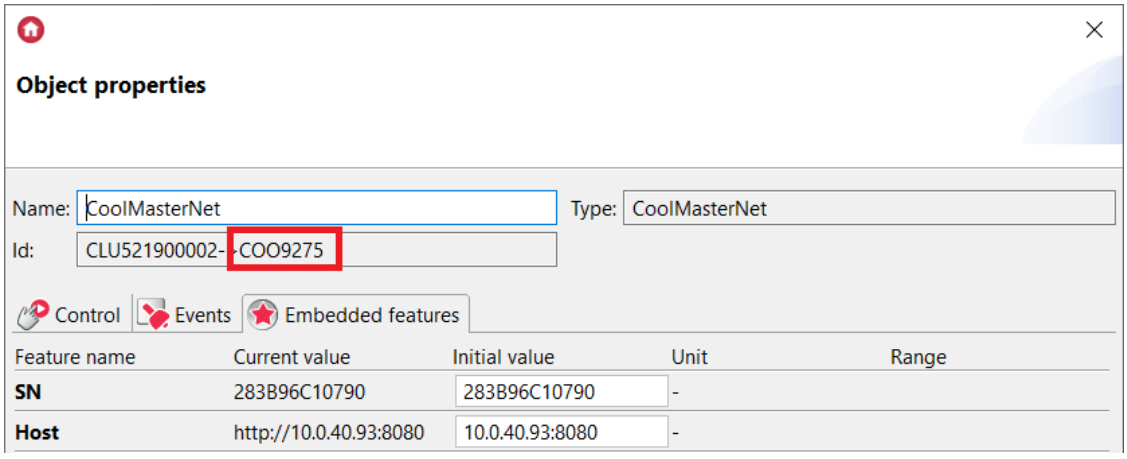
Then create a new CoolMaster virtual object:



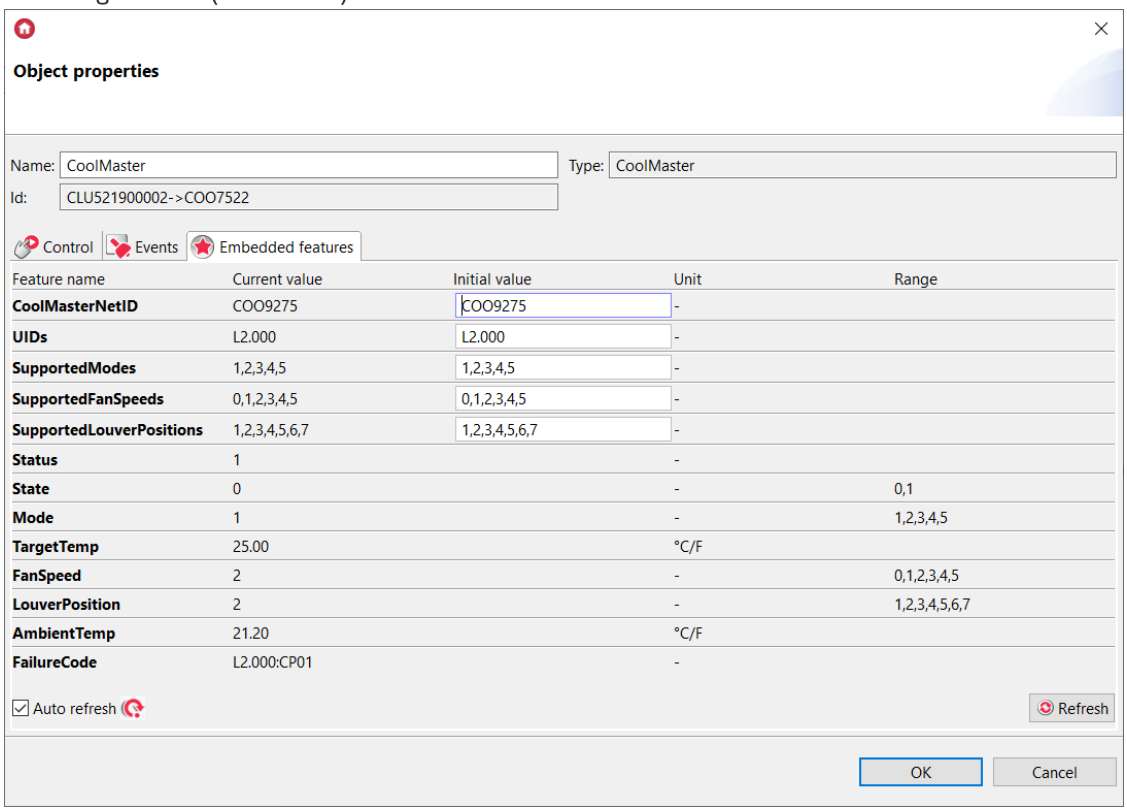


Set the parameters in the Embedded features tab:

- **CoolMasterNetID:** e.g. COO9275 (ID of the previously created CoolMasterNet object):



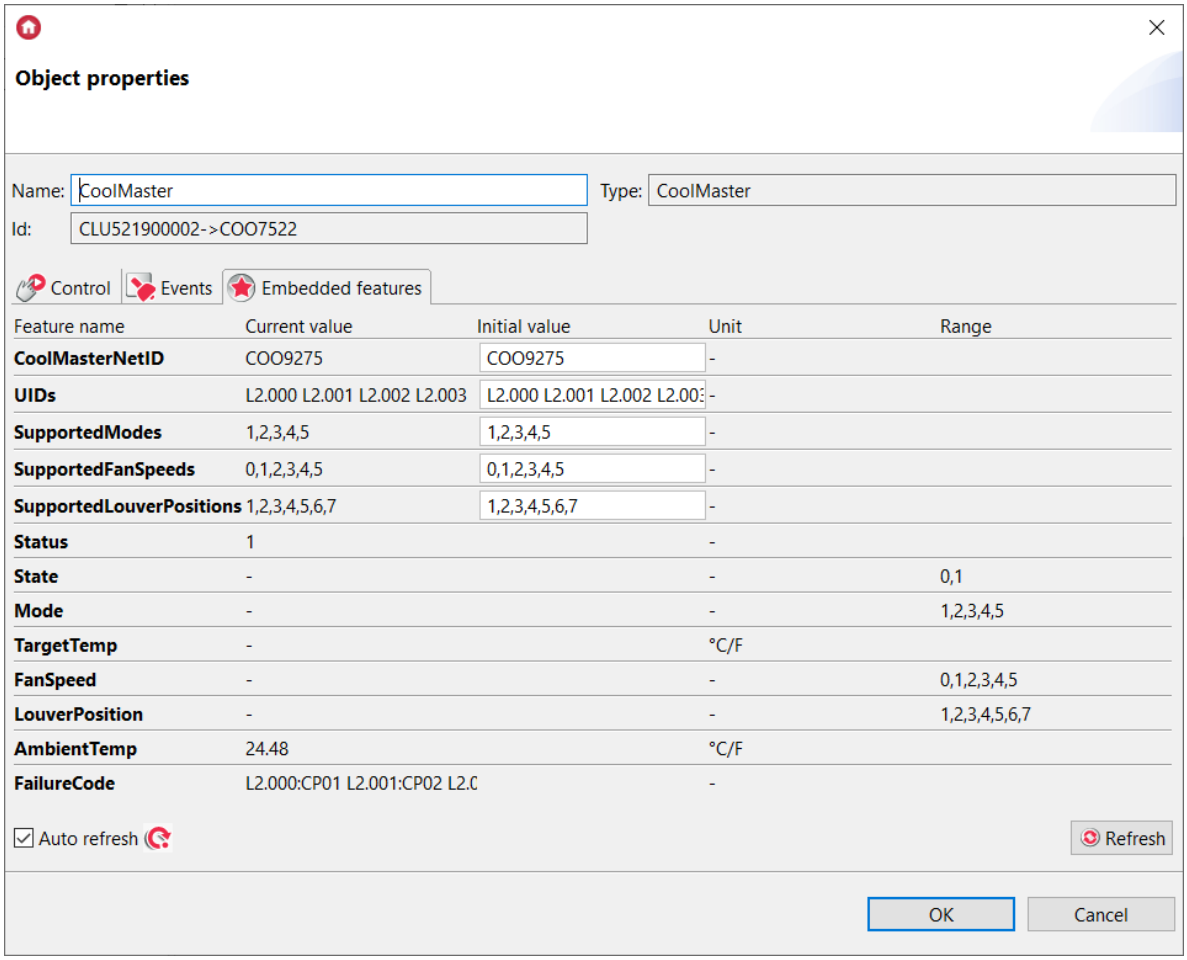
- **UIDs:** e.g. L2.000 (HVAC UIDs)



After sending the configuration, the `Status` feature should take the value `1` and the values of the other features should be loaded correctly.

Controlling multiple air conditioners with a single CoolMaster object

One CoolMaster object can work with multiple air conditioners connected to the same CoolAutomation unit. For this purpose, in the CoolMaster object in the `UIDs` feature, after the space, enter the following identifiers of the air conditioners:



Note!
The `-` value for the `State`, `Mode`, `TargetTemp`, `FanSpeed`, `LouverPosition` features means that the values of a given feature are different for at least one air conditioner from the group - desynchronization state.

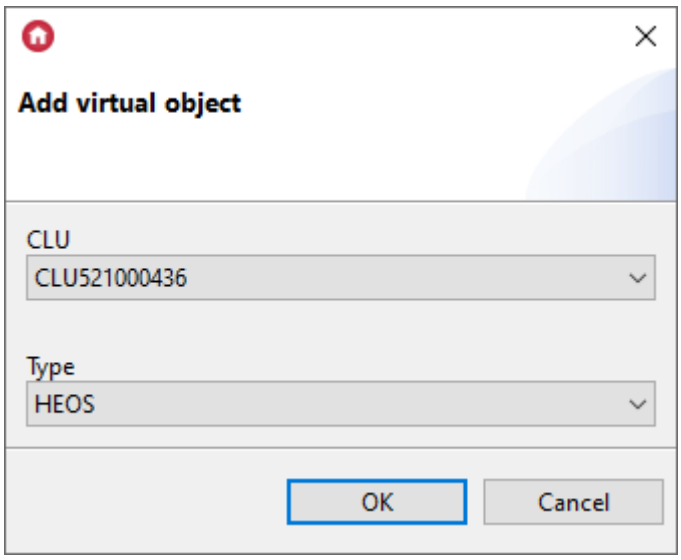
Note!
Features `State`, `Mode`, `TargetTemp`, `FanSpeed`, `LouverPosition`, `AmbientTemp`, `FailureCode` have no values before the first connection with the unit or at the moment of desynchronization state, in Object Manager they are displayed as `-`. In order to avoid errors in the scripts, before comparing such a feature, check whether it has a value:
if(feature ~= nil)

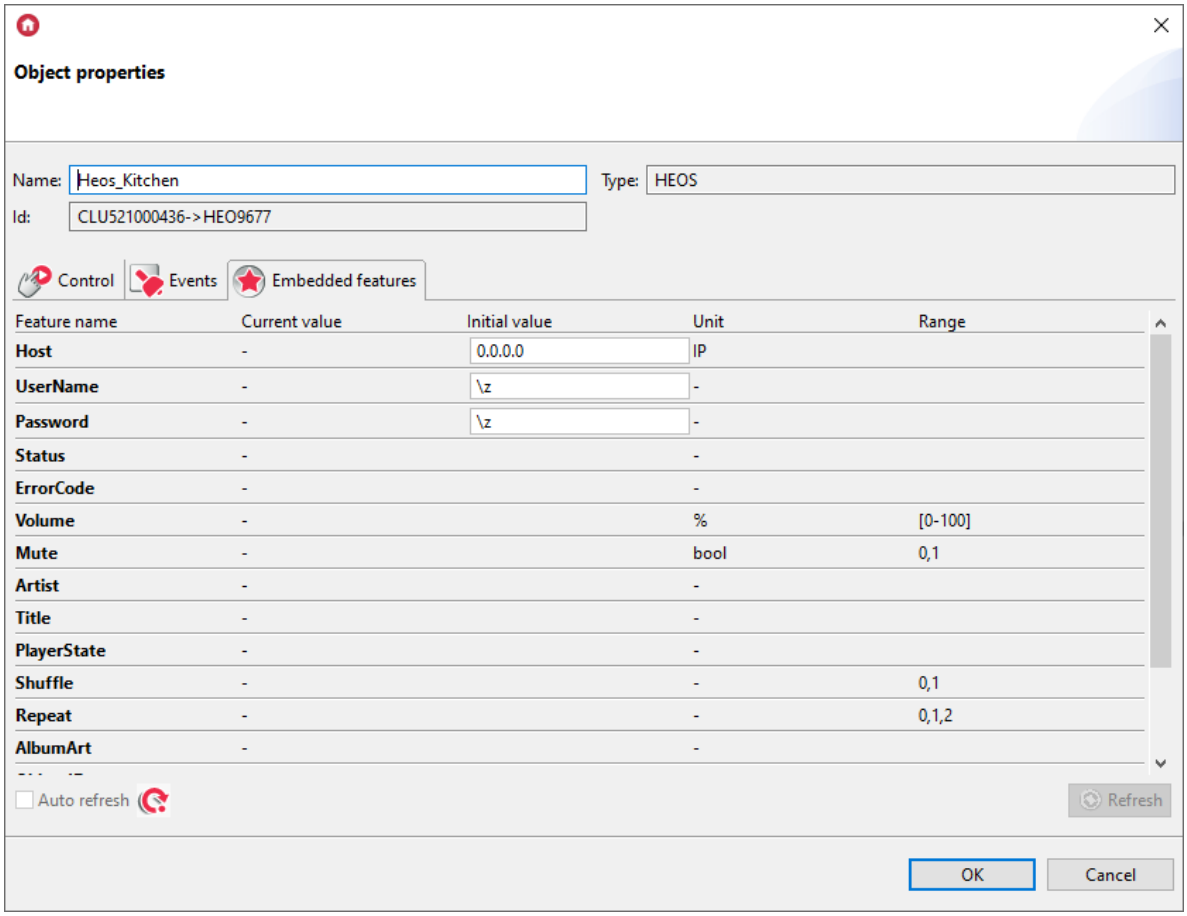
2.8.1. HEOS

The HEOS object is used to integrate HEOS speaker and Denon/Marantz AVR receivers with built-in HEOS support with the Grenton system using the GATE module.

Before integrating the speaker with the system, the speaker should be configured in the local network using dedicated application and read its assigned IP address.

To connect the speaker to system, create a new HEOS virtual object:





The following parameters should be set in the embedded features of the object:

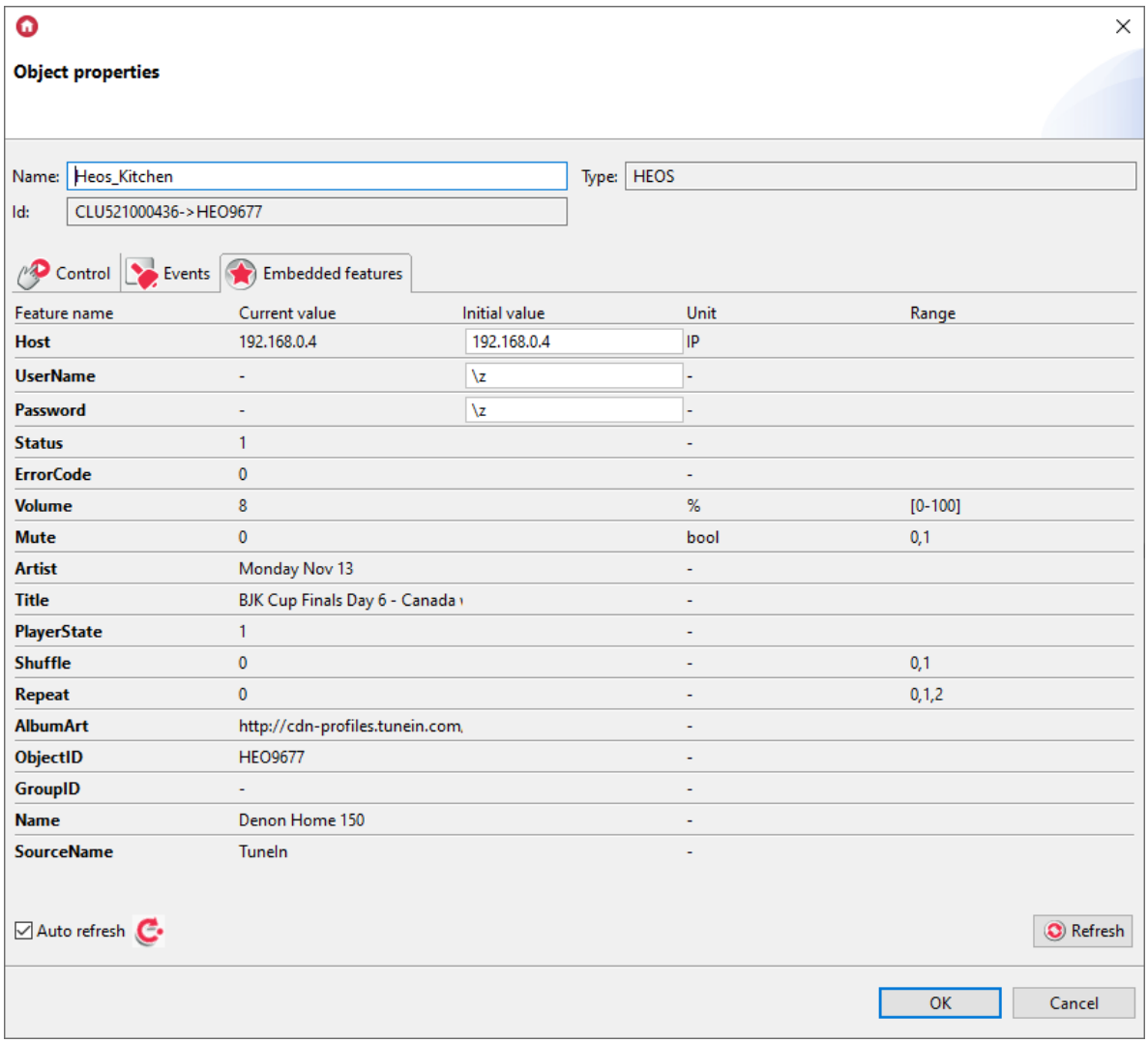
- **Host:** e.g. - 192.168.0.4 (speaker/device IP address)

Optionally can be set:

- **UserName:** Heos account username
- **Password**: Heos account user password

> **Note!**
>
> For the `PlayPresetStation` method to work properly, setting `UserName` and `Password` is required.

After sending the configuration, the `Status` feature should take the value `1`. This is information about the correct connection of the speaker to the system.

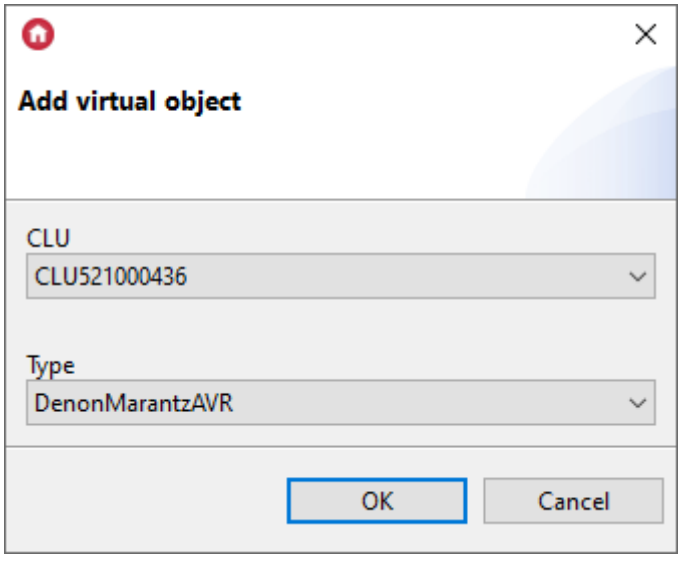


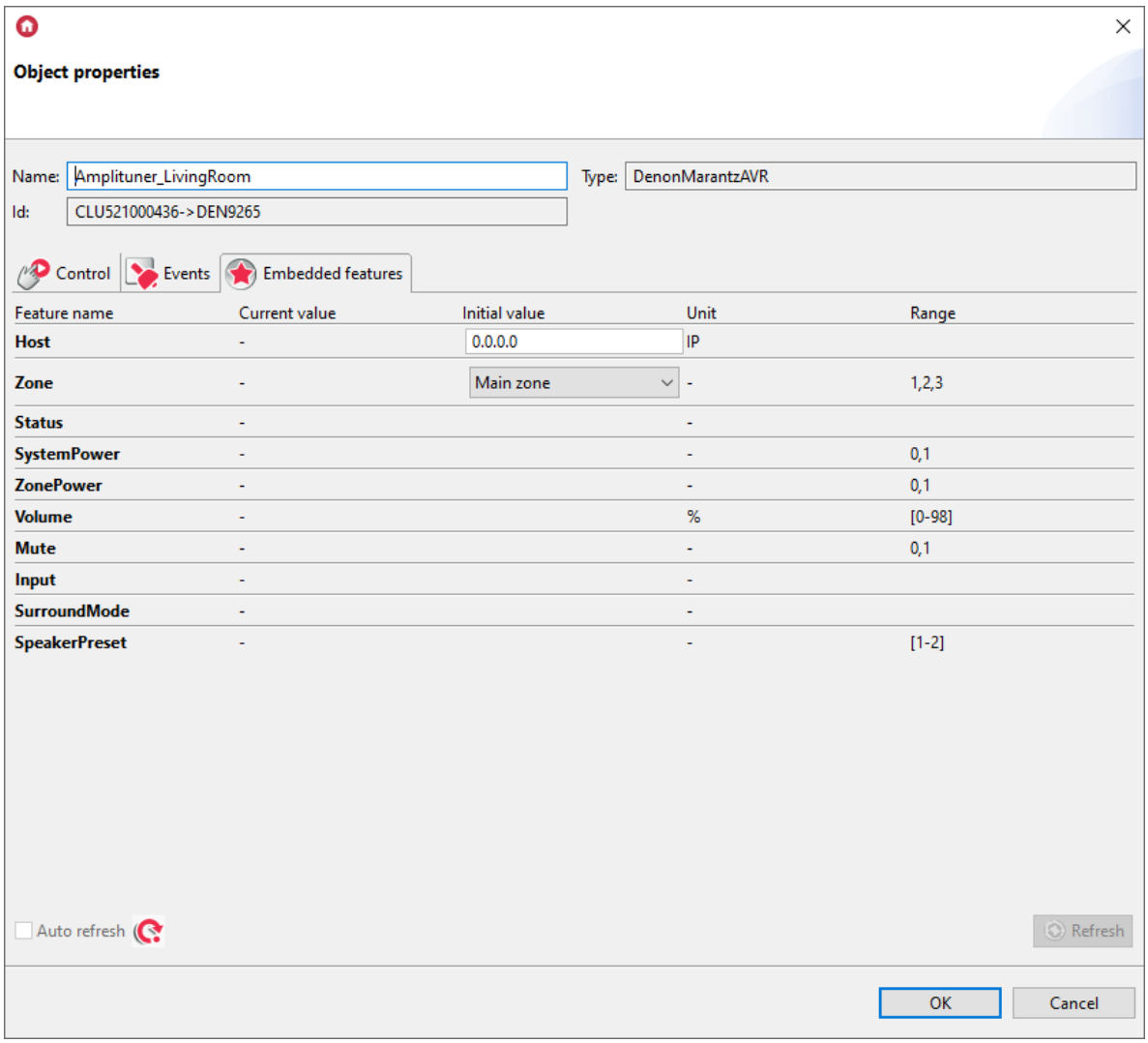
2.9.1. DenonMarantzAVR

The DenonMarantzAVR object is used to integrate devices Denon/Marantz AVR with the Grenton system using the GATE module.

Before integrating the speaker with the system, the speaker should be configured in the local network using dedicated application and read its assigned IP address.

To connect the speaker to system, create a new DenonMarantzAVR virtual object:

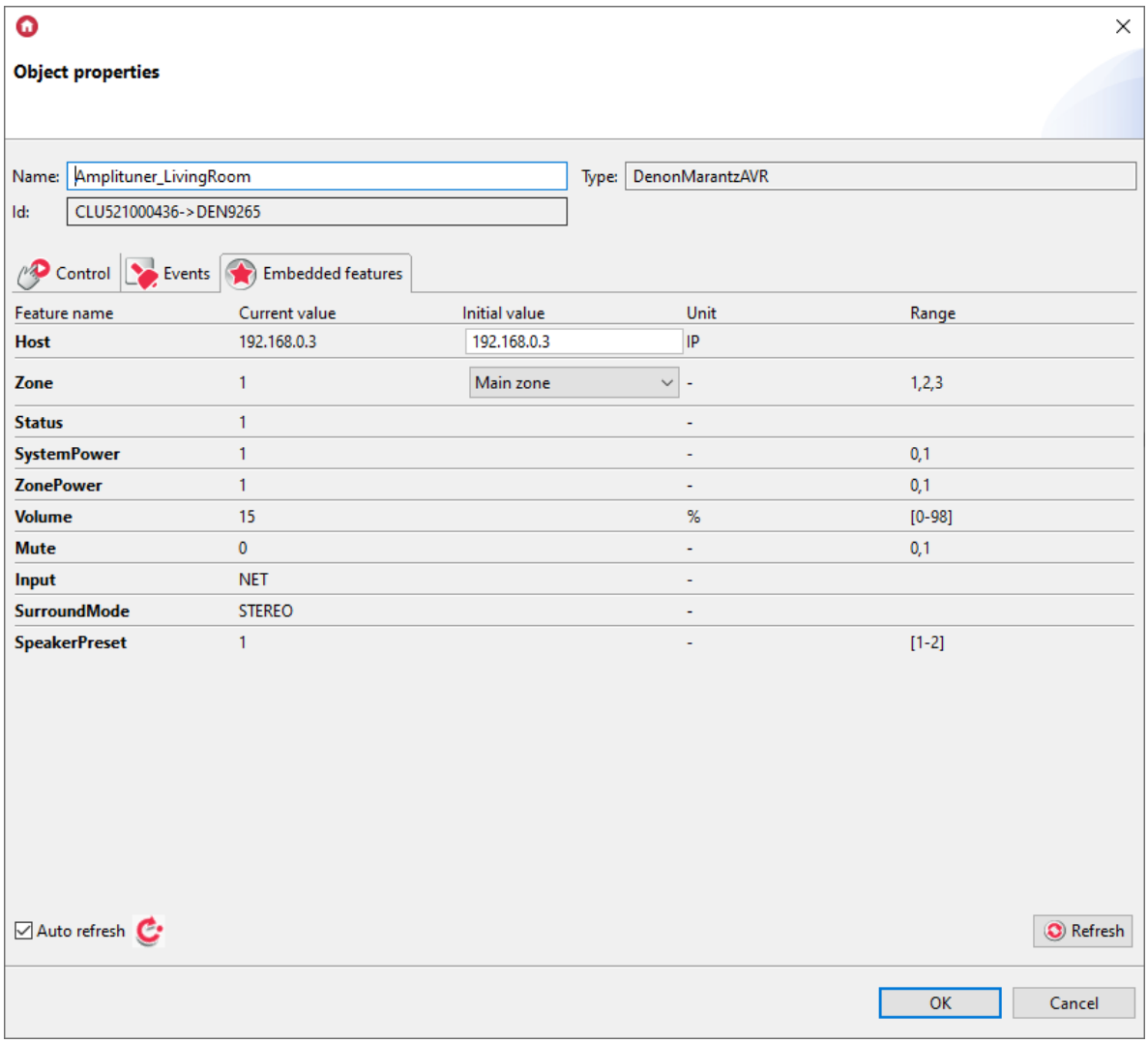




Set the parameters in the Embedded features tab:

- **Host:** e.g. - 192.168.0.4 (speaker IP address)
- **Zone:** Main zone

After sending the configuration, the `Status` feature should take the value `1`. This is information about the correct connection of the speaker to the system.



Note!

For Denon/Marantz devices that do not have zone division, set the attribute `Zone` to Main Zone.

Note!

The `SetSystemPower` method, in the case of an amplifier with multiple zones, turns off and on all available zones at once. To control the power of a specific zone, use the `SetZonePower` method.

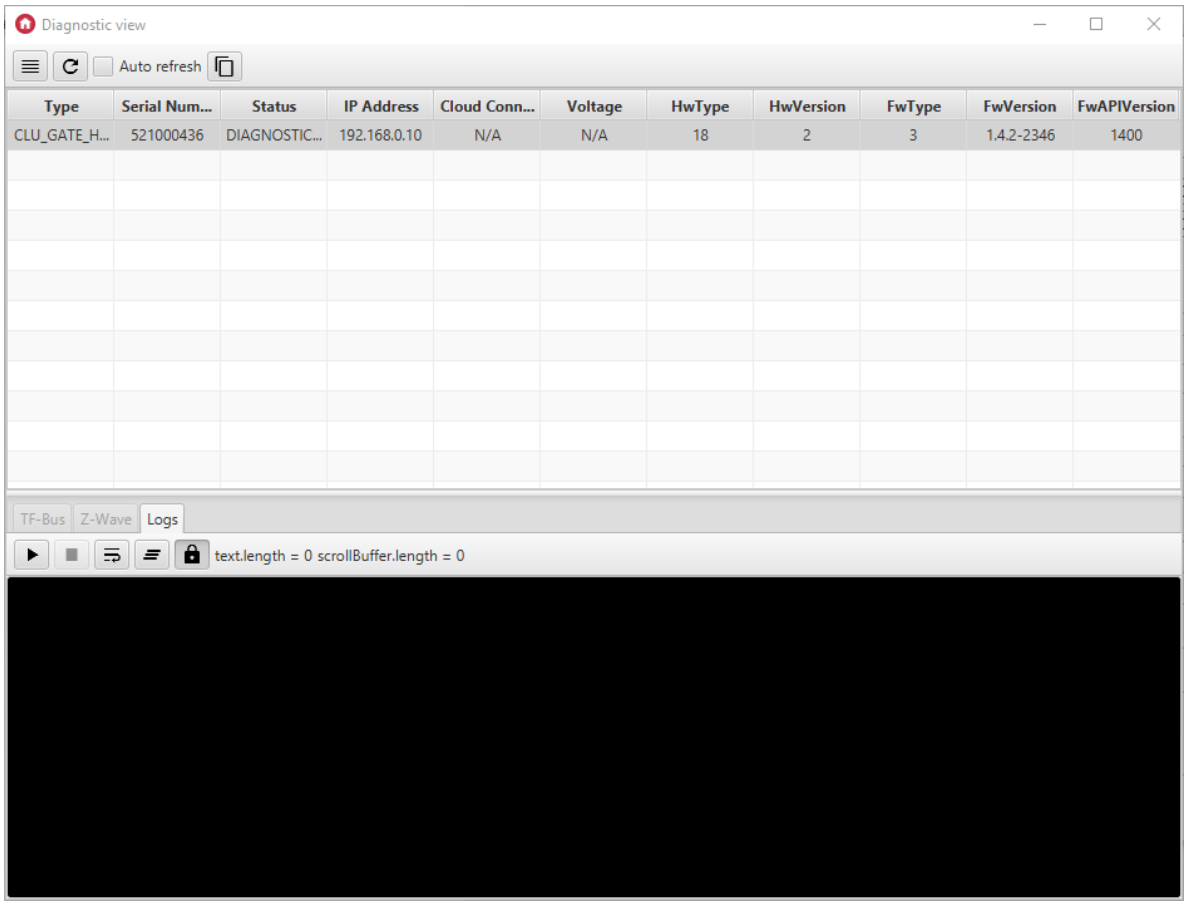
Note!

Depending on the type of device and its available features, specific methods are accessible. Before using a method, it is essential to verify its functionality support in the manual or documentation of the specific device.

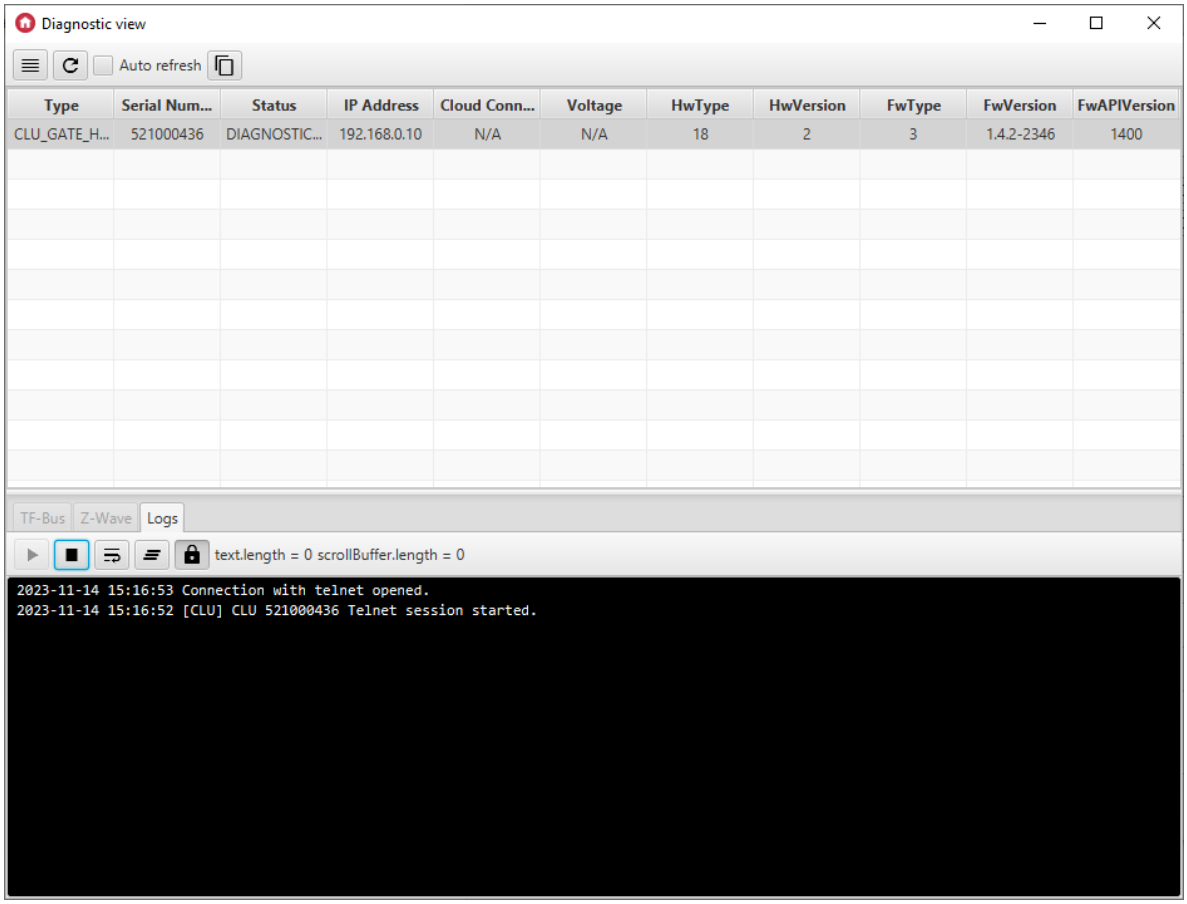
3. The ability to connect to the Gate using TELNET

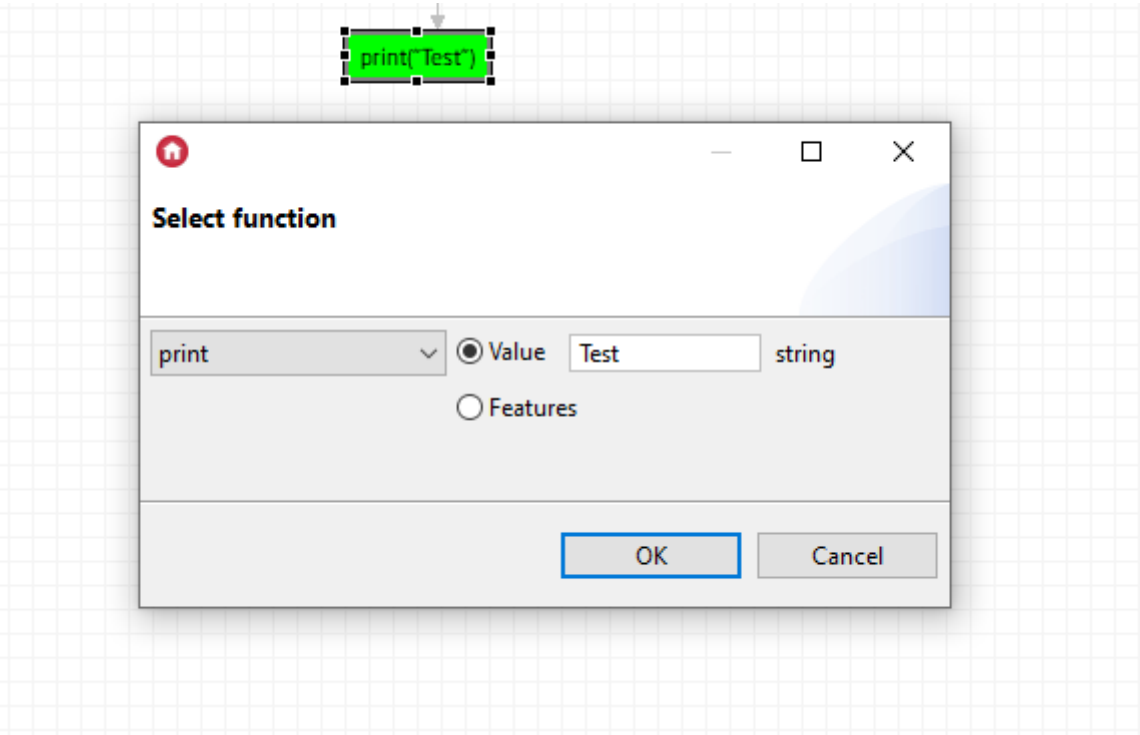
For the Http Gate module, it is possible to preview Lua scripts. In case of a configuration error (emergency mode), it is possible to view the error location in the created LUA configuration. To do this, set the logging level in the GATE object using the attribute `TelnetLogLevel` = DEBUG and send the configuration to the module.

Next, open the diagnostic view and start logging as shown in the figure below. More information about logging can be found in Chapter 11. Diagnostic View.



To display the value of a specific feature on the console, use the *Function Block* component and select the *Print* method, then choose the desired feature."





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

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(objects) 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

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
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

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
 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 `OnValueChange` 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 `OnValueChange` 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 1.4.2-2346 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
`TelnetLogLevel`	Specifies the logging level
`OverloadDetection`	Specifies whether Gate should report CPU overload using the red LED
`ResetReason`	Specifies the device reset reason: `0` - power-on `2` - configuration reload `3` - system exception

METHODS

Name	Description
`SetDateTime`	Sets the date and time
`SetClientReportInterval`	Sets the reporting period for feature changes
`SetPrimaryDNS`	Sets the PrimaryDNS feature
`SetSecondaryDNS`	Sets the SecondaryDNS feature
`SetTelnetLogLevel`	Specifies the logging level

EVENTS

Name	Description
`OnInit`	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's parameters. \z means lack of parameters
`Method`	The type of method sent in the query, e.g. GET, POST
`Timeout`	Acceptable response timeout
`RequestType`	The type of content of the query being sent. Defines the *content-type* parameter in the query header. Depending on the type selected, the contents of the `RequestBody` feature are appropriately serialized: `0 - None` - undefined. The content-type is not sent in the header. The content of the `RequestBody` feature is not serialized. `1 - Text` - *content-type: text / plain*. The content of the `RequestBody` feature is not serialized. `2 - JSON` - *content-type: application / json*. The contents of the `RequestBody` feature are serialized to JSON format. `3- XML` - *content-type: text / xml*. The contents of the `RequestBody` feature are serialized to XML format. `4 - FormData` - *content-type: application / x-www-form-urlencoded*. The contents of the `RequestBody` feature are serialized to the table. `5 - Other` - the content type (*content-type*) is different from the built-in one. The type can be defined by placing it in the header (the `RequestHeaders` attribute). The content is not serialized.
`ResponseType`	The type of expected answer. Defines the *Accept* parameter in the query header. Depending on the type chosen, the content of the received response (`ResponseBody` features) is properly parsed into the table: `0 - None` - *Accept* is not sent in the header of the query being sent. The answer (feature `ResponseBody`) is not parsed. `1 - Text` - *Accept: text / plain*. The answer (feature `ResponseBody`) is not parsed. `2 - JSON` - *Accept: application / json*. The answer (feature `ResponseBody`) is parsed with JSON. `3 - XML` - *Accept: text / xml*. The response (feature `ResponseBody`) is parsed from XML. `4 - FormData` - *Accept: application / x-www-form-urlencoded*. The answer (ResponseBode feature) is parsed. `5 - Other` - the *Accept* parameter of the header is different from the built-in one. The parameter can be defined by placing it in the header (the `RequestHeaders` attribute).
`RequestHeaders`	Additional HTTP query headers. `\ z` means no content.
`RequestBody`	The content of the message sent in the query. `\ z` means no content
`ResponseHeaders`	HTTP response headers

Name	Description
`ResponseBody`	The content of the message received after sending the query. (feature used for reading in scripts - not settable)
`StatusCode`	HTTP response status
`IsActive`	HTTP request active state

METHODS

Name	Description
`SendRequest`	Sends the request
`AbortRequest`	Breaks request's service
`Clear`	Deletes request's content
`SetHost`	Sets the host's address
`SetPath`	Sets request's path
`SetQueryStringParams`	Sets query's parameters
`SetMethod`	Sets request's method
`SetTimeout`	Sets acceptable response timeout
`SetRequestType`	Sets the content type of the request being sent (content-type)
`SetResponseType`	Sets the expected request's answer type
`SetRequestHeaders`	Sets additional HTTP request's header
`SetRequestBody`	Sets the request's message content

EVENTS

Name	Description
`OnRequestSent`	Event occurring when the request is sent
`OnResponse`	Event occurring when the answer 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`	The type of inquiry received. Depending on the type chosen, the content of the query received (the `RequestBody` attribute) is properly parsed into the table: `0 - None` - The answer is not parsed. `1 - Text` - The answer is not parsed. `2 - JSON` - The answer is parsed with JSON. `3 - XML` - The answer is parsed from XML. `4 - FormData` - The answer is parsed. `5 - Other` - The answer is not parsed. The `RequestBody` feature returns the contents of an HTTP query (a feature used to read in scripts - not settable).
`RequestBody`	Returns the content of the HTTP request (feature used for reading in scripts - non-persistent)
`RequestHeaders`	HTTP request headers (characteristic using to read the value in scripts - unchangeable)
`ResponseType`	The content type of the sent response to the query. Defines the *content-type* parameter in the response header. Depending on the type selected, the contents of the `ResponseBody` feature are appropriately serialized: `0 - None` - undefined. *Content-type* is not sent in the header. The content is not serialized. `1 - Text` - *content-type: text / plain*. The content is not serialized. `2 - JSON` - *content-type: application / json*. The `RequestBody` content is serialized to JSON format. `3 - XML` - *content-type: text / xml*. The `RequestBody` content is serialized to XML format. `4 - FormData` - *content-type: application / x-www-form-urlencoded*. The `RequestBody` content is serialized. `5 - Other` - the *Accept* parameter of the header is different from the built-in one. The parameter can be defined by placing it in the header (the `RequestHeaders` attribute).
`ResponseHeaders`	Additional HTTP response headers
`ResponseBody`	Returns the contents of the HTTP response (a feature used to read in scripts).

Name	Description
`StatusCode`	Status of the HTTP response being sent. Supported statuses: `200` - OK `201` - Created `202` - Accepted `204` - No content `205` - Reset content `400` - Bad request `403` - Forbidden `404` - Not found `405` - Method not allowed `406` - Not acceptable `408` - Request timeout `409` - Conflict `410` - Gone

METHODS

Name	Description
`SendResponse`	Sends the request's response
`Clear`	Deletes response's content
`SetPath`	Sets request's path
`SetResponseType`	Sets the expected request's answer type
`SetResponseHeaders`	Sets additional HTTP request's header
`SetResponseBody`	Sets the response's content
`SetStatusCode`	Sets Response's state

EVENTS

Name	Description
`OnRequest`	Event occurring when the request is received

D. Timer

FEATURES

Name	Description
`Time`	Counted time (in ms)
`Mode`	Timer mode: `0` - count down (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

E. Sonos

FEATURES

Name	Description
`Host`	Sonos IP Address
`UpdatePeriod`	State update period
`Status`	Communication status: `0` - no connection, `1` - connected
`ErrorCode`	Last error code: `0` - no error, negative values - negative HTTP status codes, positive values - UPnP error codes
`Volume`	Volume level ranging from 0% to 100%
`Mute`	Mute state: `0` - Off, `1` - On
`Artist`	Artist
`Title`	Title
`State`	Playback state: `0` - stopped, `1` - playing, `2` - paused, `3` - transitioning
`PlayMode`	Playback mode: `0` - normal, `1` - repeat all, `2` - repeat one, `3` - shuffle, no repeat, `4` - shuffle, repeat all, `5` - shuffle, repeat one
`AlbumArt`	Album art address
`Name`	Speaker name
`CoordinatorName`	Group coordinator name

METHODS

Name	Description
`SetUpdatePeriod`	Sets the state update period
`SetVolume`	Sets the volume level, ranging from 0% to 100%
`SetMute`	Sets the mute state
`SetPlayMode`	Sets the playback mode
`Play`	Starts playback
`Pause`	Pauses playback
`Stop`	Stops playback
`Next`	Switches to the next track
`Prev`	Switches to the previous track
`VolumeUp`	Increases the volume by a specified percentage
`VolumeDown`	Decreases the volume by a specified percentage
`SwitchMute`	Switches the mute state
`SwitchPlay`	Switches the playback state between paused and playing
`LeaveGroup`	Leaves the current speaker group, if any
`JoinGroup`	Joins the speaker group, specified by the group coordinator name

EVENTS

Name	Description
`OnConnected`	Event executed after successful speaker connection establishment
`OnDisconnected`	Event executed after the speaker connection has been lost
`OnError`	Event executed after an error has been detected
`OnChange`	Event executed after a Mute, Volume, Title, Artist, State, PlayMode, AlbumArt or CoordinatorName change occurs
`OnMuteChange`	Event executed after a Mute change occurs
`OnVolumeChange`	Event executed after a Volume change occurs
`OnTitleChange`	Event executed after a Title change occurs
`OnArtistChange`	Event executed after an Artist change occurs
`OnStateChange`	Event executed after a State change occurs
`OnPlayModeChange`	Event executed after a PlayMode change occurs
`OnAlbumArtChange`	Event executed after an AlbumArt change occurs
`OnCoordinatorNameChange`	Event executed after a CoordinatorName change occurs

F. MusicCast Object

FEATURES

Name	Description
`Host`	MusicCast IP Address
`UpdatePeriod`	State update period
`Status`	Communication status: `0` - no connection, `1` - connected
`ErrorCode`	Last error code: `0` - no error, negative values - negative HTTP status codes, positive values - Yamaha Extended Control error codes
`Volume`	Volume level ranging from 0% to 100%
`Mute`	Mute state: `0` - Off, `1` - On
`Artist`	Artist
`Title`	Title
`State`	Playback state: `1` - playing, `2` - stopped, `3` - paused
`Shuffle`	Shuffle mode: `1` - off, `2` - on, `3` - songs, `4` - albums
`Repeat`	Repeat mode: `1` - off, `2` - one, `3` - all
`Power`	Power state: `0` - standby, `1` - on
`AlbumArt`	Album art address
`ObjectID`	Object ID
`ServerID`	Server object ID
`Name`	Speaker name

Name	Description
`Role`	Grouped speaker role: `1` - not part of any group, `2` - client, `3` - server
`Input`	Input source
`AutoPowerStandby`	AutoPowerStandby state: `0` - off, `1` - on

METHODS

Name	Description
`SetUpdatePeriod`	Sets the state update period
`SetVolume`	Sets the volume level, ranging from 0% to 100%
`SetMute`	Sets the mute state
`SetShuffle`	Sets the shuffle mode
`SetRepeat`	Sets the repeat mode
`SetPower`	Sets the power state
`SetAutoPowerStandby`	Sets the AutoPowerStandby state
`Play`	Starts playback
`Pause`	Pauses playback
`Stop`	Stops playback
`Next`	Switches to the next track
`Prev`	Switches to the previous track
`VolumeUp`	Increases the volume by a specified percentage
`VolumeDown`	Decreases the volume by a specified percentage
`SwitchMute`	Switches the mute state
`SwitchPlay`	Switches the playback state between paused and playing
`DestroyGroup`	Destroys the current speaker group
`JoinGroup`	Joins the speaker group specified by the ServerID
`LeaveGroup`	Leaves the current speaker group
`SetInput`	Sets the input source

EVENTS

Name	Description
`OnConnected`	Event executed after successful speaker connection establishment
`OnDisconnected`	Event executed after the speaker connection has been lost
`OnError`	Event executed after an error has been detected
`OnChange`	Event executed after a Volume, Mute, Artist, Title, State, Shuffle, Repeat, Power, AlbumArt, Input, AutoPowerStandby, ServerID, Role change occurs
`OnMuteChange`	Event executed after a Mute change occurs
`OnVolumeChange`	Event executed after a Volume change occurs
`OnTitleChange`	Event executed after a Title change occurs
`OnArtistChange`	Event executed after an Artist change occurs
`OnStateChange`	Event executed after a State change occurs
`OnShuffleChange`	Event executed after a Shuffle change occurs
`OnRepeatChange`	Event executed after a Repeat change occurs
`OnPowerChange`	Event executed after a Power change occurs
`OnAlbumArtChange`	Event executed after an AlbumArt change occurs
`OnInputChange`	Event executed after an Input change occurs
`OnAutoPowerStandbyChange`	Event executed after an AutoPowerStandby change occurs
`OnGroupChange`	Event executed after a group related state change occurs (ServerID, Role)

G. CoolMasterNet object

FEATURES

Name	Description
`SN`	CoolMasterNet unit serial number
`Host`	CoolMasterNet unit address in http://host:port form
`UpdatePeriod`	State update period
`Status`	Connection status: `0` - no connection, `1` - connected
`ErrorCode`	CoolMasterNet last error code: `0` - no error, `1` - TCP connection error or HTTP error code

METHODS

Name	Description
`SetUpdatePeriod`	Sets the state update period
`TurnAllOn`	Turns on all the HVAC units
`TurnAllOff`	Turns off all the HVAC units

EVENTS

Name	Description
`OnConnected`	Event occurring after successful CoolMasterNet connection establishment
`OnDisconnected`	Event occurring after the CoolMasterNet connection has been lost
`OnError`	Event occurring after an error has been detected

H. CoolMaster object

FEATURES

Name	Description
`CoolMasterNetID`	CoolMasterNet object ID
`UIDs`	One or more space-separated HVAC UIDs
`SupportedModes`	List of supported operation modes separated by a comma
`SupportedFanSpeeds`	List of supported fan speeds separated by a comma, entering "-" means no support
`SupportedLouverPositions`	List of supported louver positions separated by a comma, entering "-" means no support
`Status`	Connection status: `0` - no connection, `1` - connected
`State`	Operation state: `1` - active, `0` - inactive, "-" - state desynchronized
`Mode`	Operation mode: `1` - cool, `2` - heat, `3` - fan, `4` - dry, `5` - auto, `-` - state desynchronized
`TargetTemp`	Target temperature `-` - state desynchronized
`FanSpeed`	Fan speed: 0-5, `5` - auto, `-` - state desynchronized
`LouverPosition`	Louver position: `0` - no support, `1` - auto, `2` - horizontal, `3` - 30°, `4` - 45°, `5` - 60°, `6` - vertical, `7` - stopped, `-` - state desynchronized
`AmbientTemp`	Ambient temperature or the average ambient temperature in case of a device group
`FailureCode`	Failure code

METHODS

Name	Description
`SetSupportedModes`	Sets the list of supported operation modes
`SetSupportedFanSpeeds`	Sets the list of supported fan speeds
`SetSupportedLouverPositions`	Sets the list of supported louver positions
`SetState`	Sets the working status
`SetMode`	Sets the operation mode
`SetTargetTemp`	Sets the temperature setpoint
`SetFanSpeed`	Sets the fan speed
`SetLouverPosition`	Sets the louver position
`TurnOn`	Turns on the HVAC unit or units in case of a device group
`TurnOff`	Turns off the HVAC unit or units in case of a device group
`SwitchMode`	Sets the next operation mode

EVENTS

Name	Description
`OnConnected`	Event occurring after successful CoolMasterNet connection establishment
`OnDisconnected`	Event occurring after the CoolMasterNet connection has been lost
`OnChange`	Event executed after a change occurs on State, Mode, TargetTemp, FanSpeed or LouverPosition
`OnModeChange`	Event executed after a Mode change occurs
`OnTargetTempChange`	Event executed after a TargetTemp change occurs
`OnFanSpeedChange`	Event executed after a FanSpeed change occurs
`OnLouverPositionChange`	Event executed after a LouverPosition change occurs
`OnTurnOn`	Event executed after the HVAC unit or group of units is turned on
`OnTurnOff`	Event executed after the HVAC unit or group of units is turned off
`OnFailure`	Event executed after detecting an error
`OnDesynchronization`	Event executed after desynchronisation of the grouped HVAC units properties occurs

G. HEOS object

FEATURES

Name	Description
`Host`	HEOS IP Address
`UserName`	User name
`Password`	Password
`Status`	Communication status: `0` - no connection `1` - connected
`ErrorCode`	Last HEOS CLI error code
`Volume`	Volume level ranging from 0% to 100%
`Mute`	Mute state: `0` - off, `1` - on
`Artist`	Artist
`Title`	Title
`PlayerState`	Player state: `0` - stopped, `1` - paused, `2` - playing
`Shuffle`	Shuffle mode: `0` - off, `1` - on
`Repeat`	Repeat mode: `0` - off, `1` - one, `2` - all
`AlbumArt`	Album art address
`ObjectID`	Object ID
`GroupID`	Group leader object ID
`Name`	Speaker name
`SourceName`	Input source

METHODS

Name	Description
`SetVolume`	Sets the volume level, ranging from 0% to 100%
`SetMute`	Sets the mute state
`SetShuffle`	Sets the shuffle mode
`SetRepeat`	Sets the repeat mode
`Play`	Starts playback
`Pause`	Pauses playback
`Stop`	Stops playback
`Next`	Switches to the next track
`Prev`	Switches to the previous track
`VolumeUp`	Increases the volume by a specified percentage
`VolumeDown`	Decreases the volume by a specified percentage
`SwitchMute`	Switches the mute state
`SwitchPlay`	Switches the playback state between paused and playing
`AddToGroup`	Adds the speaker specified by ObjectID to the current speaker's group
`DestroyGroup`	Destroys the current speaker group
`PlayPresetStation`	Plays the station/track specified on the favorites list in the HEOS application.
`PlayInputSource`	Sets the physical input source with the given name, in accordance with the HEOS documentation, e.g., "inputs/aux1"
`PlayUrl`	Play a stream specified by url
`PlayUSB`	Plays an audio file from a USB storage device using the full file path with the extension, e.g., "messages/leak.mp3"
`PlayUSBClip`	Plays an audio file from a USB storage device using the full file path with the extension, e.g., "messages/leak.mp3", when playing an audio file during playback of a queue/station, optionally restores the previous playback

EVENTS

Name	Description
`OnConnected`	Event executed after successful speaker connection establishment
`OnDisconnected`	Event executed after the speaker connection has been lost
`OnError`	Event executed after an error has been detected
`OnChange`	Event executed after a Mute, Volume, Title, Artist, PlayerState, Shuffle, Repeat, AlbumArt, SourceName or GroupID change occurs
`OnMuteChange`	Event executed after a Mute change occurs
`OnVolumeChange`	Event executed after a Volume change occurs
`OnTitleChange`	Event executed after a Title change occurs
`OnArtistChange`	Event executed after an Artist change occurs
`OnPlayerStateChange`	Event executed after a PlayerState change occurs
`OnShuffleChange`	Event executed after a Shuffle change occurs
`OnRepeatChange`	Event executed after a Repeat change occurs
`OnAlbumArtChange`	Event executed after an AlbumArt change occurs
`OnSourceChange`	Event executed after a SourceName change occurs
`OnGroupChange`	Event executed after a GroupID change occurs
`OnPlaybackError`	Event executed after a playback error occurs
`OnClipEnd`	Event executed after playback started using PlayUSBClip ends

H. DenonMarantzAVR object

FEATURES

Name	Description
`Host`	AV receiver IP address
`Zone`	AV receiver zone
`Status`	Communication status: `0` - no connection, `1` - connected
`SystemPower`	System power state: `0` - standby, `1` - on
`ZonePower`	Zone power state: `0` - off, `1` - on
`Volume`	Volume level ranging from 0% to 98%
`Mute`	Mute state: `0` - off, `1` - on
`Input`	Mute state: `0` - off, `1` - on
`SurroundMode`	Surround mode
`SpeakerPreset`	Speaker preset

METHODS

Name	Description
`SetSystemPower`	Sets the system power state
`SetZonePower`	Sets the zone power state
`SetVolume`	Sets the volume level, ranging from 0% to 98%
`SetMute`	Sets the mute state
`SetInput`	Sets the input source
`SetSurroundMode`	Sets the surround mode
`SetSpeakerPreset`	Sets the speaker preset
`VolumeUp`	Increases the volume by a specified percentage
`VolumeDown`	Decreases the volume by a specified percentage
`SwitchMute`	Switches the mute state
`QuickSelect`	Selects the Quick Select settings

FEATURES

Name	Description
`OnConnected`	Event executed after successful receiver connection establishment
`OnDisconnected`	Event executed after the receiver connection has been lost
`OnChange`	Event executed after a SystemPower, ZonePower, Volume, Mute, Input, SurroundMode or SpeakerPreset change occurs
`OnMuteChange`	Event executed after a Mute change occurs
`OnVolumeChange`	Event executed after a Volume change occurs
`OnSystemPowerChange`	Event executed after a SystemPower change occurs
`OnZonePowerChange`	Event executed after a ZonePower change occurs
`OnInputChange`	Event executed after a Input change occurs
`OnSurroundModeChange`	Event executed after a SurroundMode change occurs
`OnSpeakerPresetChange`	Event executed after a SpeakerPreset change occurs

XVI. DALI Controller Module

> **Note!**
> The described functionality and integration is available for **GRENTON DALI Controller DIN, Eth (INT-202-D-01)** with **2.0.7 (build 2450)** or higher.

> **Note!**
> DALI Controller is available for Object Manager in version **1.10.1 (build 242102)** and higher, and for CLU with firmware **5.13.08 (build 2442)** 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,
- `PowerSupplyVoltage` - 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` with chosen parameter:
  - GetConfiguration,
  - NewWithoutLocalAddress,
  - ResetAllLocalAddress
- Calling the method with the `GetConfiguration` parameter initiates automatic retrieval of local ballast addresses located on the bus from the range 0 to 63.

> **Note!**
> During this process all addresses are retrieved, if there are duplicates on the bus they will remain unchanged, in which case the conflicts should be resolved using the ResolveAddressDuplicate method!
>
> The controller memory is not deleted, the previous configuration will remain!

- Calling the method with the `NewWithoutLocalAddress` parameter initiates automatic addressing of new ballasts without an assigned address on the bus, which will receive local addresses from the range 0 to 63. The address assignment will be confirmed by lighting up the given luminaire for 300 ms. During DALI Discovery with the `NewWithoutLocalAddress` parameter, the addresses of ballasts that were already on the bus will remain unchanged.
- Calling the method with the `ResetAllLocalAddress` parameter initiates automatic addressing of all ballasts on the bus, which will receive local addresses from 0 to 63. The assignment of the address will be confirmed by lighting up the given luminaire for 300 ms. It should be noted that after starting the addressing, all previous addresses will be deleted. During DALI Discovery with the `ResetAllLocalAddress` parameter, addresses are assigned to 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 `ChangeLocalAddress` 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, after going to the `Control` tab, call the `ChangeLocalAddress` method with the parameters:
  - `ActualAddress` - current ballast address,
  - `AddressToSet` - new unoccupied address that will be assigned to the device,
- After calling the `ChangeLocalAddress` method, you should remove the old address using the `RemoveGear` method and then execute `DALI_Discovery` with the `GetConfiguration` parameter to retrieve the changed configuration.

- In the `DALI_MASTER` object, after going to the `Control` tab, call the `ResolveAddressDuplicate` method with the Address parameter:
  - Calling the method resolves address conflicts on the Dali bus, in the case when the address is duplicated for one of the ballasts, the first free address that is not in the module configuration is assigned
  - During `ResolveAddressDuplicate`:
    - The green diode on the DALI Controller module flashes at an interval of 200 ms,
    - The built-in feature `State` of the `DALI_MASTER` object takes the value 1.

> **Note!**
> Do not perform any operations on the DALI Controller during `ResolveAddressDuplicate`!

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`, `GroupSwitch`, `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. Replacing the Dali Controller module

In case of replacing the Dali Controller module it is possible to download the ballast configuration on the new module. To do this you need to update the new module to the latest version and then:

1. Execute `DALI_Discovery` with the `GetConfiguration` parameter to retrieve the ballast addresses,
2. After executing `DALI_Discovery` you need to call `Clu Discovery`, new Dali ballast objects will be added,
3. For new objects, you must replace the objects in the `newly found devices window`, or replace them later in the `inactive inputs/outputs rewriting window to active ones`.

> **Note!**
> In case the ballast addresses were changed using the old `SetLocalAddress` method, such information will not be retrieved by the new module and the addresses will be sorted, in this case it is possible to add new objects during `Clu Discovery`.

E. 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`	`0` - no ballast configuration `1` - DALI Discovery `2` - Short circuit/no power supply on the bus `3` - ballast configuration is on the device `4` - saving information about groups
`NumberOfGear`	Number of active ballasts connected to the device. The feature value is refreshed after restarting the system or calling the UpdateMissingGears method
`GearAddresses`	Ballast addresses given during DALI_Discovery. The feature value is refreshed after Dali Discovery
`ActiveGears`	Addresses of active ballasts in the configuration. The feature value is refreshed after system restart or calling the `UpdateMissingGears` method
`MissingGears`	Addresses of inactive ballasts in the configuration.. The feature value is refreshed after system restart or calling the `UpdateMissingGears` method

METHODS

Name	Description
`Identify`	Turns on the luminaire for 2 seconds
`ResetGear`	Restarts the ballast
`ResetLocalAddress`	Removes the LocalAddress of the selected ballast. `RemoveFromController` parameter: `True` - removes the ballast address from the controller memory `False` - does not remove the ballast address from the controller memory
`ChangeLocalAddress`	Changes the current LocalAddress of the ballast to the selected one, **address is not deleted from Dali Controller memory**
`RemoveGear`	Removes the ballast from the controller memory
`ResolveAddressDuplicate`	Resolves address conflicts on the Dali bus
`UpdateMissingGears`	Checks the activity of the ballasts that are in the configuration
`DALI_Discovery`	Searching for ballasts connected to the DALI bus and assigning them local addresses. When an address is assigned, the ballast is switched on for 300 ms. *Parameters:* `GetConfiguration`, `NewWithoutLocalAddress`, `ResetAllLocalAddress` A detailed description of the parameters can be found in the section Addressing ballasts During `DALI_Discovery`, do not perform any operations on the device!
`SetPowerOnLevel`	For the selected address, it saves in the ballast memory the DACPValue that is to be set after a restart/bus failure
`SetDAPCValue`	Sets the value of the power with which the luminaire shines. `RampTime` parameter set on a logarithmic scale 0.8 - 90 [s]
`SetLastActiveLevel`	If the ballast is off, turns on the ballast with the DACPValue value set before it was turned off
`SetGroupDAPCValue`	Sets the value of the power with which the luminaire shines for a given group. `RampTime` parameter set on a logarithmic scale 0.8 - 90 [s]
`GroupSwitch`	Changes the output state for a ballast group to the opposite `RampTime` parameter set on a logarithmic scale 0.8 - 90 [s]
`GroupSwitchOn`	Turns on the luminaire for a given group. `RampTime` parameter set on a logarithmic scale 0.8 - 90 [s]
`GroupSwitchOff`	Turns off the luminaire for a given group. `RampTime` parameter set on a logarithmic scale 0.8 - 90 [s]

EVENTS

Name	Description
`OnDALI_DiscoveryCompleted`	Event occurring after the ballasts have been found and given local addresses
`OnDALI_BusPowerFailure`	Event generated: when there is no power supply to the DALI bus for more than 1s, when there is a short circuit on the DALI bus

B. DALI_GEAR

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

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 `SetHSVValue` method
`HSVSaturation`	Color 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 `SetHSVSaturation` method
`HSVHue`	Color hue value as per the HSV model (0-360). The feature does not get the actual color of the luminaire! Set according to called `SetHSVHue` method
`ColourTemperature`	The feature does not retrieve the actual color state of the luminaire! Set based on the `SetColourTemperature` method being called.

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]
`SetColorTemperature`	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. PowerSupplyVoltage

FEATURES

Name	Description
`Value`	Current output value taking into account the scalar
`Value%`	Current percentage input value of the maximum value (`MaxValue` characteristic)
`Sensitivity`	Minimum change of input state when the `OnValueChange` , `OnValueLower` or `OnValueRise` event is generated
`MinValue`	Minimum value of the Value characteristic after exceeding which the `OnOutOfRange` event is generated
`MaxValue`	Maximum value of the Value characteristic after exceeding which the `OnOutOfRange` event is generated

METHODS

Name	Description
`SetSensitivity`	Sets input sensitivity value
`SetMinValue`	Sets `MinValue`
`SetMaxValue`	Sets `MaxValue`

EVENTS:

Name	Description
`OnValueChange`	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 (`MinValue` : `MaxValue`)
`OnInRange`	Event occurs when value returns to `MinValue` / `MaxValue` 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
`Value`	Returns the input state
`HoldDelay`	The time after which pressing and holding the button will trigger the `OnHold` event
`HoldInterval`	The cyclic interval (in ms), after which the next `OnHold` events are triggered while holding the button

METHODS

Name	Description
`SetHoldDelay`	Sets `HoldDelay` value
`SetHoldInterval`	Sets `HoldInterval` value

EVENTS

Name	Description
`OnValueChange`	The cyclic interval (in ms), after which the next `OnHold` events are triggered while holding the button
`OnSwitchOn`	An event triggered when the high state is set on input
`OnSwitchOff`	An event triggered when the low state is set on input
`OnShortPress`	The event is triggered after pressing the button for 500-2000ms
`OnLongPress`	The event is triggered after pressing the button for 2000-5000ms
`OnHold`	Event triggered when the input is in the high state, the first time after the `holdDelay` time has elapsed, and then cyclically every `HoldInterval` value
`OnClick`	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
`Value`	The value of the input
`MinValue`	The minimum value of the input
`MaxValue`	The maximum value of the input
`Discovered`	Information returned during CLU Discovery about connecting the sensor to the module

EVENTS

Name	Description
`OnValueChange`	An event triggered when the output value is changed
`OnRise`	Event triggered when the upper hysteresis threshold is exceeded (rising edge)
`OnLower`	Event triggered when the lower hysteresis threshold is exceeded (falling edge)
`OnOutOfRange`	Event triggered when the output value is outside the specified range (`MinValue`:`MaxValue`)
`OnInRange`	An event triggered when the value returns to the interval within the threshold values (`MinValue`:`MaxValue`)

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
`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`	Information on blocking Z-Wave communication with the module: `0` - communication with the module is not blocked, `1` - communication with the module is blocked (banned module). 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
`FailCount`	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.
`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`	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. **Note!** *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!*
`ClearFailCount`	Clears the number of unsuccessful communication attempts
`Set`	Sets the value of a given configuration register (parameter): `1 - Register` (register or parameter number), `2 - Value` (the value of the register or parameter), `3 - 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 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
`Value`	Returns the input status: `0` - no violation, `1` - violation

EVENTS

Name	Description
`OnValueChange`	An event triggered when the status changes to the opposite
`OnSwitchOn`	An event triggered when the high state is set on input
`OnSwitchOff`	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
`Value`	The current value of the sensor
`MinValue`	The value below which the `OnOutOfRange` event is generated
`MaxValue`	The value above which the `OnOutOfRange` event is generated

METHODS

Name	Description
`SetMinValue`	Sets the low threshold value of the `OnOutOfRange` event
`SetMaxValue`	Sets the upper threshold value of the `OnOutOfRange` event

EVENTS

Name	Description
`OnValueChange`	An event triggered when the sensor value is changed
`OnValueRise`	An event is triggered when the sensor value changes to a higher one than the previous one
`OnValueLower`	An event triggered when the sensor value is changed to a lower one than the previous one
`OnOutOfRange`	An event triggered when one of the threshold values `MinValue` / `MaxValue` is exceeded
`OnInRange`	An event triggered when the value returns to the interval within the threshold values (`MinValue` : `MaxValue`)

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
`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 battery level changes
`OnLowBattery`	An event triggered when a battery drop is detected below the warning level
`OnBatteryGood`	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
`Interval`	Time of self-awakening of the Z-Wave module from sleep mode (in seconds)
`LastWakeUp`	Time of the last awakening of the Z-Wave module from sleep mode

METHODS

Name	Description
`SetInterval`	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`	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). 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
`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) **Note!** *Parameter 2, 3, 5 and 8 refer to the association of modules that is not supported by the Grenton system!* **Note!** *Parameter 3 - changing the parameter value does not cause sending it during motion detection!* **Note!** *Parameter 4 - correct setting of the parameter value, however the module itself does not change the operating mode !* **Note!** *Parameter 7 and 9 - correct setting of the parameter value, however the set value has not been tested due to the faulty sensor!* **Note!** *Parameter 1, 6 - no noticeable changes in module work after the change of value!* **Note!** *Parameter 9 - smaller range of set values (up to 100 lux)!* **Note!** *There is no information on the register number 11 (Motion Event Report One Time Enable) in the documentation!*

METHODS

Name	Description
`RemoveBan`	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.. **Note!** `RemoveBan` *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!*
`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) **Note!** *Calling the* `Set` *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!*
`Get`	Gets the value of a given configuration (parameter) register **Note!** *Calling the* `Get` *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!*
`SetDefault`	Sets the default value for a given configuration (parameter) register **Note!**! *Calling the* `SetDefault` *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!*

EVENTS

Name	Description
`OnBanned`	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
`Value`	Returns the input status: `0` - closing, `1` - opening

EVENTS

Name	Description
`OnValueChange`	An event triggered when the status changes to the opposite
`OnSwitchOn`	An event triggered when the high state is set on input
`OnSwitchOff`	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
`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 battery level changes
`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

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
`Interval`	The period of automatic 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 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`	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). 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
`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) **Note!** *Parameters 1 and 2 refer to the association of modules, which is not supported by the Grenton system!*

METHODS

Name	Description
`RemoveBan`	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. **Note!** `RemoveBan` *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!*
`ClearFailCount`	Clears the number of unsuccessful communication attempts
`Set`	Sets the value of a given configuration register (parameter): `Register` (register or parameter number), `Value` (register or parameter value), `Size` (size of the sent register or parameter value - in bytes) **Note!** *Calling the* `Set` *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!*
`Get`	Gets the value of a given configuration (parameter) register **Note!** *Calling the* `Get` *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!***
`SetDefault`	Sets the default value for a given configuration (parameter) register **NOTE**! *Calling the* `SetDefault` *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!*

EVENTS

Name	Description
`OnBanned`	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
`Value`	Returns the input status: `0` - no violation, `1` - violation

EVENTS

Name	Description
`OnValueChange`	An event triggered when the status changes to the opposite
`OnSwitchOn`	An event triggered when the high state is set on input
`OnSwitchOff`	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
`Value`	The current value of the sensor
`MinValue`	The value below which the `OnOutOfRange` event is generated
`MaxValue`	The value above which the `OnOutOfRange` event is generated

METHODS

Name	Descriptrion
`SetMinValue`	Sets the low threshold value of the `OnOutOfRange` event
`SetMaxValue`	Sets the upper threshold value of the `OnOutOfRange` event

EVENTS

Name	Description
`OnValueChange`	An event triggered when the sensor value is changed
`OnValueRise`	An event triggered when the sensor value changes to a higher one than the previous one
`OnValueLower`	An event triggered when the sensor value is changed to a lower one than the previous one
`OnOutOfRange`	An event triggered when one of the threshold values `MinValue` / `MaxValue` is exceeded
`OnInRange`	An event triggered when the value returns to the interval within the threshold values (`MinValue`:`MaxValue`)

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
`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 battery level changes
`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

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
`Interval`	The period of automatic 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

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`	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). 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
`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) **Note!** *Parameter 2, 3, 5 and 8 refer to the association of modules that is not supported by the Grenton!* **Note!** *Parameter 1, 6 and 7 - no noticeable changes in the module's work after the change of value!* **Note!**Parameter 9 - smaller range of set values (up to 100 lux)!

METHODS

Name	Description
`RemoveBan`	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. **Note!** `RemoveBan` *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!*
`ClearFailCount`	Clears the number of unsuccessful communication attempts
`Set`	Sets the value of a given configuration (parameter) register: `Register` (register or parameter number), `Value` (register or parameter value), `Size` (size of the registry value sent or parameter - in bytes) **Note!** *Calling the* `Set` *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!*
`Get`	Gets the value of a given register (parameter) configuration **Note!** *Calling the* `Get` *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!*
`SetDefault`	Sets the default value for a given register (parameter) configuration **Note!** *Calling the SetDefault 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!*

EVENTS

Name	Description
`OnBanned`	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
`Value`	Returns the input state: `0` - closing, `1` - opening

EVENTS

Name	Description
`OnValueChange`	An event triggered when the status changes to the opposite
`OnSwitchOn`	An event triggered when the high state is set on input
`OnSwitchOff`	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
`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 battery level changes
`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

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
`Interval`	The period of automatic 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 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`	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). 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
`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) **Note!** *Parameters 1 and 2 refer to the association of modules, which is not supported by the Grenton system!*

METHODS

Name	Description
`RemoveBan`	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. **Note!** *`RemoveBan` 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!*
`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) **Note!** *Calling the `Set` 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!*
`Get`	Gets the value of a given register (parameter) configuration **Note!** *Calling the `Get` 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!*
`SetDefault`	Sets the default value for a given register (parameter) configuration **Note!** *Calling the SetDefault 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!*

EVENTS

Name	Description
`OnBanned`	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
`Value`	Returns the input status: `0` - dry, `1` - flooded

EVENTS

Name	Description
`OnValueChange`	An event triggered when the status changes to the opposite
`OnSwitchOn`	An event triggered when the high state is set on input
`OnSwitchOff`	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
`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 battery level changes
`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

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
`Interval`	Z-Wave module awakening period from sleep mode
`LastWakeUp`	Time from the last Z-Wave module awakening

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 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`	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). 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
`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) **Note!** *Parameter 7 refers to the association of modules that is not supported by the Grenton system!*

METHODS

Name	Description
`RemoveBan`	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. **Note!** `RemoveBan` *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!*
`ClearFailCount`	Clears the number of unsuccessful communication attempts
`Set`	Sets the value of a given configuration (parameter) register `Register` (register or parameter number), `Value` (the value of the register or parameter), `Size` (size of the sent register or parameter value - in bytes) **Note!** *Calling the* `Set` *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!*
`Get`	Gets the value of a given configuration (parameter) register **Note!** *Calling the* `Get` *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!*
`SetDefault`	Sets the default value for a given configuration (parameter) register **Note!** *Calling the* `SetDefault` *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!*

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
`Value`	Returns the input status: `0` - no violation, `1` - violation (smoke)

METHODS

-

EVENTS

Name	Description
`OnValueChange`	An event triggered when the status changes to the opposite
`OnSwitchOn`	An event triggered when the high state is set on input
`OnSwitchOff`	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
`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 battery level changes
`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

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
`Interval`	The period of automatic 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 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`	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). 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
`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`	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. **Note!** `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!*
`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
`Value`	Returns the output state : `0` - low, `1` - high

METHODS

Name	Description
`SetValue`	Sets the output state as `1` or `0`
`Switch`	Switches the output. The Time parameter determines how long the state change takes place, for 0 it is constant
`SwitchOn`	Turns on the output. The Time parameter determines how long the state change takes place, for 0 it is constant
`SwitchOff`	Turns off the output. The Time parameter determines how long the state change takes place, for 0 it is constant

EVENTS

Name	Description
`OnValueChange`	An event triggered when the status changes to the opposite
`OnSwitchOn`	An event triggered when the high state is set to output
`OnSwitchOff`	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 battery level changes
`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`	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). 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
`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). **Note!**After restarting the CLU, the *Switch Binary Switch* command is sent to the module, to which the module does not respond, so that `FailCount` is increased by 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`	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. **Note!** `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!
`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) **Note!** *Calling the* `Set` *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!*
`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
`SetWarningLevel`	Sets the warning level of the Z-Wave module battery

EVENTS

Name	Description
`OnChange`	An event triggered when the battery level changes
`OnLowBattery`	An event triggered when a battery drop is detected below the warning level
`OnBatteryGood`	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
`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 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`	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). 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
`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`	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. **Note!** `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!
`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
`SetWarningLevel`	Sets the warning level of the Z-Wave module battery

EVENTS

Name	Description
`OnChange`	An event triggered when the battery level changes
`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 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
`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.

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`	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). 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
`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`	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. **Note!** `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!*
`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: `0` - lack of movement, `1` - upward movement, `2` - downward movement
`Percent`	Percentage value of the awning opening, where: `0%` - window closed, `100%` - window opened **Note!** *The value of the* `Percent` *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: `0 - Manual` - Manual, `1 - Semiauto` - Semiautomatic, `2 - Auto` - Automatic
`SeasonMode`	Seasonal mode of the device `0 - Summer` - Summer, `1 - Winter` - Winter **Note!** *Parameter does not apply to manual mode* `Mode = 0`
`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 wwhen the awning blind 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`	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). 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
`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`	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. **Note!** `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!*
`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

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 **Note!** The value of the `Percent` feature is refreshed when the roller completes the work - it should be taken into account when using this feature eg for the Slider component. **Note!** *Calling the `Stop` method while roller is in movement does not refresh the `Percent` 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
`OnChange`	An event triggered when the roller state is changed
`OnUp`	An event triggered when the state changes 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

B. 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`	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). 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
`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`	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. **Note!** *`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!*
`ClearFailCount`	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: `0` - Lack of movement, `1` - opening, `2` - closing
`Percent`	The percentage of window opening where: `0%` - window closed, `100%` - window opened **Note!** *The value of the `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
`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`	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). 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
`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`	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. **Note!** `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!*
`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

14. FAKRO ZWMR 24

> **Note!**
> The module service is available from CLU version 05.11.01 *(build 2302A)*.

14.1. General information

FAKRO ZWMR24 module includes the ability to control the roller shutter - both maximum opening/closing and setting the percentage of the roller shutter opening.

Roller shutter movement time is based on the detection of current overload or underload on the control circuit. These values are set via specific registers of the device. Detailed information on parameter settings and modes of operation of S1 and S2 inputs can be found in the manufacturer's documentation of the FAKRO ZWMR24 module.

How to add / remove: Adding / deleting a device is done by pressing the button on the device during inclusion / exclusion (called on CLU).

> **Note!**
> The object does not take over information about the real state of the device controlled by inputs S1 and S2.

14.2. Objects

Object enabling the roller shutter control (up / down / stop). The condition of the roller shutter is determined on the basis of the called methods.

A. ZWAVE_ROLLER_SHUTTER

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

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 down
`HoldDown`	Hold always up
`SetPosition`	Shutter opening percentage setting: `0` % - fully closed, `1` 00% - fully open

FEATURES

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.

FEATURES

Name	Description
`Register`	Register (parameter) number
`Value`	Register (parameter) value
`NodeID`	Module's number (node) in the Z-Wave network
`Banned`	Returns information about communication with module: `0` - communication with the module is not blocked, `1` - blocked communication with the module (module banned)
`FailCount`	The number of failed communication attempts with the Z-Wave module

METHODS

Name	Description
`Set`	Sets the value of the register (parameter)
`Get`	Gets the value of a given register (parameter)
`SetDefault`	Sets the default value for register (parameter)
`RemoveBan`	Removes the blockade of communication with the Z-Wave module
`ClearFailCount`	Cleans the number of failed communication attempts

EVENTS

Name	Description
`OnBanned`	Occurs when Z-Wave device is banned

15. FAKRO ZWS 230

> **Note!**
> The module service is available from CLU version 05.11.01 *(build 2302A)*.

15.1. General information

FAKRO ZWS 230 module includes the ability to control the window and detect water using a rain sensor.

How to add / remove: Adding / removing a device is done by pressing the red button on the device during inclusion / exclusion (triggered on CLU).

15.2. Objects

A. ZWAVE_FAKRO_ZWS

> **Note!**
> The object does not take over information about the real state of the device controlled from the button located on the module.

FEATURES

Name	Description
`State`	Device state: `0` - no movement, `1` - opening, `2` - closing
`WaterSensor`	Water sensor value

METHODS

Name	Description
`Open`	Opening the window
`Close`	Closing the window
`Stop`	Stop if window is opening or closing
`Start`	Opening the window if the preceding motion was closing or closing the window if the preceding motion was opening

EVENTS

Name	Description
`OnChange`	Result from a change in the window state
`OnOpen`	Occurs when changing the Stop state to the Up state
`OnClose`	Occurs when changing the Stop state to the Down state
`OnStart`	Occurs when Start is requested
`OnStop`	Occurs when Stop is requested
`OnRainChange`	Occurs when a change in the sensor state takes place (regardless of the value)
`OnRainOn`	Occurs when the high state is set at sensor
`OnRainOff`	Occurs when the low state is set at sensor

B. ZWAVE_CONFIG

The object displays information about communication parameters with the module in the Z-Wave network.

FEATURES

Name	Description
`NodeID`	Module's number (node) in the Z-Wave network
`Banned`	Returns information about communication with module: `0` - communication with the module is not blocked, `1` - blocked communication with the module (module banned)
`FailCount`	The number of failed communication attempts with the Z-Wave module

METHODS

Name	Description
`RemoveBan`	Removes the blockade of communication with the Z-Wave module
`ClearFailCount`	Cleans the number of failed communication attempts

EVENTS

Name	Description
`OnBanned`	Occurs when Z-Wave device is banned

16. Fibaro RGBW

Module version: ***FGRGBWM-441 v2/5 EU***

16.1. 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).

16.2. 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: `0` - communication with the module is not blocked, `1` - blocked communication with the module (banned module). Blocking occurs when 3 consecutive attempts to communicate with the module fail (incrementing the `FailCount` 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 (`Banned` = 1)
`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 `Banned` = 1 feature). Calling the method enables the command to be sent again to the module. **Note!** *The `RemoveBan` feature is not synonymous with correct 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

17. Remotec ZXT-120

Module version: ZXT-120EU V1.0

17.1. 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).

17.2. 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.

17.3. Objects

A. ZWAVE_IR

The object allows reading and writing of configuration parameters and sending IR codes.

FEATURES

Name	Description
`AcDeviceNumber`	Returns the number of the AC device from the internal library of IR codes (number from the ZXT-120 Code List)
`EmitterPower`	Returns the power of the external (connected) infrared transmitter: `0` - normal power `255` - high power
`LearningStatus`	Returns the status of learning the IR codes: `0` - IR channel idle, `1` - learning successful, `2` - the learning procedure is in progress, `4` - learning failed
`SurroundIrControl`	Multidirectional IR signal transmission: `0` - Disabled, `255` - 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
`SendCode`	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)
`LearnCode`	Invokes the learning mode of the IR code with a specific number (code number in the range 0-22)
`SetAcDeviceNumber`	Sets the AC device number from the internal IR code library (number from the ZXT-120 Code List)
`SetEmitterPower`	Sets the power of the external infrared transmitter
`SetSurroundIrControl`	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 battery level changes
`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
`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.

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`	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). 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
`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`	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. **Note!** `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!*
`ClearFailCount`	Clears the number of unsuccessful communication attempts

EVENTS

Name	Description
`OnBanned`	An event triggered when the device is banned

18. Remotec ZXT-310

Module version: ***ZXT-310EU HW: 00 FW: 1.10***

18.1. 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.

18.2. 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.

18.3. 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 **Note!** *Parameter* `EmitterPower` *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
`SendCode`	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)
`LearnCode`	Invokes the learning mode of the IR code with a specific number (code number in the 1-384 range)
`SetPortRouting`	Sets the IR port number to be assigned to the currently selected endpoint
`SetAvDeviceNumber`	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)
`SetEmitterPower`	Sets the power of the external infrared transmitter **Note!** *Parameter* `EmitterPower` *is not configurable for port 1*
`SetTransmissionMode`	Sets the IR code transmission mode
`SetEndpointNumber`	Sets the endpoint number to be controlled (1 ÷ 6)

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 IR code changes to *"OK"*
`OnLearning`	An event triggered when the IR learning mode status changes to "*Learning*"
`OnLearning`	An event triggered when the IR learning mode status changes to "*Command Full*"
`OnLearningFail`	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
`PortRouting`	Returns the number of the IR port assigned to the 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 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 **Note!** *The* `EmitterPower` *parameter is not configurable for port 1*
`TransmissionMode`	Returns the IR code transmission mode: `0` - continuous transmission, `255` - single pulse

METHODS

Name	Description
`SendCode`	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)
`SetPortRouting`	Sets the IR port number to be assigned to the currently selected endpoint
`SetAvDeviceNumber`	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)
`SetEmitterPower`	Sets the power of the external infrared transmitter **Note!** *Parameter* `EmitterPower` *is not configurable for port 1*
`SetTransmissionMode`	Sets the IR code transmission mode

EVENTS

Name	Description
`OnIrSend`	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
`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.

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`	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). 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
`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`	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. **Note!** `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!*
`ClearFailCount`	Clears the number of unsuccessful communication attempts

EVENTS

Name	Description
`OnBanned`	An event triggered when the device is banned

19. Aeotec Nano Switch

Module version: ZW116-C

19.1. 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.

19.2. 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`; `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
`OnValueChange`	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`	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). 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
`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`	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. **Note!** *`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!*
`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

20. Aeotec Dual Nano Switch

Module version: ZW140-C

20.1. 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
`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 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.

20.2. 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`; `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
`OnValueChange`	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`	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). 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
`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`	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. **Note!** `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!*
`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

21. Aeotec Nano Dimmer

Module version: ZW111-C

21.1. 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.

21.2. 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`	Determines 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 continuous 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
`OnValueChange`	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
`OnValueLower`	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 `HoldDelay` value
`SetHoldInterval`	Sets `HoldInterval` value

EVENTS

Name	Description
`OnValueChange`	Occurs when a change in the input state takes place (regardless of the value)
`OnSwitchOn`	Occurs when the high state is set at input
`OnSwitchOff`	Occurs when the low state is set at input
`OnShortPress`	Occurs after pressing the button for 500 - 2000ms
`OnLongPress`	Occurs after pressing the button for at least 2000ms
`OnHold`	Occurs for the first time after HoldDelay time and then cyclically every HoldInterval value
`OnClick`	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
`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`	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). 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
`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`	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. **Note!** *`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!*
`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

22. Aeotec Nano Shutter

Module version: ZW141-C

22.1. 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
`35` - moving time from up to down in seconds	`11`
`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.

22.2. 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
`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
`MaxTime`	Default Time parameter value. 0 if not specified

METHODS

Name	Description
`Up`	Roller shutter up or STOP if moving. Parameter Time: `number` - output is active for specified timer `0` - output is active for the time specified in MaxTime
`Down`	Roller shutter down or STOP if moving. Parameter Time: `number` - output is active for specified timer `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: `number` - output is active for specified timer `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
`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`	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). 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
`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`	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. **Note!** *`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!*
`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

23. Aeotec Nano Shutter (V2)

Module version: ZW141-C

23.1. 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.

23.2. 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`	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). 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
`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`	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. **Note!** `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!*
`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

24. Aeotec Smart Switch 7

Module version: ZW175-C16

> **Note!**
> The module service is available from CLU version 05.11.01 *(build 2302A)*.

24.1. General information

Aeotec Smart Switch 7 support includes the ability to control the socket using the Grenton system. The module can be controlled by the methods of the ZWAVE_DOUT object or by the button located on the module.

How to add / remove: To add the Aeotec Smart Switch 7 module, hold the add button until the diode turns blue. When the button is released, the LED will flash blue. At this point, make two short presses. All activities should be performed during Inclusion called on CLU. To remove a module, hold down the button on the module until it turns blue. When released, the LED will flash blue. During this time, then make two short presses. These activities should be performed during the Exclusion invoked on the CLU.

> **Note!**
> In order to unlock the module when it is blocked due to exceeding the Overload value, the `SetProtectionState(Off)` method should be used.

24.2. Objects

A. ZWAVE_DOUT

Object enabling the socket control.

FEATURES

Name	Description
`Value`	Returns: `1` for output set at `On`; `0` for output set at `Off` state
`Overload`	Device power value beyond which the device is blocked and the `OnOverload` event is generated at this time
`ProtectionState`	Device lock status: `0` - off `2` - on
`Power`	Returns power in watts
`PowerConsumption`	Returns power consumption in kilowatt-hour

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 mode spec `number` - switches output for a time specified by a parameter (in milliseconds)
`SwitchOn`	Sets output value to `1`
`SwitchOff`	Sets output value to `0`
`SetOverload`	Sets the power value beyond which the OnOverload event is generated
`SetProtectionState`	Sets device lock status

EVENTS

Name	Description
`OnValueChange`	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
`OnOverload`	Occurs when a device is blocked due to exceeding the Overload value

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.

FEATURES

Name	Description
`Register`	Register (parameter) number
`Value`	Register (parameter) value
`NodeID`	Module's number (node) in the Z-Wave network
`Banned`	Returns information about communication with module: `0` - communication with the module is not blocked, `1` - blocked communication with the module (module banned)
`FailCount`	The number of failed communication attempts with the Z-Wave module

METHODS

Name	Description
`Set`	Sets the value of the register (parameter)
`Get`	Gets the value of a given register (parameter)
`SetDefault`	Sets the default value for register (parameter)
`RemoveBan`	Removes the blockade of communication with the Z-Wave module
`ClearFailCount`	Cleans the number of failed communication attempts

EVENTS

Name	Description
`OnBanned`	Occurs when Z-Wave device is banned

25. Aeotec Multisensor 6

Module version: ZW100-C

25.1. General information

The Aeotec Multisensor 6 Z-Wave module allows you to read: temperature, light level, humidity, UV rays, battery level and has a tamper alarm. In addition, it gives the ability to set/read the time of waking up the module.

> **Attention!**
> There may be numerous delays and anomalies in the execution of commands by modules, especially those that are within the operating range of other Z-WAVE devices or devices using the 868.42 MHz radio frequency, such as remote controls for gates.

How to add / remove: In order to add a module, short press the action button in the Aeotec Multisensor 6 Z-Wave module during Inclusion called on the CLU. To remove the module, short press the action button during Exclusion.

The Aeotec Multisensor 6 Z-Wave module can be powered by a battery or through a suitable USB adapter. The manufacturer recommends using USB power during setup. Configuration for battery power must be done after waking up the module. In order to wake up the module, hold down the button on the module for 3 seconds. Wake-up mode persists for 10 minutes.

25.2. Objects

A. BINARY_SENSOR

The object allows you to read the status of the motion sensor.

FEATURES

Name	Description
`Value`	Returns the input status: `0` - no violation, `1` - violation

EVENTS

Name	Description
`OnValueChange`	Occurs when a change in the input state takes place (regardless of the value)
`OnSwitchOn`	Occurs when the high state is set at input
`OnSwitchOff`	Occurs when the low state is set at input

B. ANALOG_SENSOR

The object allows you to read the temperature (ANALOG_SENSOR_01), illuminance measured in lux (ANALOG_SENSOR_02), humidity (ANALOG_SENSOR_03), UV rays (ANALOG_SENSOR_04).

FEATURES

Name	Description
`Value`	Current sensor value
`MinValue`	Value below which OnOutOfRange event is occurred
`MaxValue`	Value above which OnOutOfRange event is occurred

METHODS

Name	Description
`SetMinValue`	Sets MinValue
`SetMaxValue`	Sets MaxValue

EVENTS

Name	Description
`OnValueChange`	Event occurring when sensor Value changes
`OnValueRise`	Event occurring when the set value is higher than the current value
`OnValueLower`	Event occurring when the set value is lower than the current value
`OnOutOfRange`	Event occurring when setting a value which is higher than the maximum value or lower than the minimum value
`OnInRange`	Event occurring when setting a value which is lower than the maximum value or higher than the minimum value

C. ZWAVE_TAMPER_ALARM

The object displays information about the tamper alarm status.

FEATURES

Name	Description
`AlarmDetected`	Alarm detection status: `0` - no tamper detected, `1` - tamper detected

METHODS

Name	Description
`ClearAlarm`	Cancels active alarm

EVENTS

Name	Description
`OnChange`	Occurs after changing the alarm state in AlarmDetected
`OnAlarmDetected`	Occurs after changing AlarmDetected from `0` to `1`
`OnAlarmCleared`	Occurs after changing AlarmDetected from `1` to `0`

D. ZWAVE_WAKEUP

The object allows you to set and read the battery wake-up time of the Z-Wave module. The default value set by the CLU is 3600s (60 minutes). The minimum value is 240s (4 minutes); maximum 7200s (2 hours).

FEATURES

Name	Description
`Interval`	Z-Wave module awakening period from sleep mode
`LastWakeUp`	Time from the last Z-Wave module awakening

METHODS

Name	Description
`SetInterval`	Sets Z-Wave module awakening time from sleep mode

EVENTS

Name	Description
`OnWakeUp`	Event occurring after detecting Z-Wave module waking up from sleep mode

E.ZWAVE_BATTERY

The object allows you to read the battery status. The reading is carried out cyclically, every set time, for the `Interval` feature of the **`ZWAVE_WAKEUP`** object (default 3600s).

FEATURES

Name	Description
`BatteryLevel`	Z-Wave module battery level (in %)
`WarningLevel`	Z-Wave module battery level below which the warning event is occurring

METHODS

Name	Description
`SetWarningLevel`	Sets Z-Wave battery level below which the warning event is occurring

EVENTS

Name	Description
`OnChange`	Event occurring after Z-Wave module battery level change
`OnLowBattery`	Event occurring after Z-Wave module battery level drop below the WarningLevel
`OnBatteryGood`	Event occurring when the Z-Wave module battery level increase above the WarningLevel

F. 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`	Information on blocking Z-Wave communication with the module: `0` - communication with the module is not blocked, `1` - communication with the module is blocked (banned module). 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
`FailCount`	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.
`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)

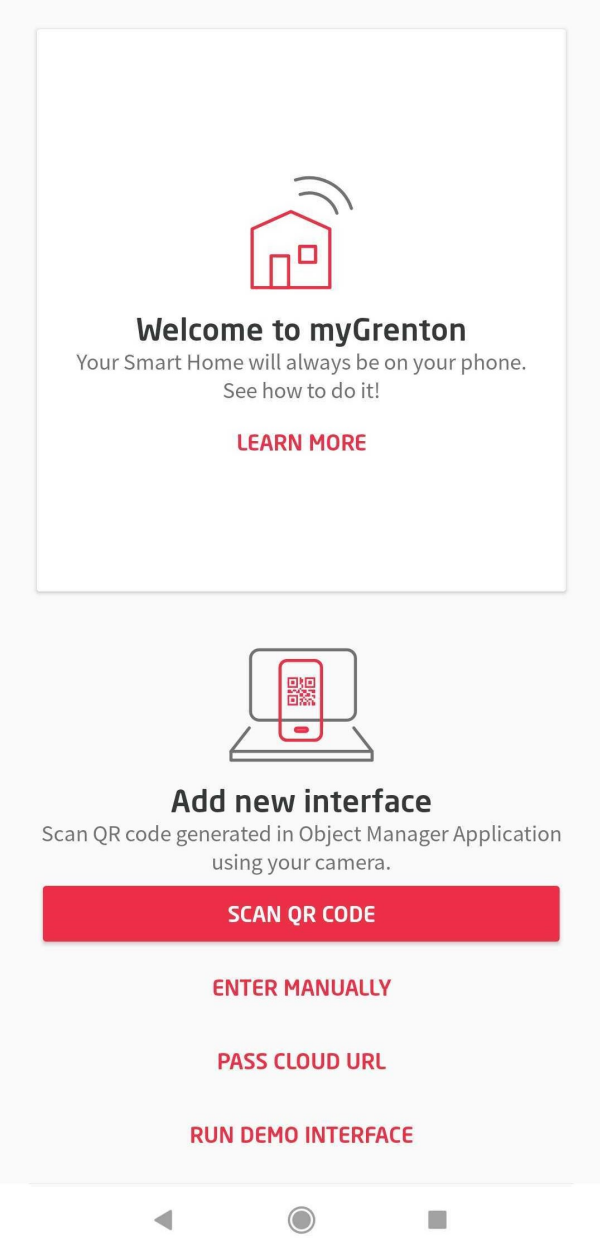
XVIII. myGrenton mobile application

1. Installation and first launch of the myGrenton application

1.1. Installation

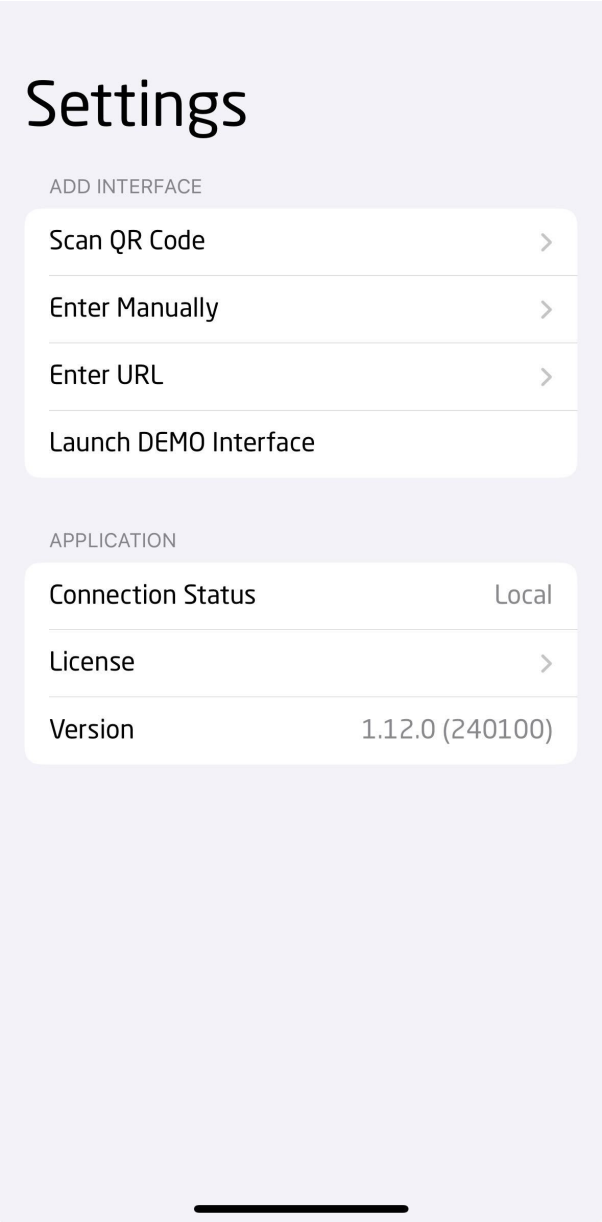
A. Android

The current version of the application is available in the Play Store for phones or tablets with the **Android system (version 7.0 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 (13 or higher)**. After opening the application, a welcome window with the possibility of adding a new interface will appear.



Note!
Minimum version of CLU **05.06.04** is required for correct operation of the application. For selected widgets, the minimum CLU and Object Manager versions required for correct operation are given in their descriptions.

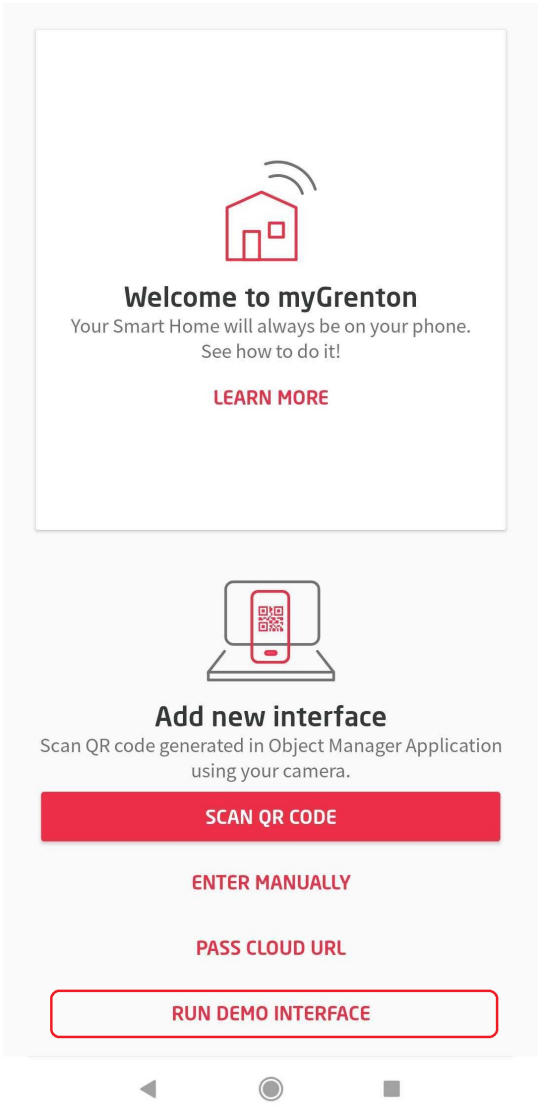
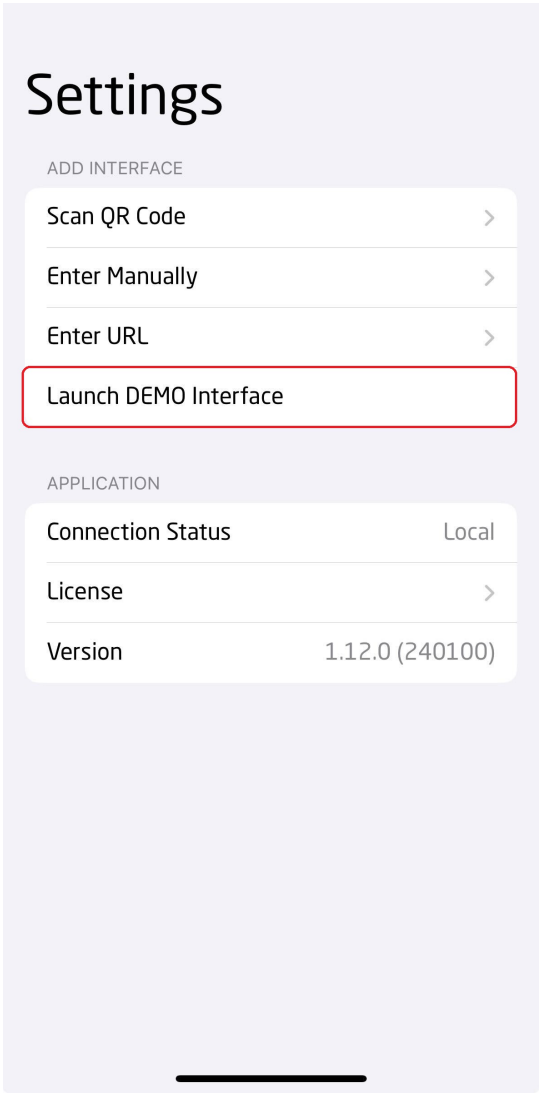
Note!
The application may ask, among others for permission to take photos and videos, full access to the network, display network connections, prevent the phone from going to sleep, receive data from the Internet. For correct working of the application, you must agree to the above conditions.

1.2. First start-up, demonstration interface

The myGrenton application allows you to experience the application interface, widget capabilities and key functionalities without connecting to the actual Grenton system using the DEMO interface.

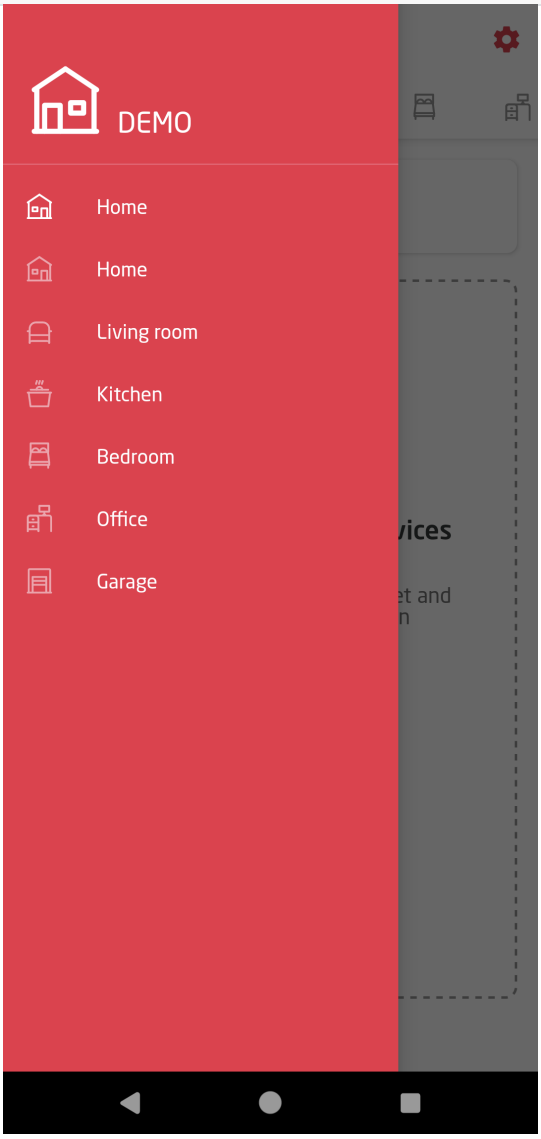
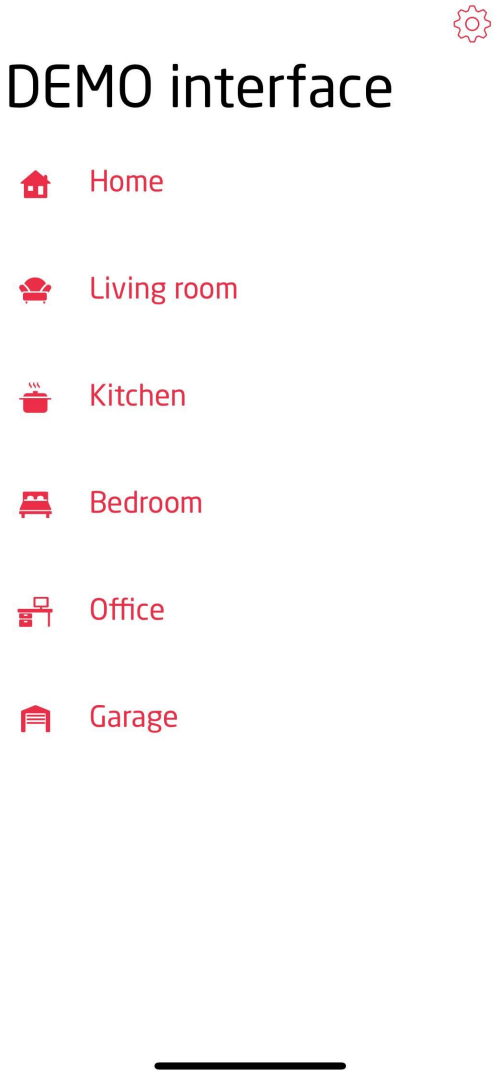
Adding a DEMO interface is possible from the first run view of the application or subsequent ones (Settings view), only if the user has not added an interface connected to the real system. The interface includes several sample pages with fully functional widgets.

Note!
The DEMO interface is available for myGrenton application version 1.5.0 (Android) / 1.9.0 (iOS) or higher.

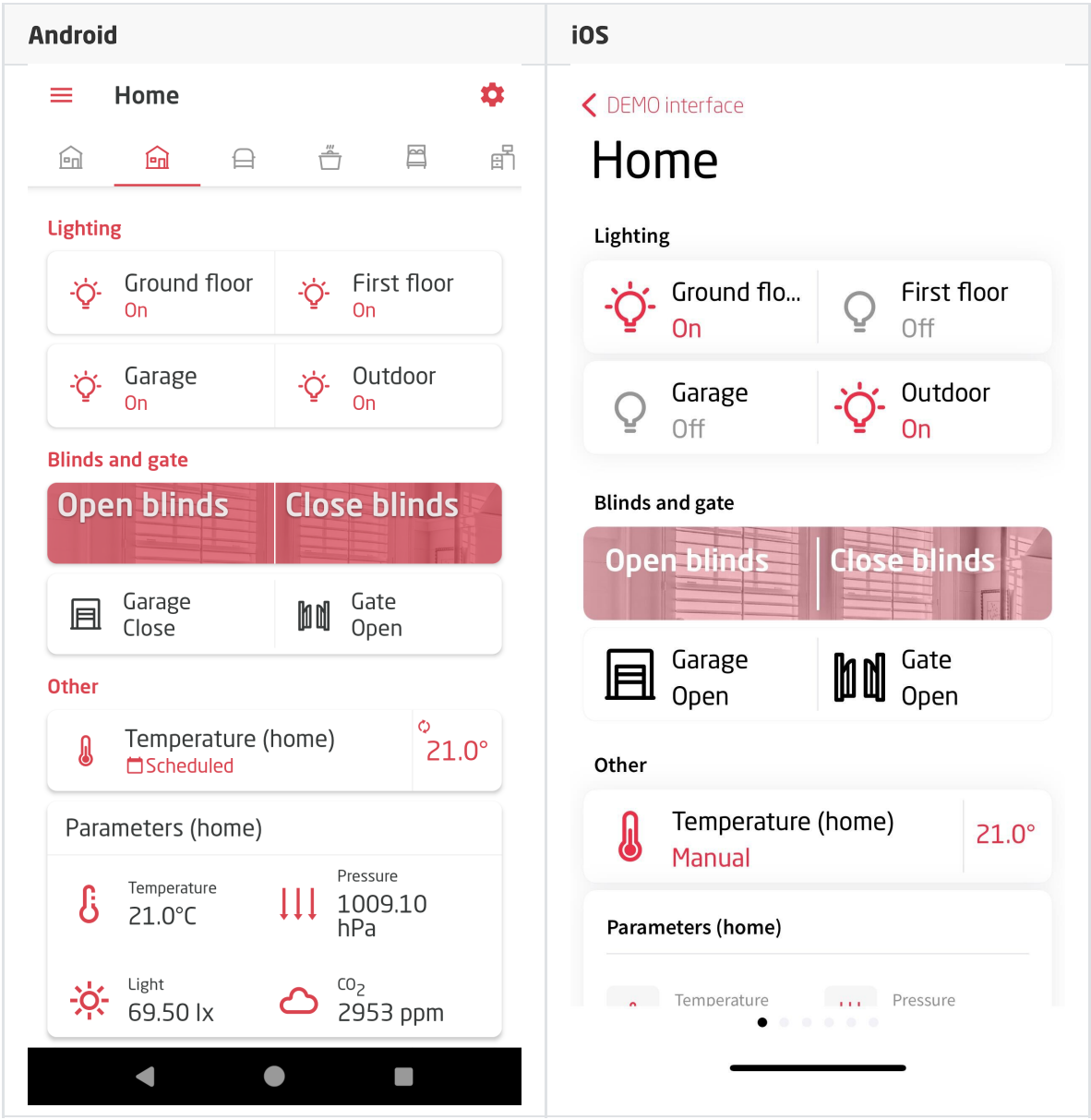
Android	iOS
	

Welcome to myGrenton
Your Smart Home will always be on your phone.
See how to do it!
LEARN MORE
Add new interface
Scan QR code generated in Object Manager Application using your camera.
SCAN QR CODE
ENTER MANUALLY
PASS CLOUD URL
RUN DEMO INTERFACE

Settings
ADD INTERFACE
Scan QR Code
Enter Manually
Enter URL
Launch DEMO Interface
APPLICATION
Connection Status
Local
License
Version
1.12.0 (240100)

Android	iOS
	

DEMO
Home
Home
Living room
Kitchen
Bedroom
Office
Garage
DEMO interface
Home
Living room
Kitchen
Bedroom
Office
Garage

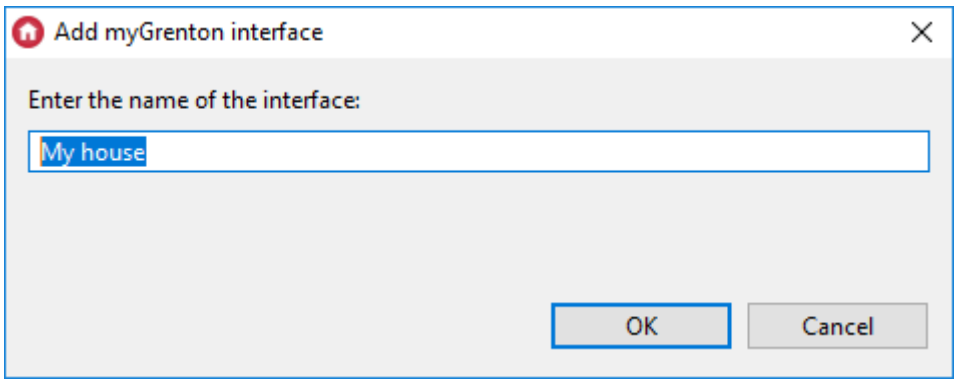


The DEMO interface will be automatically deleted after adding an new interface. It is possible to delete DEMO interface in the standard way - in the interface management window.

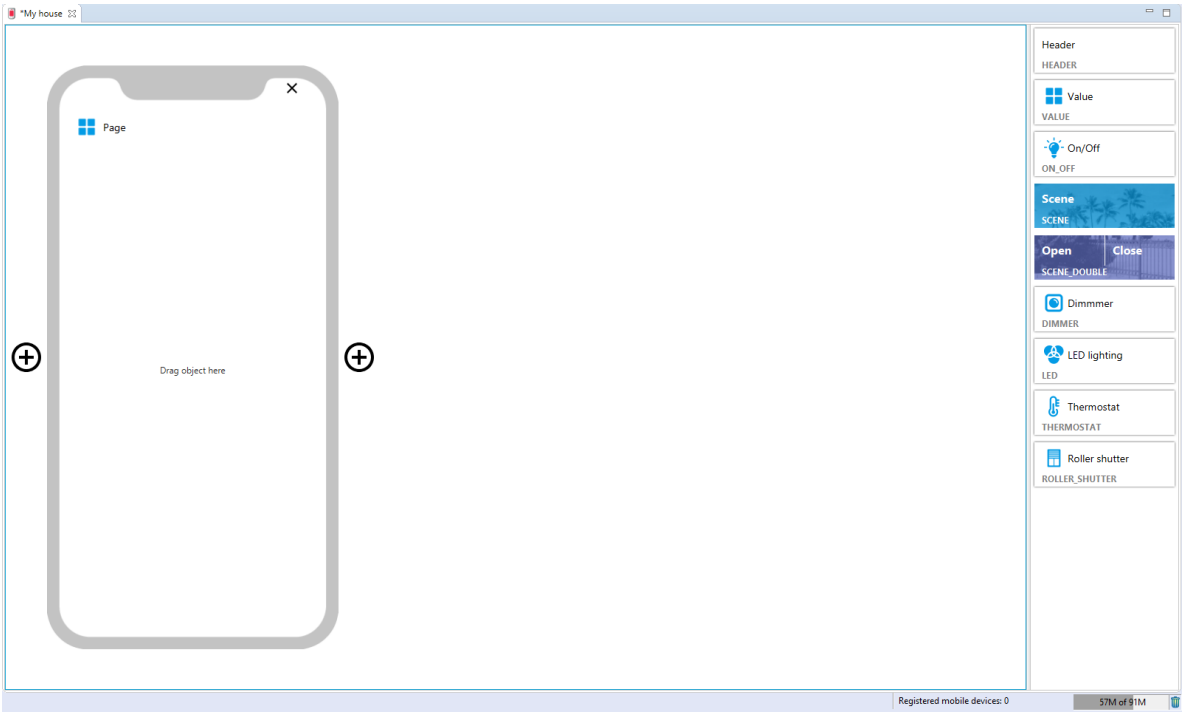
2. Creating the interface

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

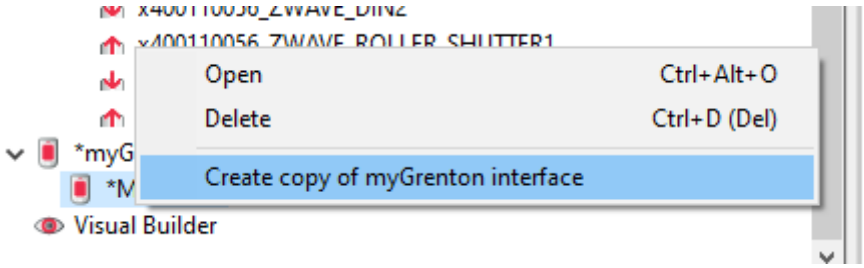
To add another interface page, click on the + icon next to the phone graphic. Up to 30 pages can be created.

2.2. Deleting a page from the interface

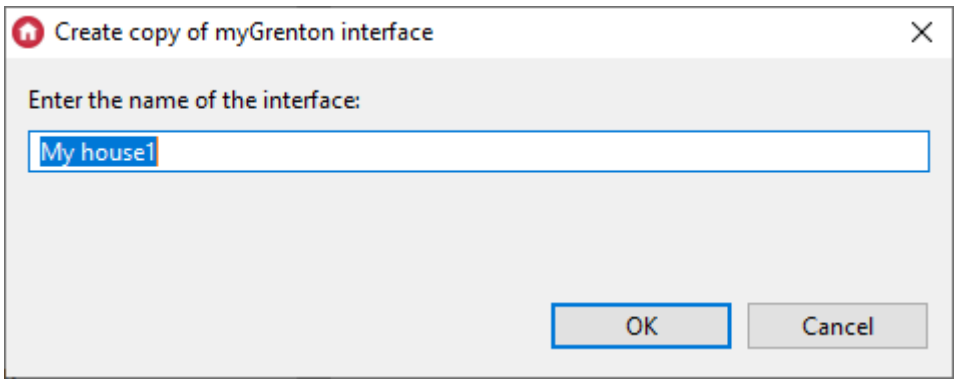
To remove a page from the interface, click on the ⨉ icon, which is located in the upper right corner of the phone and confirm the removal of the page.

2.3. Copying the interface

To copy an interface, right-click on the interface and select "Create copy of myGrenton interface" from the menu:



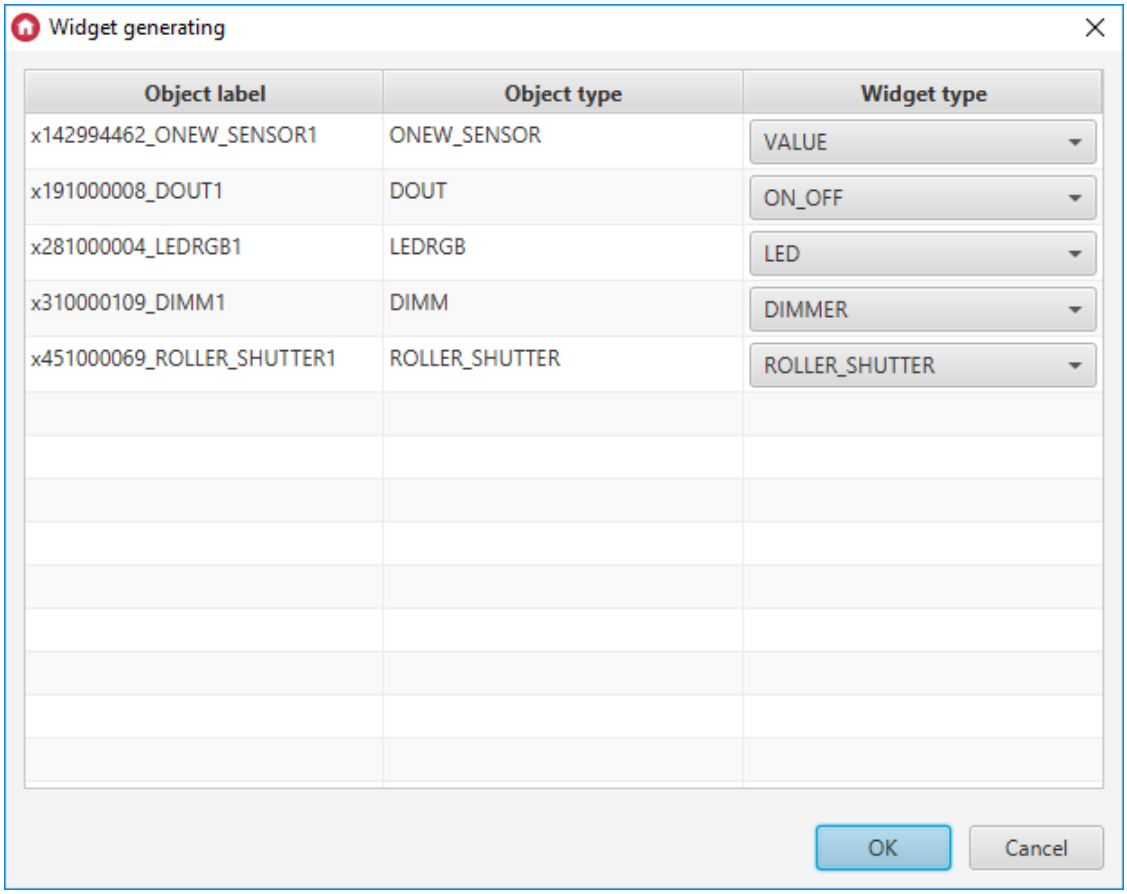
Then choose a name for the new interface:



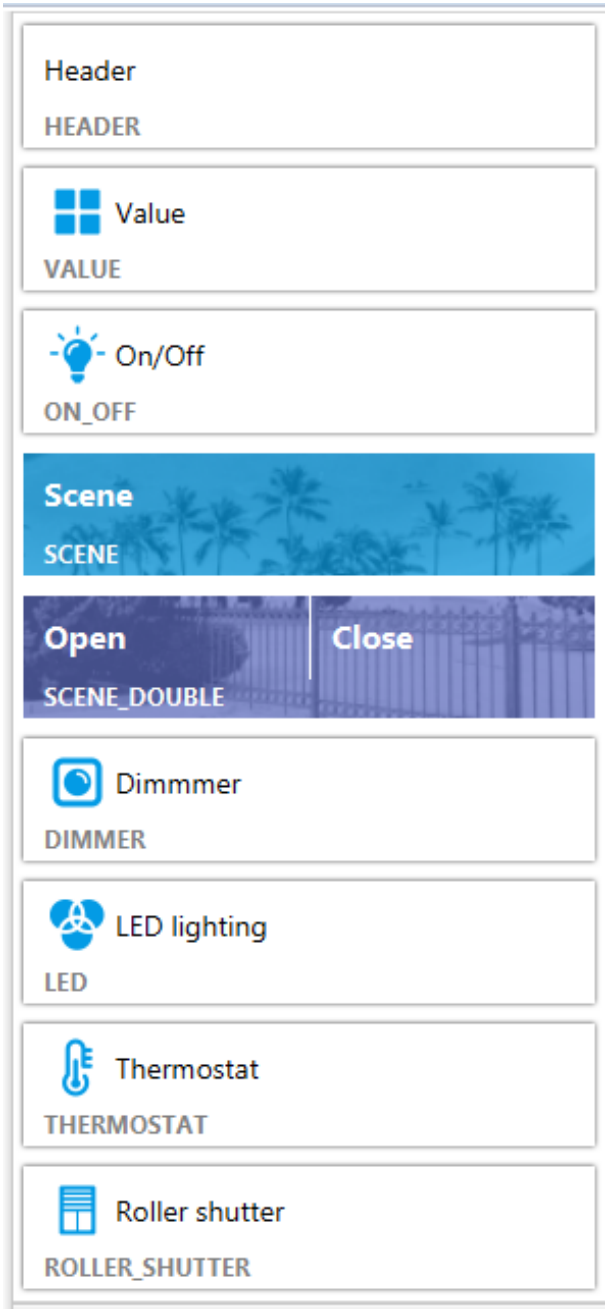
3. Widgets

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)



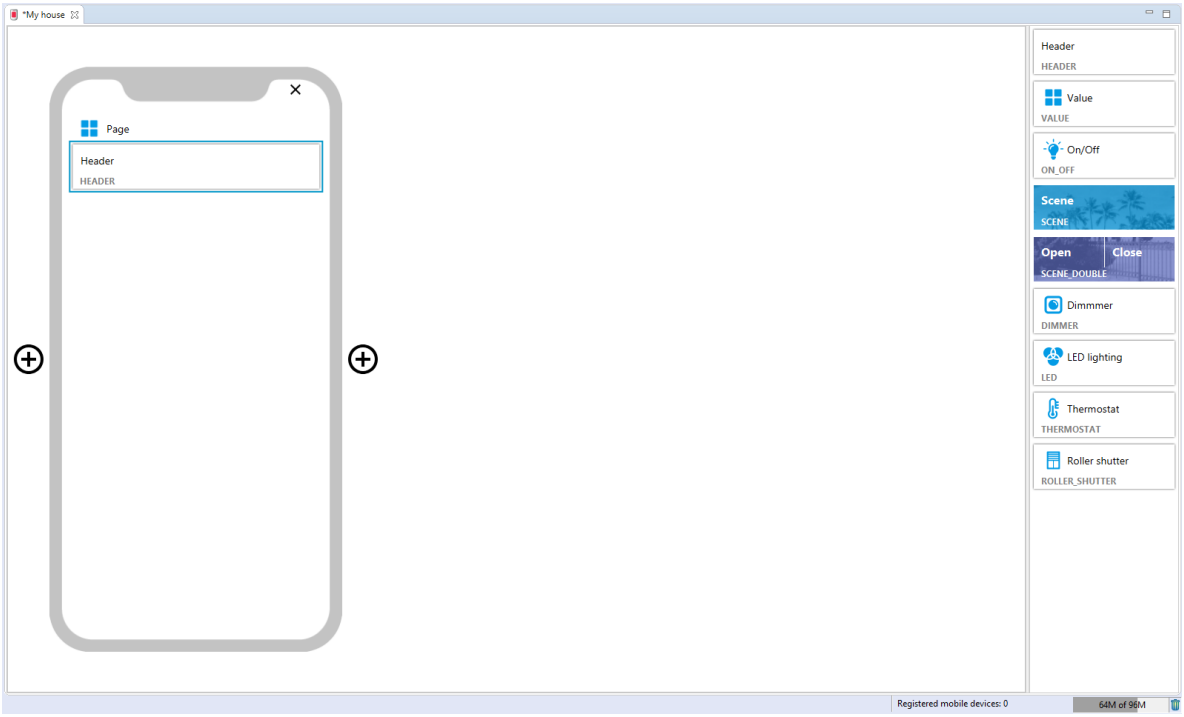
2. Dragging a specific widget from the tab on the right side and then completing its desired parameters.



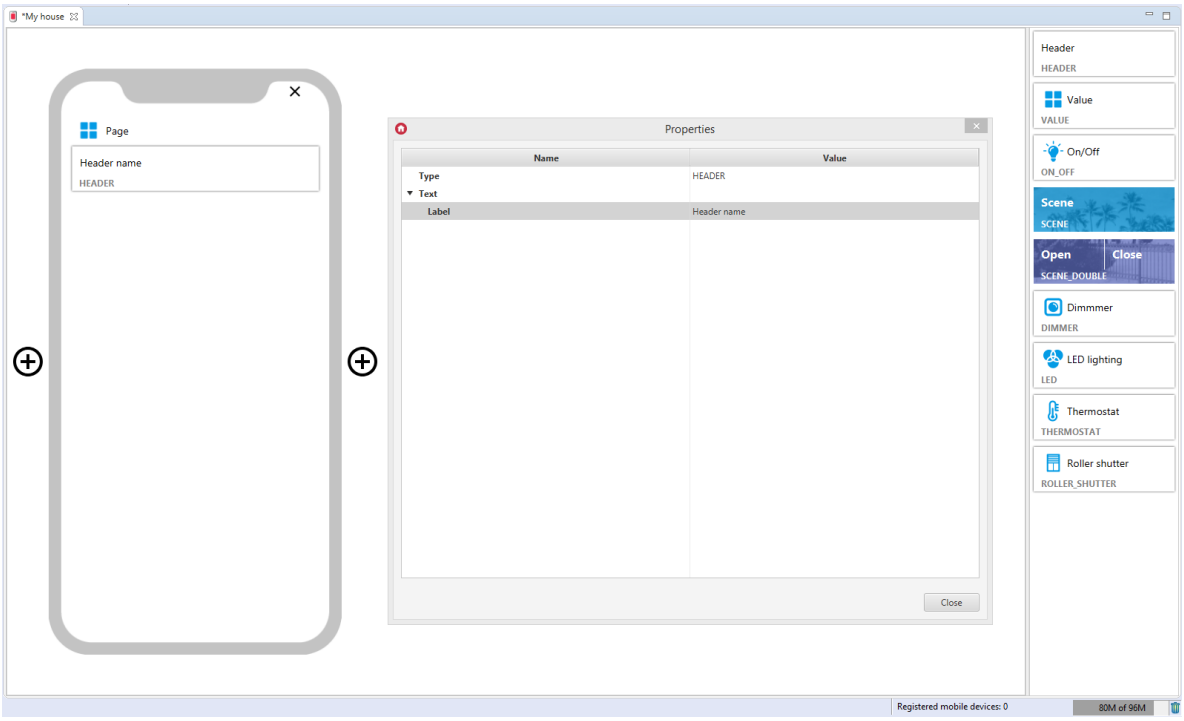
Note!
The maximum number of widgets per page is 30.

3.1. Header (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.



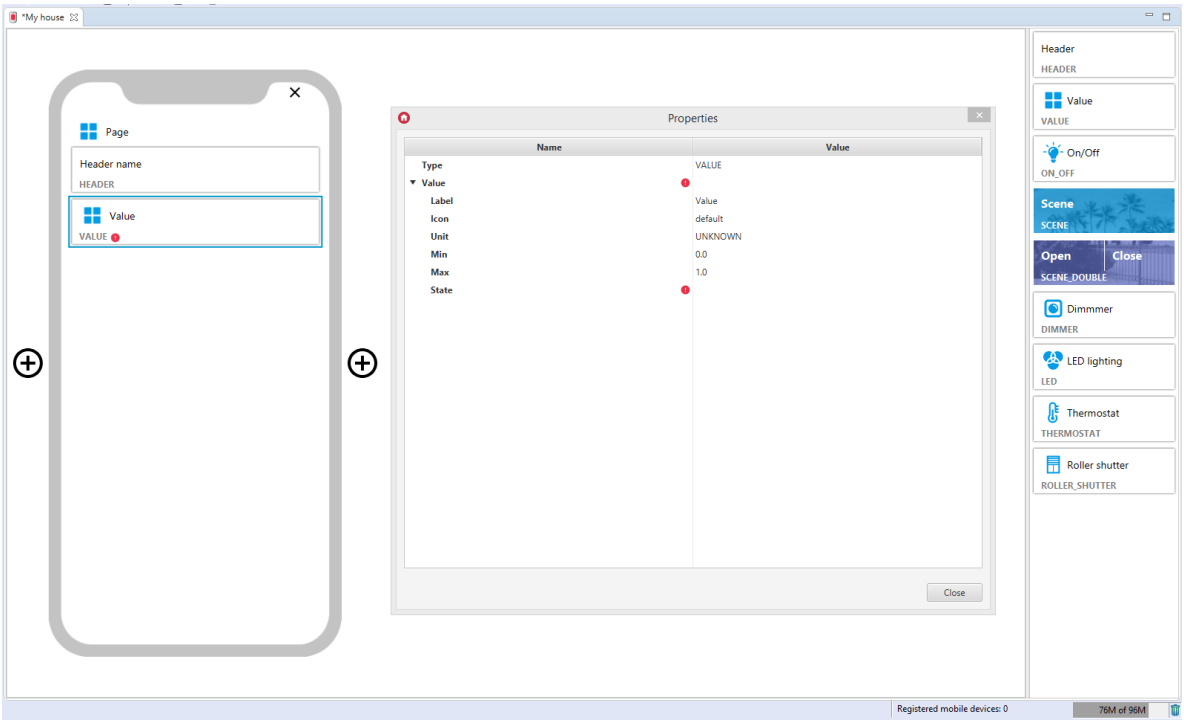
3.2. Value (VALUE)

Note!

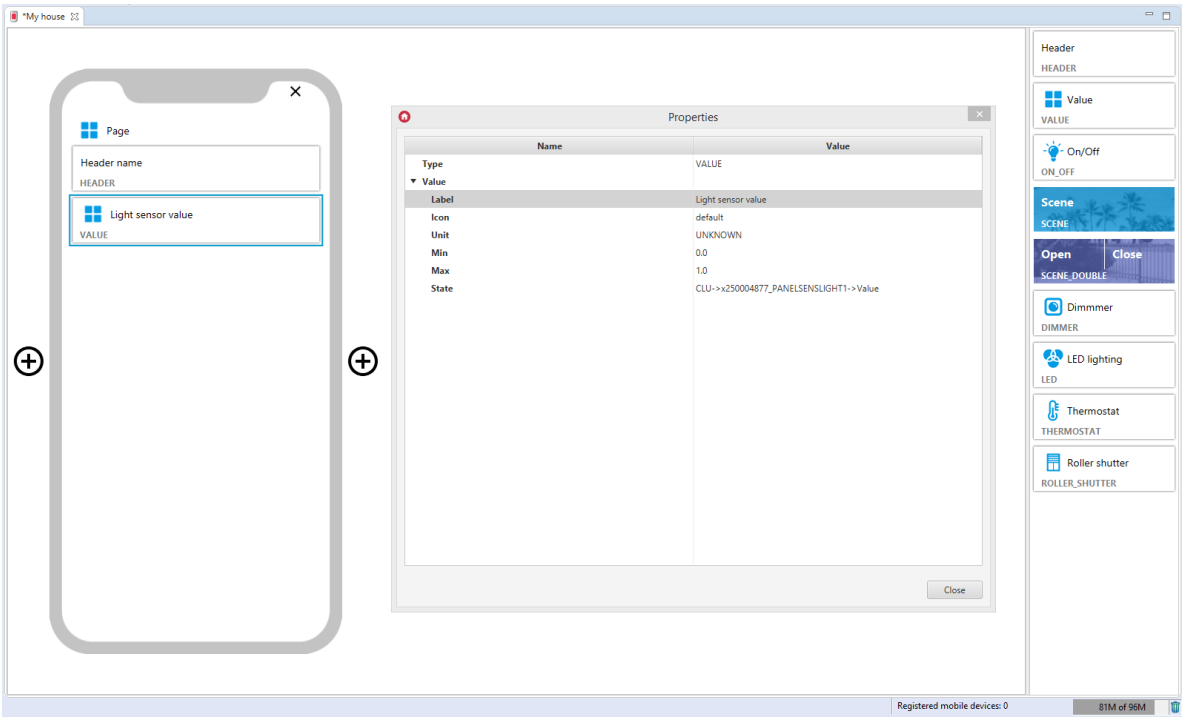
From the version of Object Manager 1.6.0, the VALUE widget and the possibility of using it as a pre-defined template will not be available. It is replaced with the VALUE_V2 widget.

VALUE widgets included in projects created on previous versions of Object Manager will still be properly supported and displayed in the myGrenton application.

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 VALUE widget:

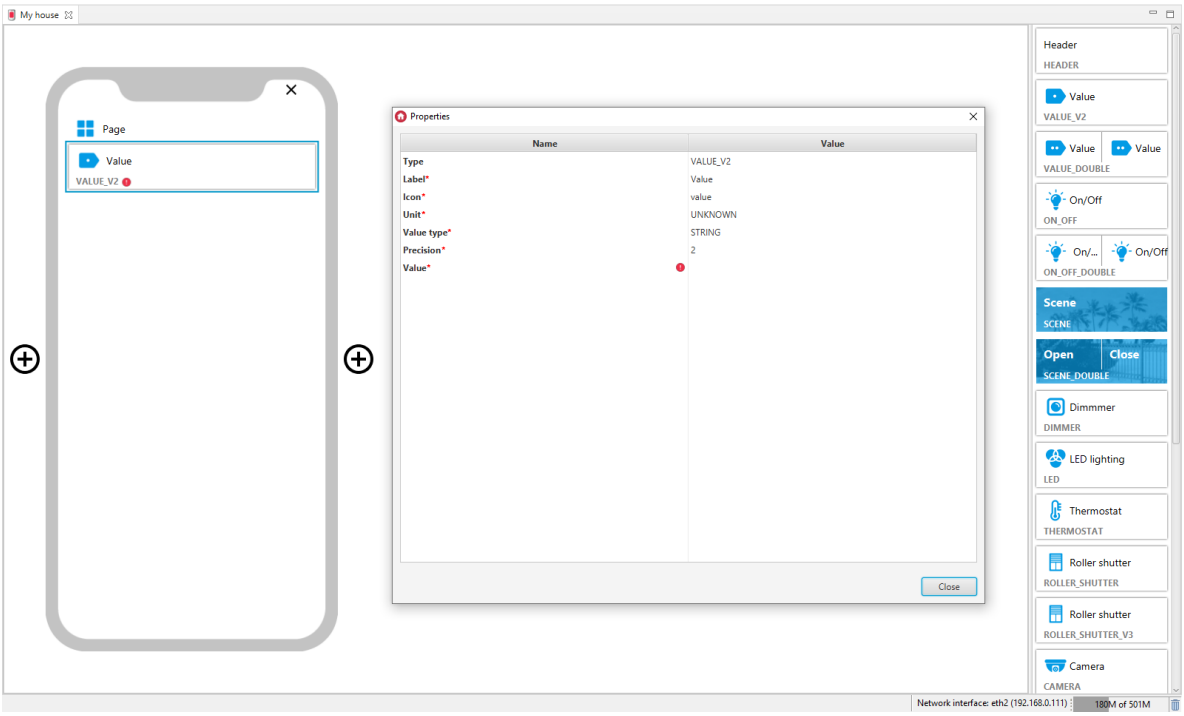


3.3. Value v2 (VALUE_V2)

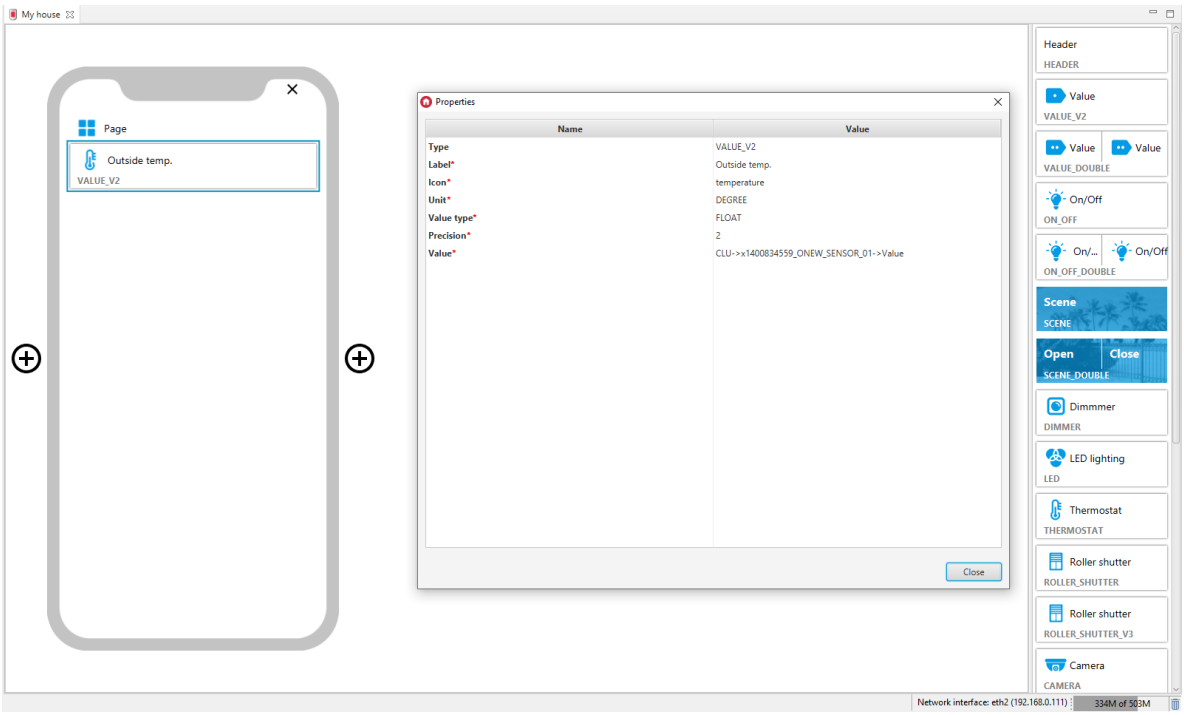
Note!
The VALUE_V2 widget is available for Object Manager version 1.6.0 or higher, and for myGrenton application version 1.4.0 or higher (Android) and version 1.8.0 or higher (iOS).

The widget is dedicated to display the value of a user feature or an embedded feature of a given object. This widget has three units to choose from: unknown, percent and degree. The selected unit is displayed next to the value. In addition, you should choose the type:

- STRING - text,
- FLOAT - floating point, for this type of value it is possible to define the number of displayed decimal places (property `Precision`),
- INTEGER - only integer values are displayed.



Filled VALUE_V2 widget:

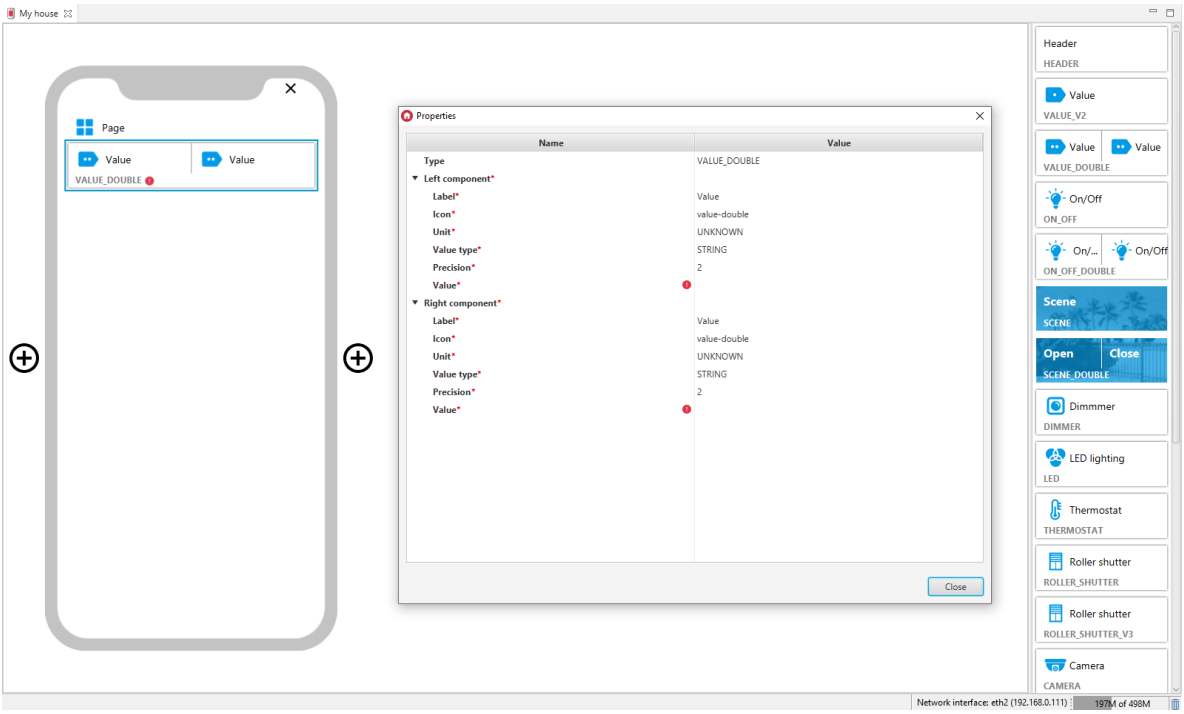


3.4. Value Double (VALUE_DOUBLE)

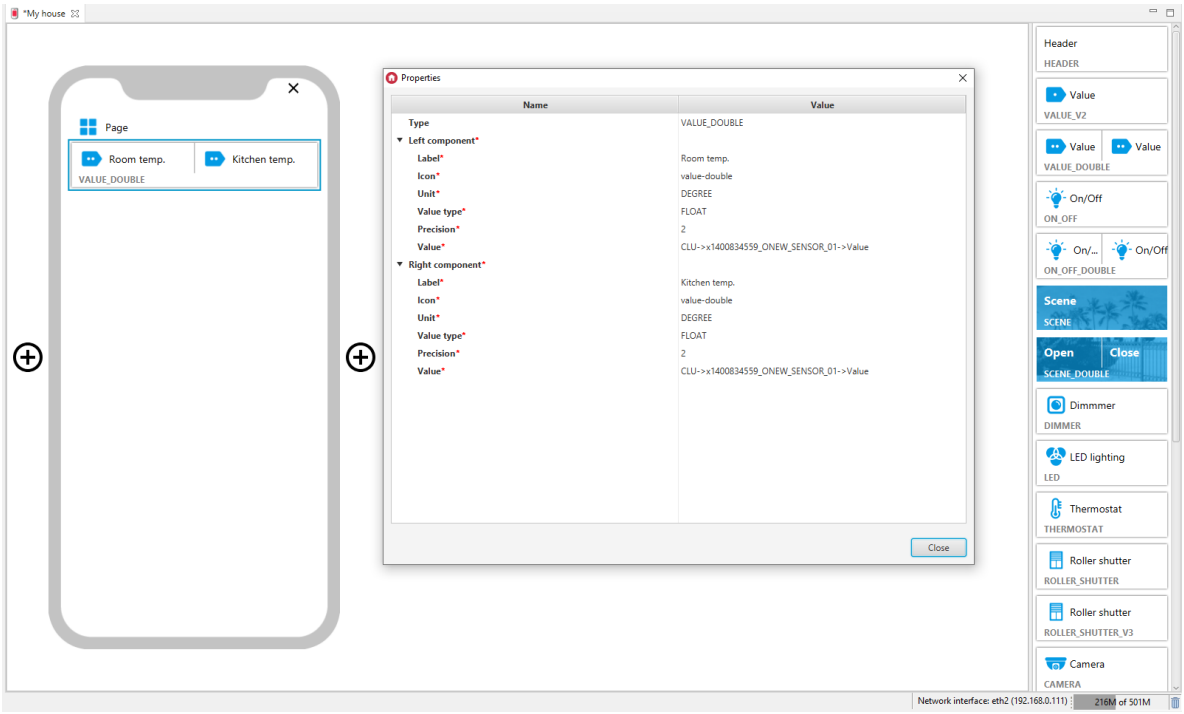
Note!
The VALUE_DOUBLE widget is available for Object Manager version 1.6.0 or higher, for myGrenton application version 1.4.0 (Android) / 1.8.0 (iOS) or higher.

This is the Value v2 double-version widget. The widget is dedicated to display the value of a user features or an embedded features of a given object. This widget has three units to choose from: unknown, percent and degree. The selected unit is displayed next to the value. In addition, you should choose the type:

- STRING - text,
- FLOAT - floating point, for this type of value it is possible to define the number of displayed decimal places (property `Precision`),
- INTEGER - only integer values are displayed.

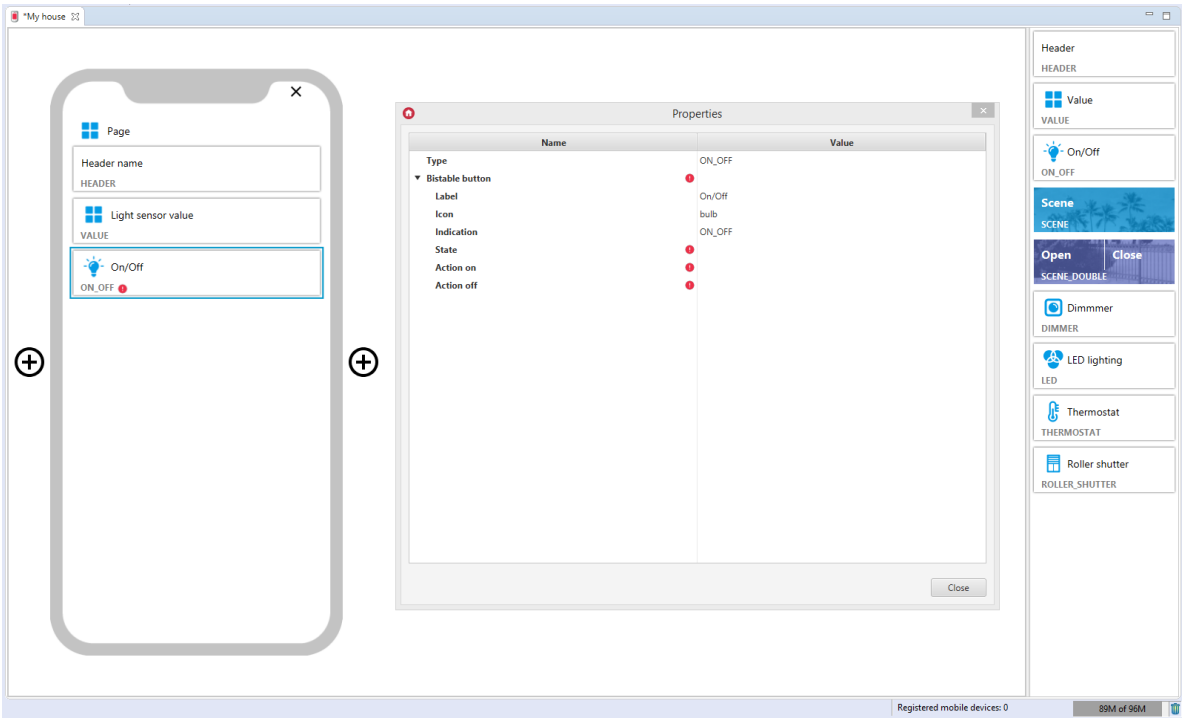


Filled VALUE_DOUBLE widget:

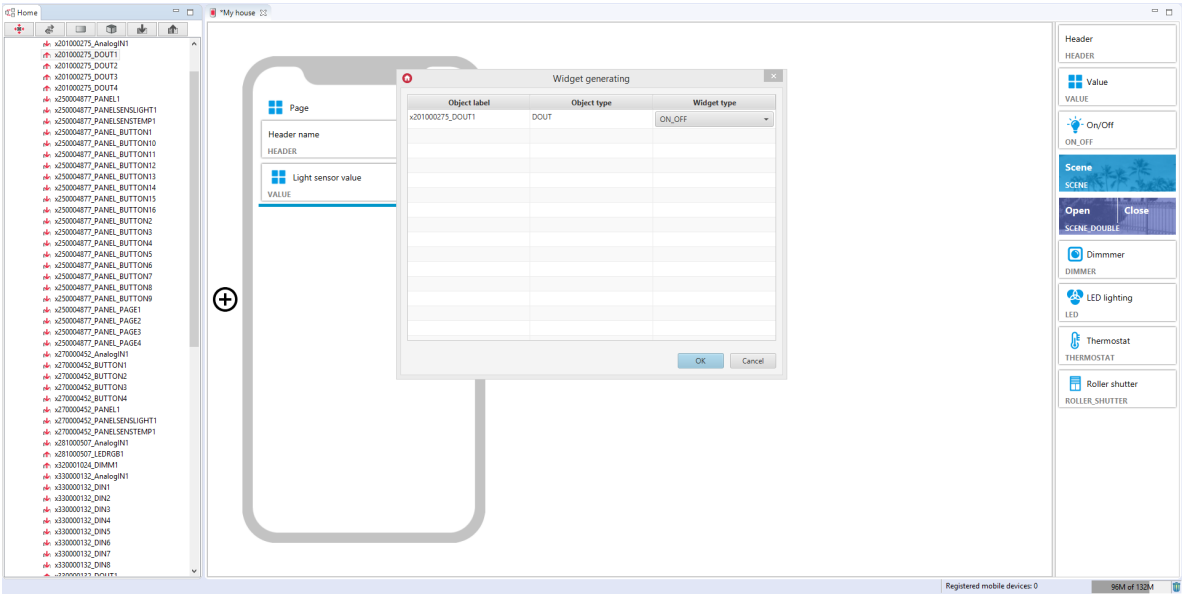


3.5. On/Off (ON_OFF)

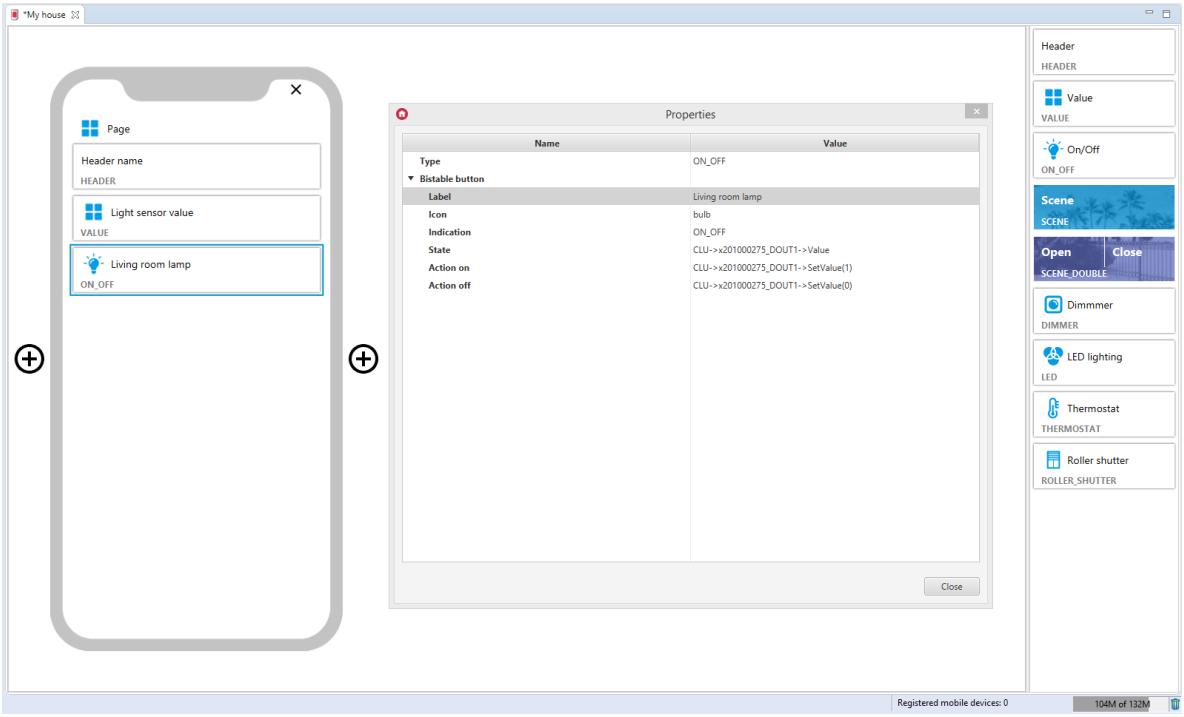
The widget is used to control the executive elements. When dragging the widget from the tab on the right side of the screen, it must be completed with the values in the `State`, `Action on`, `Action off` fields. Widget dedicated mainly to control relay outputs, however it can be used, among others, for switching LED lighting, starting virtual objects.



For the digital output objects, pre-defined templates for the ON_OFF widget are defined. To add the ON_OFF widget with the desired digital output object, drag the DOUT object from the list of objects to the interface page:



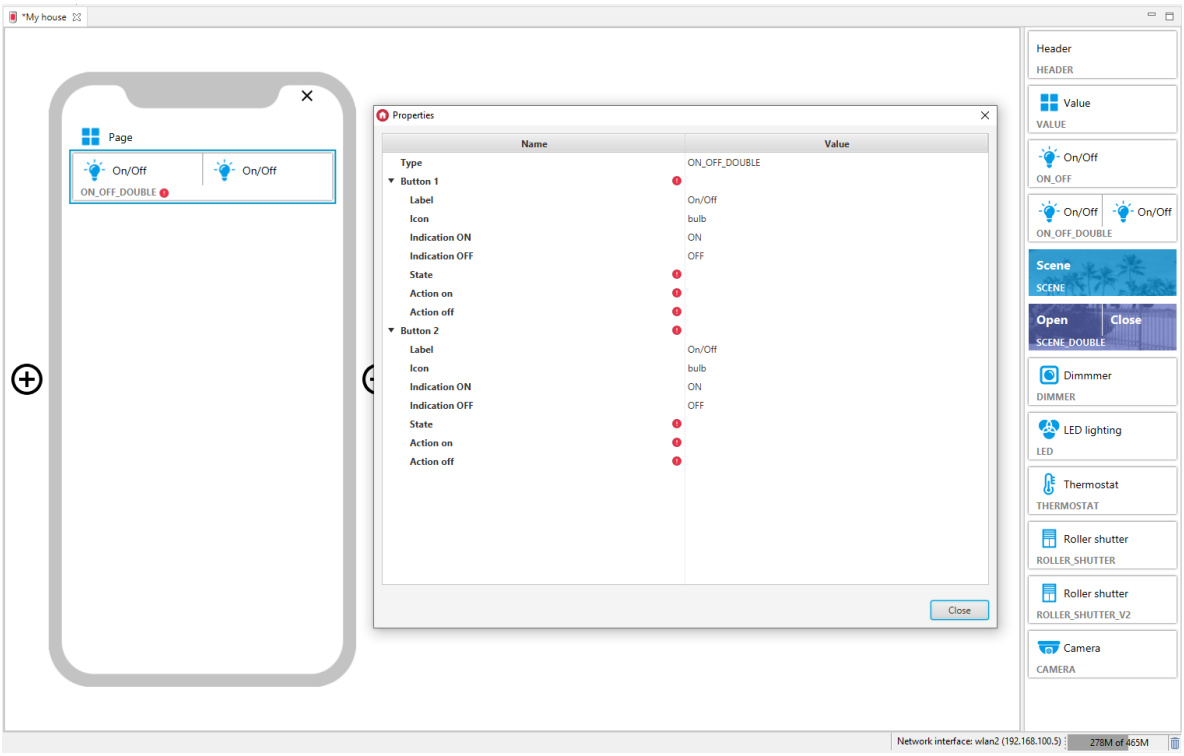
Filled ON_OFF widget:



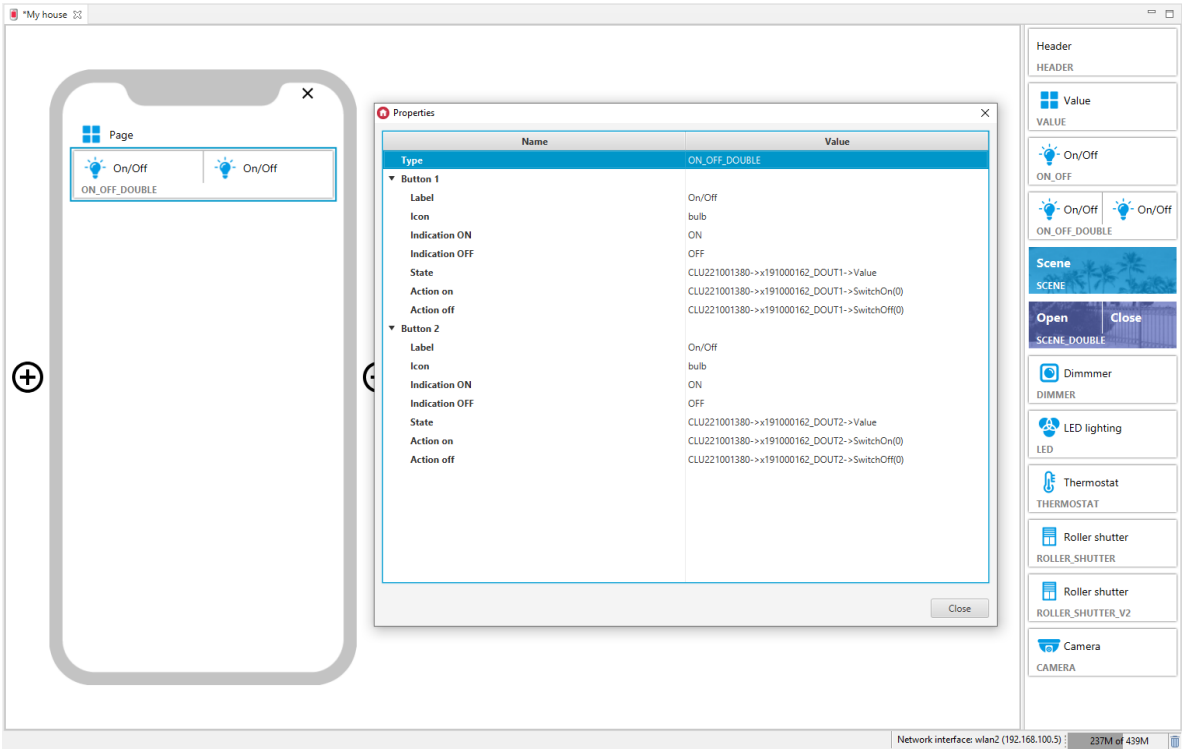
3.6. On/Off Double (ON_OFF_DOUBLE)

> **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.

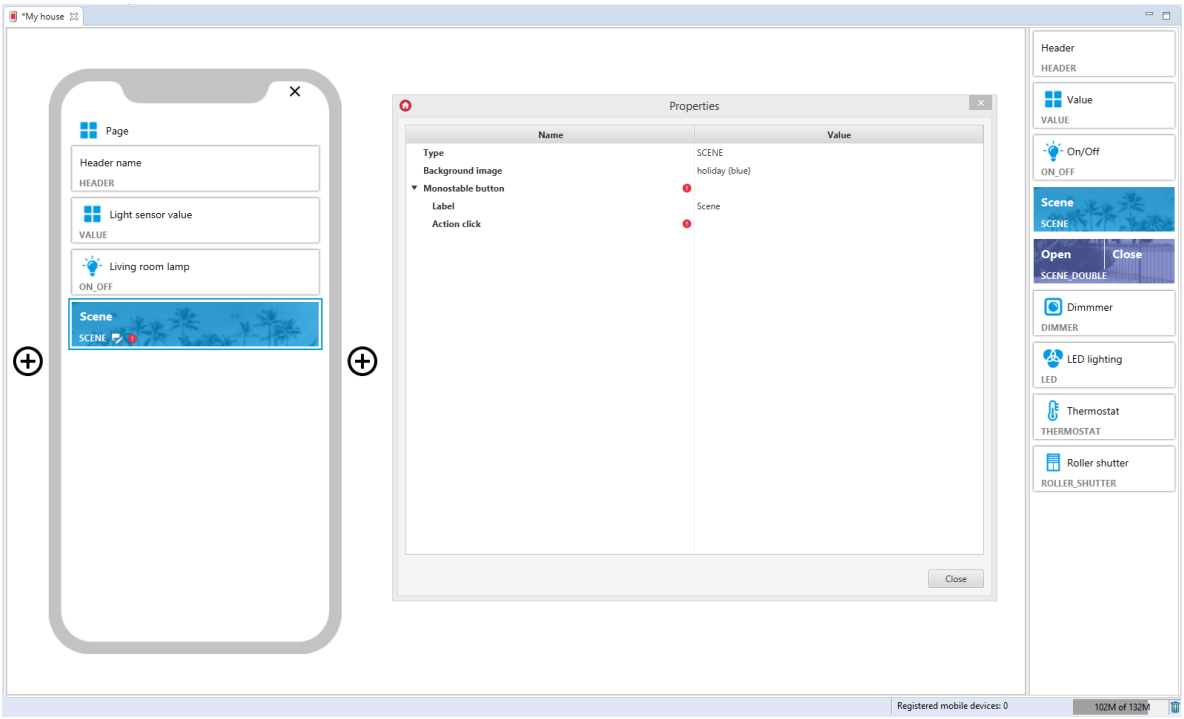


Filled ON_OFF_DOUBLE widget:

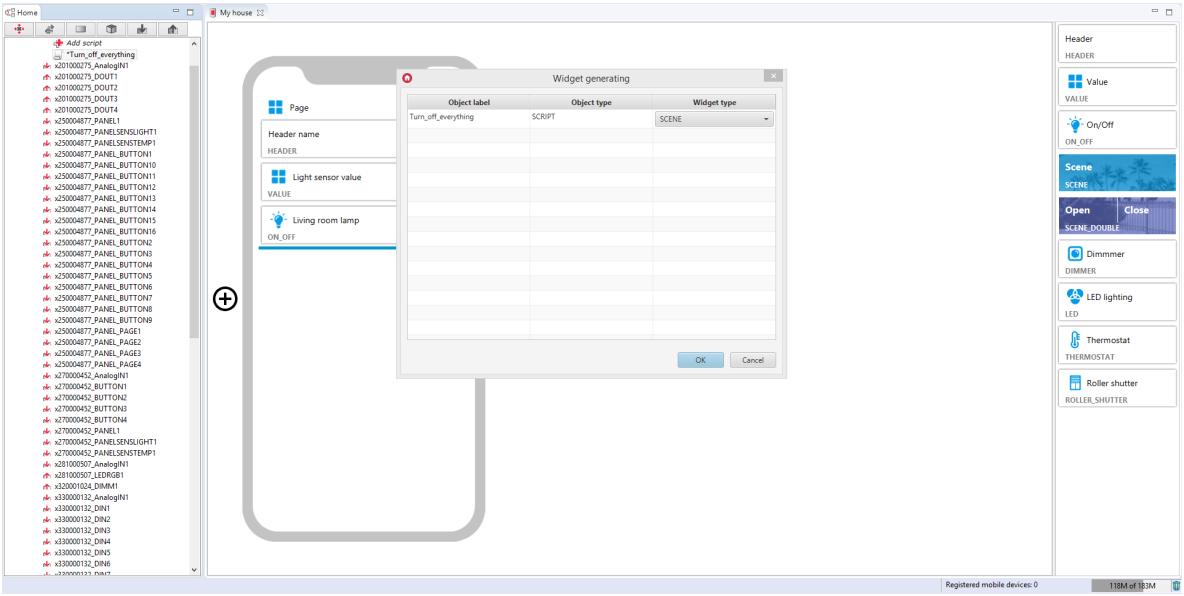


3.7. Scene (SCENE)

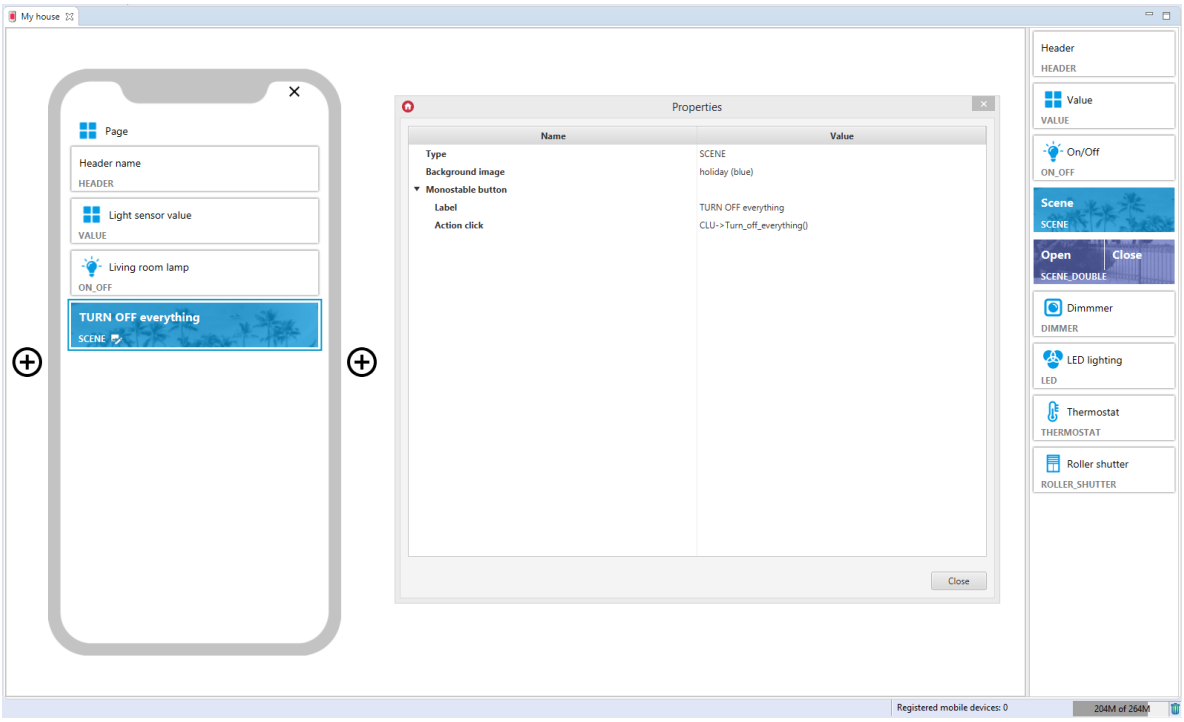
The widget is dedicated to calling created scripts.



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:

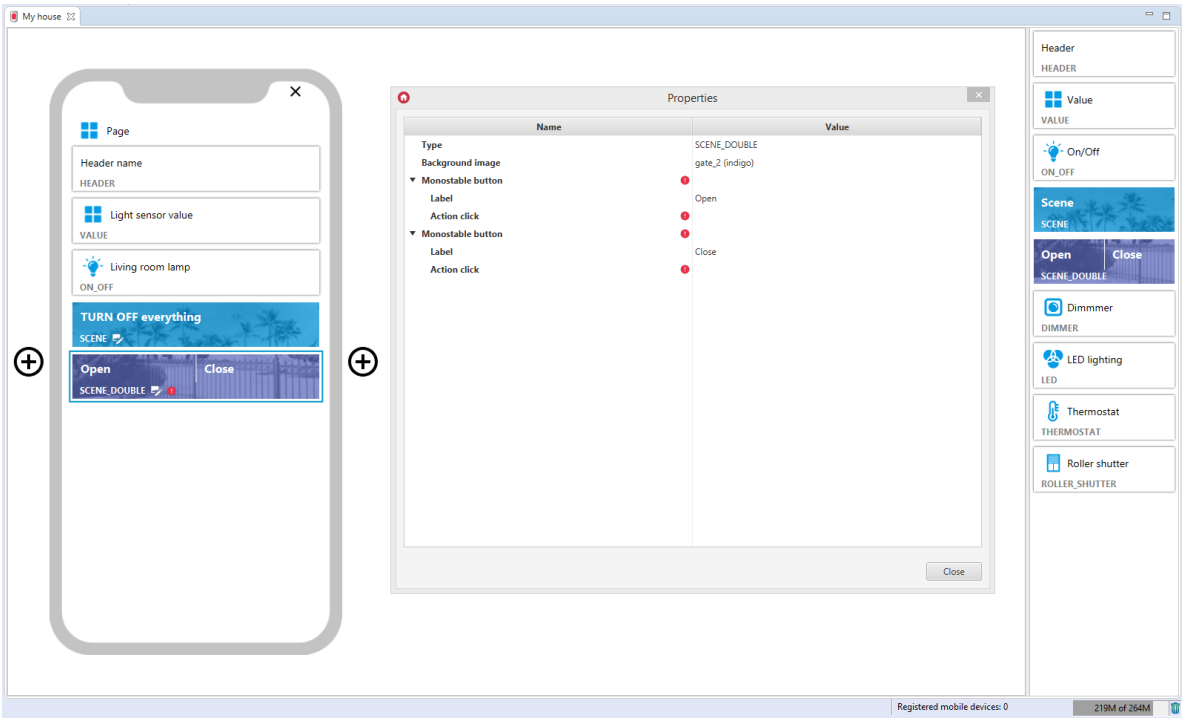


Filled SCENE:

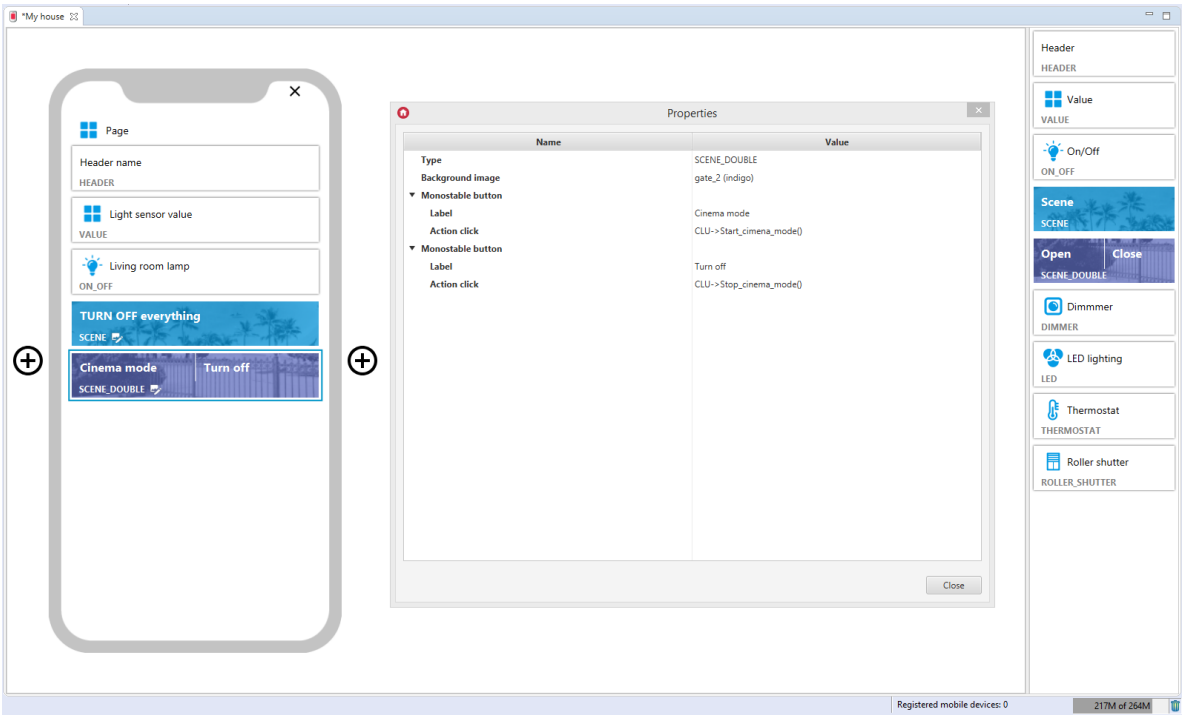


3.8. Scene Double (SCENE_DOUBLE)

The widget is used to call scripts (object methods). In contrast to the widget SCENE has 2 buttons for actions.



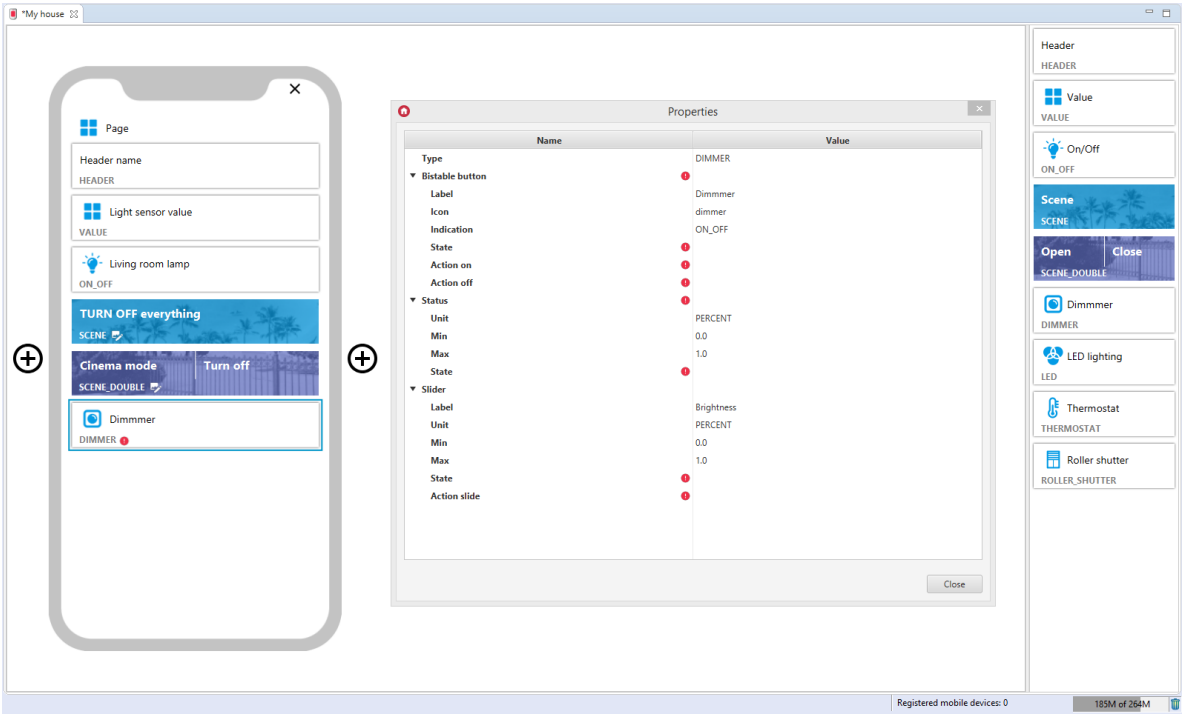
Filled SCENE_DOUBLE widget:



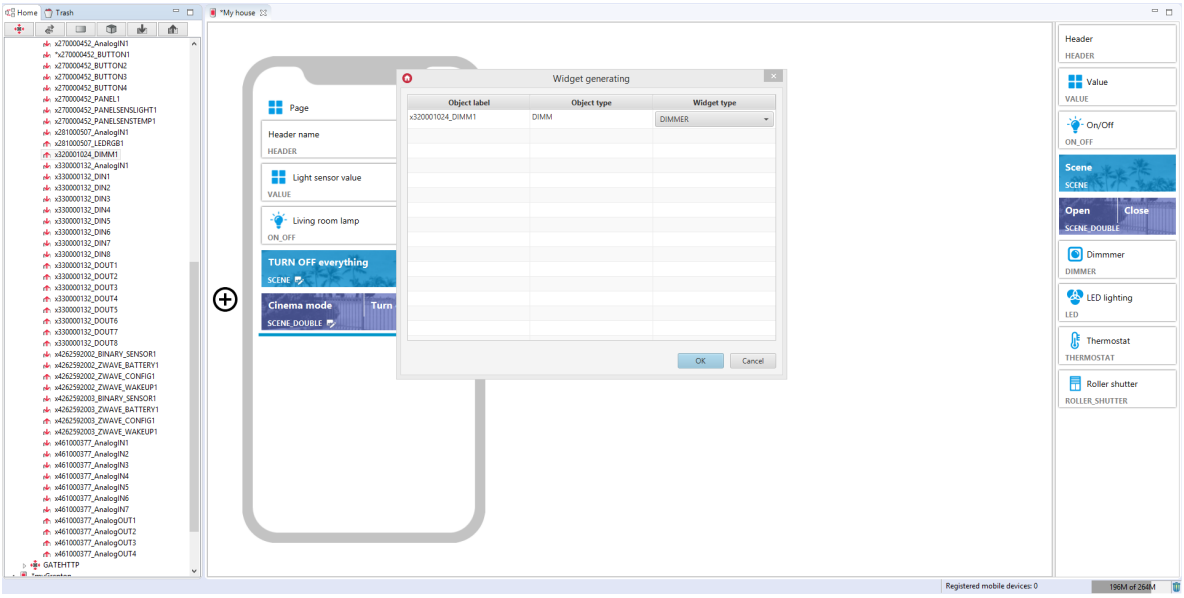
3.9. Dimmer (DIMMER)

> **Note!**
> From the version of Object Manager 1.7.0, the DIMMER widget and the possibility of using it as a pre-defined template will not be available. It is replaced with the DIMMER_V2 widget.
>
> DIMMER widgets included in projects created on previous versions of Object Manager will still be properly supported and displayed in the myGrenton application.

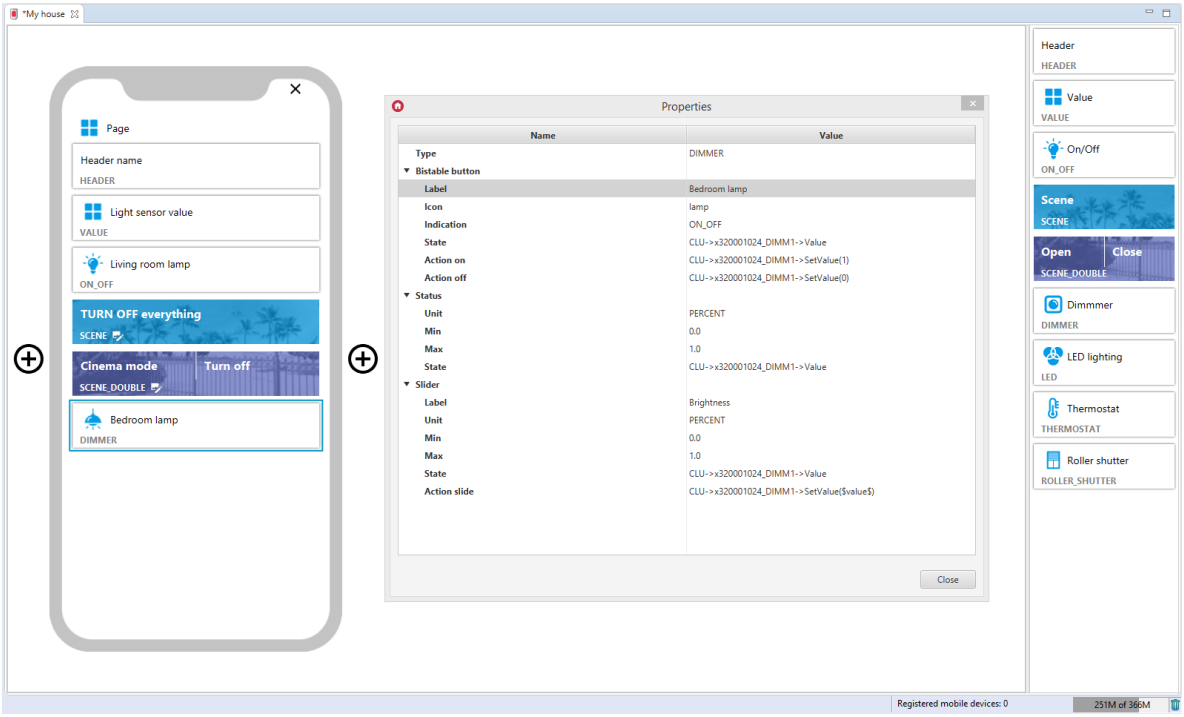
Widget dedicated for DIMMER and LEDRGB modules. When dragging the widget from the tab on the right side of the screen, it should be filled in with the values in the `State`, `Action on`, `Action off`, `Unit`, `Min`, `Max`, `Action slide` fields. This widget has a slider, which allows it to be controlled in a given range.



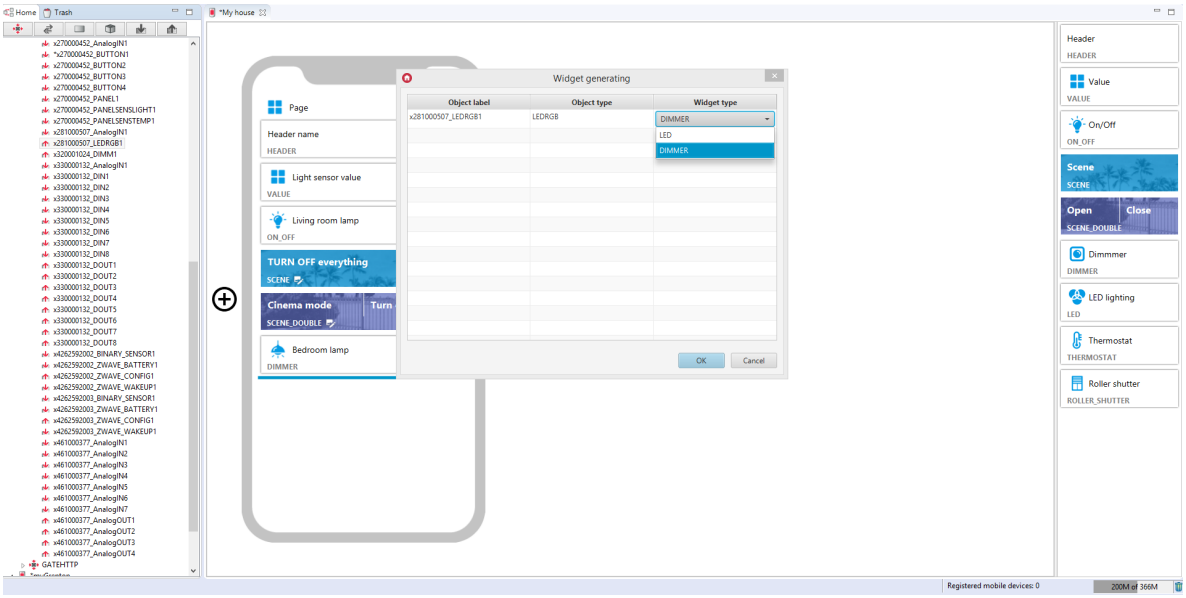
For the DIMM object, pre-defined templates for the DIMMER widget are defined. To add a DIMMER widget with a pre-made template, drag a DIMM from the list of objects to the interface page:



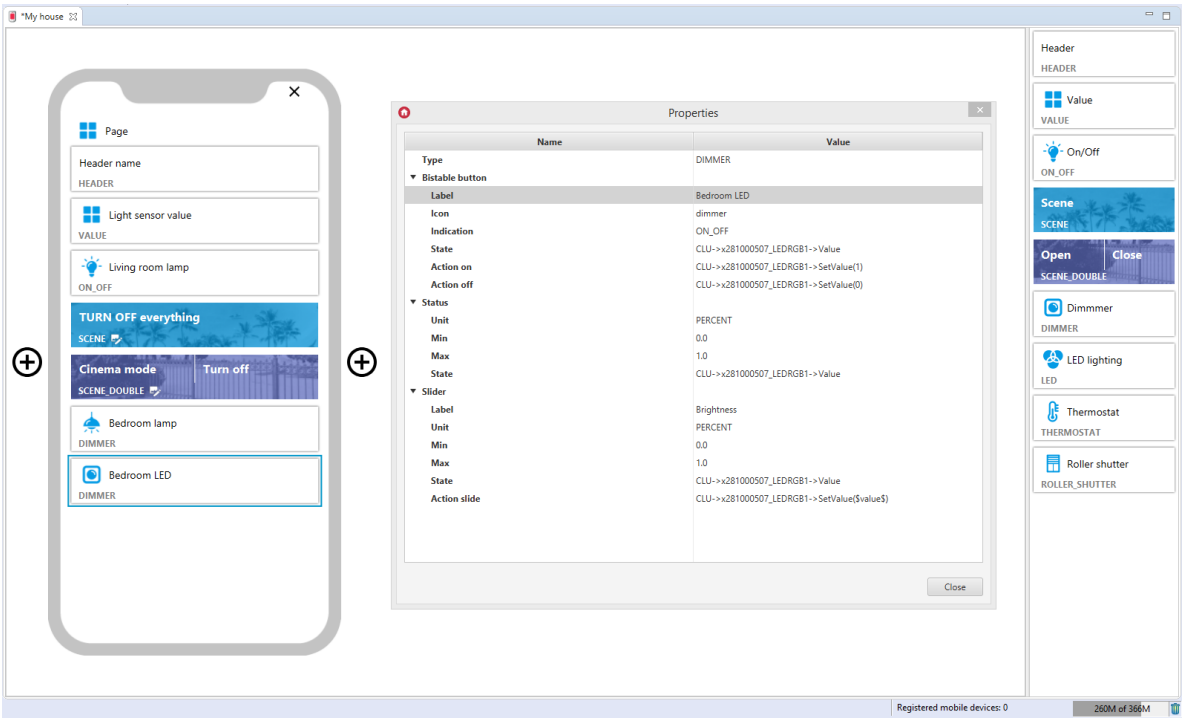
Filled DIMMER widget:



The DIMMER widget has a ready template also for the LEDRGB object:



Widget created:

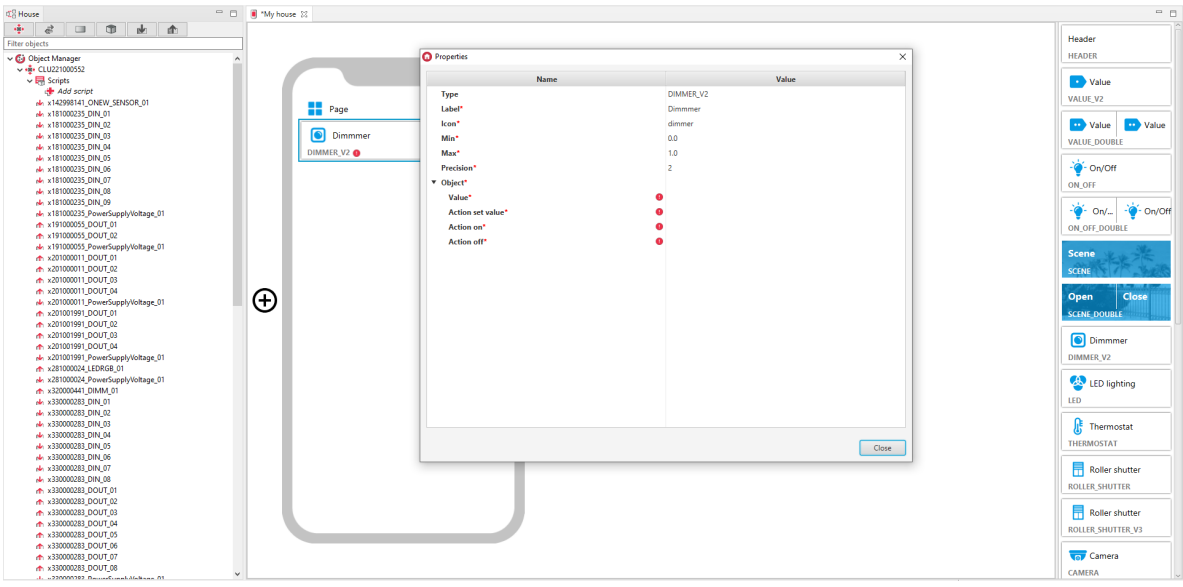


3.10. Dimmer v2 (DIMMER_V2)

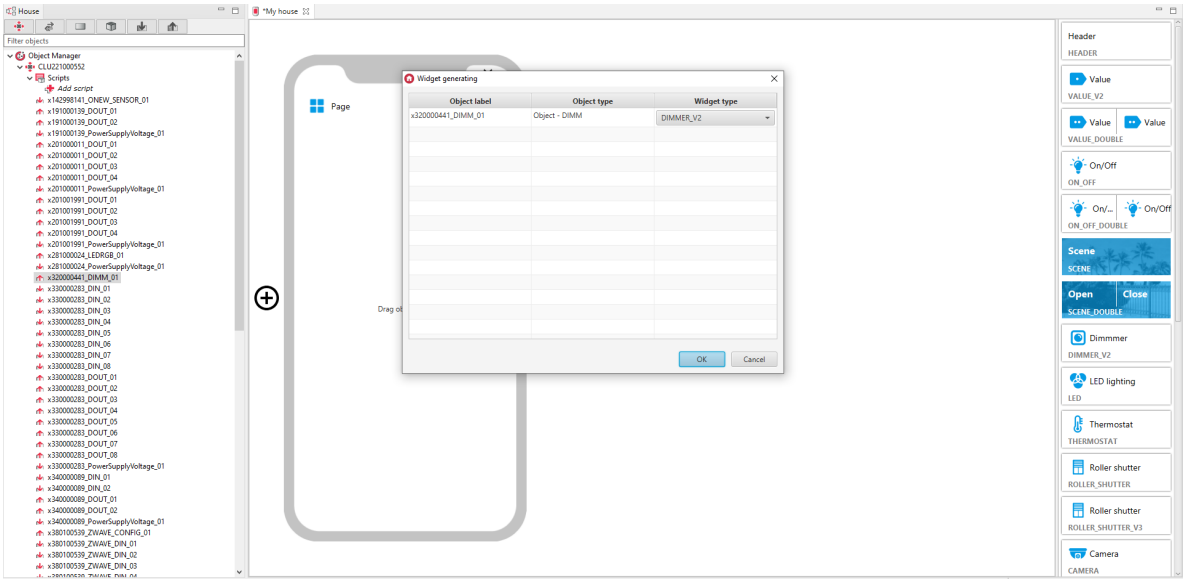
> **Note!**
> The DIMMER_V2 widget is available for Object Manager version 1.7.0 or higher, and for myGrenton application version 1.5.0 or higher (Android) and version 1.9.0 or higher (iOS).

Widget dedicated for DIMMER and LEDRGBW modules, enables smooth lighting control. The DIMMER_V2 widget contains:

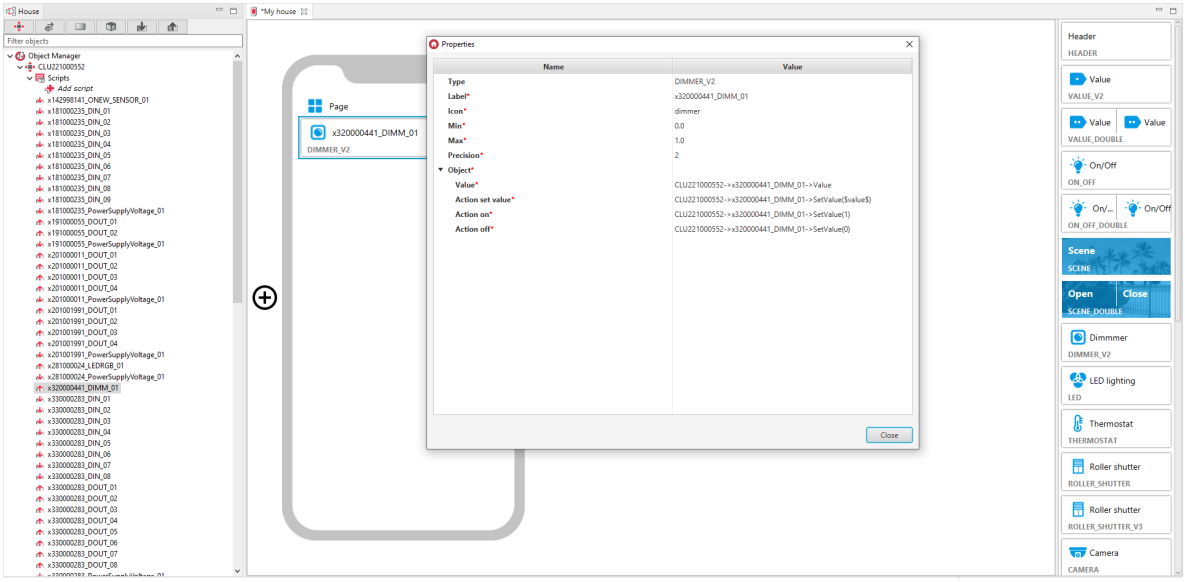
- value - the current output value expressed as a percentage, displayed in the right part of the widget (calculated on the basis of the set `Min`, `Max` properties),
- actions on / off - actions triggered when clicking on the widget to activate / deactivate the output,
- output control slider - works in percentage mode, the output value is set on the basis of the given range (`Min`, `Max` properties) and precision (the `Precision` property determines the number of decimal places of the set value).



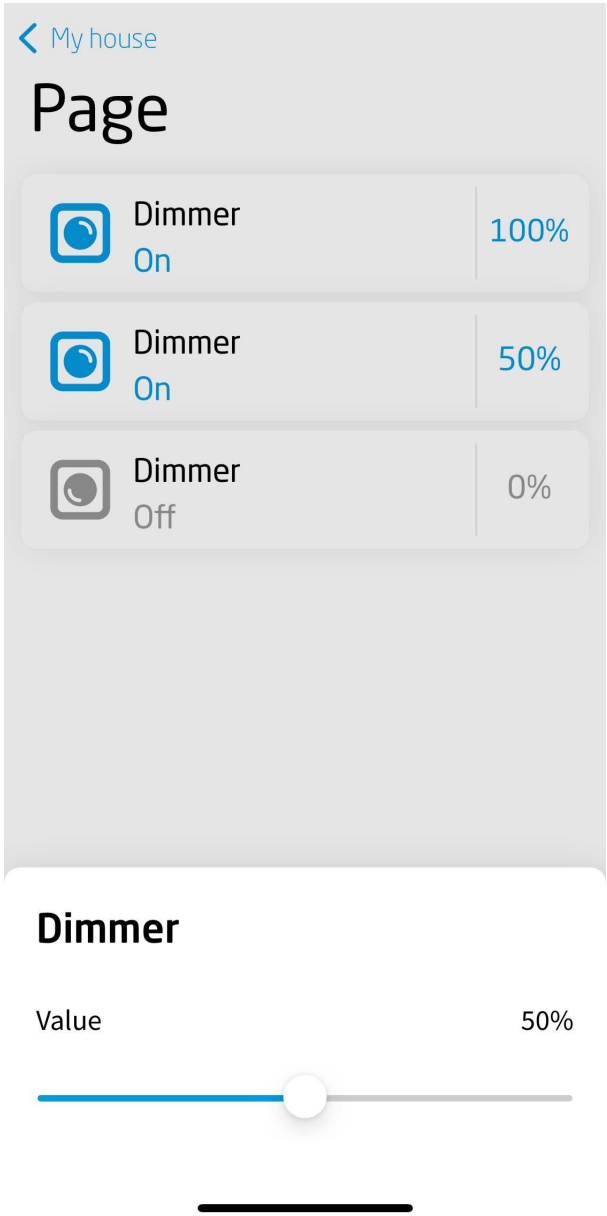
For DIMM, LEDRGB, LEDRGBW objects, pre-defined templates for the DIMMER_V2 widget are defined. To add a DIMMER_V2 widget with a pre-made template, drag an object from the list of objects to the interface page:



Filled DIMMER_V2 widget:

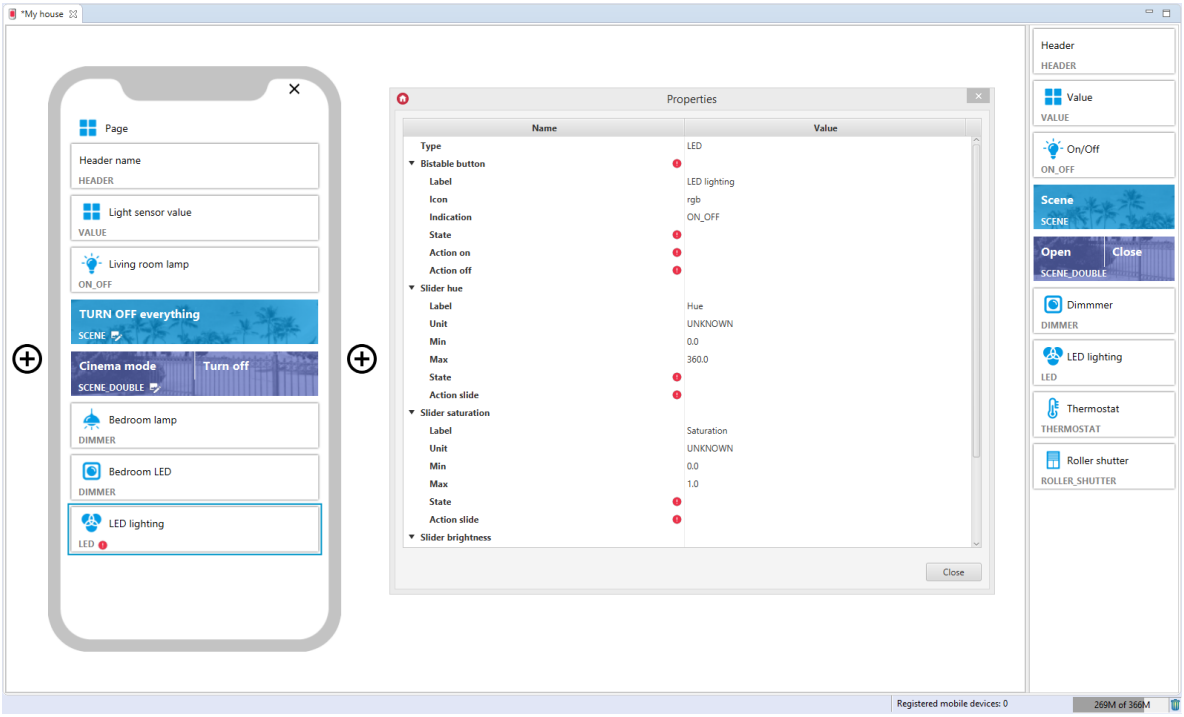


Widget appearance in the myGrenton application:

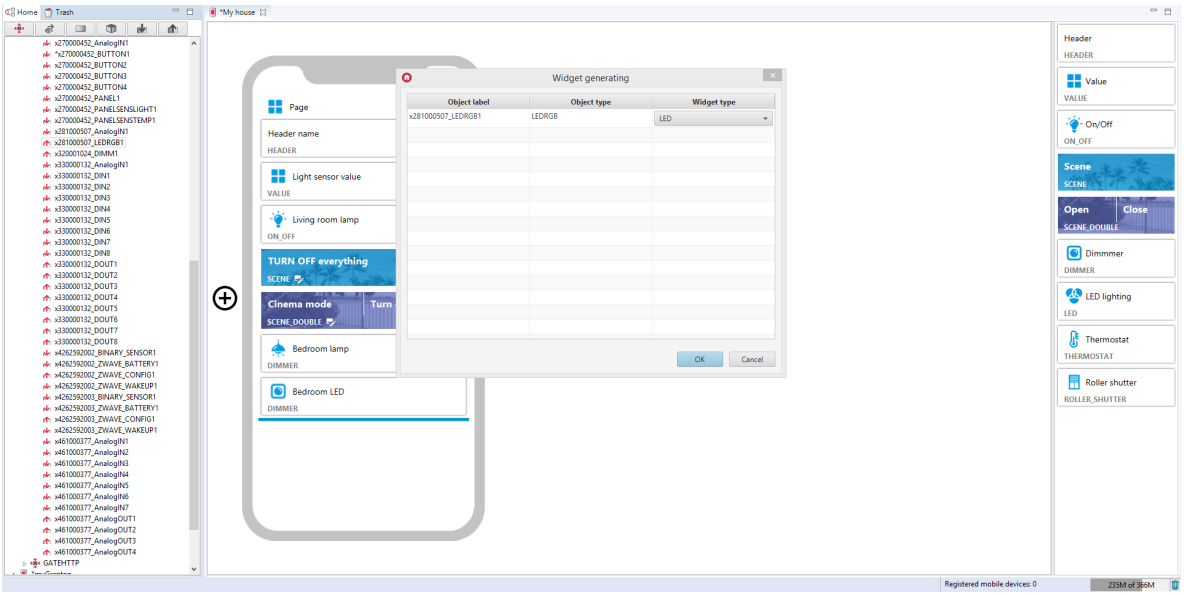


3.11. LED Lighting (LED)

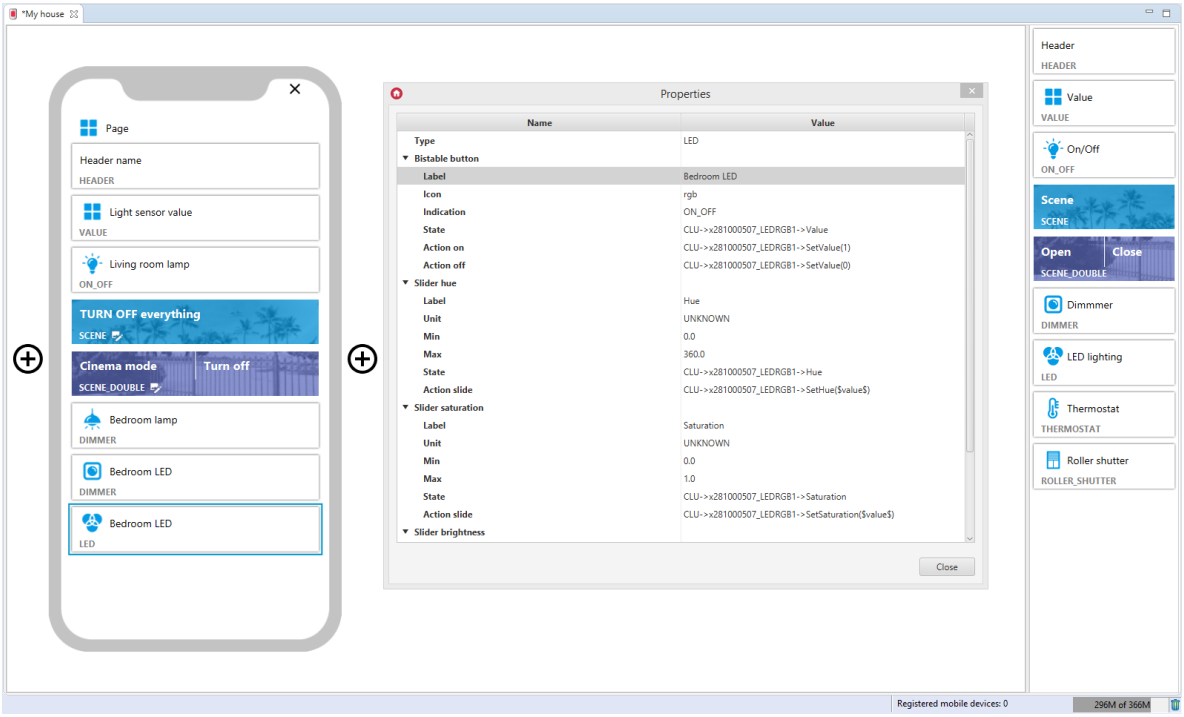
Widget dedicated to LED lighting. It has one bistable on/off switch and 3 sliders: Color control slider, Saturation control slider, Brightness control slider.



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:



Filled LED widget:

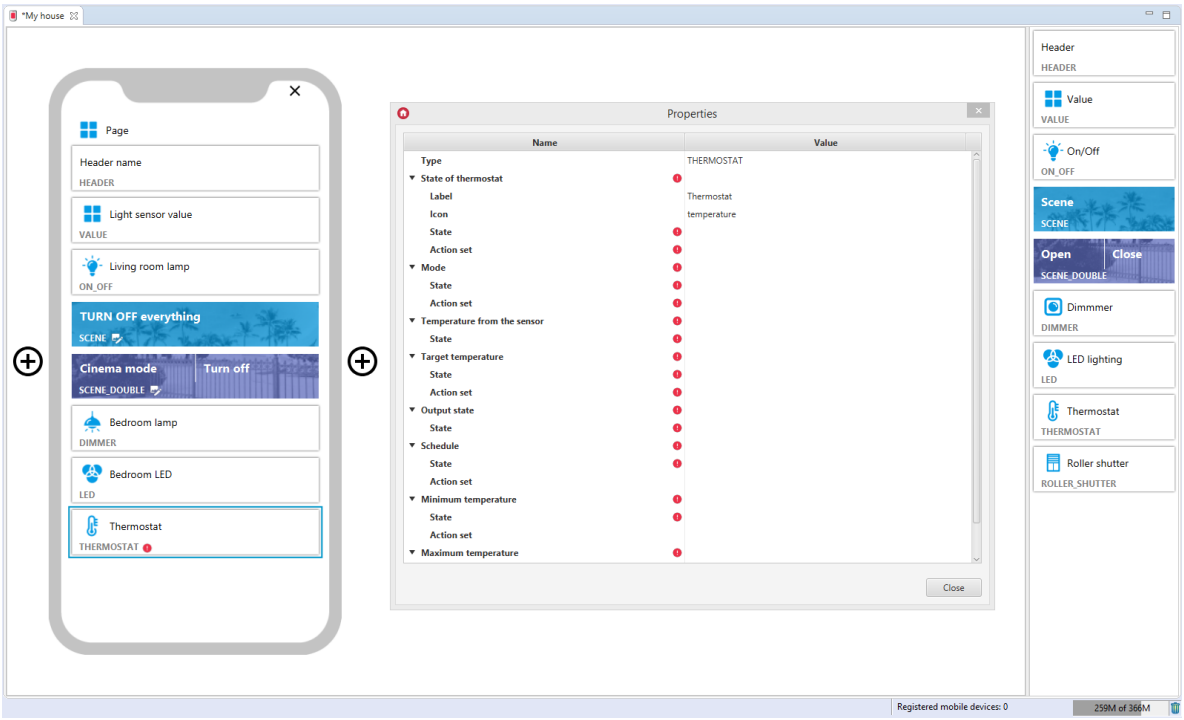


3.12. Thermostat (THERMOSTAT)

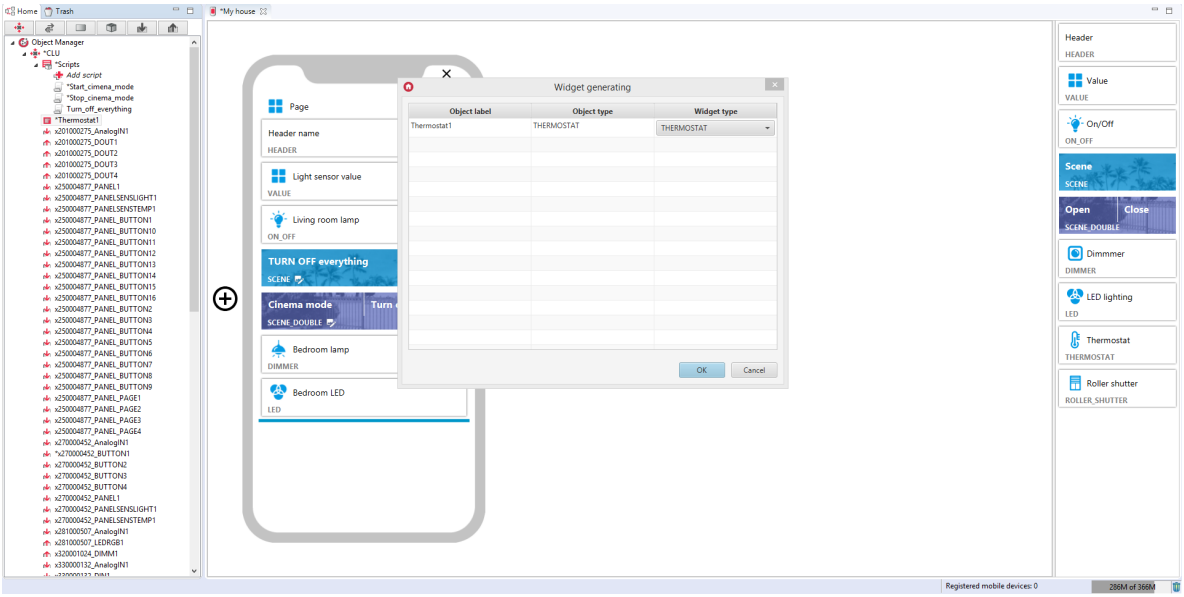
Note!
Widget is supported for thermostats created in **CLU 2.0!**

Note!
From the version of Object Manager 1.11.0, the THERMOSTAT widget and the possibility of using it as a pre-defined template will not be available. It is replaced with the THERMOSTAT_V2 widget. THERMOSTAT widgets included in projects created on previous versions of Object Manager will still be properly supported and displayed in the myGrenton application.

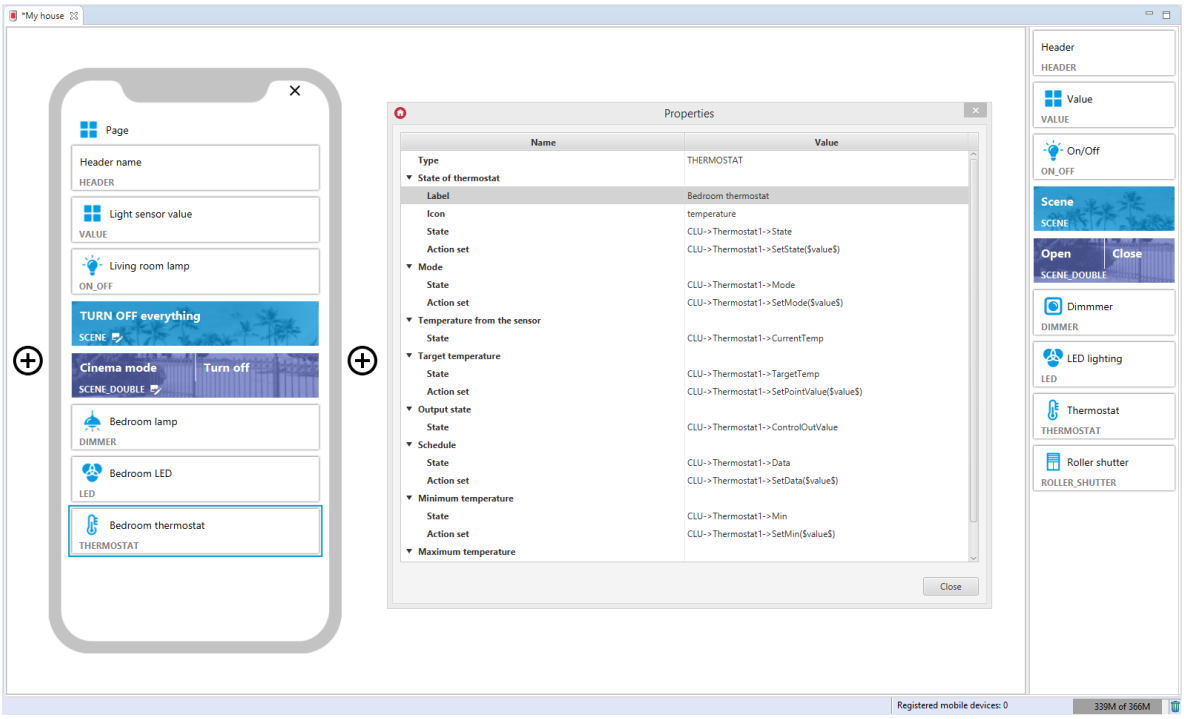
Widget dedicated for virtual objects of the thermostat type. In the case when we draw from the list of objects already defined thermostat to the interface, the created widget is completed on the basis of given thermostat input and output features.



For thermostats, ready-made templates for the THERMOSTAT widget are defined. To add a THERMOSTAT widget with a ready template, drag the virtual thermostat from the list of objects to the interface page:



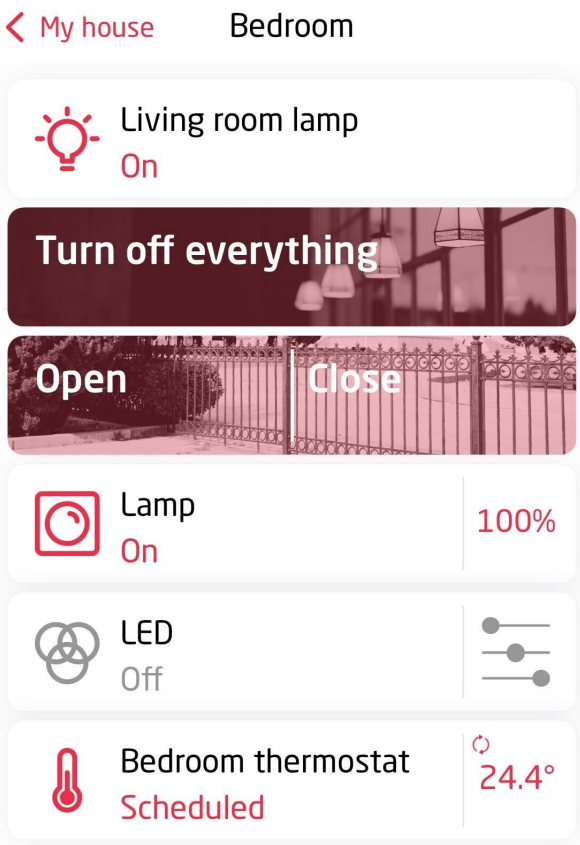
Filled THERMOSTAT widget:



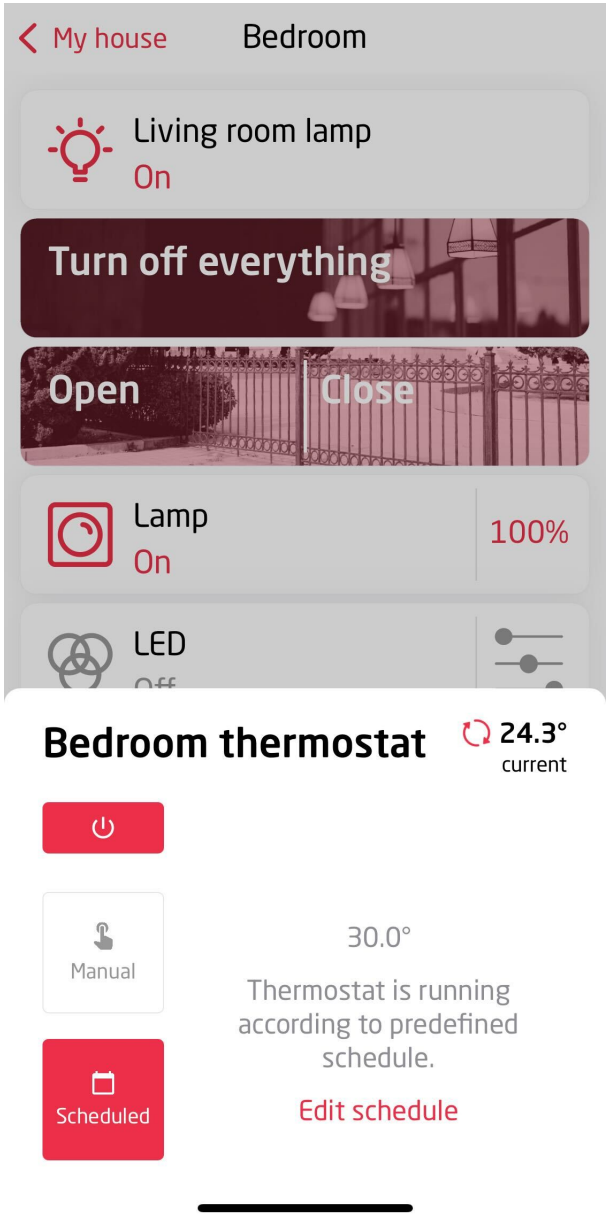
A. Schedule configuration in the application

> **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).

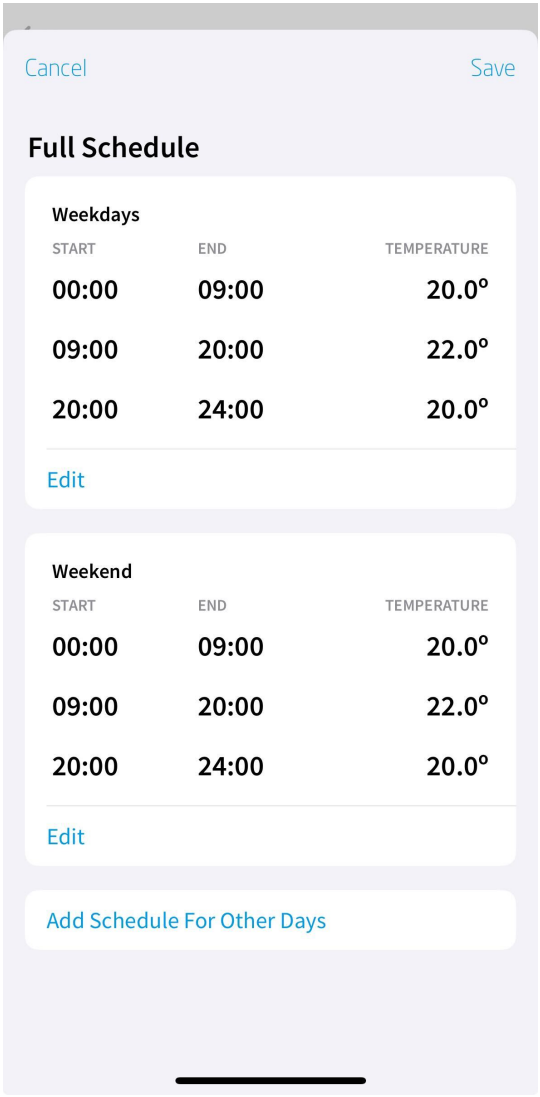
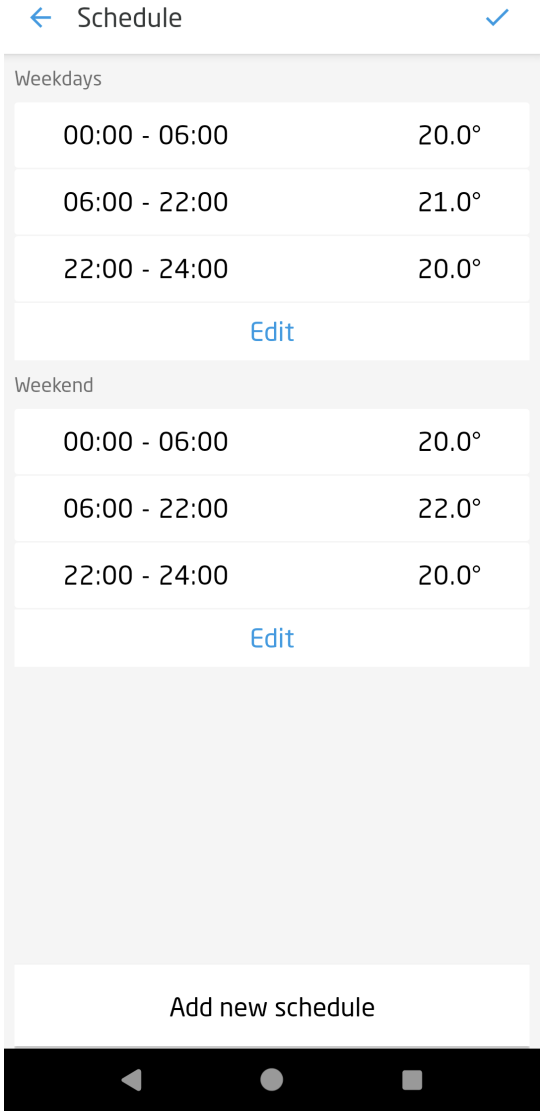
In the myGrenton application, you can edit the thermostat schedule. To do this, click on the widget temperature field:



Then select the schedule mode and the `Edit schedule` option:

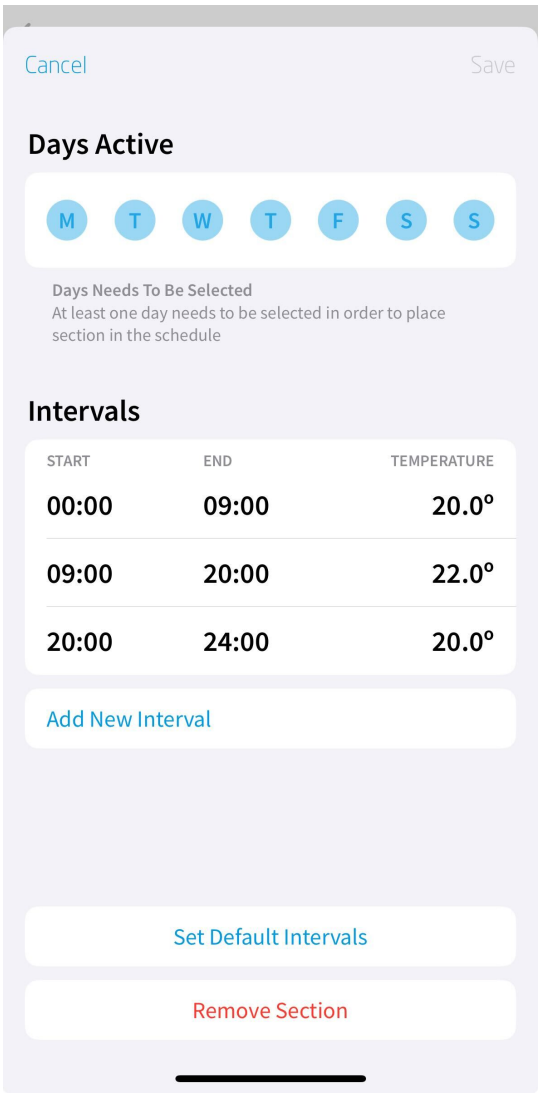
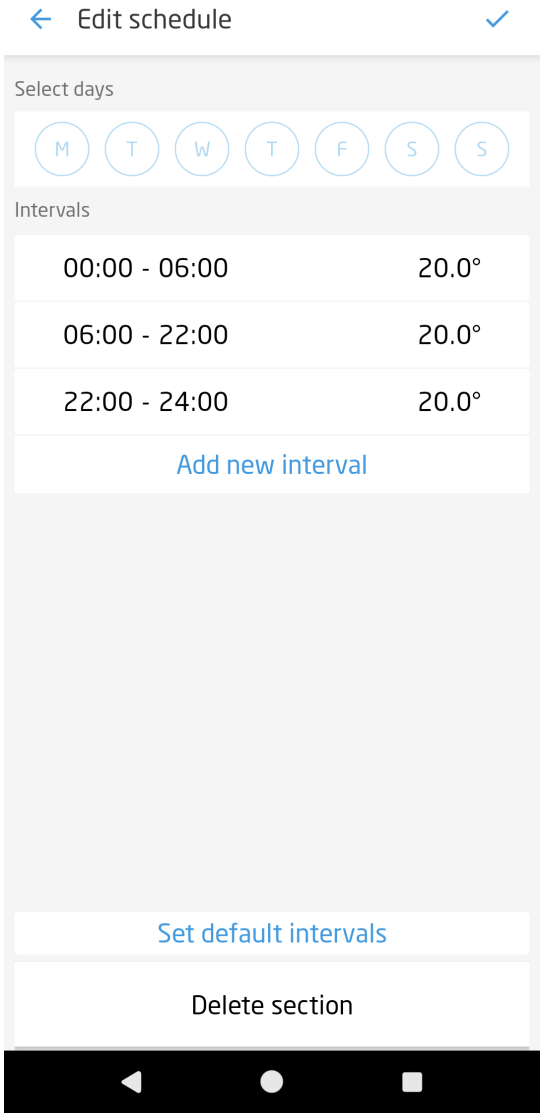


The window will display the schedule downloaded from the CLU. You can edit this schedule or add new schedules for each day of the week:

iOS	Android
	

Adding a new schedule

After selecting `Add Schedule For Other Days` (iOS) or `Add new schedule` (Android), the adding schedule window will open:

iOS	Android
	

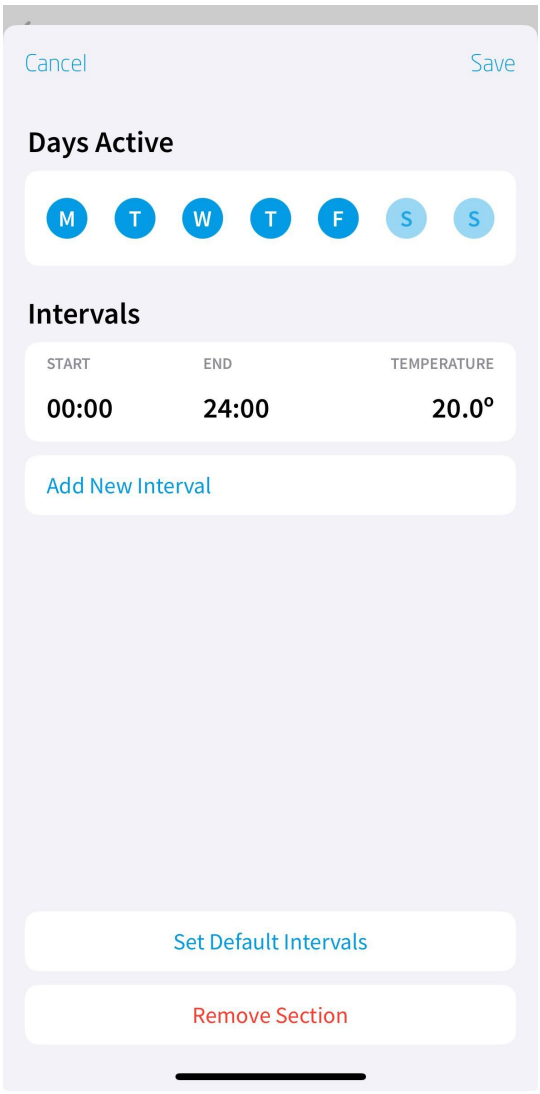
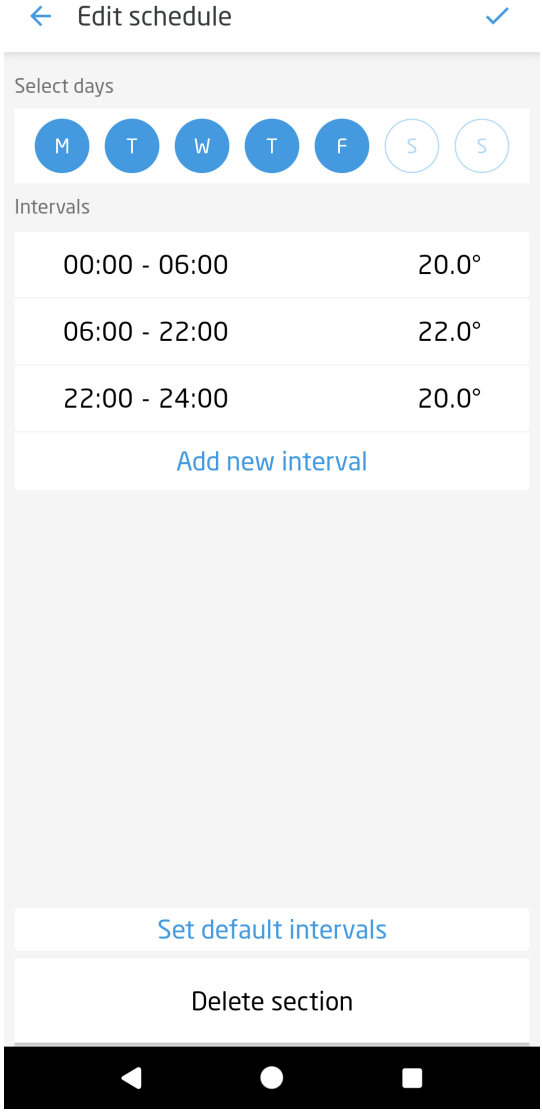
Then:

- 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:

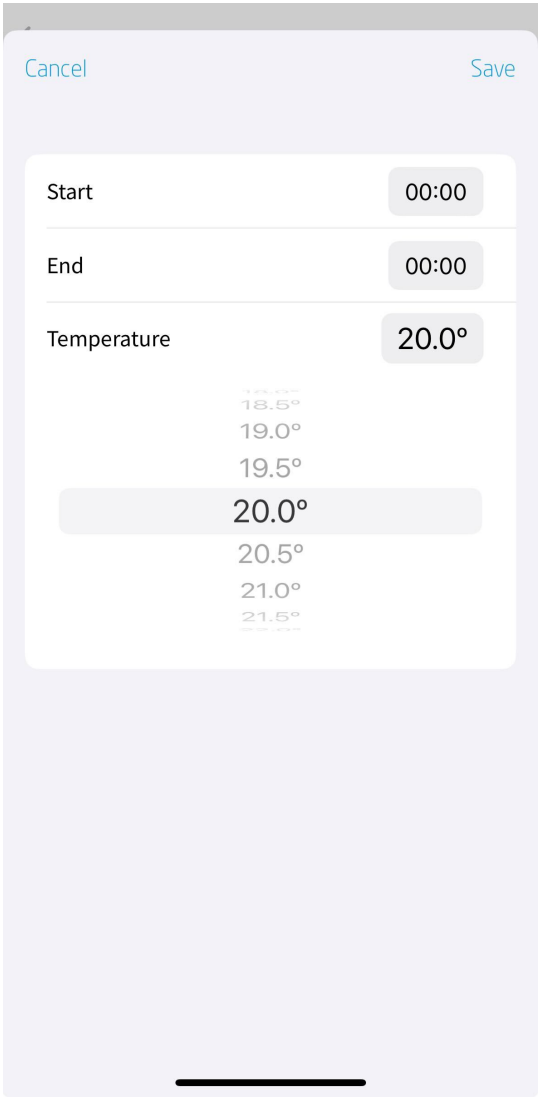
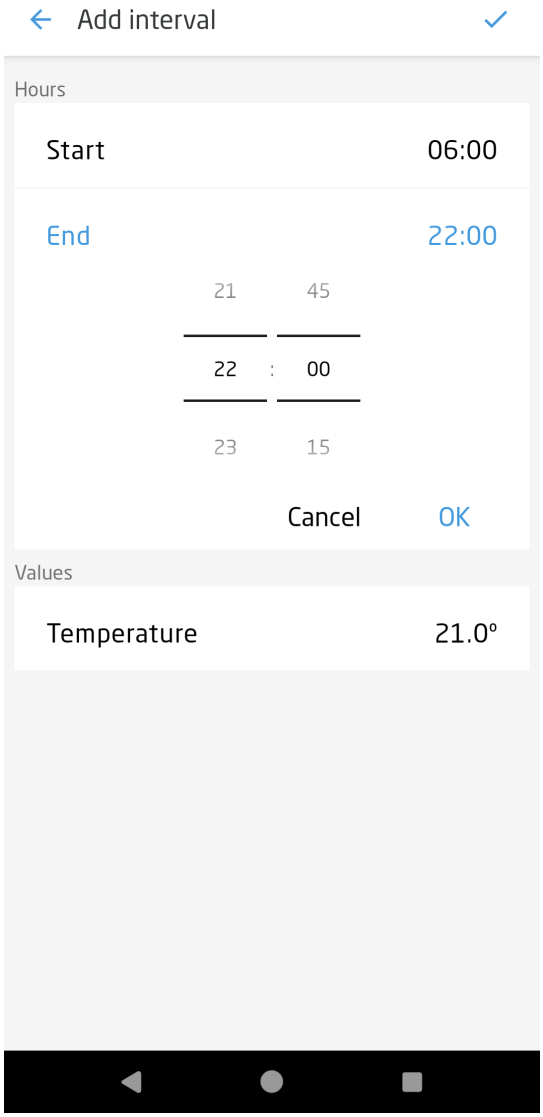
iOS	Android
	

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:

iOS	Android
	

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`.

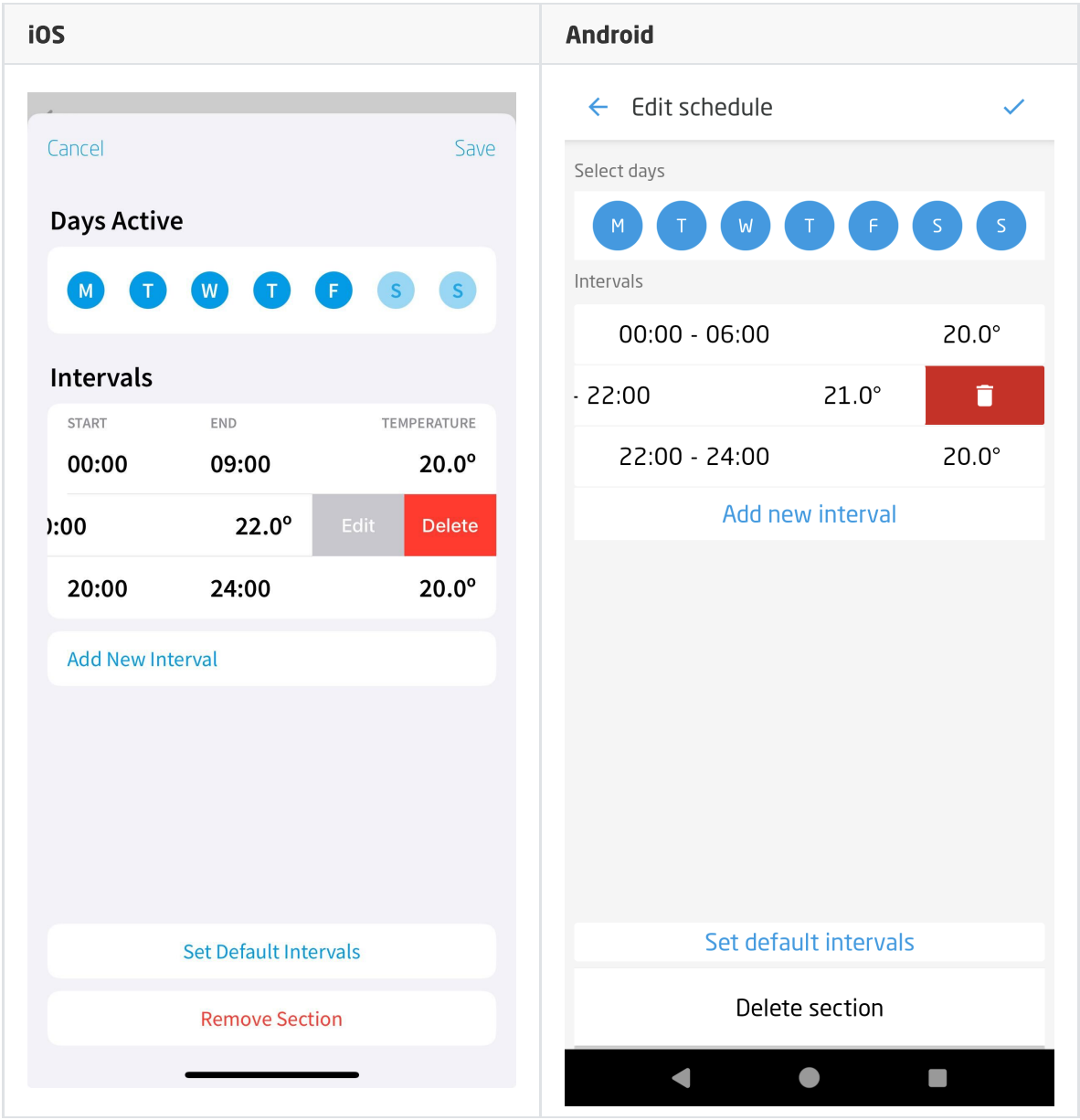
The application will automatically add intervals for unaccounted hours to correctly fill in the values for the whole day.

> **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`.

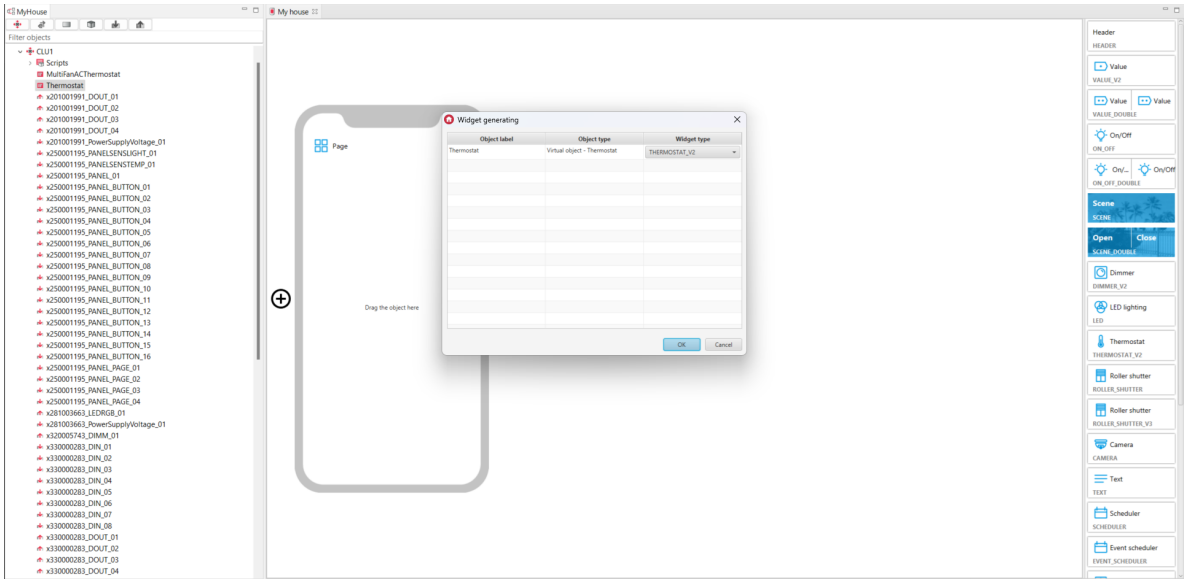


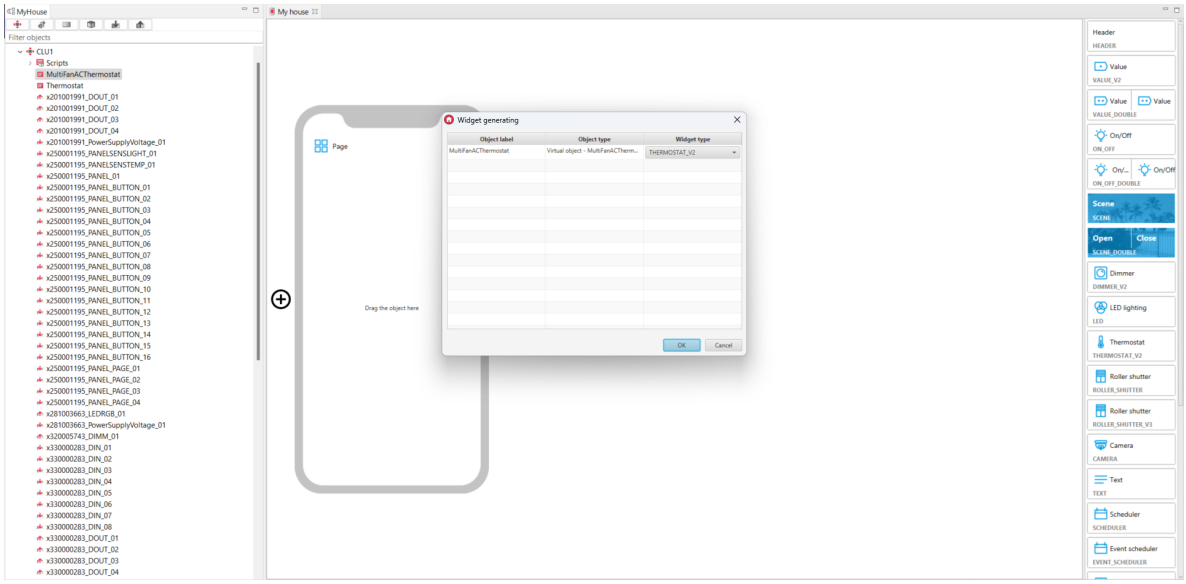
3.13. Thermostat v2 (THERMOSTAT_V2)

> **Note!**
> The THERMOSTAT_V2 widget is available for Object Manager version 1.11.0 or higher, and for myGrenton application version 1.11.9 or higher (Android) and version 1.14.0 or higher (iOS).

Widget dedicated for virtual objects of the Thermostat and MultiFanACThermostat type.

For thermostats, ready-made templates for the THERMOSTAT_V2 widget are defined. To add the THERMOSTAT_V2 widget with a ready-made template, drag the Thermostat or MultiFanACThermostat virtual object from the list of objects to the interface page:





Configured THERMOSTAT_V2 widget for the Thermostat virtual object::

Properties

Name	Value
Type	THERMOSTAT_V2
Label*	Thermostat
Icon*	temperature
Number of fan speeds*	0
▼ Object*	
State*	CLU1->Thermostat->State
Set state*	CLU1->Thermostat->SetState($value$)
Operating mode*	CLU1->Thermostat->Mode
Set operating mode*	CLU1->Thermostat->SetMode($value$)
Current temperature*	CLU1->Thermostat->CurrentTemp
Target temperature*	CLU1->Thermostat->TargetTemp
Set target temperature*	CLU1->Thermostat->SetPointValue($value$)
Control out value*	CLU1->Thermostat->ControlOutValue
Schedule data*	CLU1->Thermostat->Data
Set the schedule data*	CLU1->Thermostat->SetData($value$)
Minimum temperature*	CLU1->Thermostat->Min
Set the minimum temperature*	CLU1->Thermostat->SetMin($value$)
Maximum temperature*	CLU1->Thermostat->Max
Set the maximum temperature*	CLU1->Thermostat->SetMax($value$)
Control direction	
Set the control direction	
Fan mode	
Set the fan mode	

Close

Configured THERMOSTAT_V2 widget for the MultiFanACThermostat virtual object:

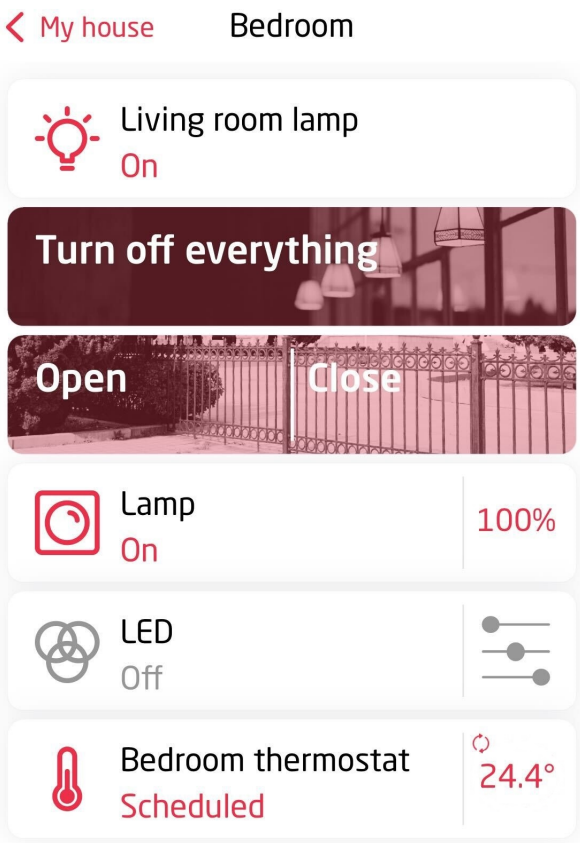
Properties

Name	Value
Type	THERMOSTAT_V2
Label*	MultiFanACThermostat
Icon*	temperature
Number of fan speeds*	3
▼ Object*	
State*	CLU1->MultiFanACThermostat->State
Set state*	CLU1->MultiFanACThermostat->SetState($value$)
Operating mode*	CLU1->MultiFanACThermostat->Mode
Set operating mode*	CLU1->MultiFanACThermostat->SetMode($value$)
Current temperature*	CLU1->MultiFanACThermostat->CurrentTemp
Target temperature*	CLU1->MultiFanACThermostat->TargetTemp
Set target temperature*	CLU1->MultiFanACThermostat->SetPointValue($value$)
Control out value*	CLU1->MultiFanACThermostat->ControlOutValue
Schedule data*	CLU1->MultiFanACThermostat->Data
Set the schedule data*	CLU1->MultiFanACThermostat->SetData($value$)
Minimum temperature*	CLU1->MultiFanACThermostat->Min
Set the minimum temperature*	CLU1->MultiFanACThermostat->SetMin($value$)
Maximum temperature*	CLU1->MultiFanACThermostat->Max
Set the maximum temperature*	CLU1->MultiFanACThermostat->SetMax($value$)
Control direction	CLU1->MultiFanACThermostat->ControlDirection
Set the control direction	CLU1->MultiFanACThermostat->SetControlDirection($value$)
Fan mode	CLU1->MultiFanACThermostat->FanMode
Set the fan mode	CLU1->MultiFanACThermostat->SetFanMode($value$)

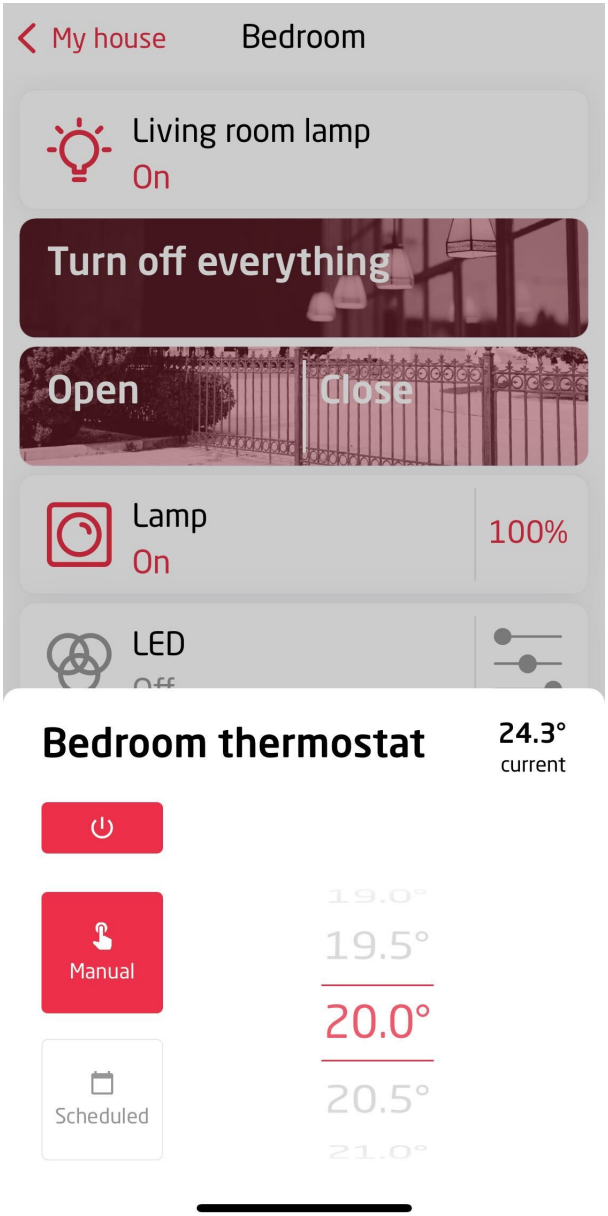
Close

Widget appearance in the myGrenton application:

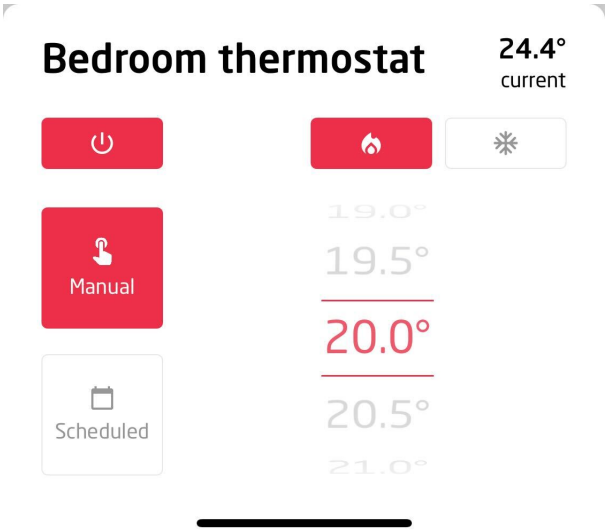
- page view:



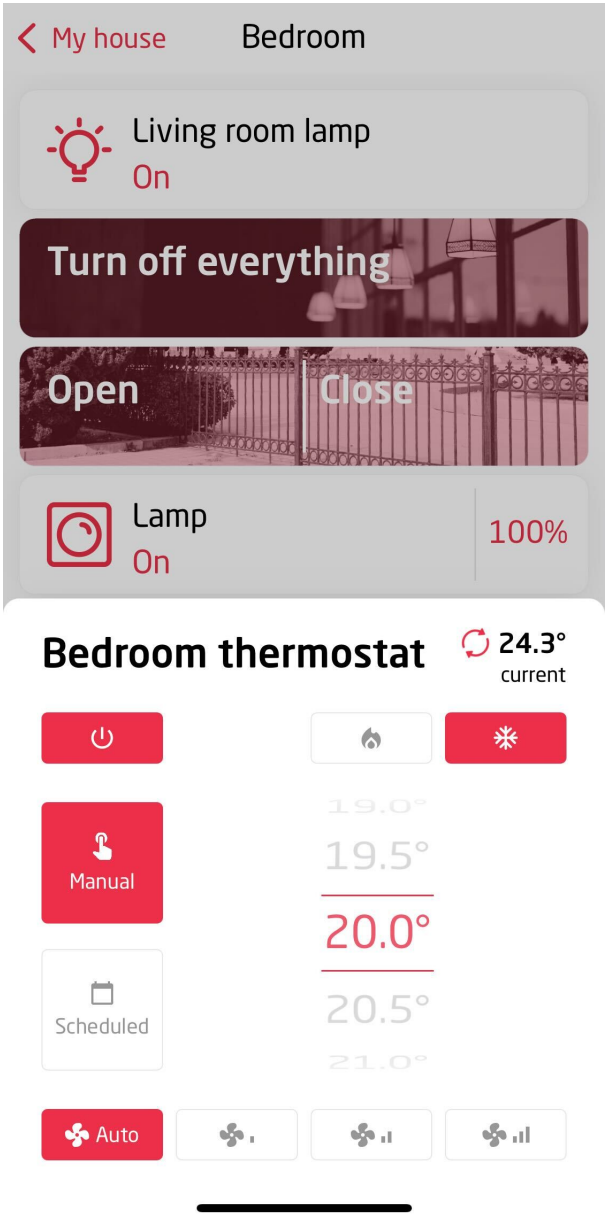
- Thermostat virtual object:



- Thermostat virtual object after entering the `Control direction` and `Set the control direction` properties:



- MultiFanACThermostat virtual object:



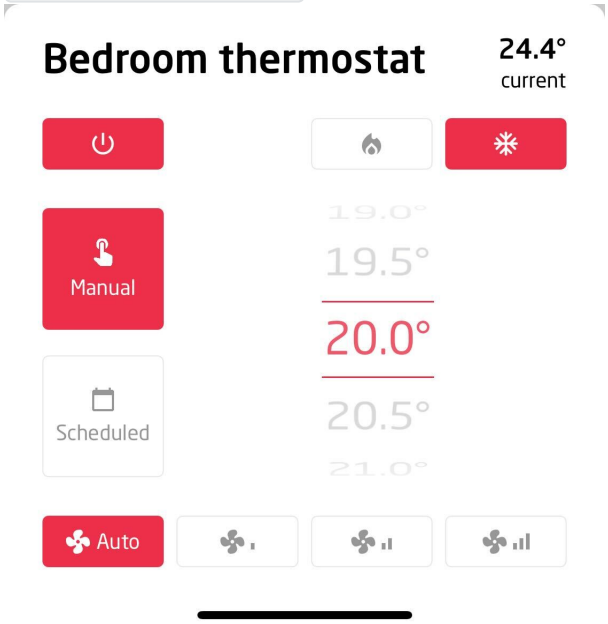
A. Schedule configuration in the application

Editing the schedule in the application is done in the same way as for the THERMOSTAT widget.

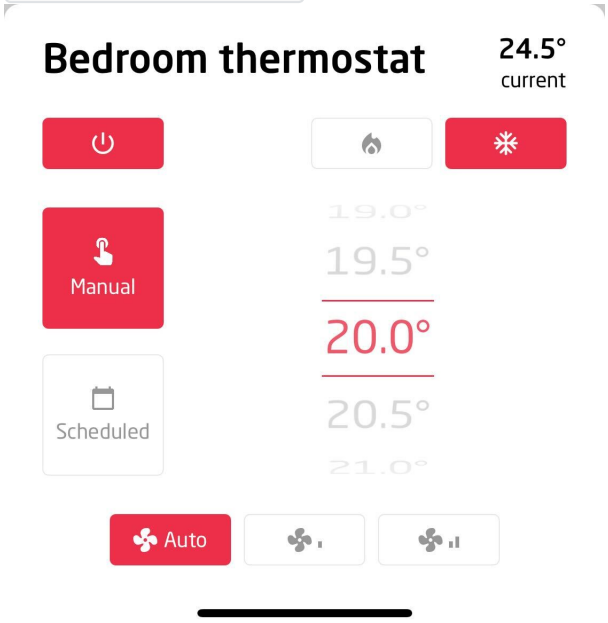
B. Fan mode configuration for the MultiFanACThermostat virtual object

By changing the `Number of fan speeds` property in the widget configuration, it is possible to display buttons for fan control:

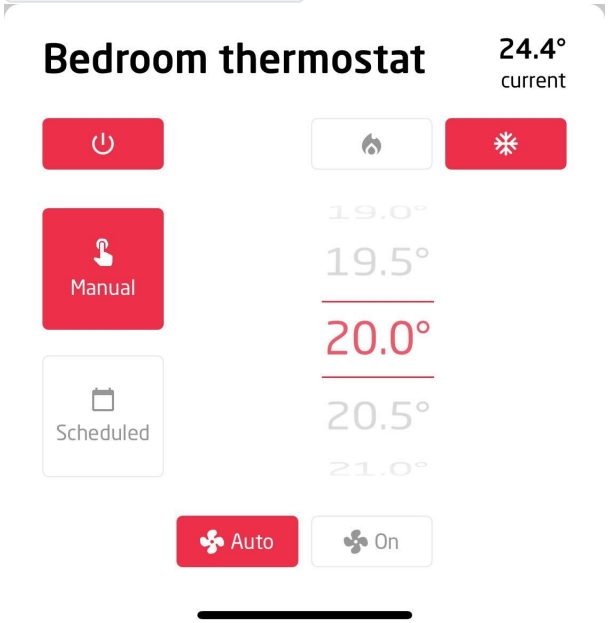
- `Number of fan speeds` = 3 - available buttons are Auto, Low, Medium, High:



- `Number of fan speeds` = 2 - available buttons are Auto, Low, Medium:

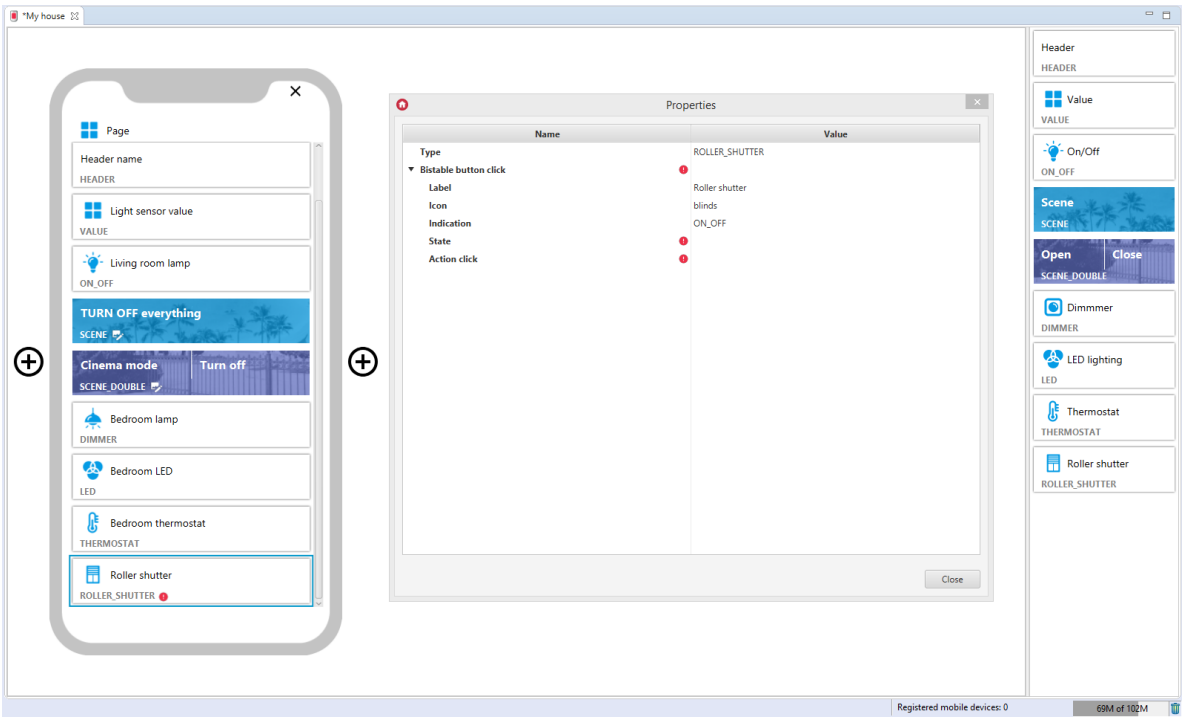


- `Number of fan speeds` = 1 - available buttons are Auto, On:

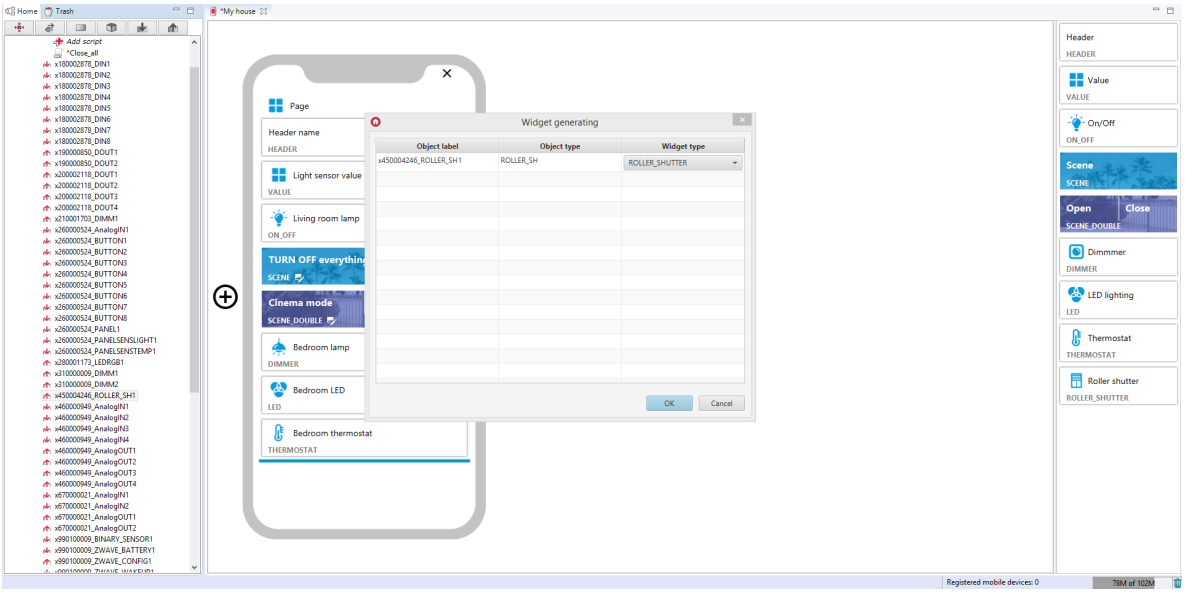


3.14. Roller Shutter (ROLLER_SHUTTER)

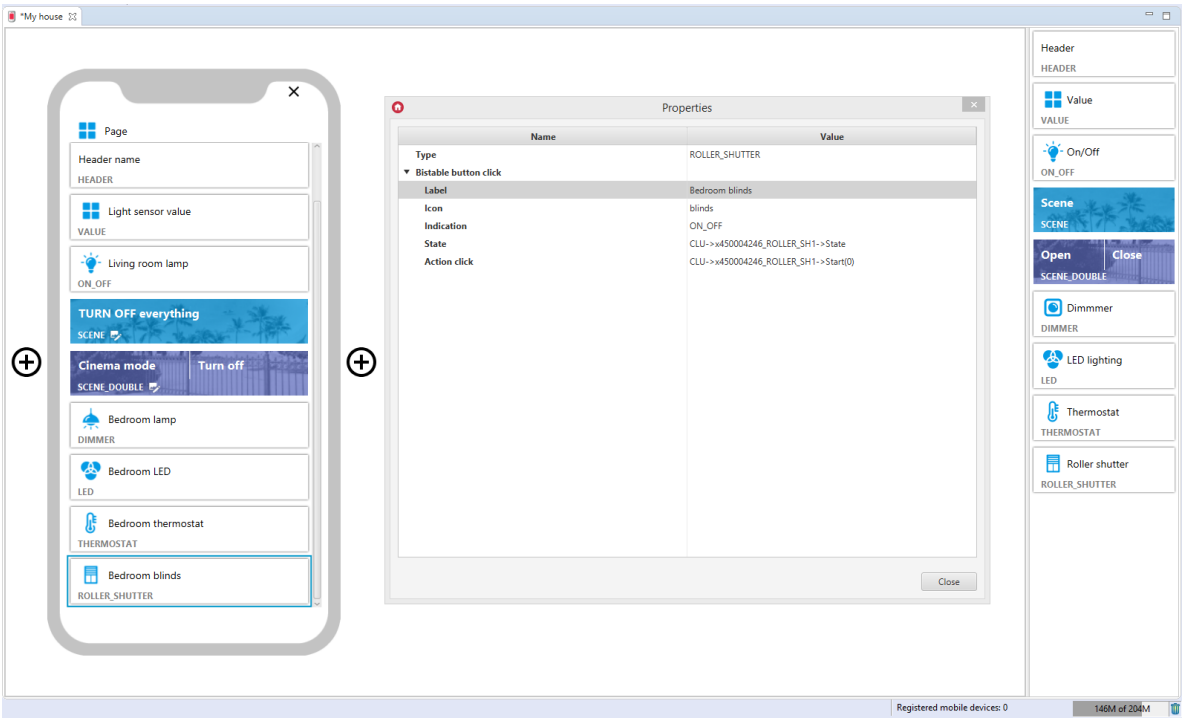
The widget is used to control the blinds. The application displays the current status of the blinds (STOPPED, CLOSED, OPEN).



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:



Filled ROLLER_SHUTTER widget:



3.15. Roller Shutter v2 (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.

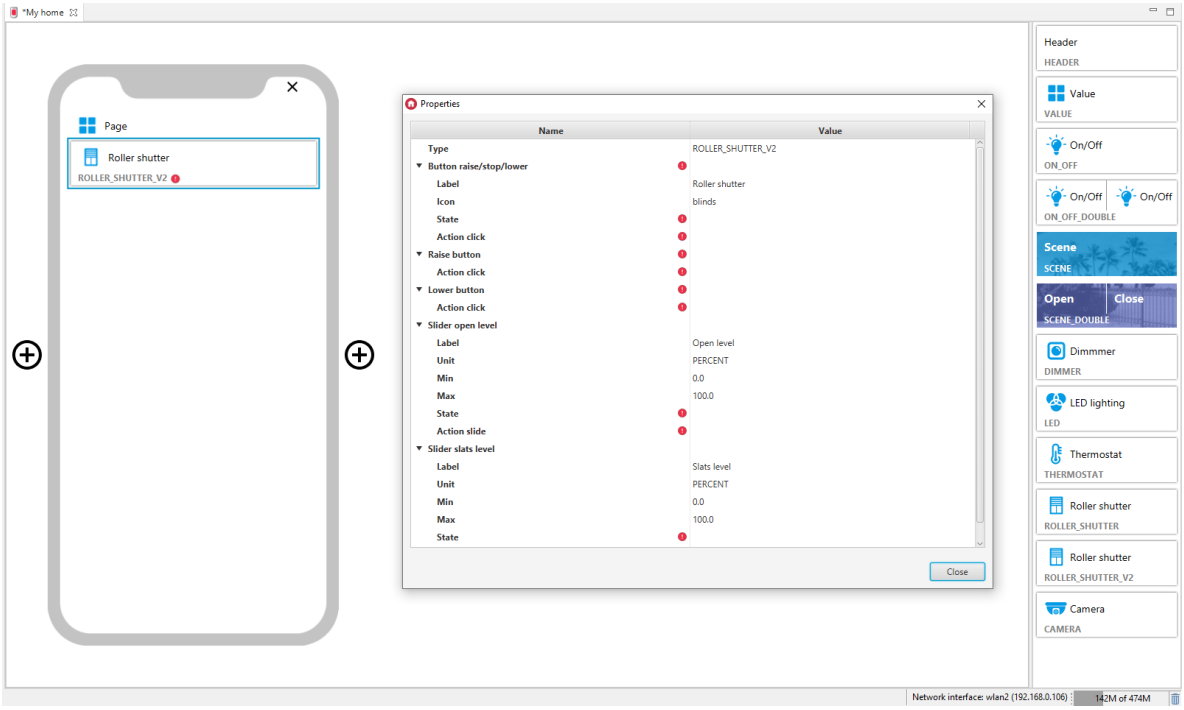
Note!

From the version of Object Manager 1.6.0, the ROLLER_SHUTTER_V2 widget and the possibility of using it as a pre-defined template will not be available. It is replaced with the ROLLER_SHUTTER_V3 widget.

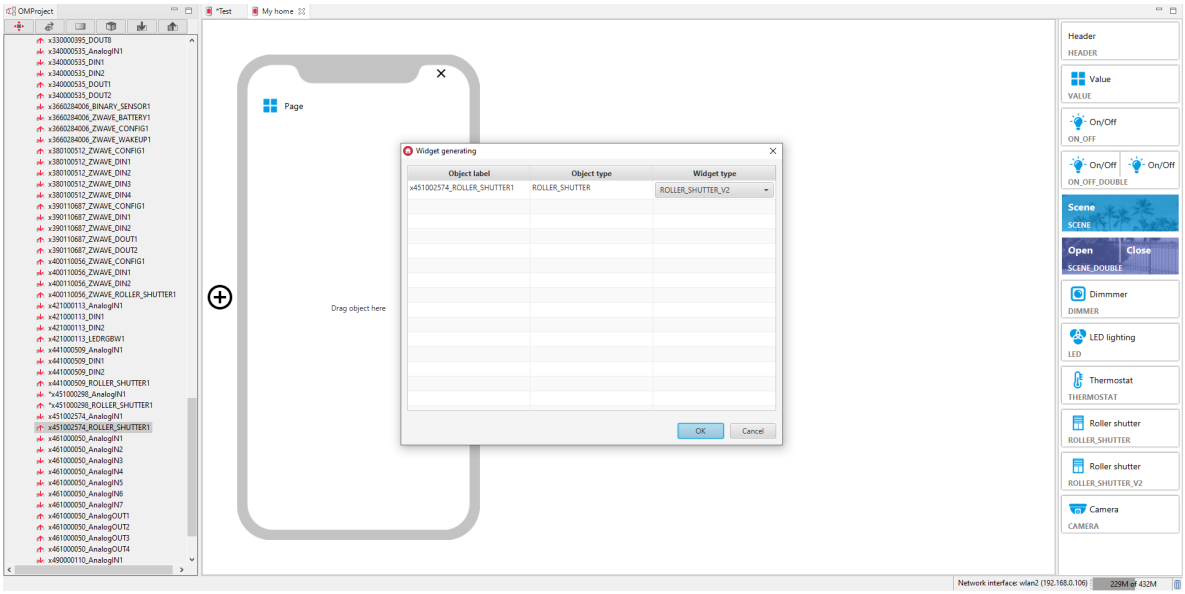
ROLLER_SHUTTER_V2 widgets included in projects created on previous versions of Object Manager will still be properly supported and displayed in the myGrenton application.

Widget dedicated to advanced control of blinds and slats. The ROLLER SHUTTER_V2 widget supports:

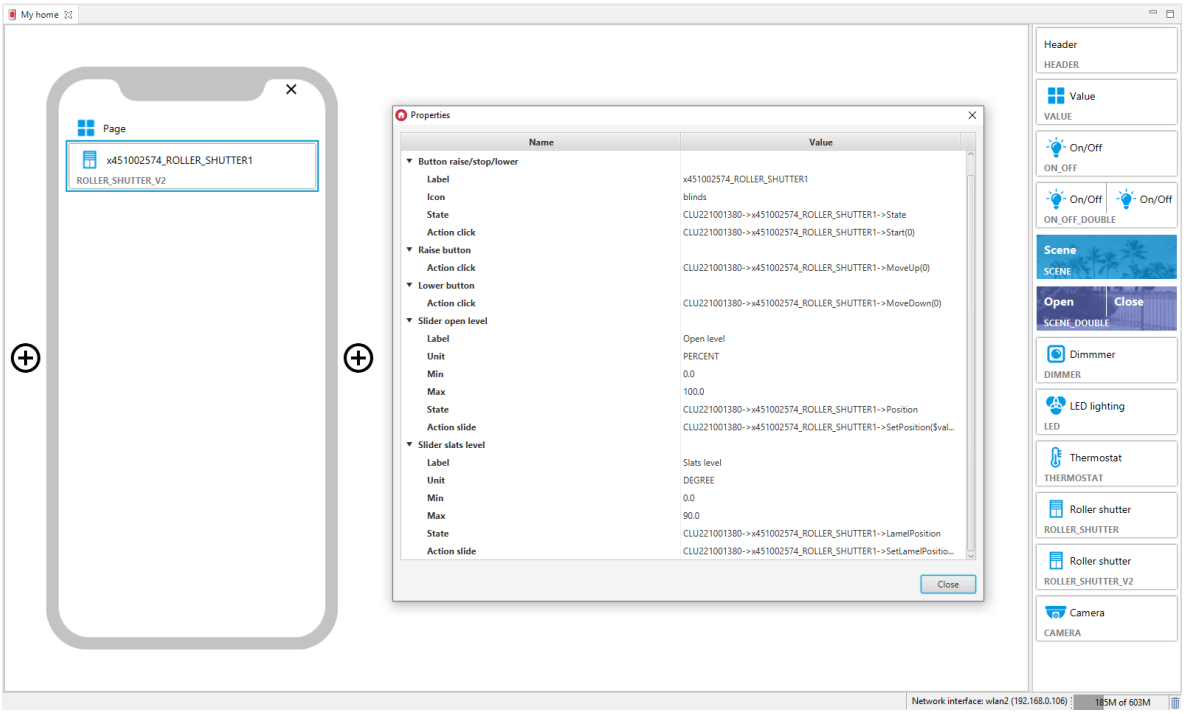
- Buttons enabling the actions of OPENING / CLOSING / STOP the roller shutter,
- Percentage of shutter opening level,
- Sliders for controlling and displaying the shutter and slat opening level.



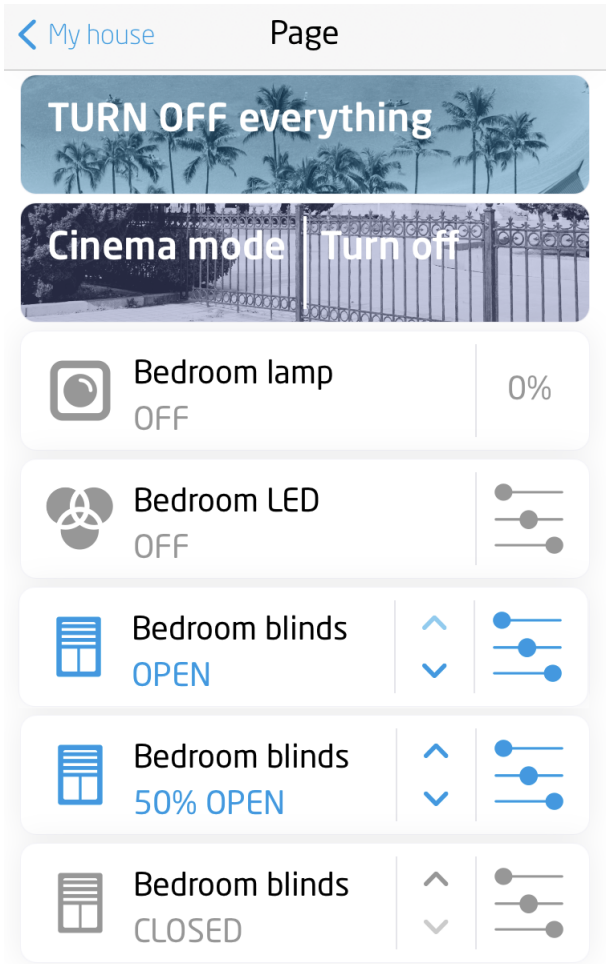
For blind controllers, pre-defined templates for the ROLLER_SHUTTER_V2 widget are defined. To add the ROLLER_SHUTTER_V2 widget with a ready template, drag the ROLLER_SHUTTER object from the list of objects to the phone:

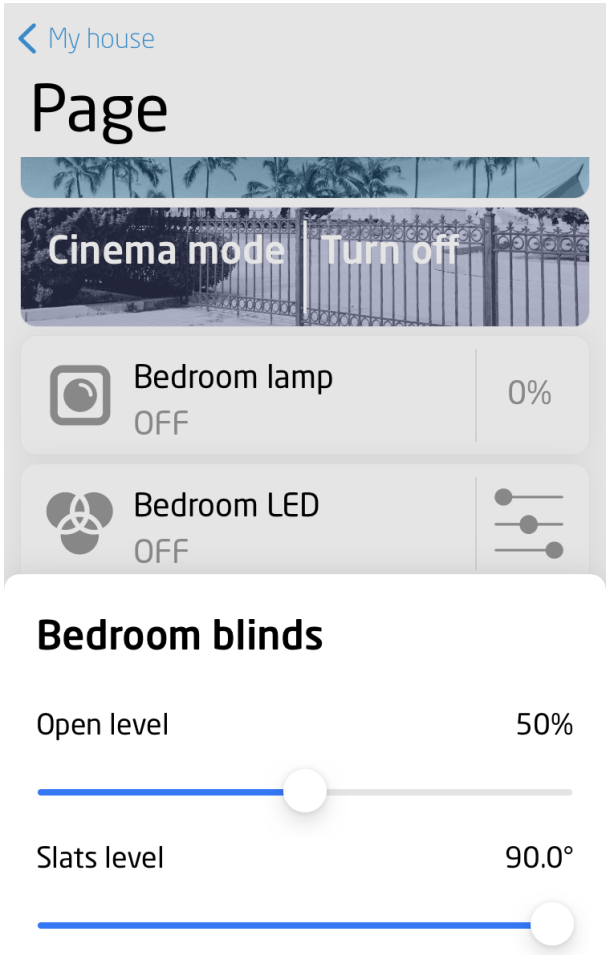


Filled ROLLER_SHUTTER_V2 widget:



Widget appearance in myGrenton application:



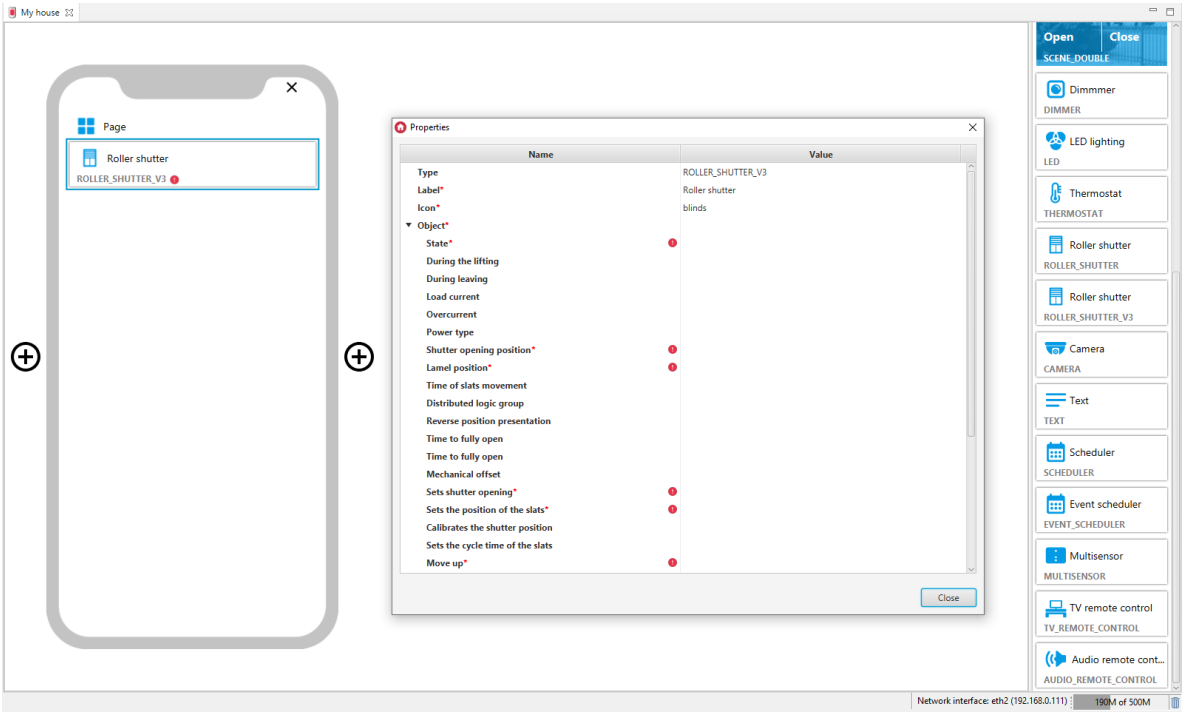


3.16. Roller Shutter v3 (ROLLER_SHUTTER_V3)

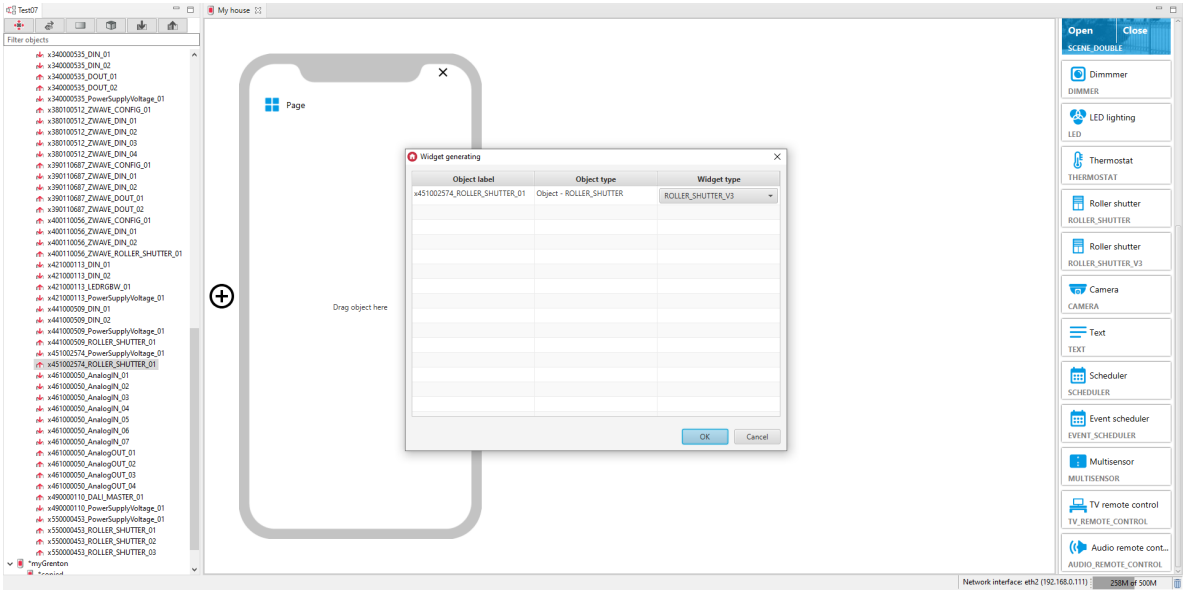
> **Note!**
> The ROLLER_SHUTTER_V3 widget is available for Object Manager version 1.6.0 or higher, CLU version 5.7.1 or higher, Roller Shutter x1 DIN / Roller Shutter x3 DIN / Roller Shutter FM module in version 2.1.1 or higher and for myGrenton application version 1.4.0 (Android) / 1.8.0 (iOS) or higher.

Widget dedicated to advanced control of blinds and slats. The ROLLER SHUTTER_V3 widget supports:

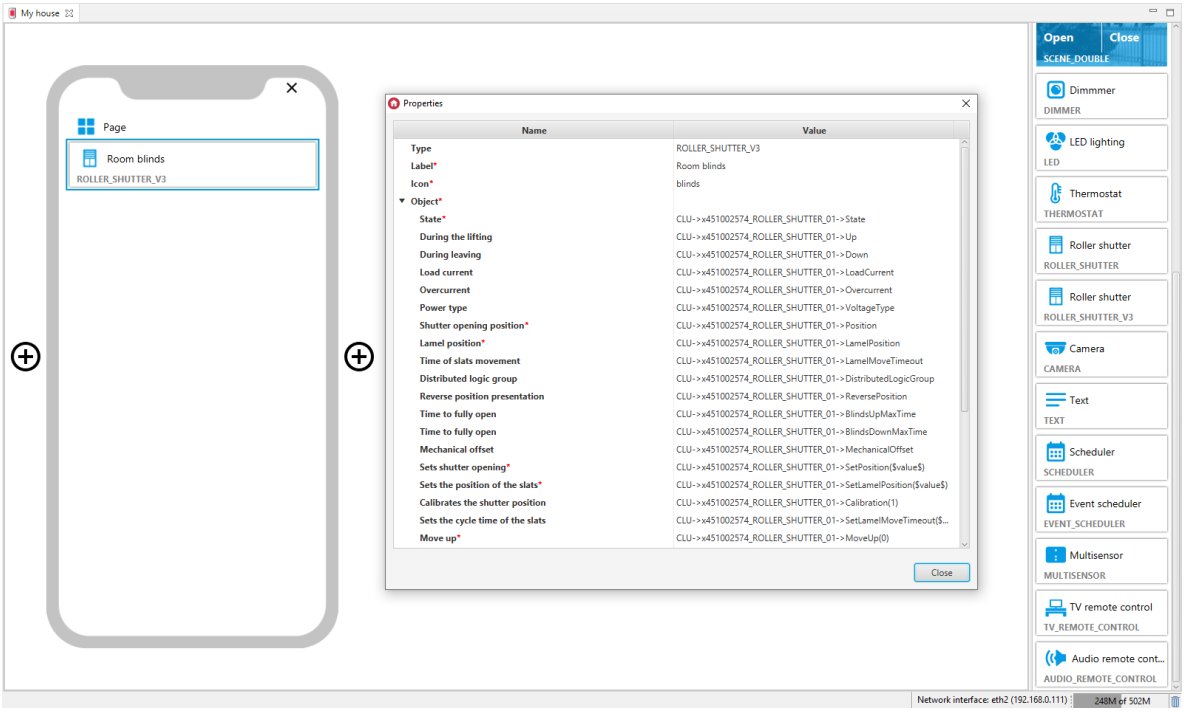
- Buttons enabling the actions of OPENING / CLOSING / STOP the roller shutter,
- Percentage of shutter opening level,
- Sliders for controlling and displaying the shutter and slat opening level,
- Display of the state of `Calibration` / `Locked`.



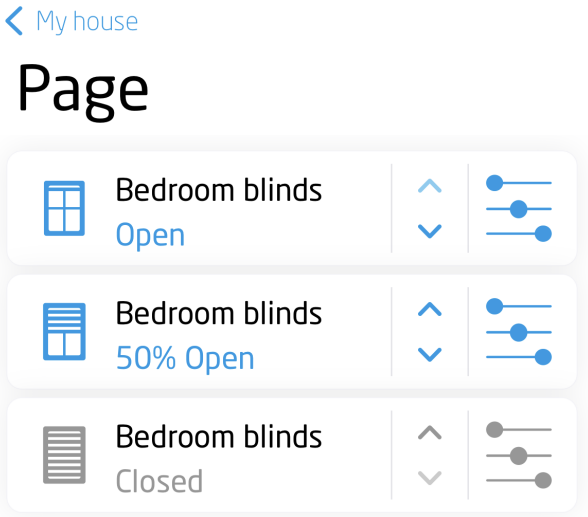
For blind controllers, pre-defined templates for the ROLLER_SHUTTER_V3 widget are defined. To add the ROLLER_SHUTTER_V3 widget with a ready template, drag the ROLLER_SHUTTER object from the list of objects to the phone:

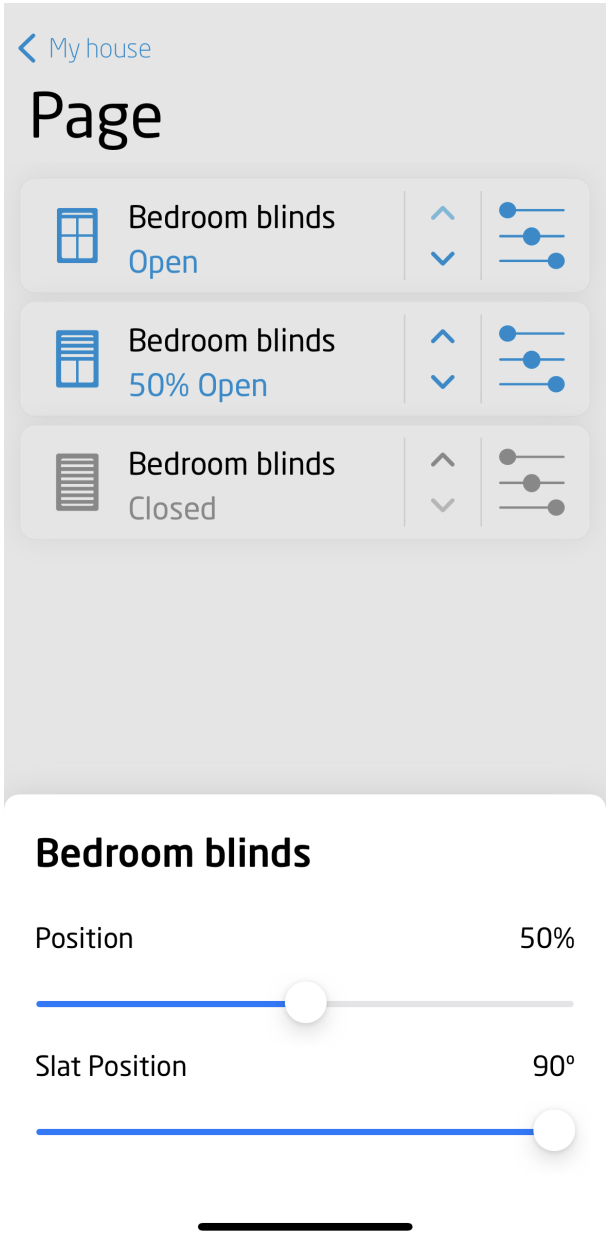


Filled ROLLER_SHUTTER_V3 widget:



Widget appearance in myGrenton application:

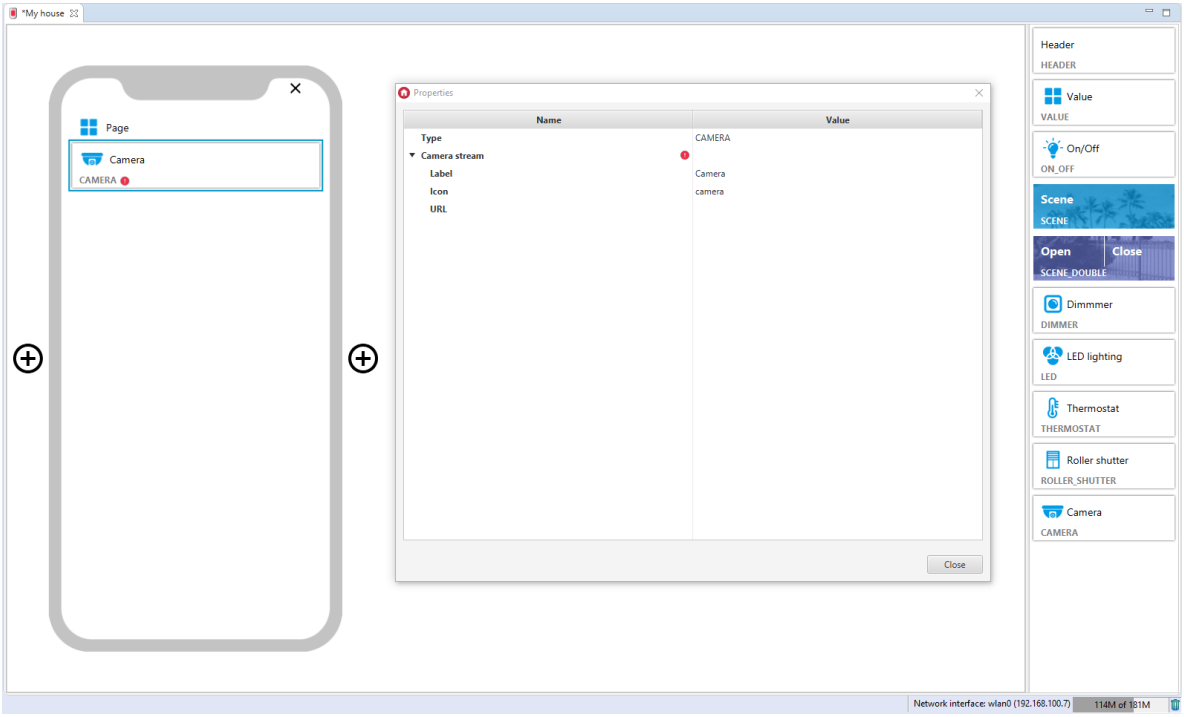




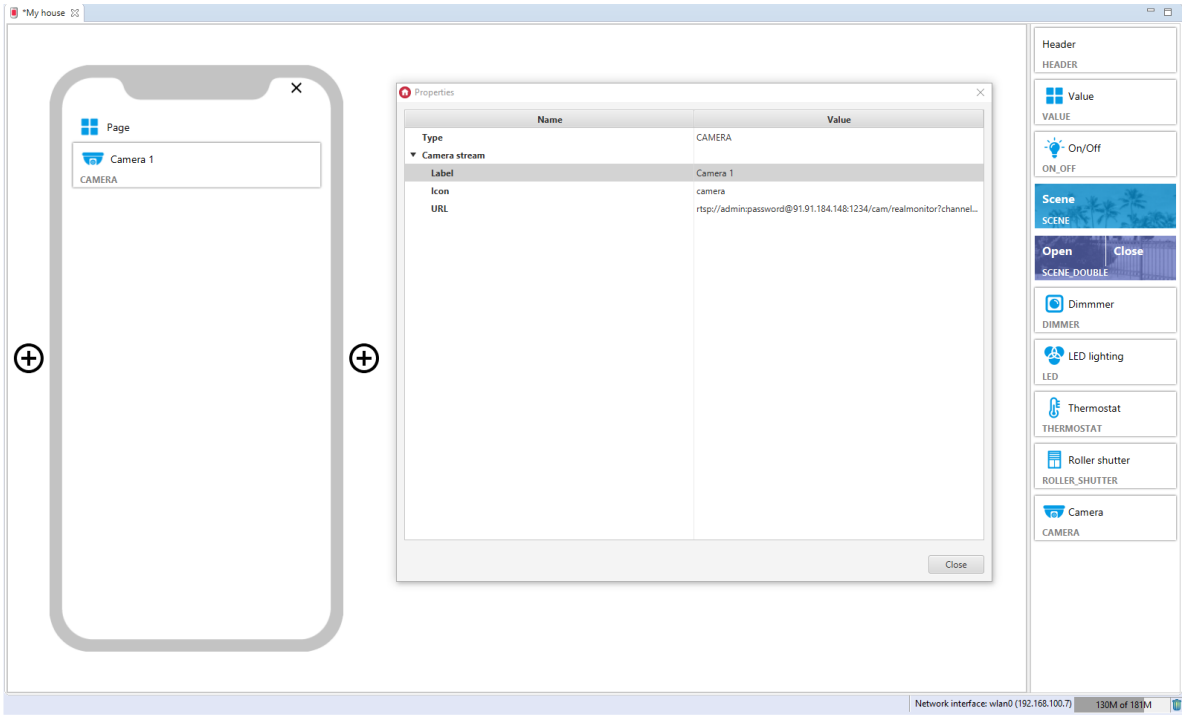
3.17. Camera (CAMERA)

> **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).



Filled CAMERA 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 >`

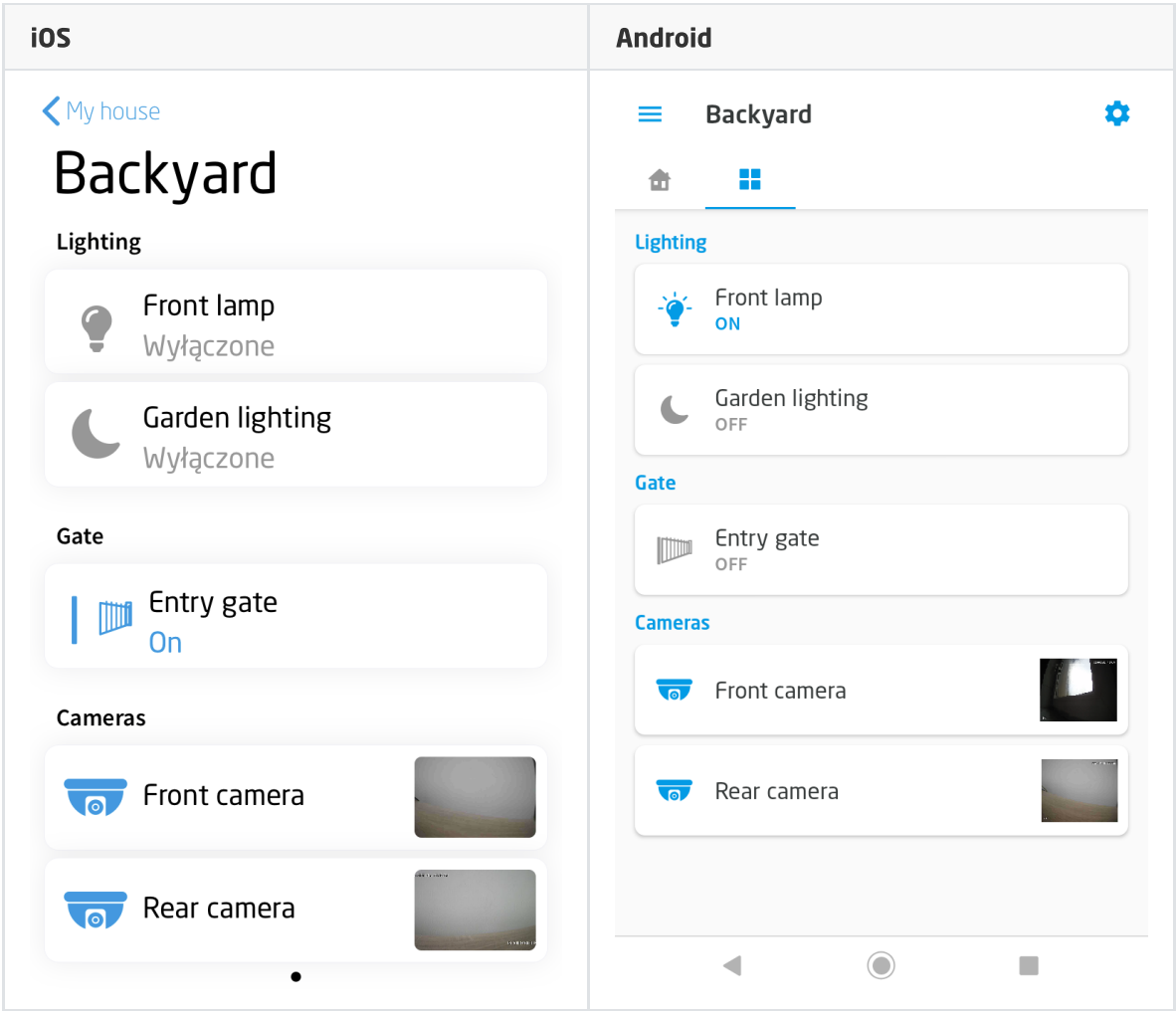
Example:

`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.

Widget appearance in the myGrenton application:



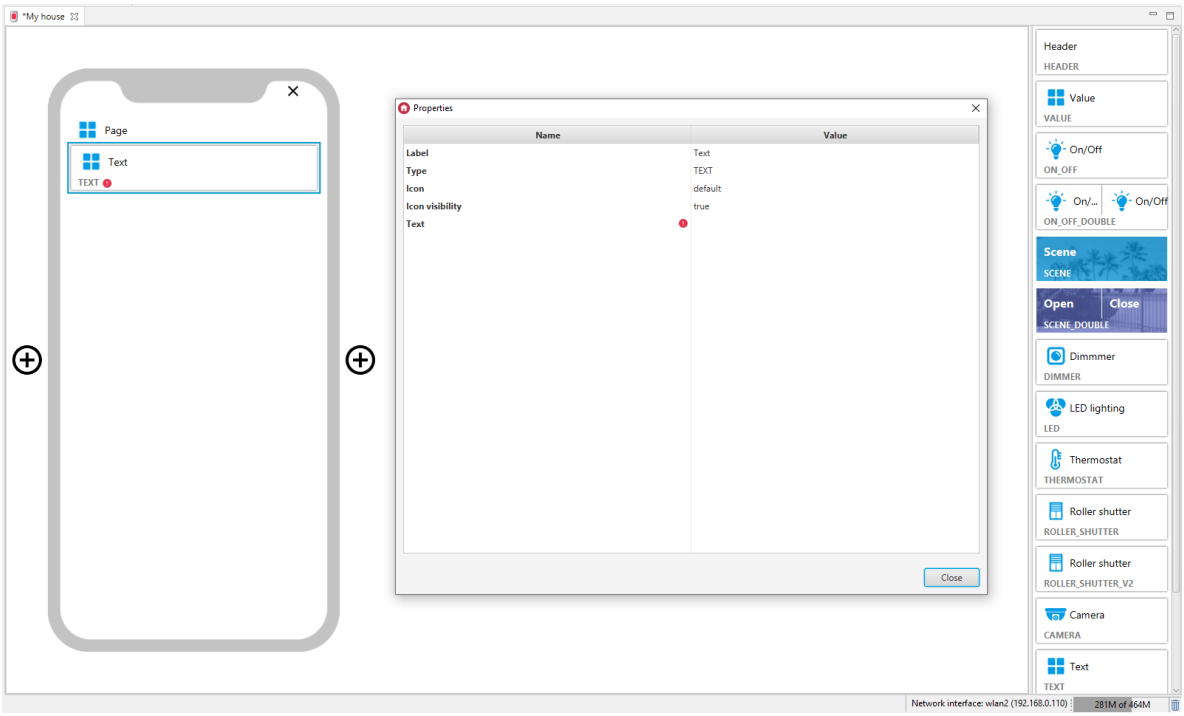
Camera stream preview view:

iOS	Android

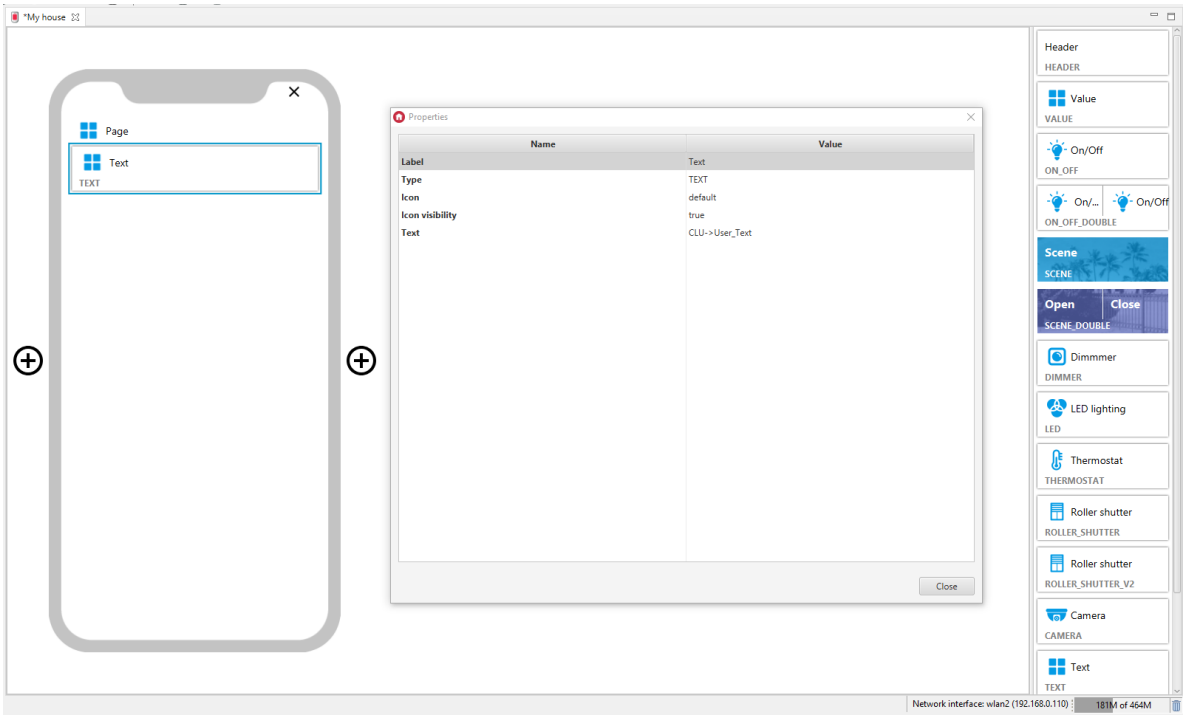
3.18. Text (TEXT)

> **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.

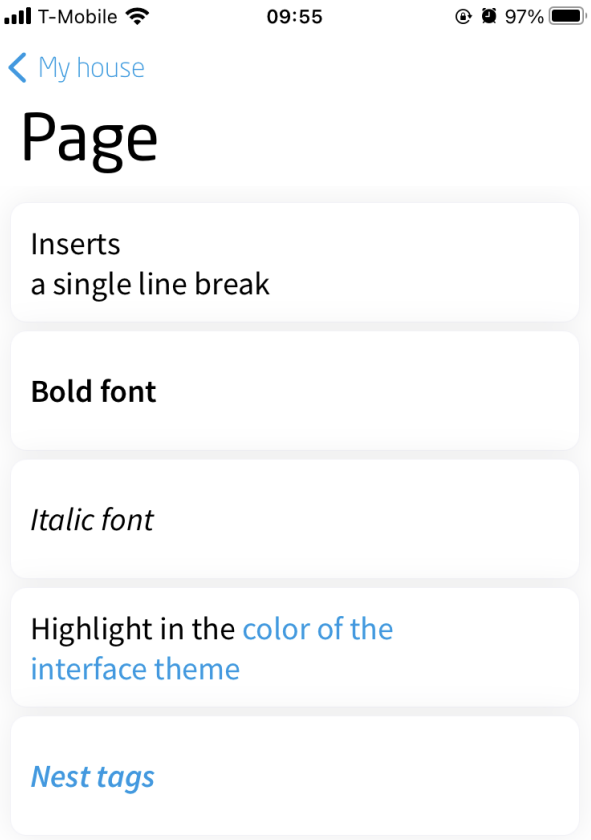
Examples of the use of tags:

- 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.

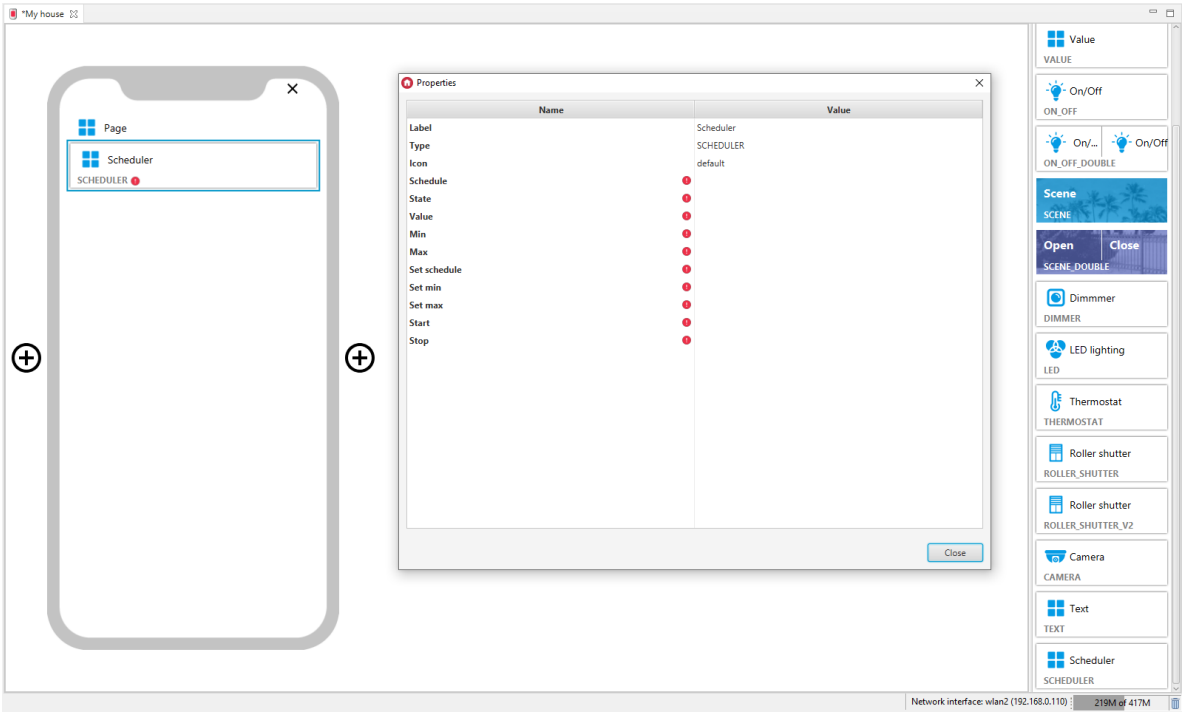
Widget appearance in the myGrenton application:



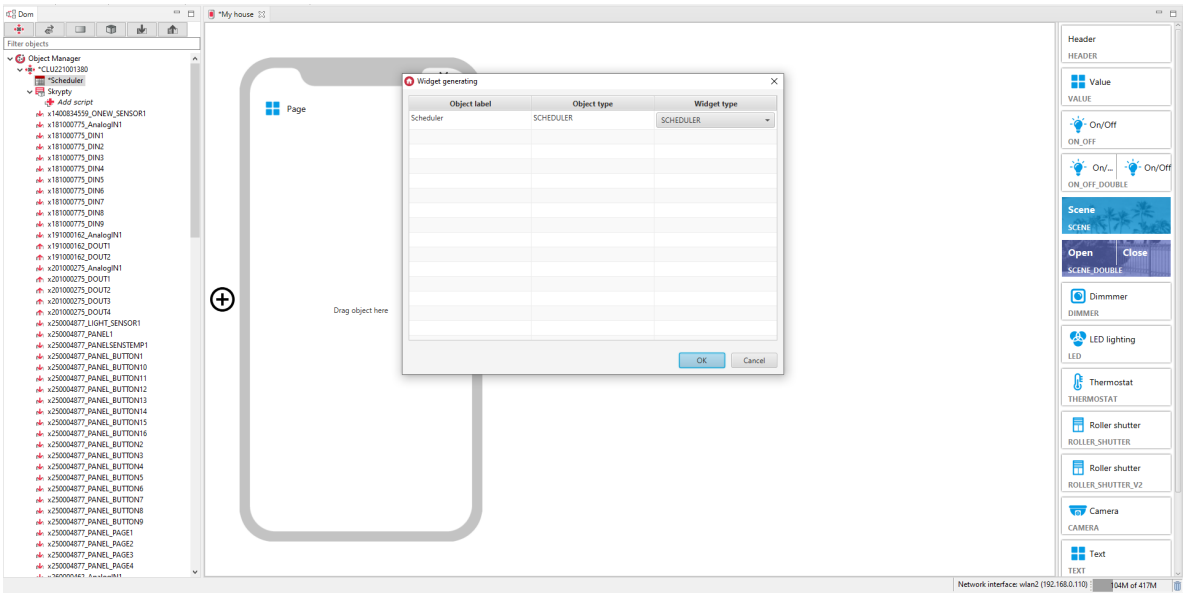
3.19. Scheduler (SCHEDULER)

> **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.

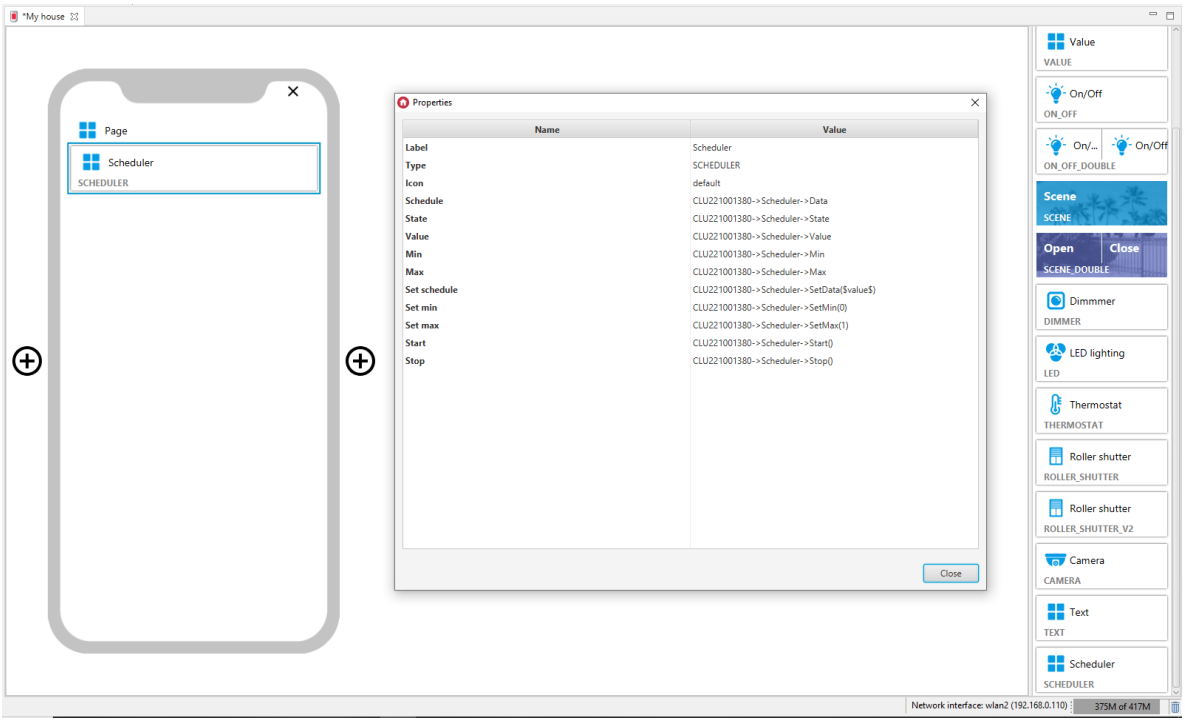
Widget dedicated for virtual objects of the scheduler type.



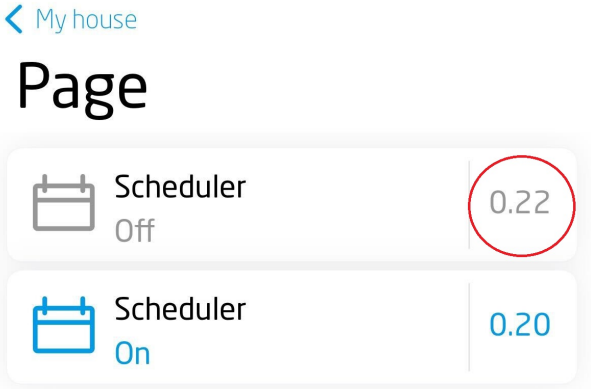
For scheduler, ready-made templates for the SCHEDULER widget are defined. To add a SCHEDULER widget with a ready template, drag the virtual scheduler from the list of objects to the interface page:



Filled SCHEDULER widget:



In the myGrenton application, you can edit the schedule. To do this, click on the widget value field:

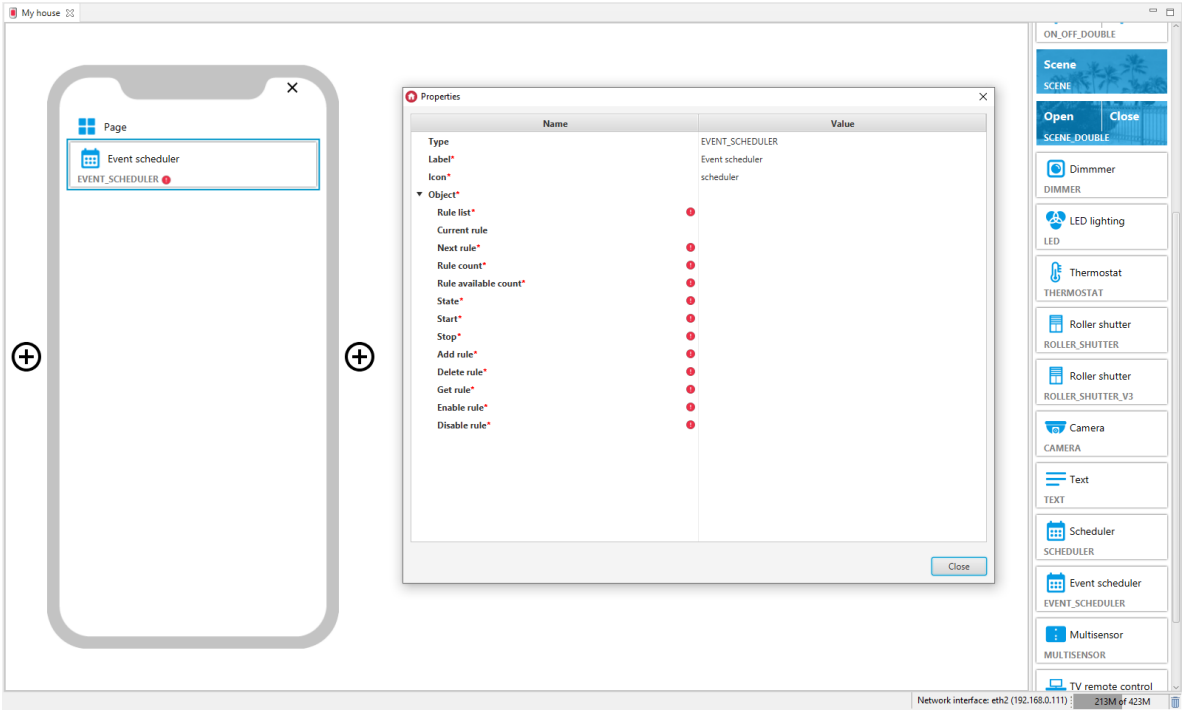


Schedule configuration has been described for the THERMOSTAT widget look up XVIII.3.9.A.. Slight differences are due to the different characteristics of the virtual scheduler object.

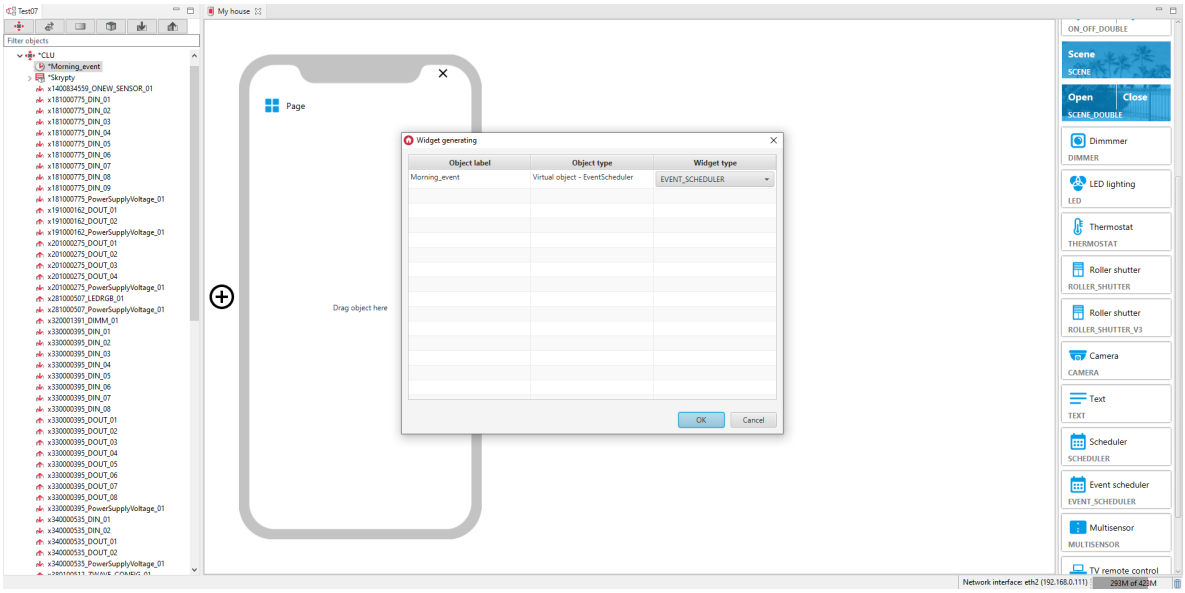
3.20. Event Scheduler (EVENT_SCHEDULER)

> **Note!**
> The EVENT_SCHEDULER widget is available for Object Manager version 1.6.0 or higher, CLU version 5.9.1 or higher, for myGrenton application version 1.4.0 (Android) / 1.8.0 (iOS) or higher.

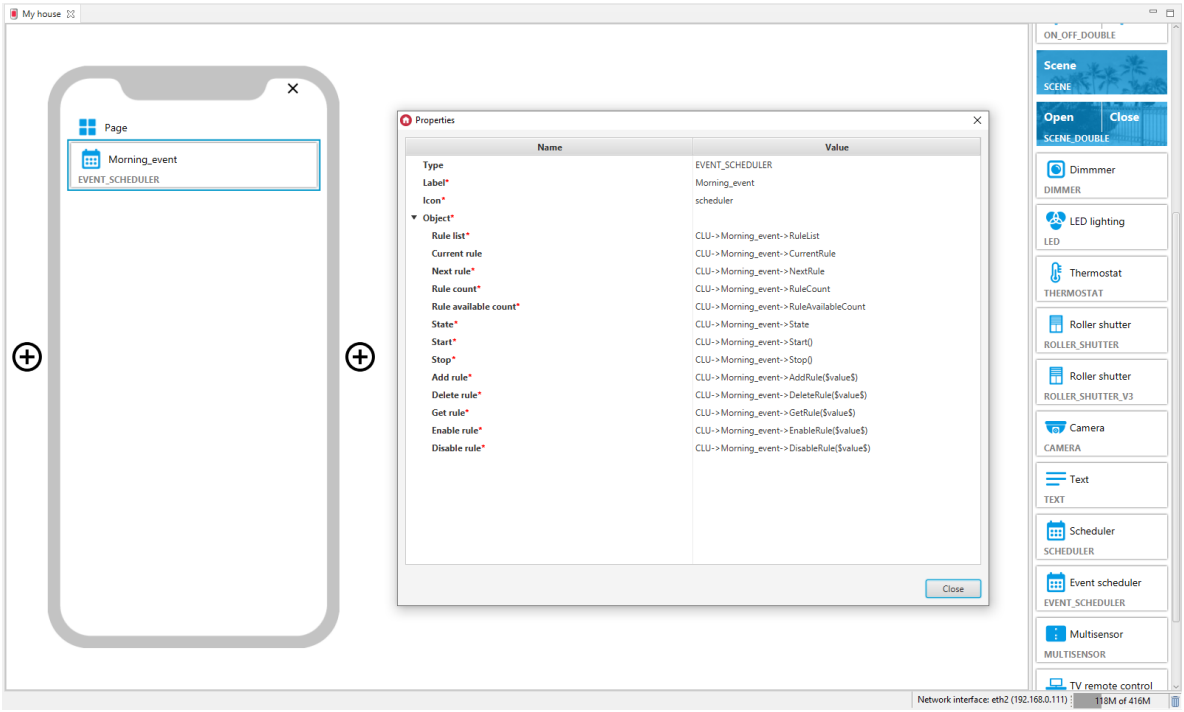
Widget dedicated for virtual objects of the event scheduler type.



For event scheduler, ready-made templates for the EVENT_SCHEDULER widget are defined. To add a EVENT_SCHEDULER widget with a ready template, drag the virtual event scheduler from the list of objects to the interface page:

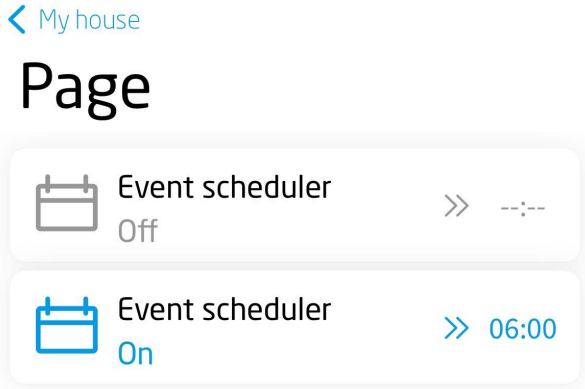


Filled EVENT_SCHEDULER widget:

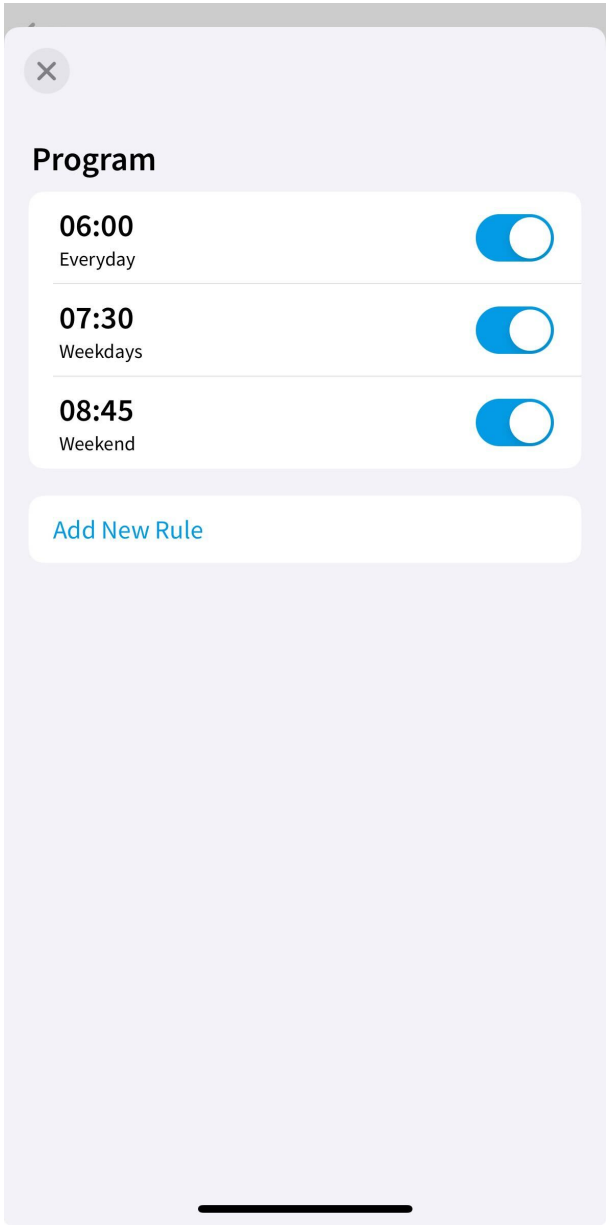


A. Event scheduler configuration in the application

In the myGrenton application, you can edit the rules of the event scheduler. To do this, click on the widget value field:

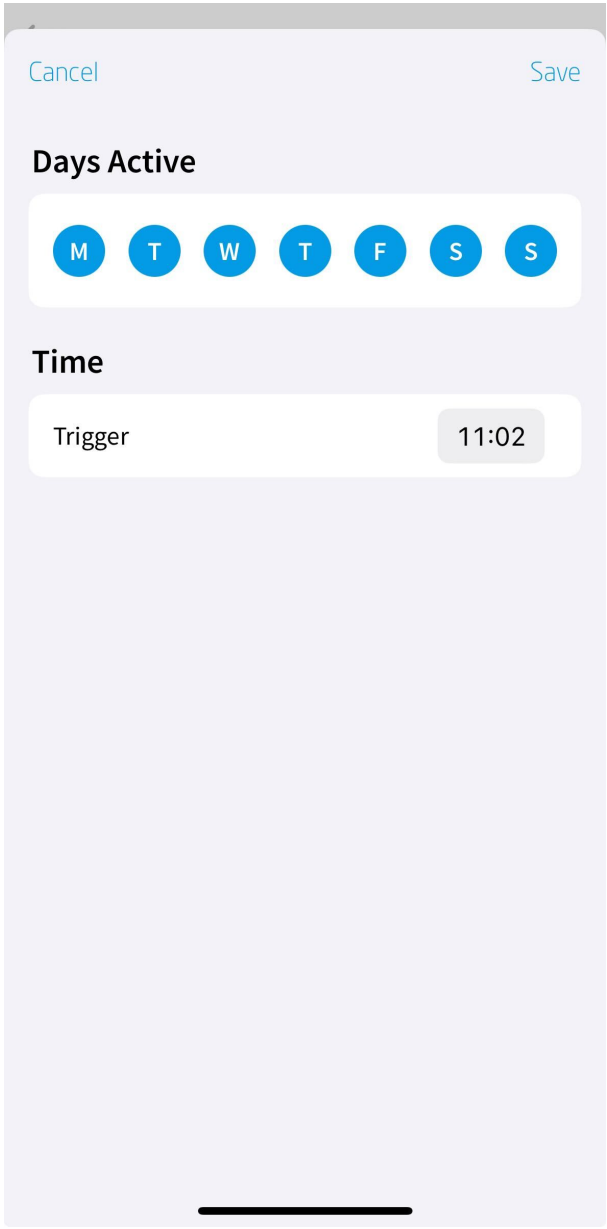


The window will display the rules downloaded from the CLU. You can edit these rules or add new ones for each day of the week:



Adding a new rule

After selecting `Add New Rule`, the adding rule window will open:

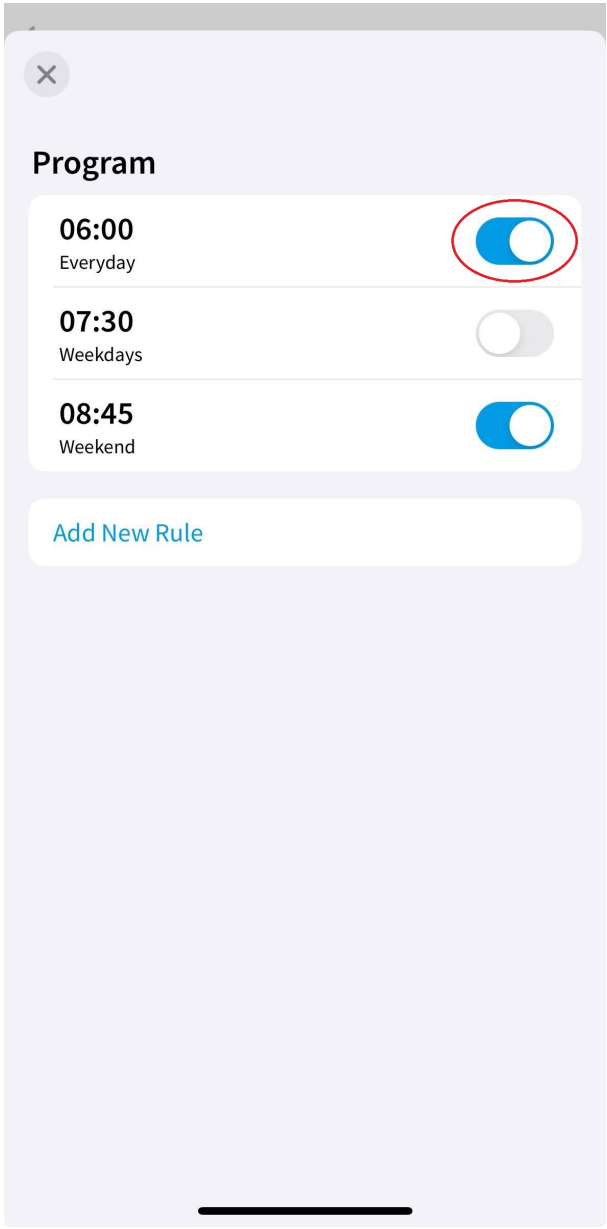


Then:

- Mark the desired days of the week (at least one day must be selected),
- Set the trigger time,
- Accept the changes by clicking on `Save`.

Activate / deactivate the rule

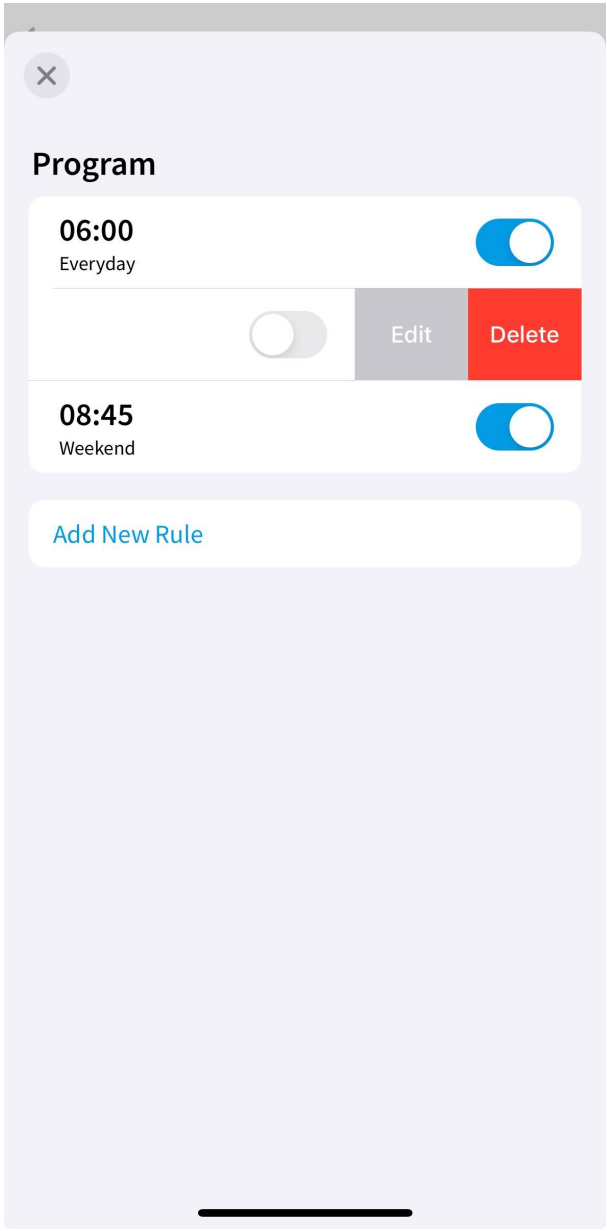
To activate / deactivate a rule, click on the slider in the right part of the rule:



Delete / edit rule

To edit an existing rule, click on the rule (Android) or make a left-swipe gesture on the rule, and then click on the `Edit` option (available only for iOS).

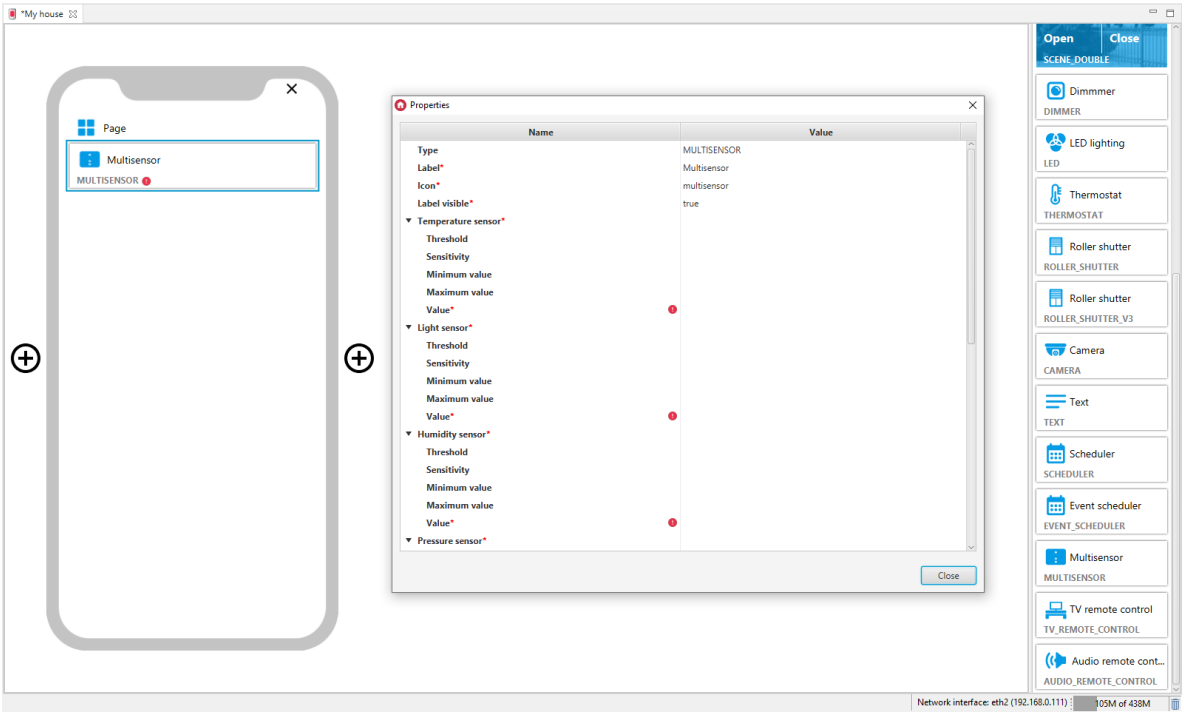
To delete a rule, perform a left-swipe gesture on the rule, and then click `Delete`.



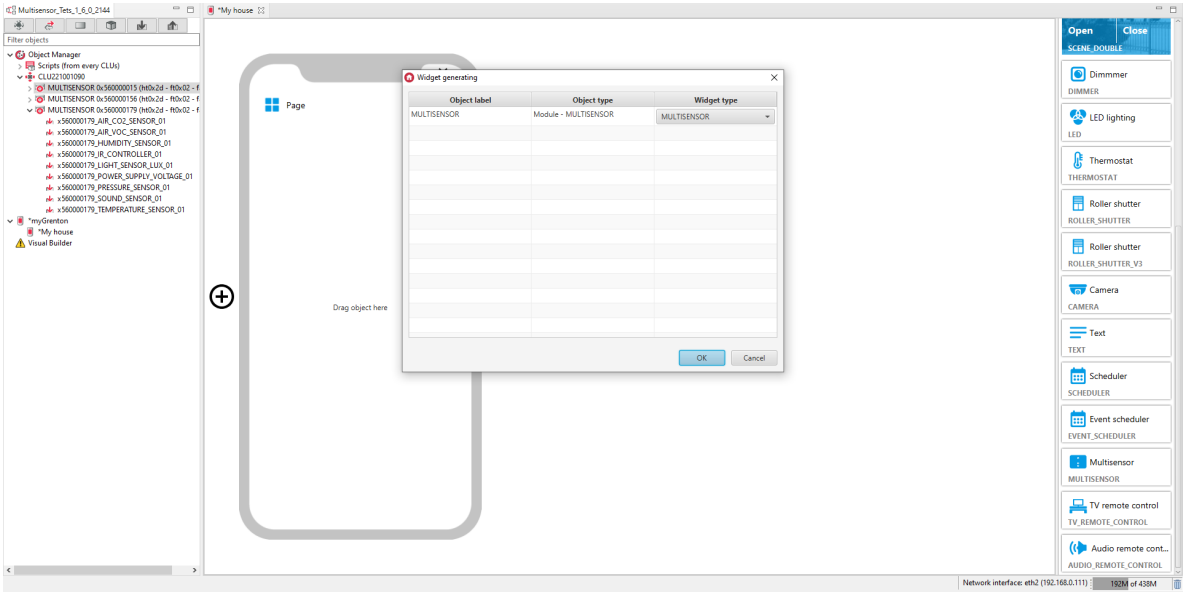
3.21. Multisensor (MULTISENSOR)

> **Note!**
> The TV_REMOTE_CONTROL widget is available for Object Manager version 1.6.0 or higher, CLU version 5.9.1 or higher, Multisensor IR module in version 1.2.6 or higher, for myGrenton application version 1.4.0 (Android) / 1.8.0 (iOS) or higher.

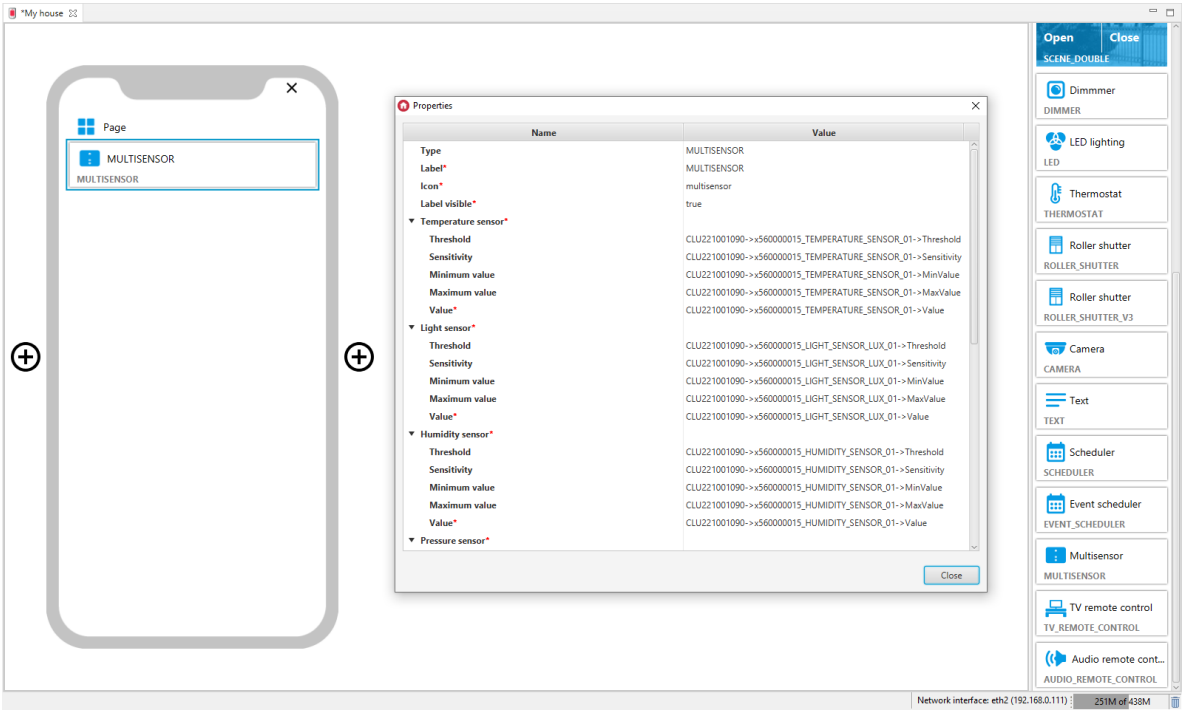
Widget dedicated to displaying the parameters of the GRENTON MULTISENSOR IR device. It allows you to display environmental parameters such as: temperature, humidity, CO2, VOC (Volatile Organic Compounds), air pressure, sound intensity and light intensity. The widget has a `Label Visibility` property that allows you to display or hide the widget label in the application. The order of displayed parameters can be changed in the myGrenton application.



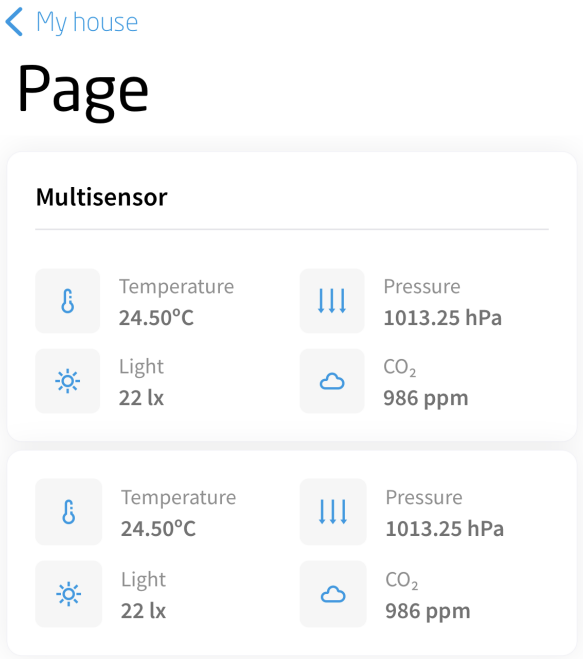
For MULTISENSOR object, pre-defined templates for the MULTISENSOR widget are defined. To add the MULTISENSOR widget with a ready template, drag the MULTISENSOR object from the list of objects to the phone:

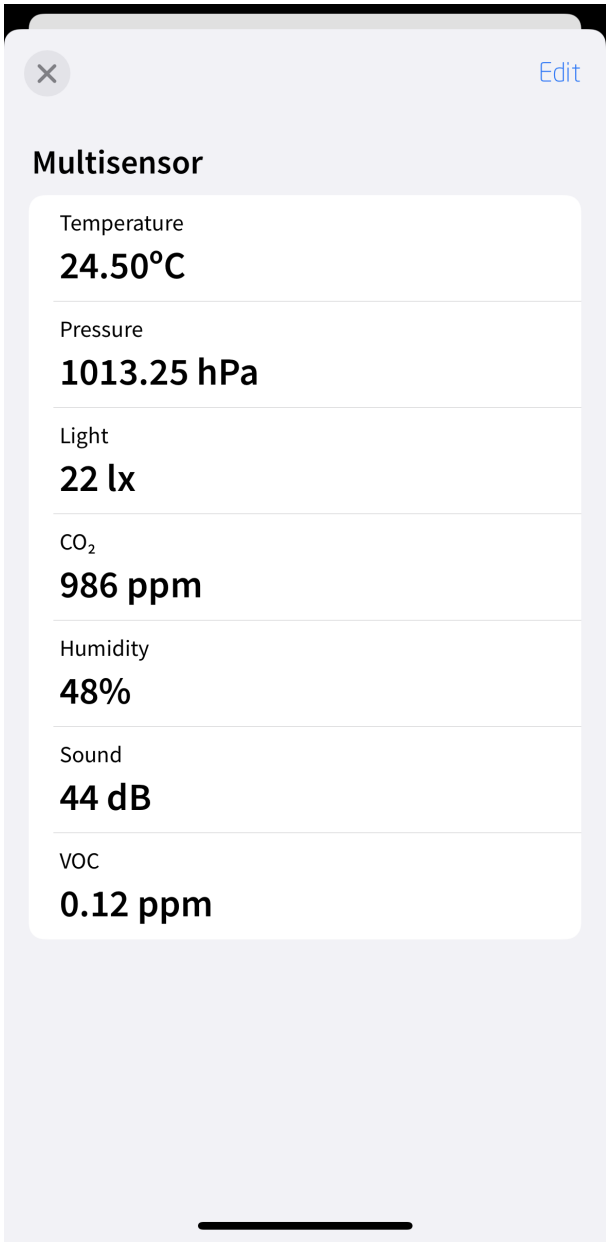


Filled MULTISENSOR widget:



Widget appearance in the myGrenton application, versions with visible and hidden label:





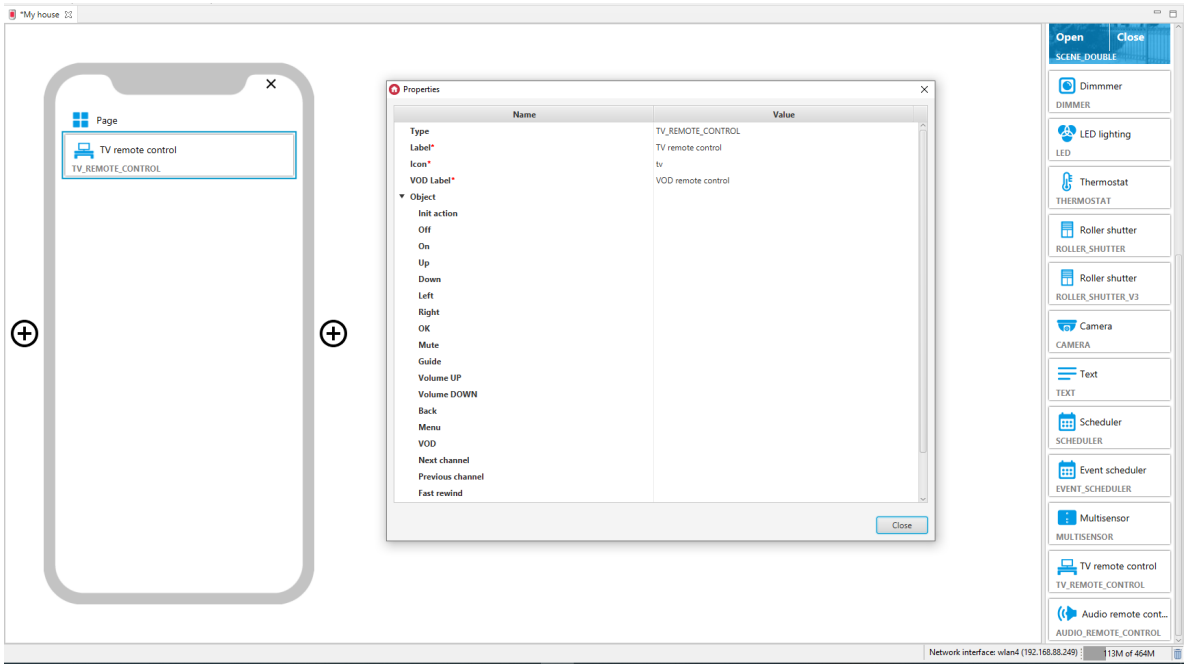
3.22. TV Remote Control (TV_REMOTE_CONTROL)

> **Note!**
> The TV_REMOTE_CONTROL widget is available for Object Manager version 1.6.0 or higher, for myGrenton application version 1.4.0 (Android) / 1.8.0 (iOS) or higher.

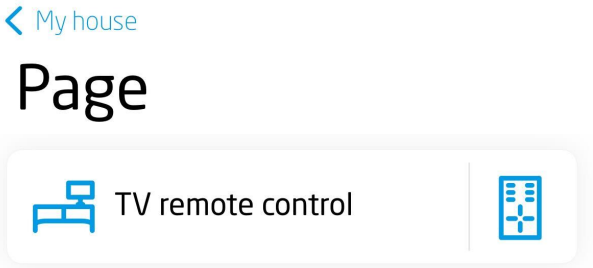
The widget is modeled on the appearance of the TV Remote Control. It allows you to work with devices integrated with the Grenton system and replace the traditional remote control. The control is carried out by means of actions assigned to the selected buttons. The Tv Remote Control widget contains:

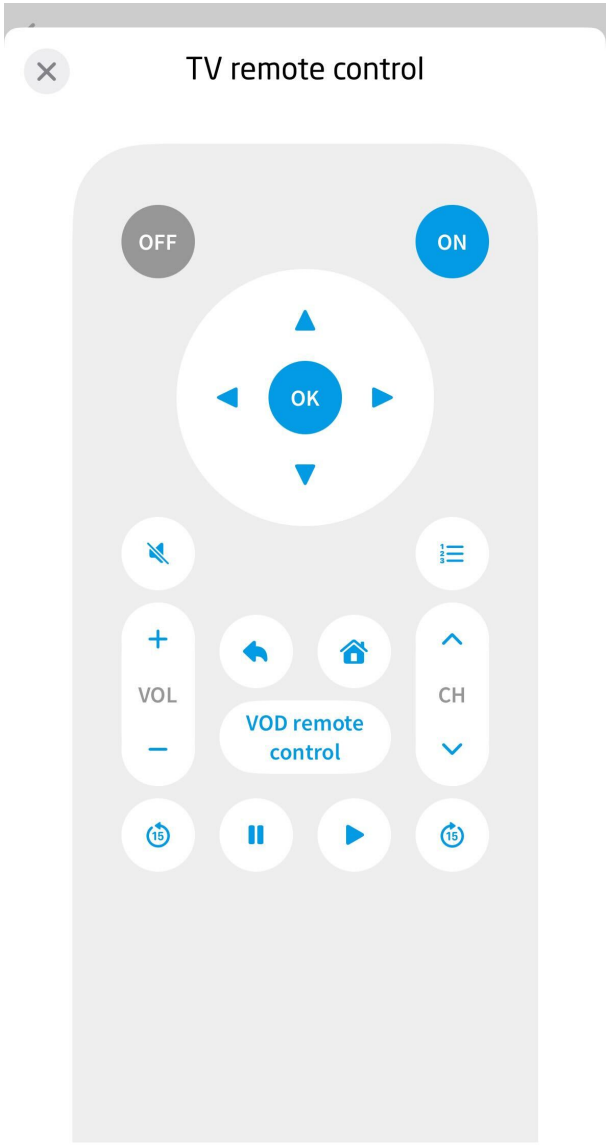
- initialization action - action triggered when clicking on the widget on the myGrenton application (any action can be assigned),
- buttons to control the activation / deactivation of the device,
- buttons for changing the channel (program) and changing the device volume,
- buttons to control playback (pause, run, forward, rewind),
- function buttons (menu, VOD, return, etc.).

Operation in the Remote TV widget is one-way, it is not possible to read the device status.



Widget appearance in the myGrenton application:





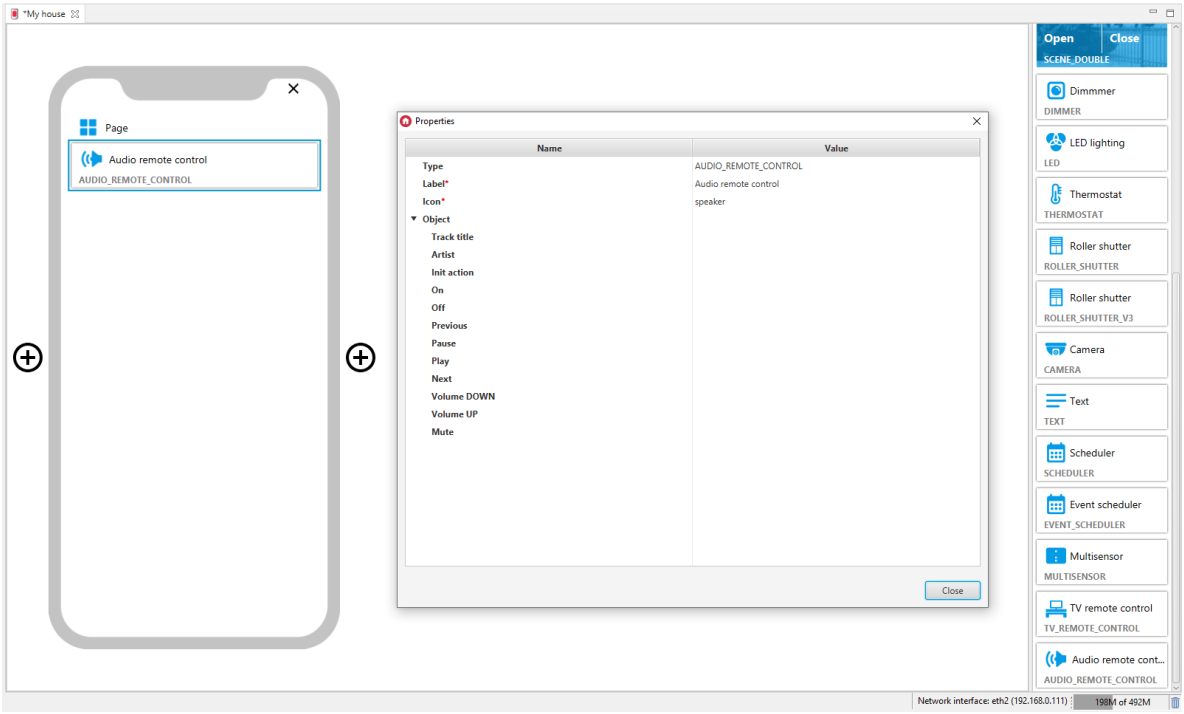
3.23. Audio Remote Control (AUDIO_REMOTE_CONTROL)

> **Note!**
> The AUDIO_REMOTE_CONTROL widget is available for Object Manager version 1.6.0 or higher, for myGrenton application version 1.4.0 (Android) / 1.8.0 (iOS) or higher.

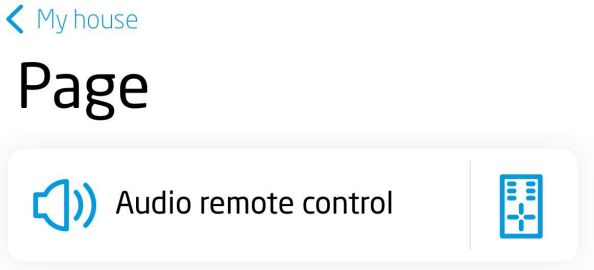
Widget dedicated to controlling audio systems. It allows you to work with devices integrated with the Grenton system. The control is carried out by means of actions assigned to the selected buttons. The AUDIO_REMOTE_CONTROL widget contains:

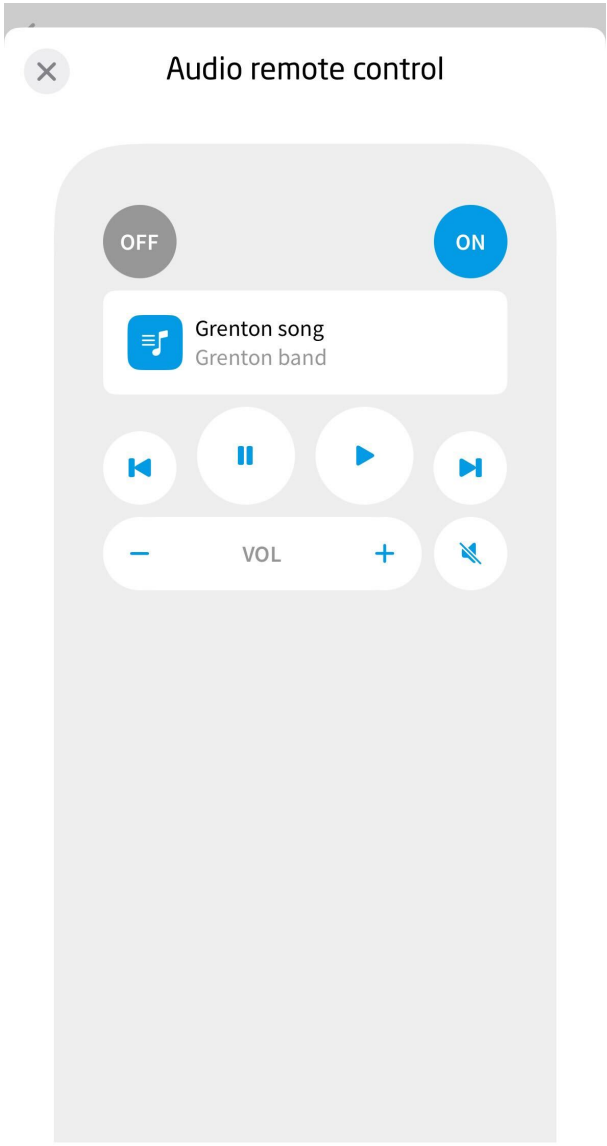
- initialization action - action triggered when clicking on the widget on the myGrenton application (any action can be assigned),
- buttons to control the activation / deactivation of the device,
- buttons for changing the device volume,
- buttons to control playback (pause, run, next, previous),
- artist and title display fields.

Operation in the Remote Audio widget is one-way, it is not possible to read the device status. It is possible to retrieve information from the selected feature (e.g. user's feature) and display it in the Artist and / or Title field.



Widget appearance in the myGrenton application:





3.24. Contact Sensor (CONTACT_SENSOR)

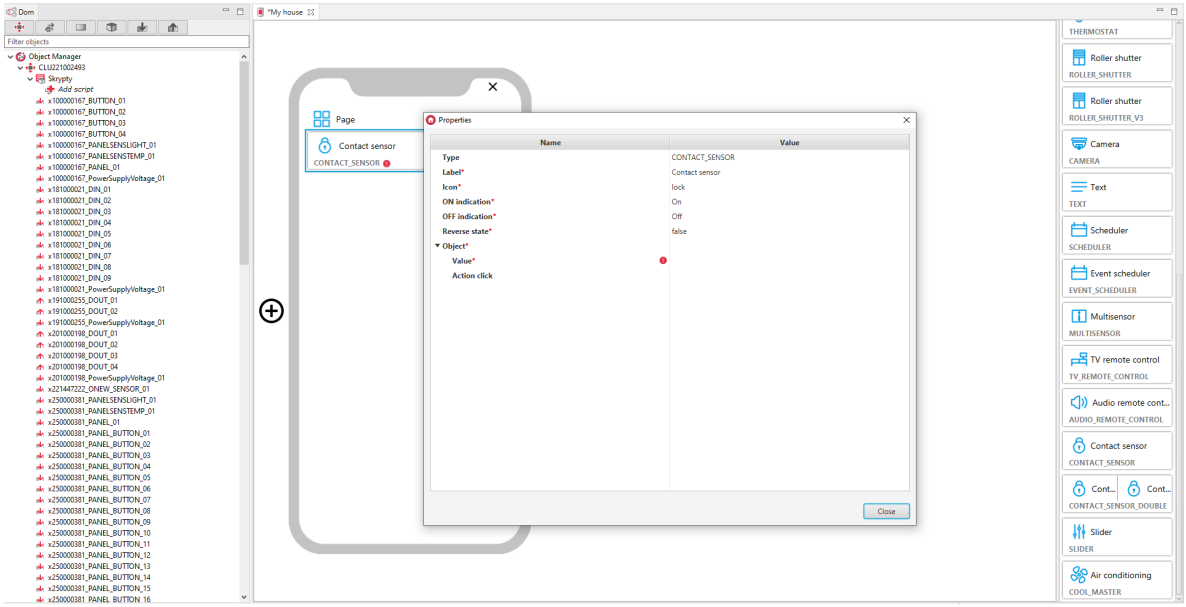
> **Note!**
> The CONTACT_SENSOR widget is available for Object Manager version 1.7.0 or higher, for myGrenton application version 1.5.0 (Android) / 1.9.0 (iOS) or higher.

Widget dedicated to displaying On / Off states based on the value of objects, features. The CONTACT_SENSOR widget contains:

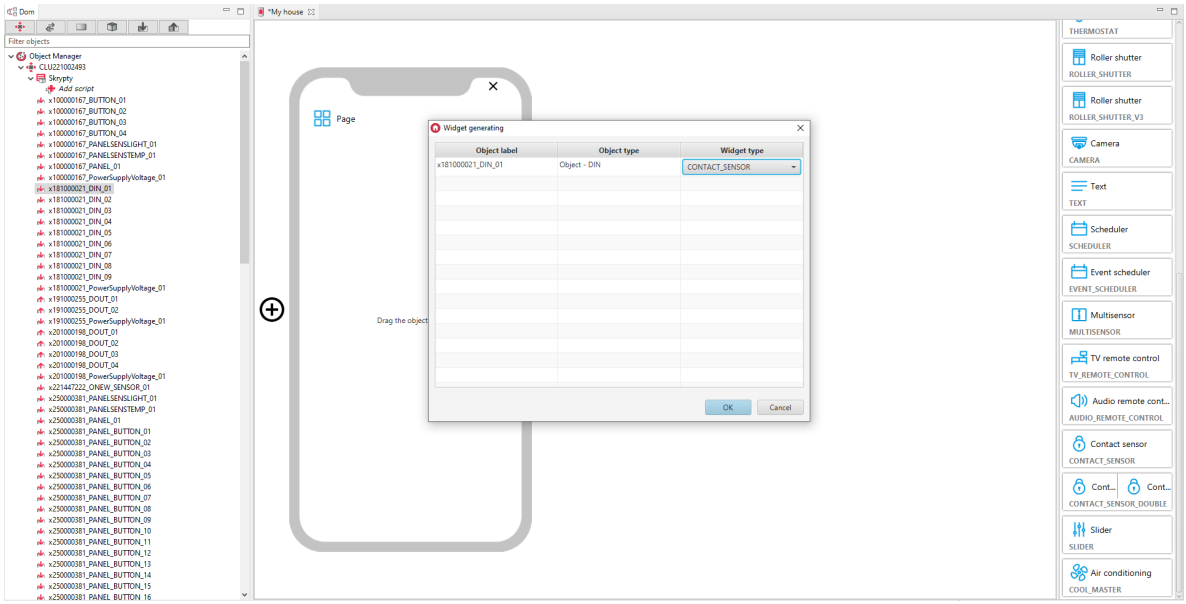
- support for On / Off states - based on the value of objects, features (`0/1` or `true/false`), the widget status is properly displayed,
- indications - it is possible to set any name of On / Off states through the properties `Indication ON` , `Indication OFF` ,
- support for state reversal (`0` - `Enabled` , `1` - `Disabled`)

> **Note!**
> Functionality is available for Object Manager version 1.10.0 or higher, myGrenton application 1.9.17 (Android) / 1.12.1 (iOS) or higher.

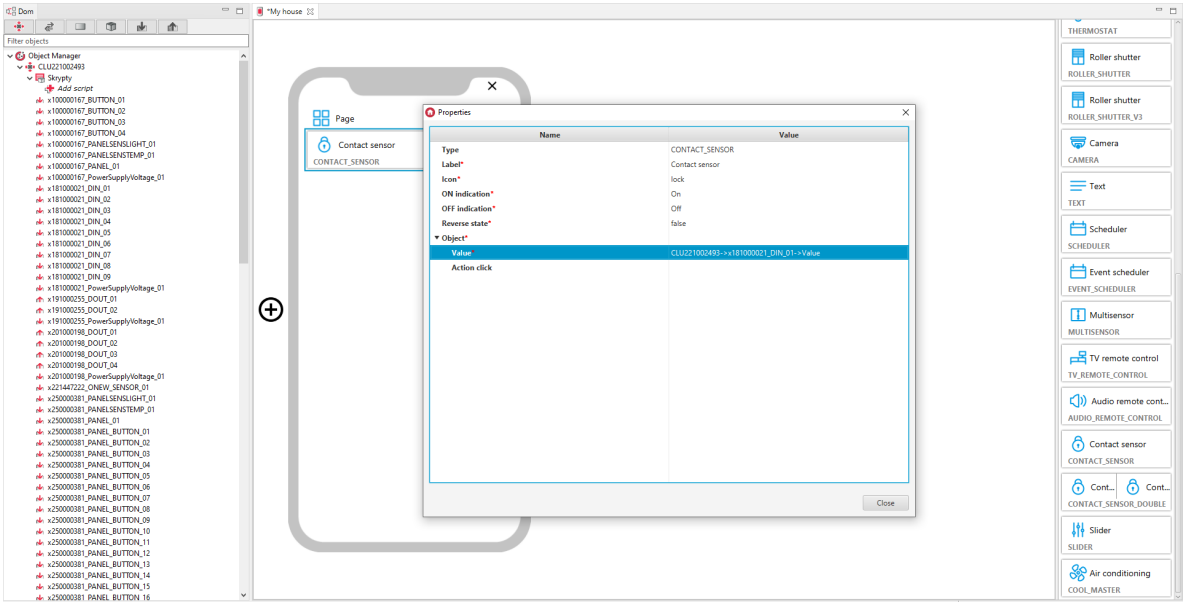
- optional click action - action triggered when clicking on the widget (any action can be assigned).



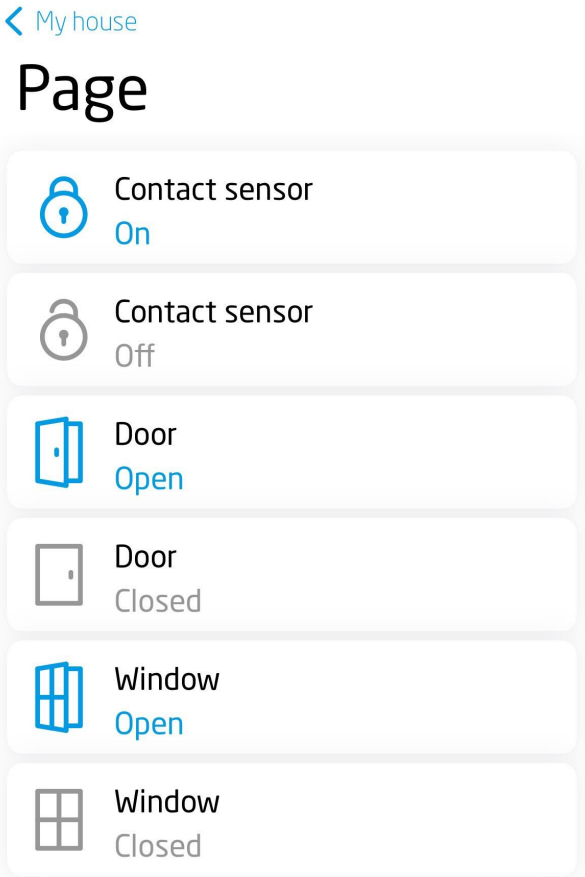
For DIN objects, pre-defined templates for the CONTACT_SENSOR widget are defined. To add a CONTACT_SENSOR widget with a pre-made template, drag an object from the list of objects to the interface page:



Filled CONTACT_SENSOR widget:



Widget appearance in the myGrenton application:



3.25. Contact Sensor Double (CONTACT_SENSOR_DOUBLE)

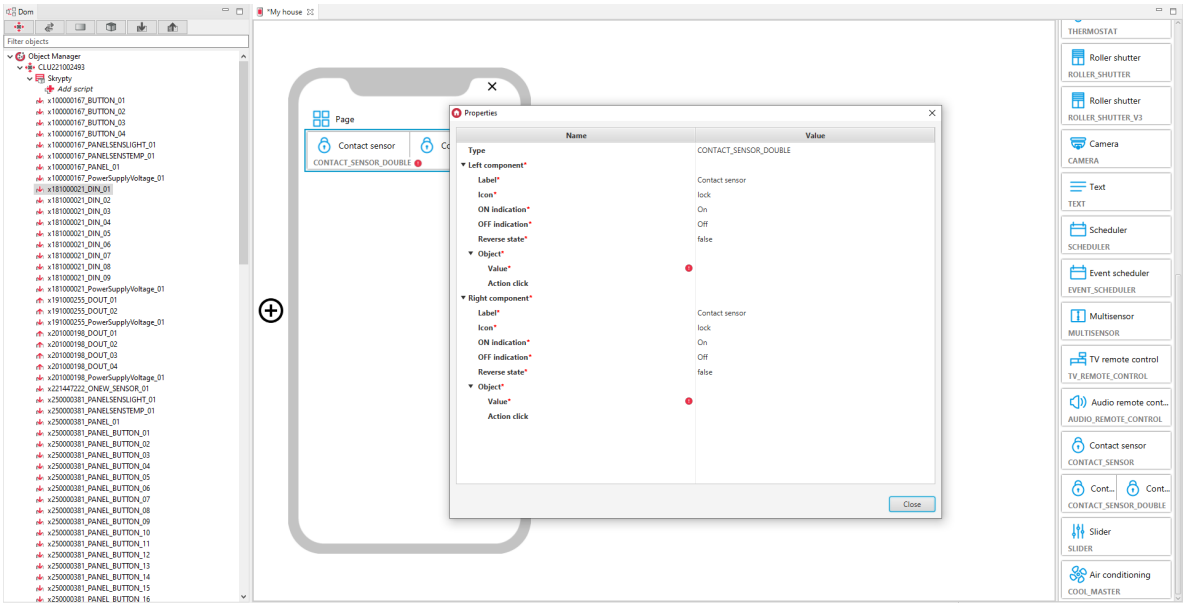
> **Note!**
> The CONTACT_SENSOR_DOUBLE widget is available for Object Manager version 1.7.0 or higher, for myGrenton application version 1.5.0 (Android) / 1.9.0 (iOS) or higher.

This is the Contact Sensor double-version widget. Widget dedicated to displaying On / Off states based on the value of objects, features. The CONTACT_SENSOR_DOUBLE widget contains:

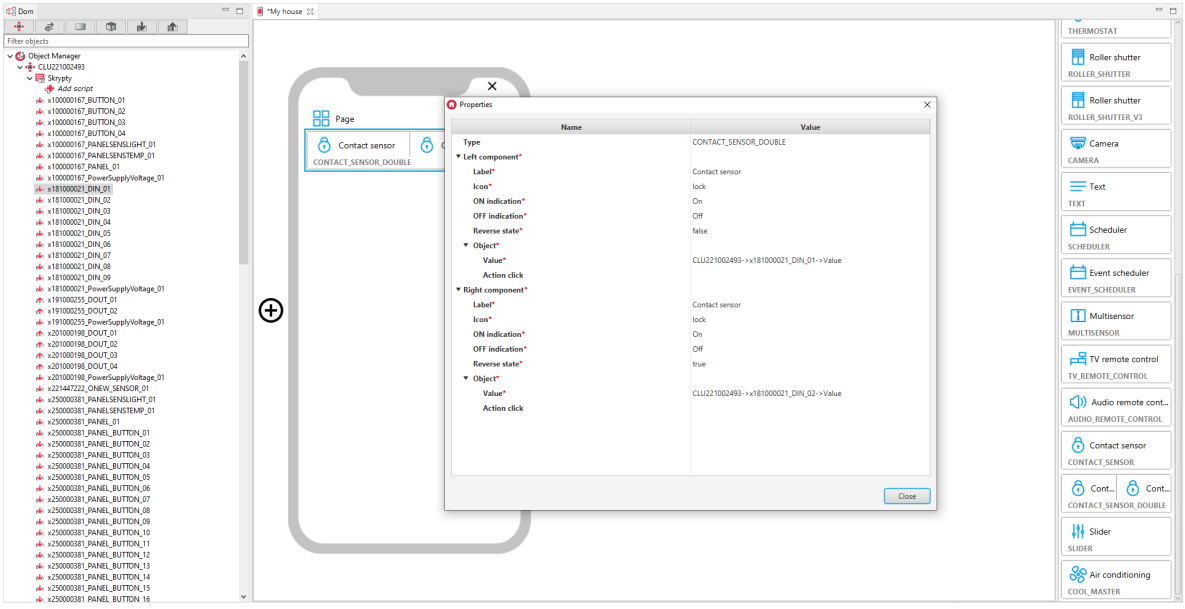
- support for On / Off states - based on the value of objects, features (`0/1` or `true/false`), the widget status is properly displayed,
- indications - it is possible to set any name of On / Off states through the properties `Indication ON` , `Indication OFF` ,
- support for state reversal (`0` - `Enabled` , `1` - `Disabled`)

> **Note!**
> Functionality is available for Object Manager version 1.10.0 or higher, myGrenton application 1.9.17 (Android) / 1.12.1 (iOS) or higher.

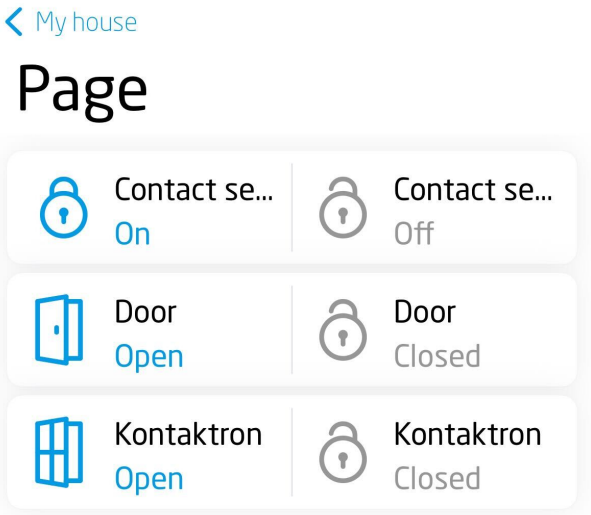
- optional click action - action triggered when clicking on the widget (any action can be assigned).



Filled CONTACT_SENSOR_DOUBLE widget:



Widget appearance in the myGrenton application:

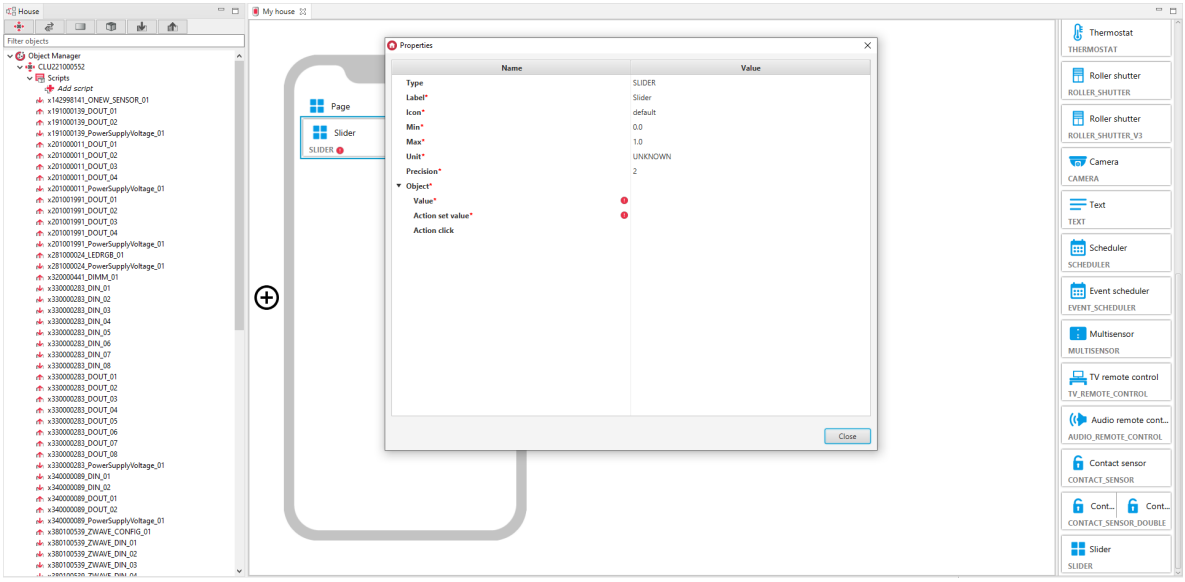


3.26. Slider (SLIDER)

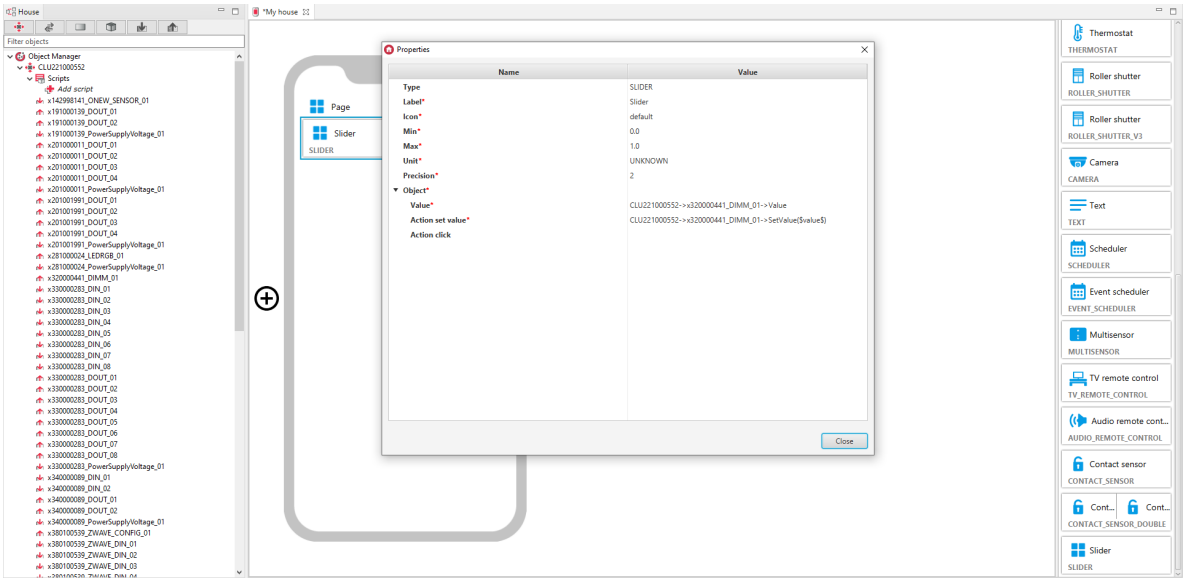
> **Note!**
> The SLIDER widget is available for Object Manager version 1.7.0 or higher, for myGrenton application version 1.5.0 (Android) / 1.9.0 (iOS) or higher.

Widget dedicated to smooth value control. The SLIDER widget contains:

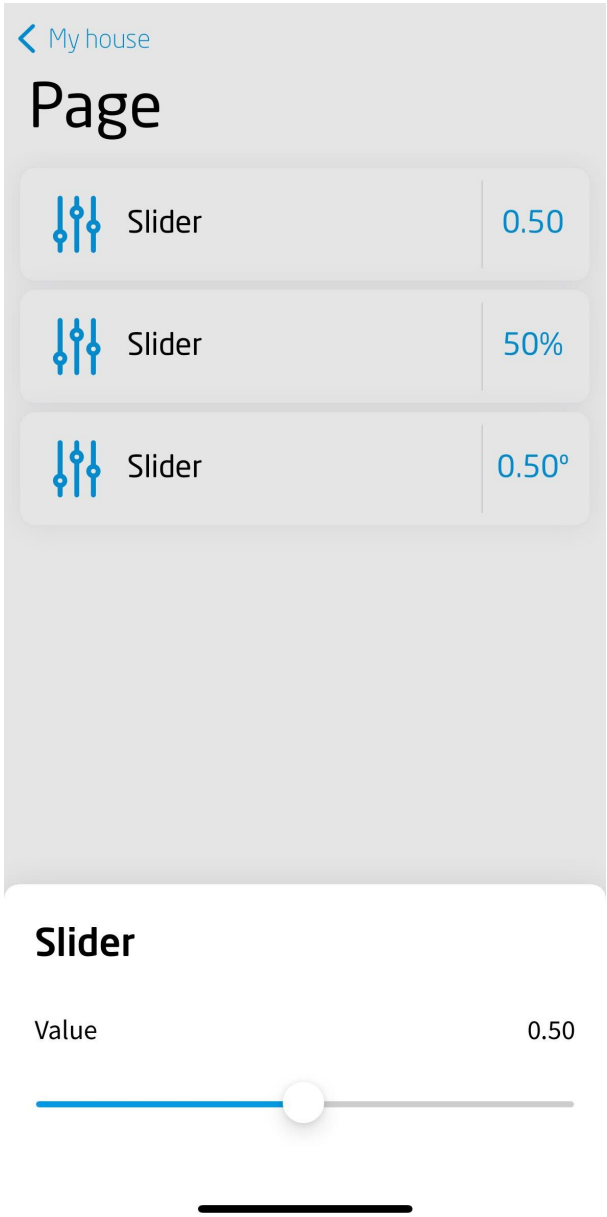
- unit - units to choose from: `UNKNOWN`, `PERCENT`, `DEGREE`,
- value - current output value displayed in the right part of the widget. For the `PERCENT` unit, calculated on the basis of the `Min`, `Max` properties set, for the `UNKNOWN`, `DEGREE` units, the value is displayed with the number of decimal places specified in the `Precision` property,
- slider - value set on the basis of the given range (`Min`, `Max` properties) and precision (the `Precision` property determines the number of decimal places of the set value),
- optional click action - action triggered when clicking on the widget (any action can be assigned).



Filled SLIDER widget:



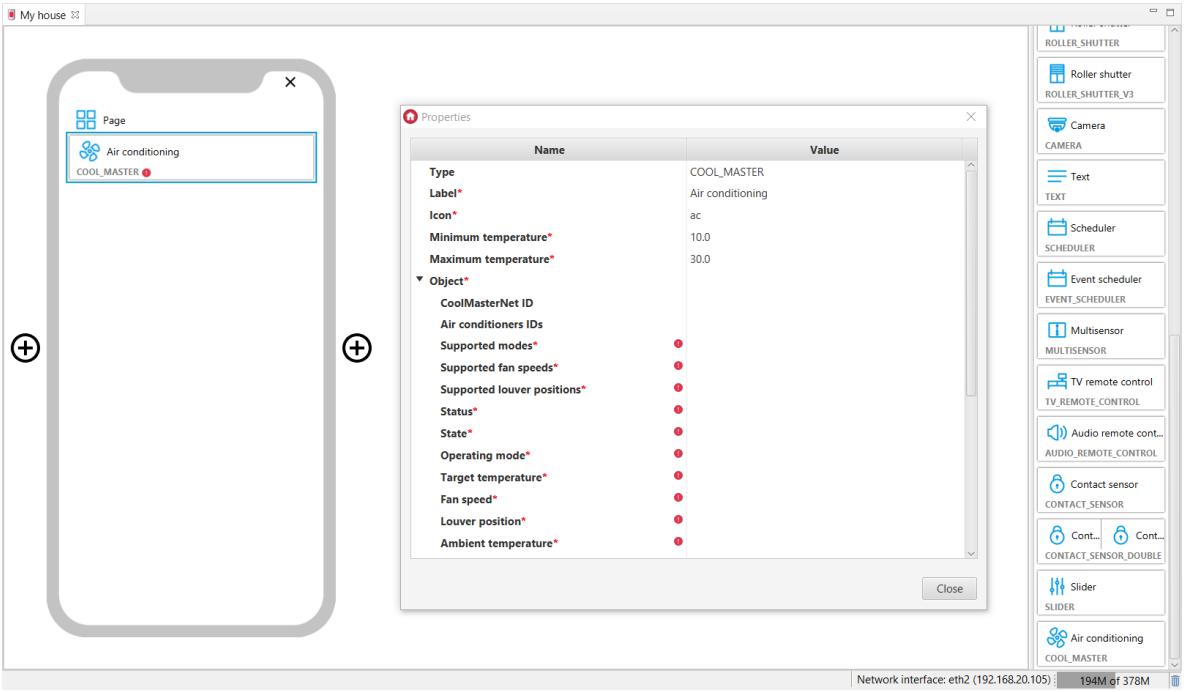
Widget appearance in the myGrenton application:



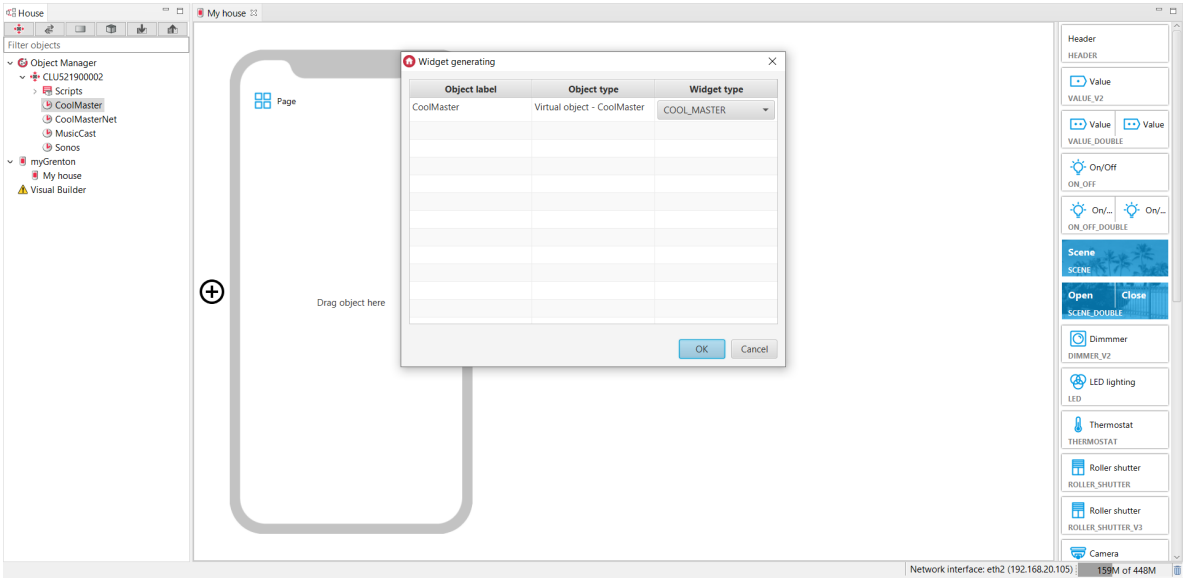
3.27. Air conditioning (COOL_MASTER)

> **Note!**
> The COOL_MASTER widget is available for Object Manager version 1.8.0 or higher, for myGrenton application version 1.6.1 (Android) / 1.10.0 (iOS) or higher.

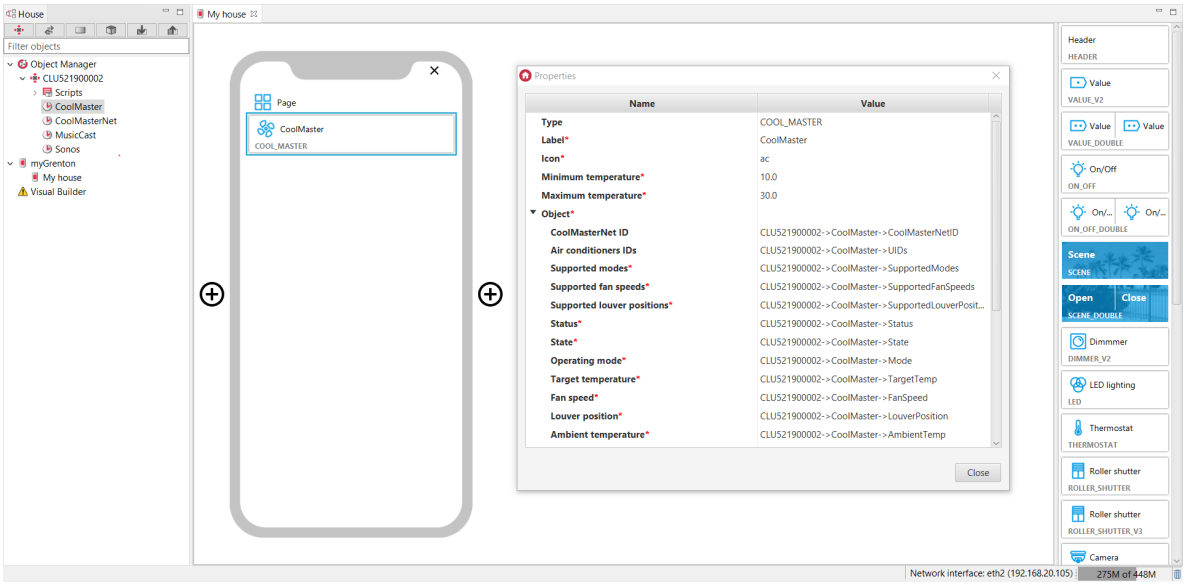
Widget dedicated to work with the CoolMaster virtual object. It is used to control the air conditioning. Allows turn on/off the air conditioner, control such parameters as: set temperature `TargetTemp`, `Mode` operating mode, `FanSpeed` fan speed, `LouverPosition` level of the air flow regulating slot and displays the current ambient temperature `AmbientTemp`.



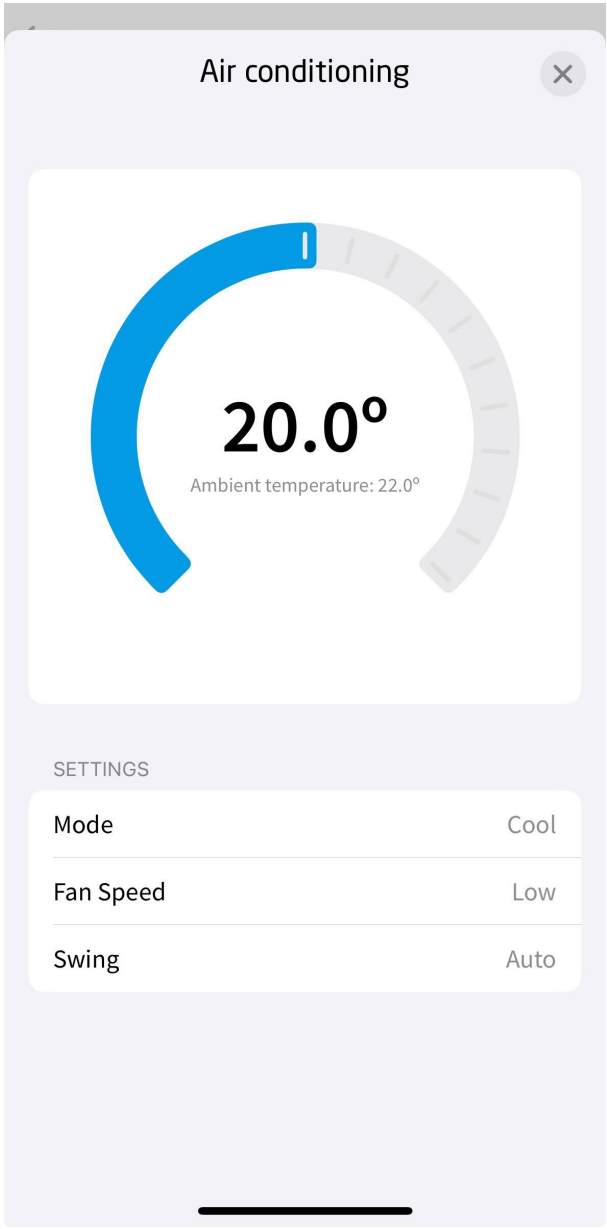
For thermostats, ready-made templates for the COOL_MASTER widget are defined. To add a COOL_MASTER widget with a ready template, drag the virtual object CoolMaster from the list of objects to the interface page:



Filled COOL_MASTER widget:



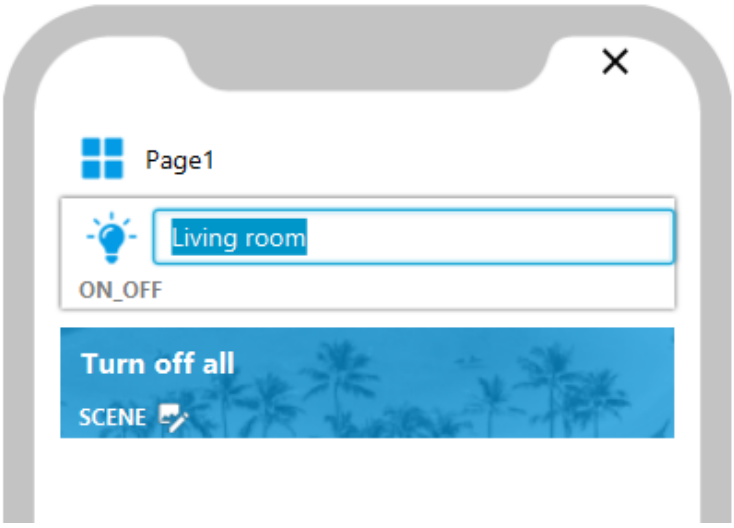
Widget appearance in the myGrenton application:



3.28. Personalization of the widget

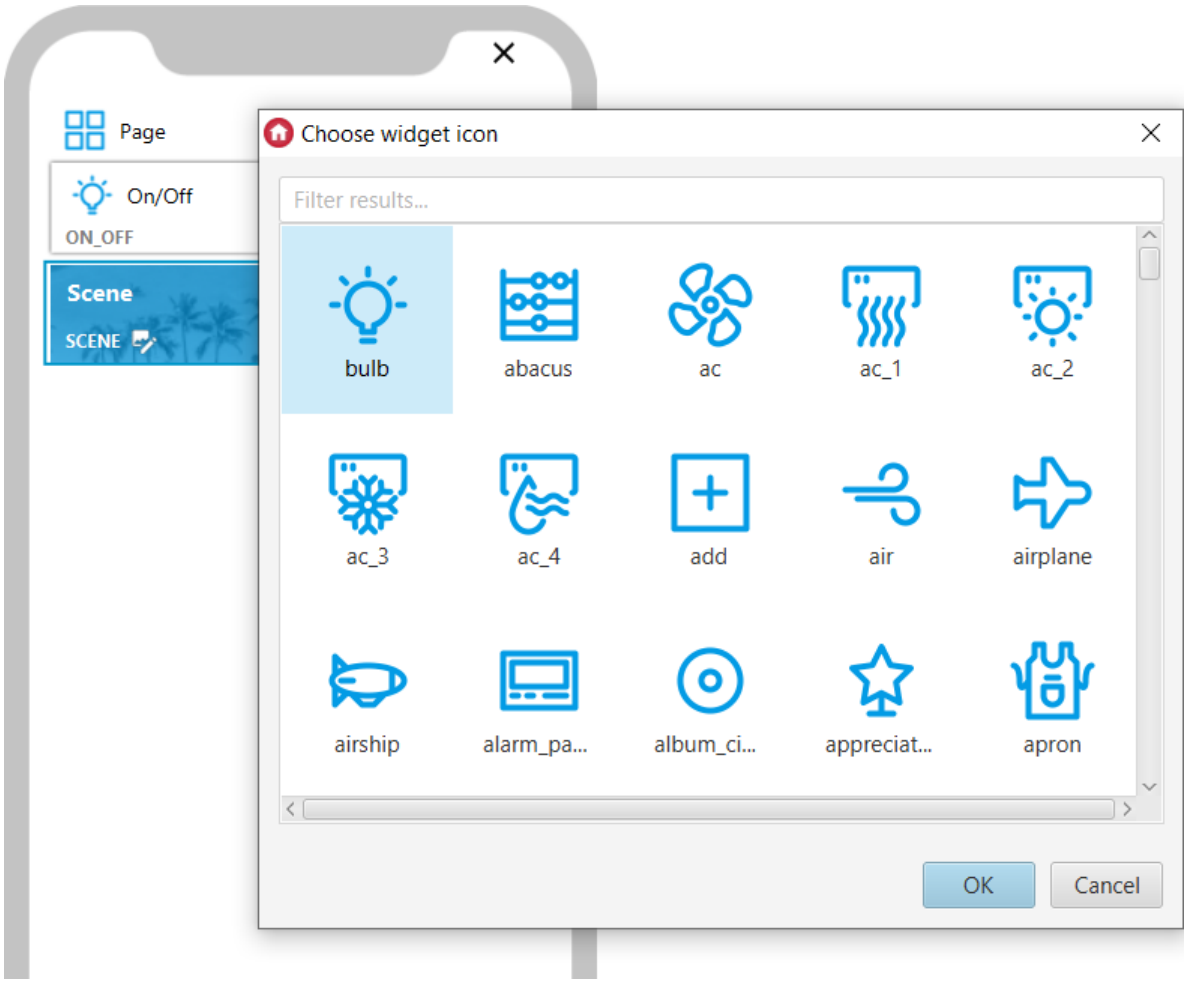
A. Change the name

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.

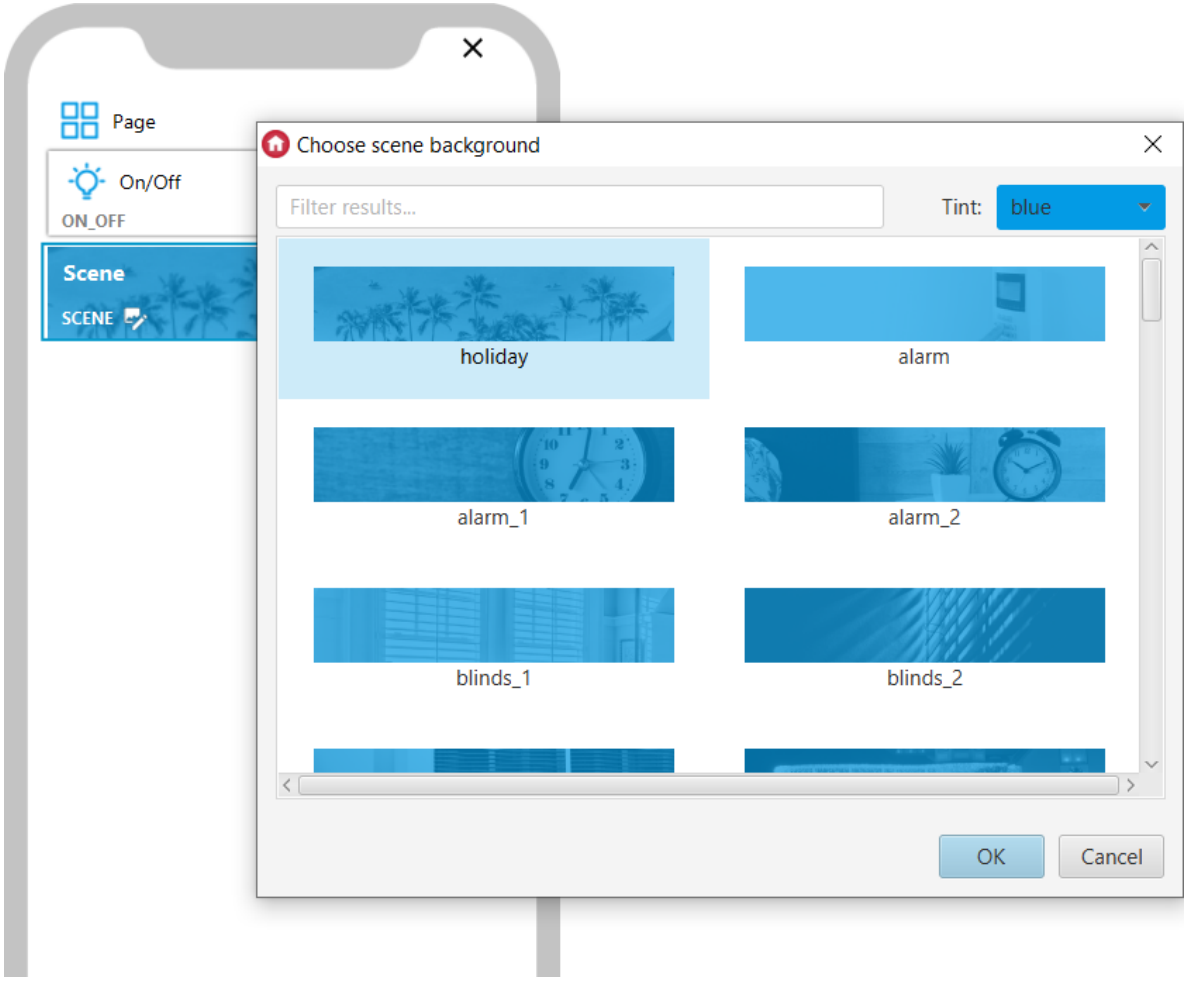


B. Change icon / background picture

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.



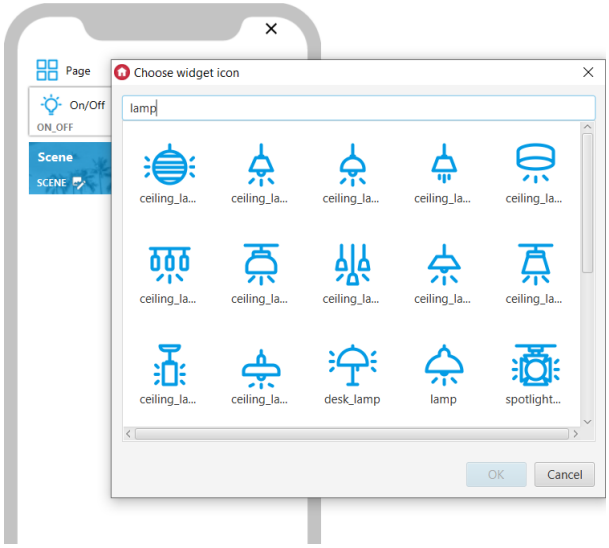
The background image of the Scene or Open / Close widget can be quickly changed by clicking on the edit icon under the widget name. Additionally, it is possible to choose the shade of the picture from the available 15 colors.

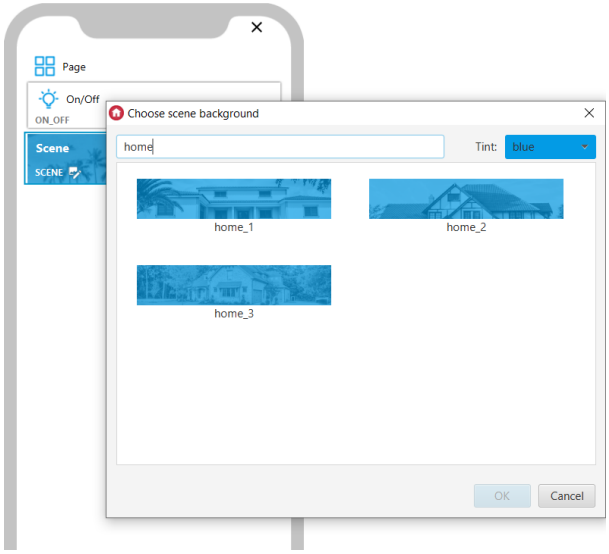


Note!
If an icon or background image is not supported in a given version of myGrenton, it will be replaced with the default icon / background image.

Icon/background filtering

In the selection window, it is possible to filter icons or backgrounds based on the entered phrase.





Note!
Icon filtering functionality is available for Object Manager version 1.8.0 or higher.

3.29. 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.

3.30. 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.

3.31. Run the SCENE widget using Shortcuts

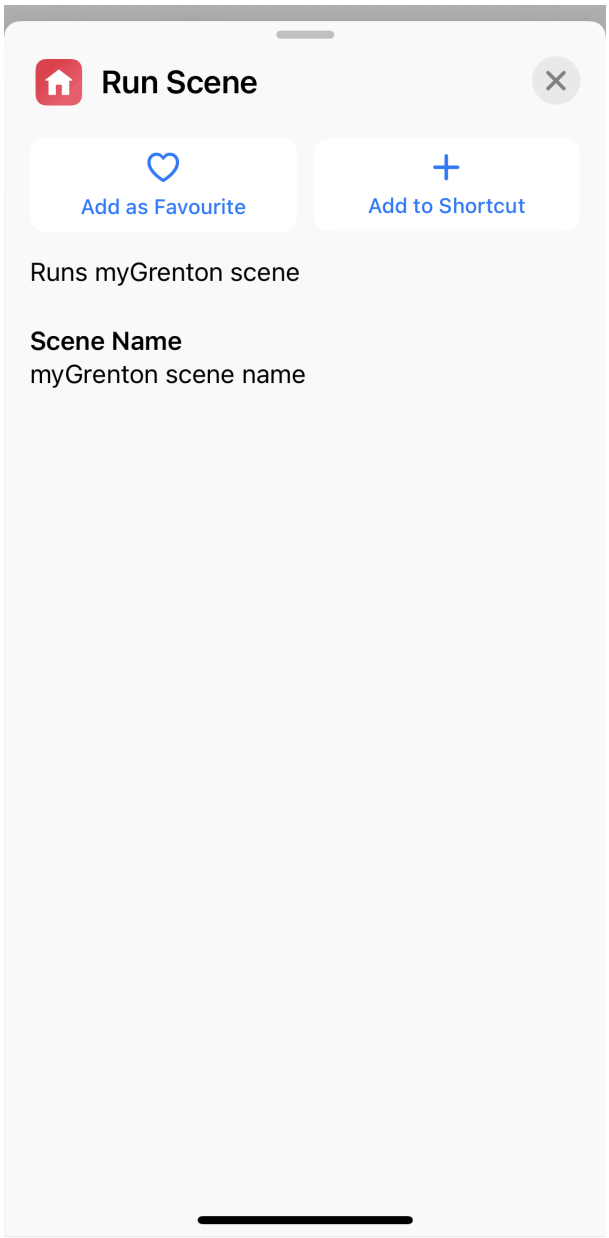
Note!
Functionality not available for PIN-locked widgets and interfaces.

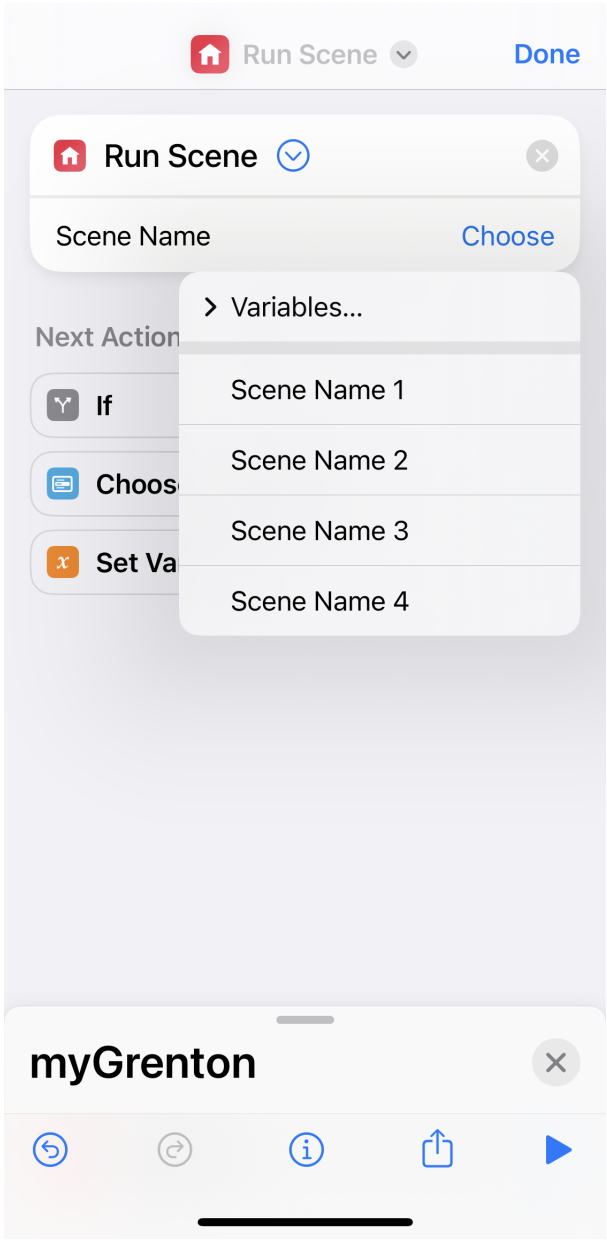
A. Using shortcuts in myGrenton iOS

Note!
This functionality is available for myGrenton application version 1.11.0 iOS or higher.

Note!
This functionality is available for mobile devices with iOS 16 higher.

The Shortcuts application allows you to create a shortcut for the Run Scene action, which is the equivalent of the SCENE widget from the myGrenton application.





Adding the SCENE widget in a shortcut is only possible for the interface with Cloud access checked (in the Interface Settings, the "Use cloud" switch is turned on). SCENE widgets are available from all interfaces in the myGrenton application, both active and inactive.

Shortcuts are launched only by the Cloud, regardless of the current connection type of the myGrenton application to the system.

Running a scene is also possible using Siri. Suggested phrases for invoking the myGrenton scene with Siri:

- "Run scene with myGrenton",
- "Invoke scene with myGrenton",
- "Run (Scene Name - name of the SCENE widget from the myGrenton application) with myGrenton",
- "Invoke (Scene Name - name of the SCENE widget from the myGrenton application) with myGrenton".

More information on how shortcuts work, their functionality and how to call them in Siri can be found in the documentation of the Shortcuts application.

Note!
If communication is lost as a result of an unstable connection, the shortcut will make another attempt to establish a connection and call the scene. It is not recommended to run scenes operating on the principle of switching to the opposite state.

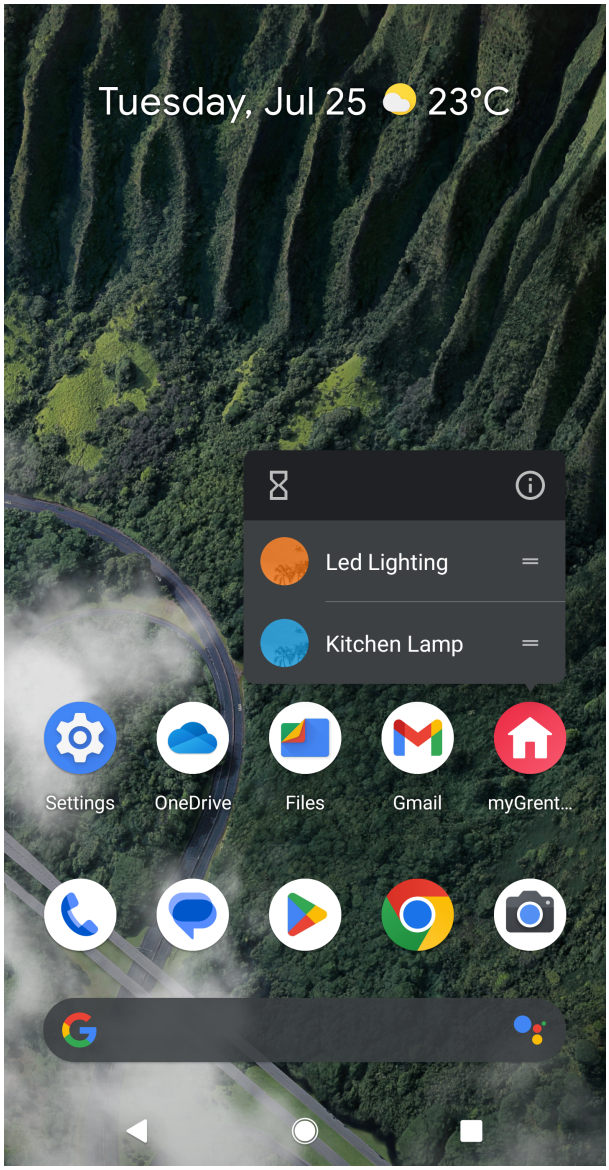
The functionality should not be treated as a replacement for the myGrenton application.

B. Using shortcuts in myGrenton Android

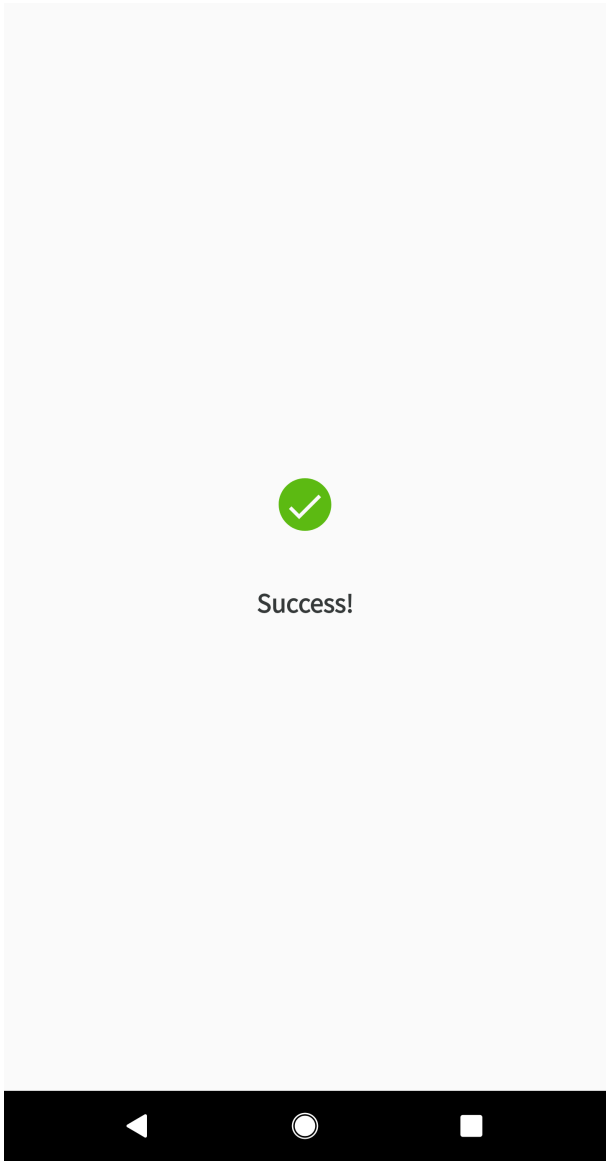
Note!
This functionality is available for myGrenton application version 1.7.0-232901 Android or higher.

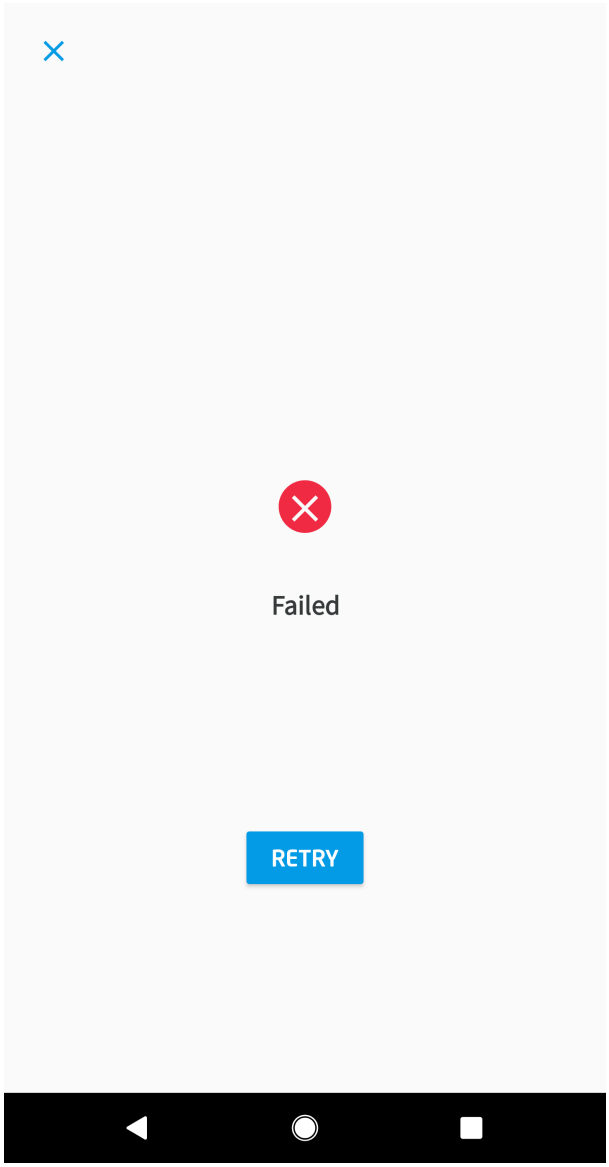
Note!
This functionality is available for devices in which the desktop manager (launcher) supports application shortcuts.

To use the shortcut, launch the SCENE widget in the previously uploaded interface. Next, exit the application and hold down the myGrenton application icon until the context menu appears. The shortcut associated with the SCENE widget will appear in the list. To add it to the desktop, hold down the shortcut and move it to the desired location on the desktop.

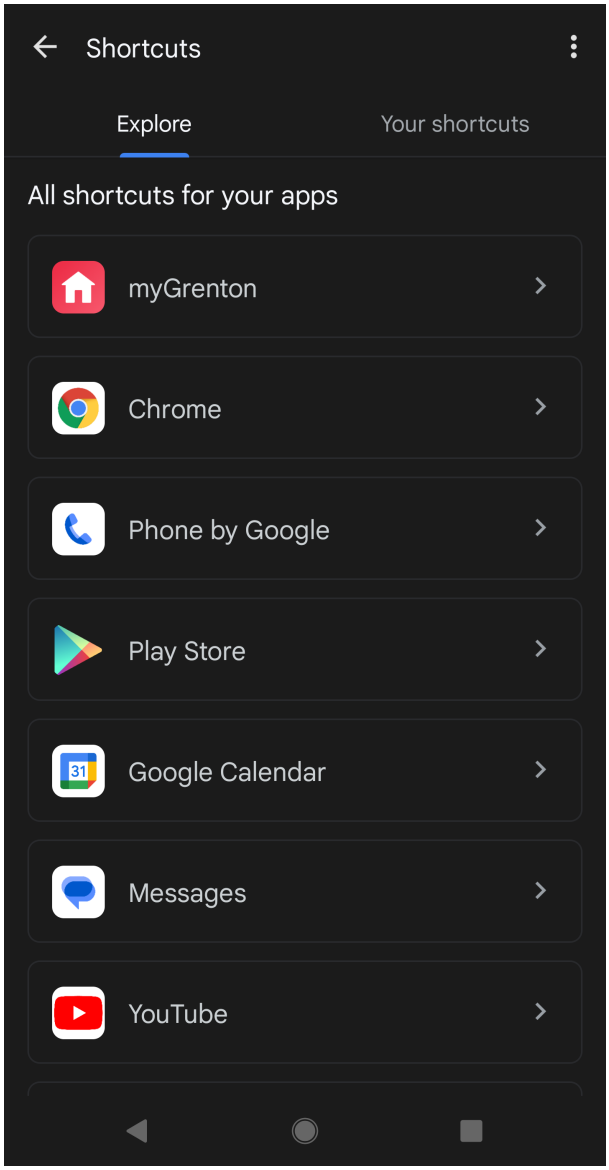


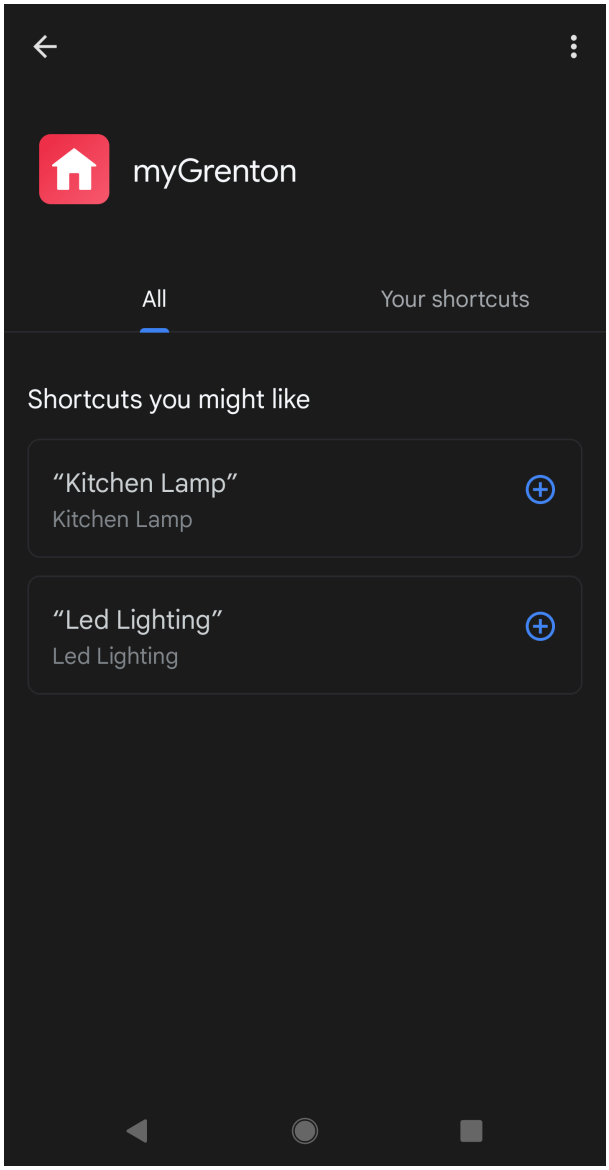
After clicking on the added shortcut on the desktop, if the application has a correct connection with the CLU, the method will be called and the message *Success!* will be displayed, which will disappear after about 1 second. If the shortcut is not invoked correctly, the message *Failure!* will appear on the screen.

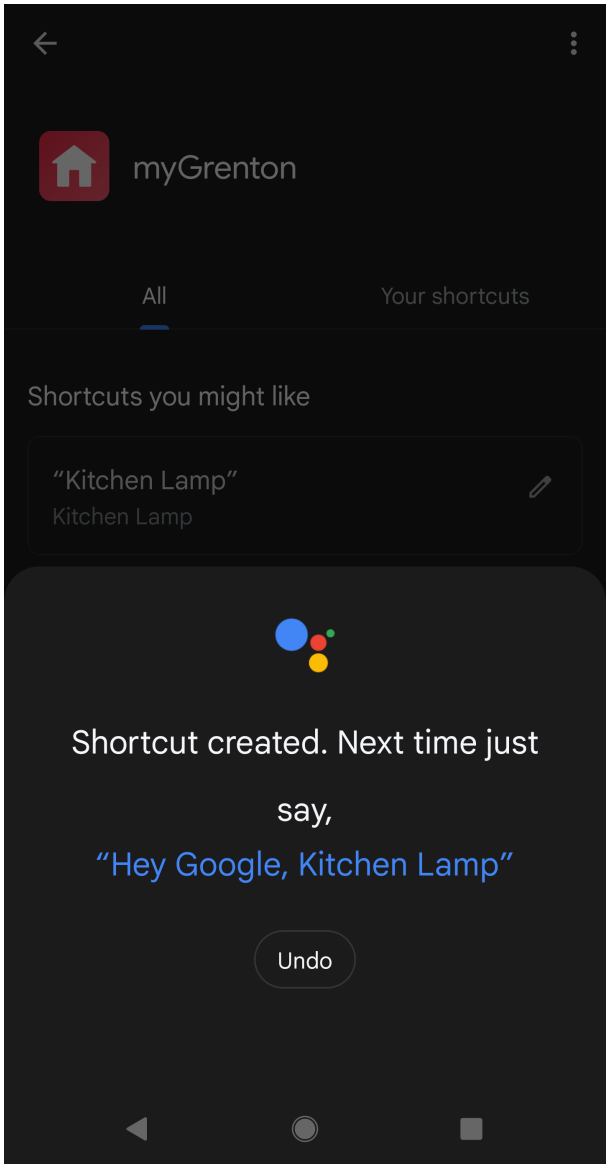

Success!



Running a shortcut is also possible using Google Assistant. To set a given shortcut, use the Shortcuts option in the Google Assistant settings. Then, find the myGrenton application on the list and add a shortcut with your own or default name using the - button.


Shortcuts
Explore
Your shortcuts
All shortcuts for your apps
myGrenton
Chrome
Phone by Google
Play Store
Google Calendar
Messages
YouTube


myGrenton
All
Your shortcuts
Shortcuts you might like
“Kitchen Lamp”
Kitchen Lamp
“Led Lighting”
Led Lighting

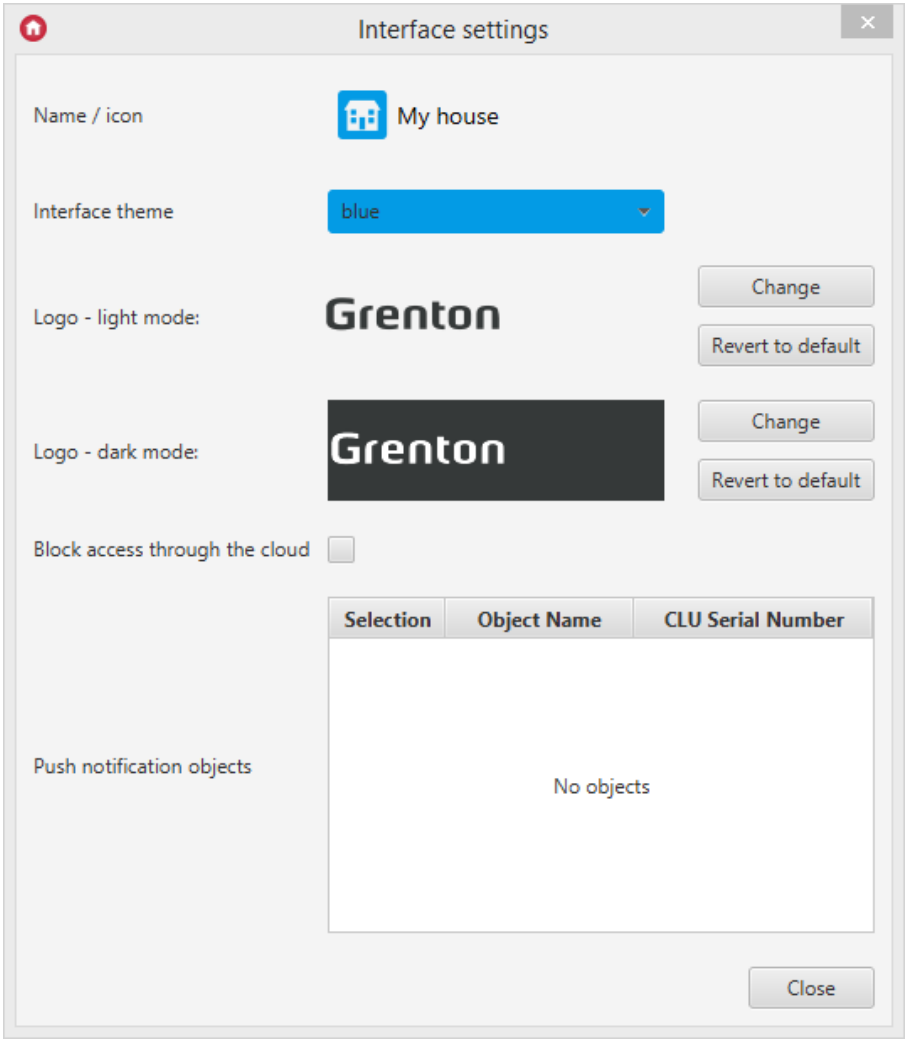


After performing the above steps and saying the name of the shortcut to the assistant, the method assigned to the SCENE widget will be invoked.

4. Personalization of the interface

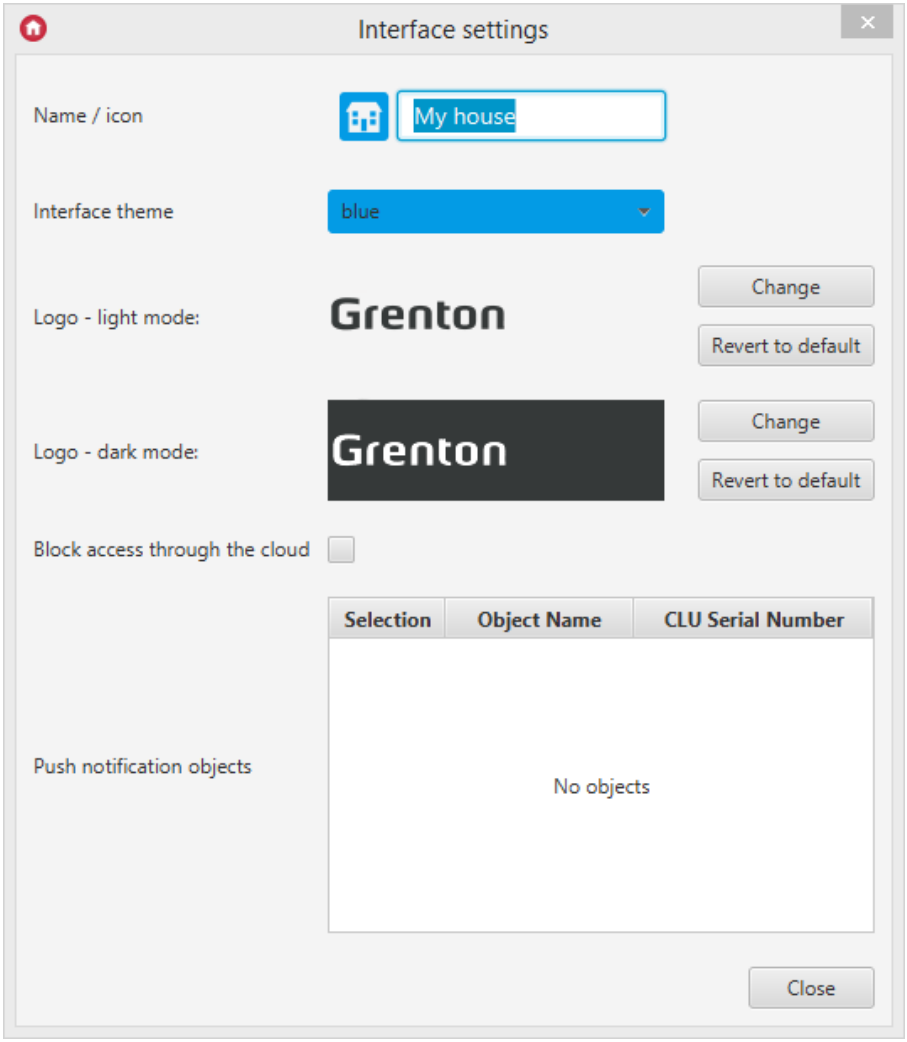
You can customize the appearance of the interface to your preferences. To personalize the interface, click the `MyGrenton interface settings` icon in the toolbar:

After clicking the icon, a window with the interface settings will appear:



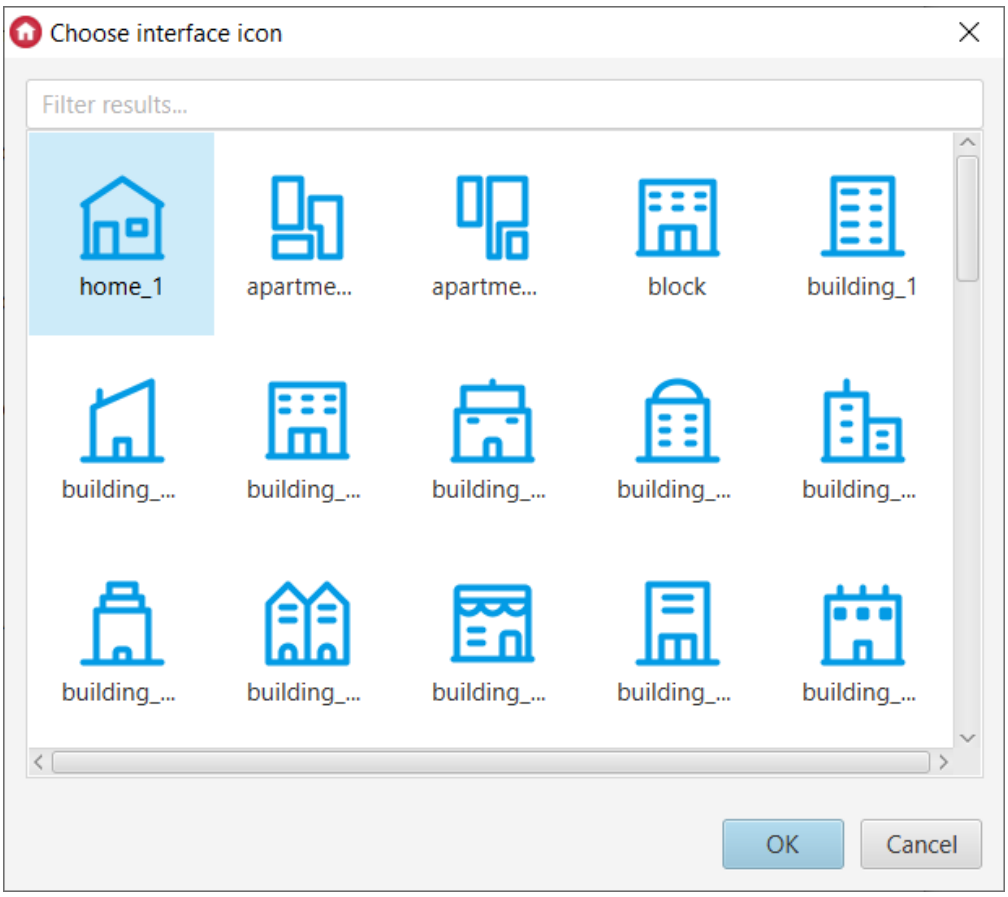
4.1. Change the name of the interface

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.

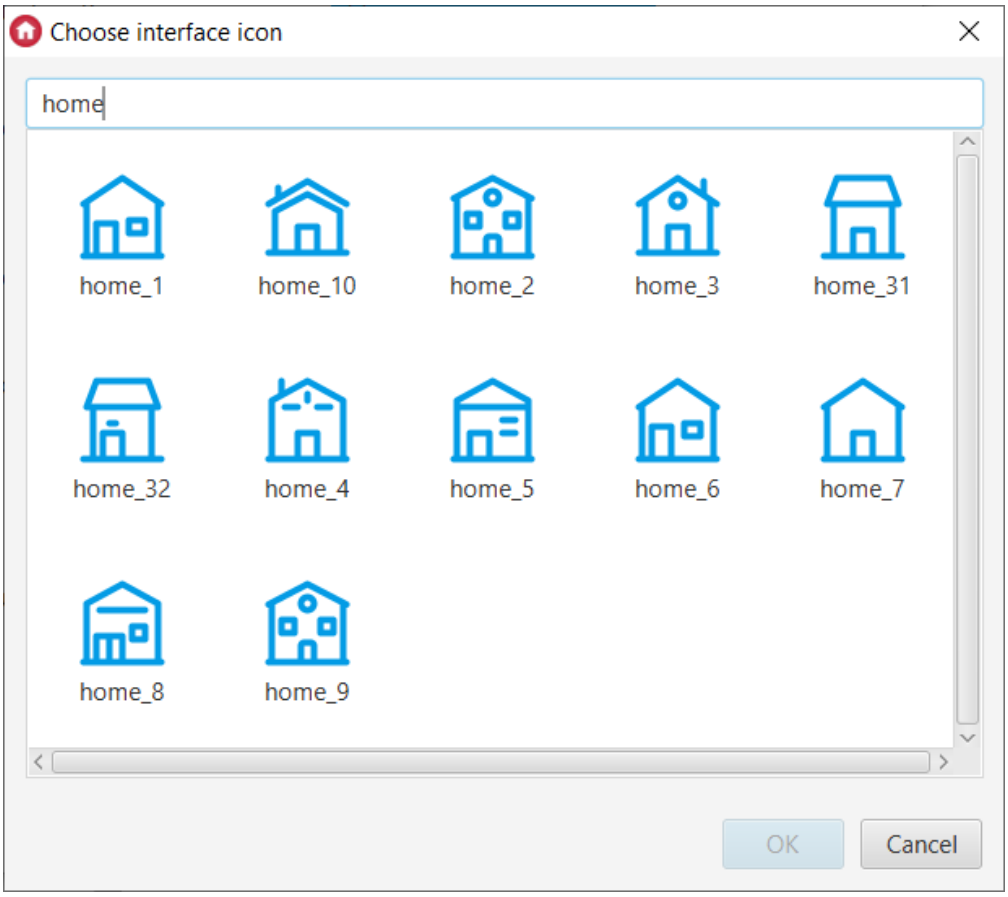


4.2. Change the interface icon

To change the interface icon, click on the icon image next to the interface name. A window with available icons will appear.



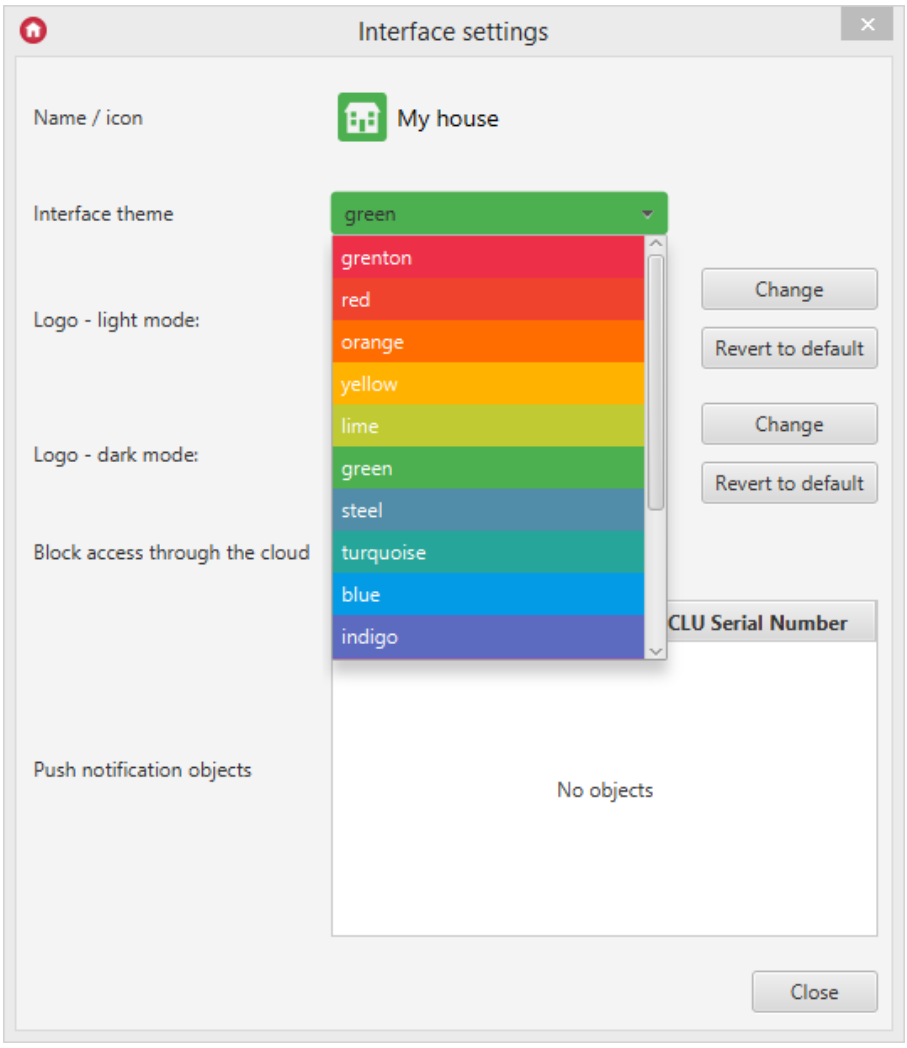
In the window, it is possible to filter icons based on the entered phrase.



Note!
Icon filtering functionality is available for Object Manager version 1.8.0 or higher.

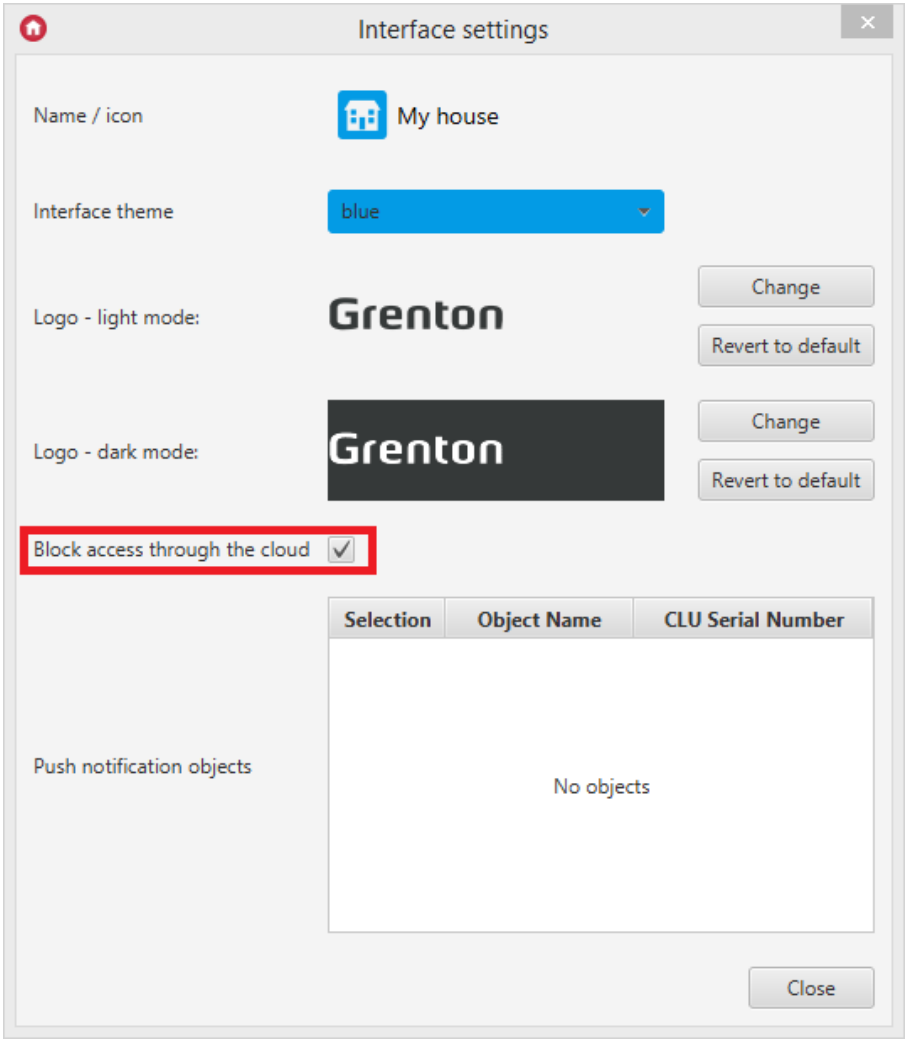
4.3. Change the color of the interface

It is also possible to change the color of the interface theme. There are 15 different theme colors available from the drop-down list.



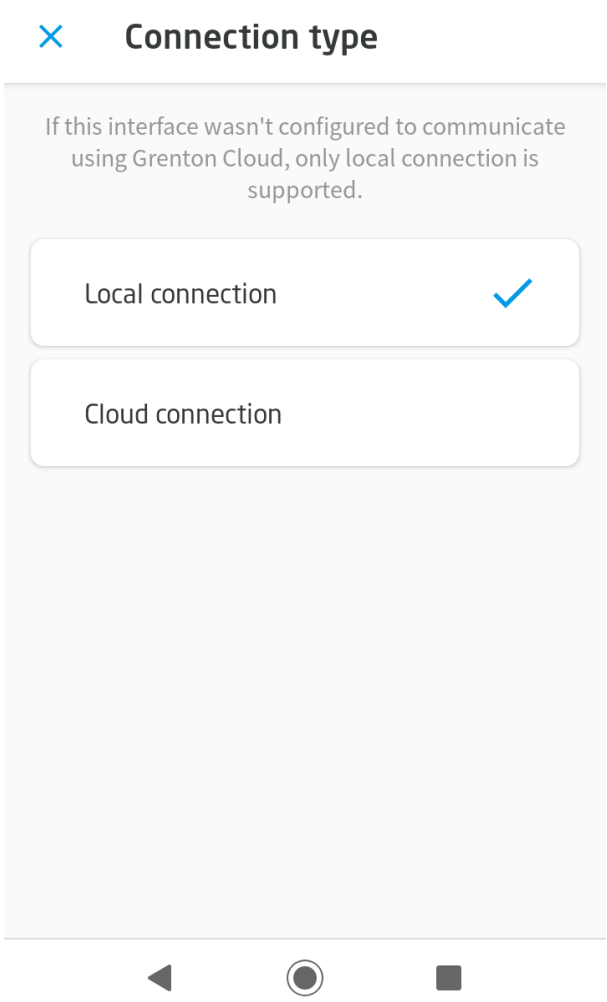
4.4. Blocking access through the cloud

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.

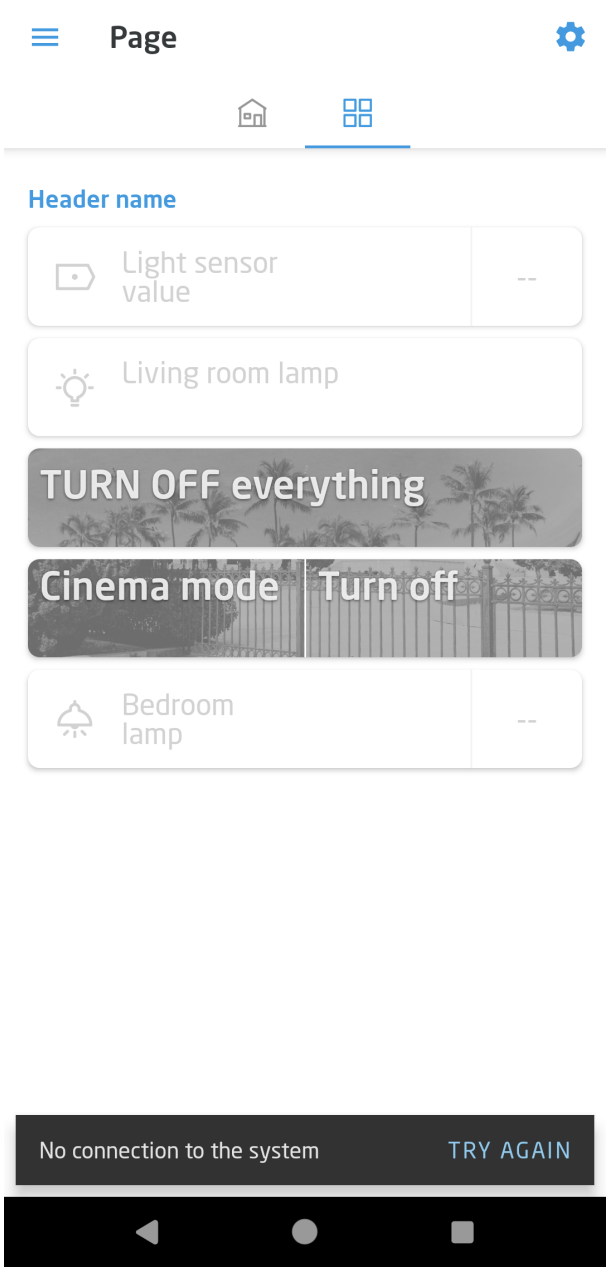


A. Android application behavior

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).

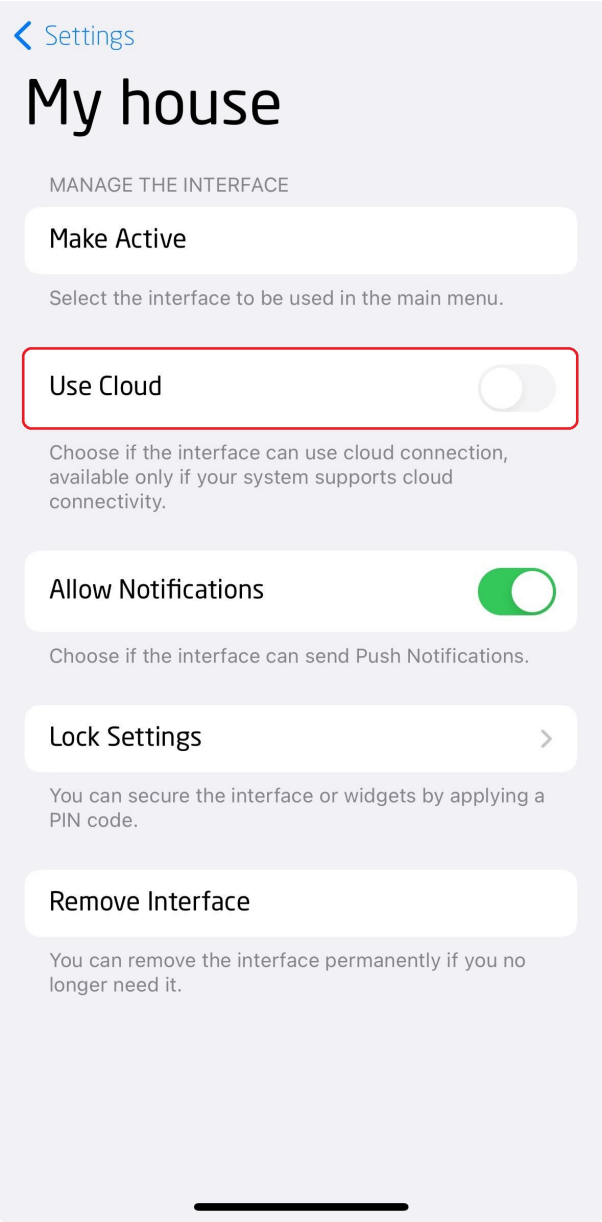


When changing the connection mode to connection via the cloud, connection with CLU will be impossible. A message will be displayed that there is no connection to the system.



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.

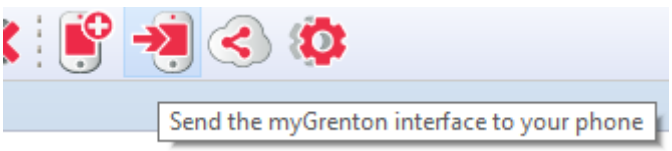


5. Sending the interface to the device

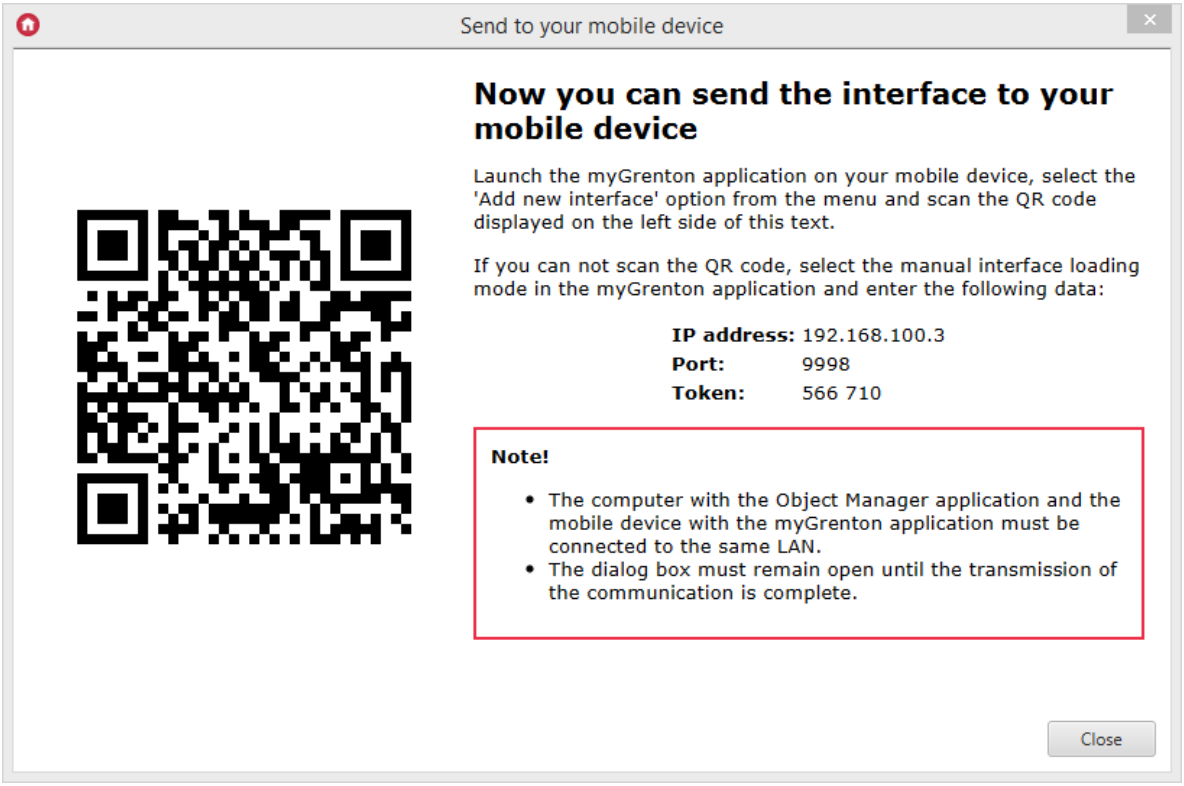
It is possible to send it to a mobile device in 2 ways:

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:



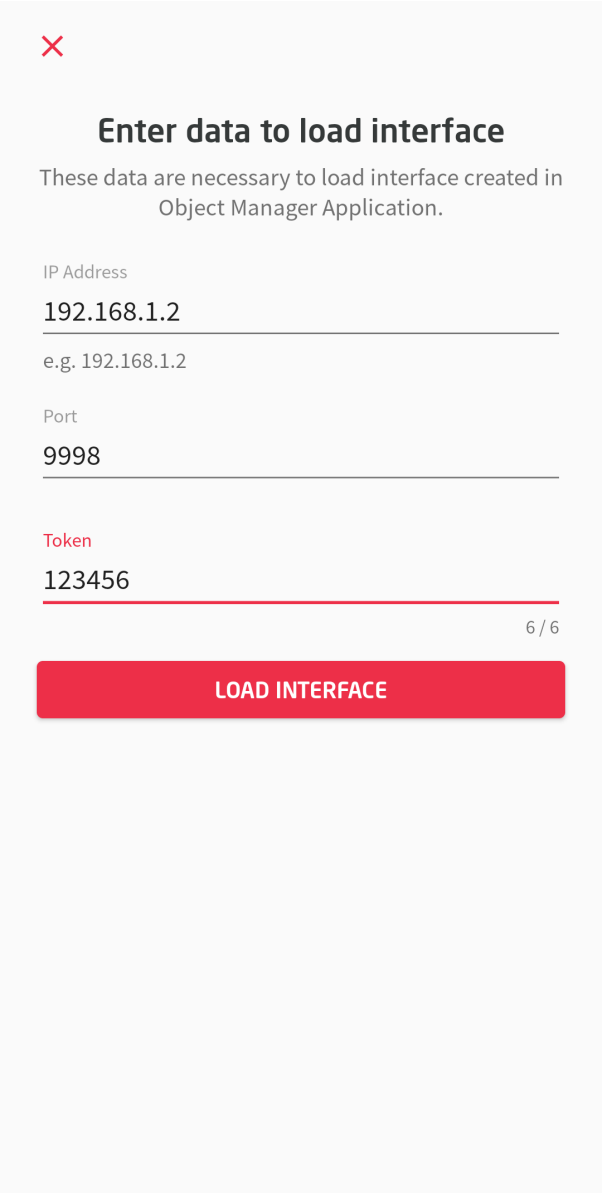
The window displayed allows you to send the interface by scanning the QR code or by providing interface parameters:



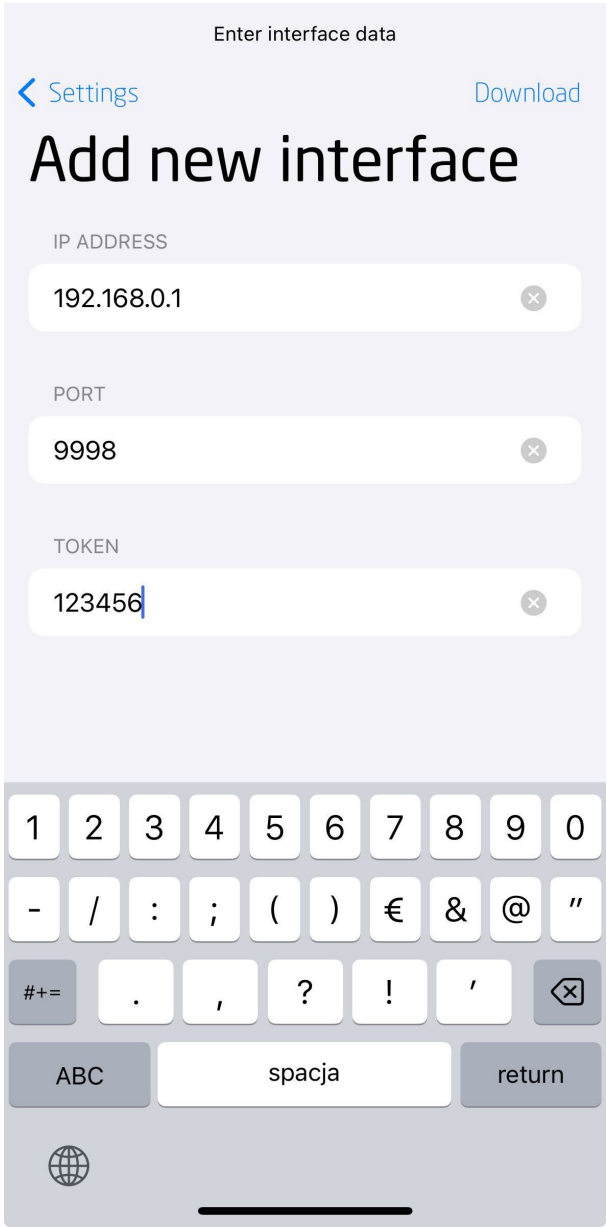
Note!
The computer with the Object Manager application and the mobile device with the myGrenton application must be connected to the same LAN.
The dialog window must remain open until the interface transfer is completed.

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`:



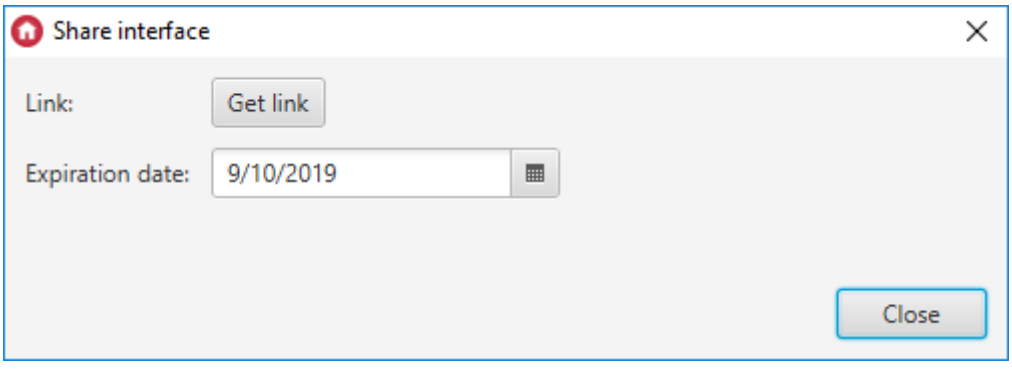
In the application for iOS devices, select the `Download` option:



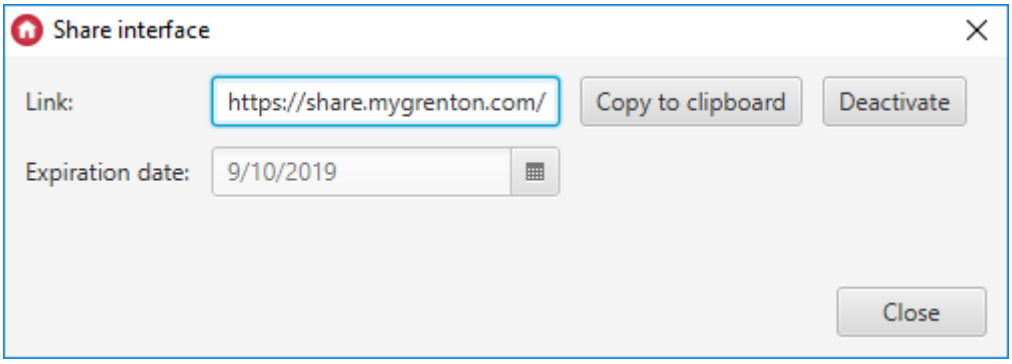
After correctly uploading the application will automatically launch the loaded interface.

5.2. Sharing myGrenton interface via the cloud

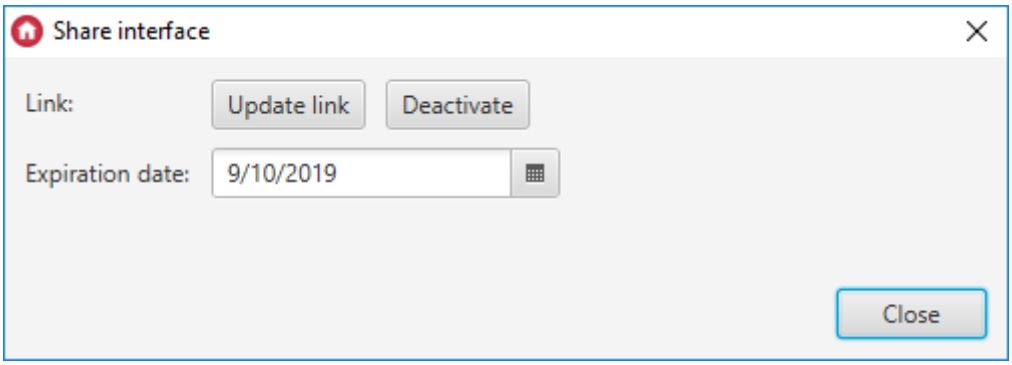
After clicking the icon `Share myGrenton interface in the cloud` the window for sharing the interface to the Grenton cloud will appear:



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

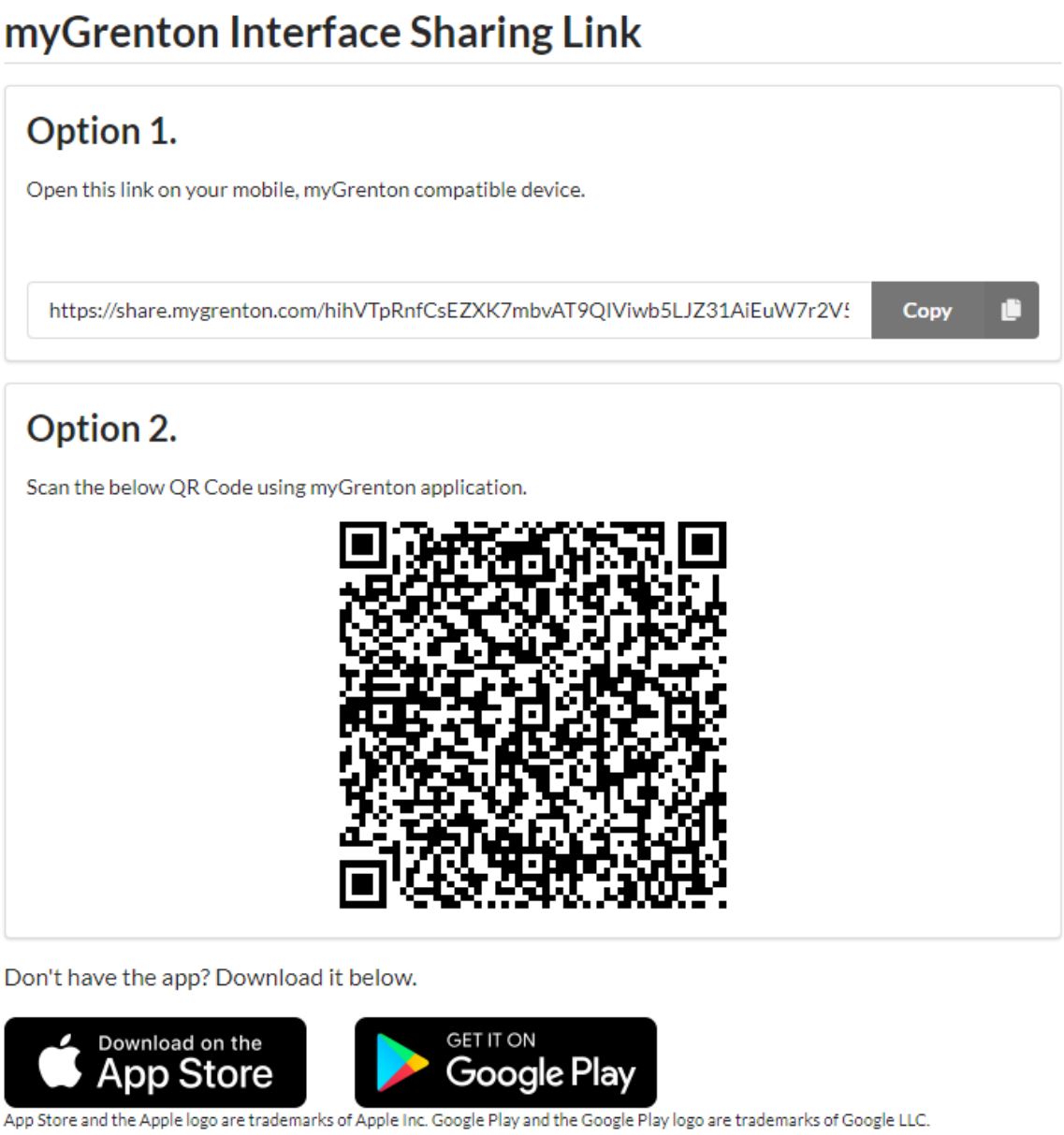


If the user has made changes to the interface, it is possible to update the shared interface:



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.



Note!
Sending an interface that contains widgets not supported in a given version of the application will execute correctly, unsupported widgets will be skipped and will not be visible.

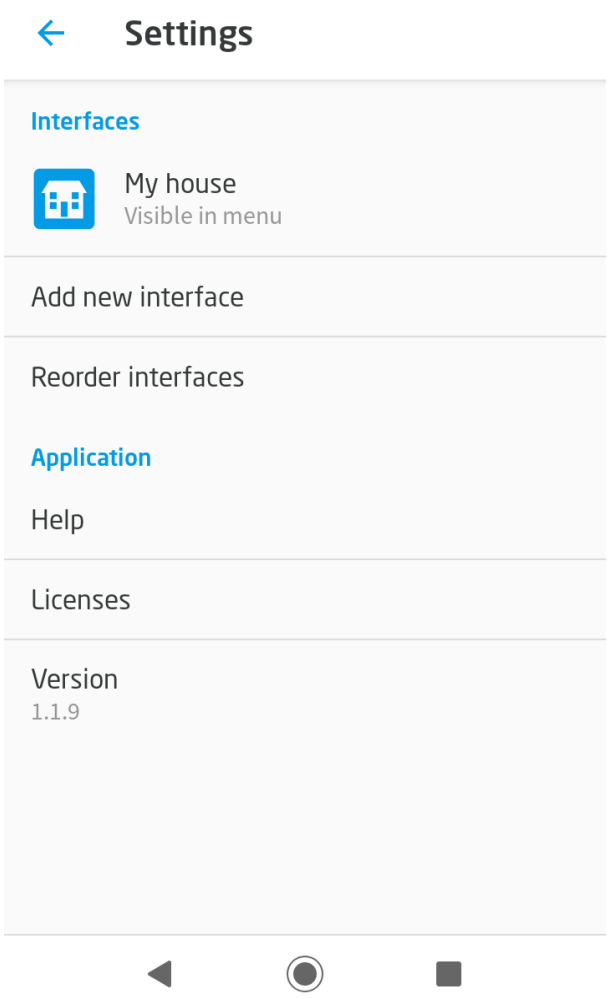
Functionality available with myGrenton application version 1.5.0 (Android) / 1.9.0 (iOS) or higher.

6. Application and interface settings

6.1. Application settings

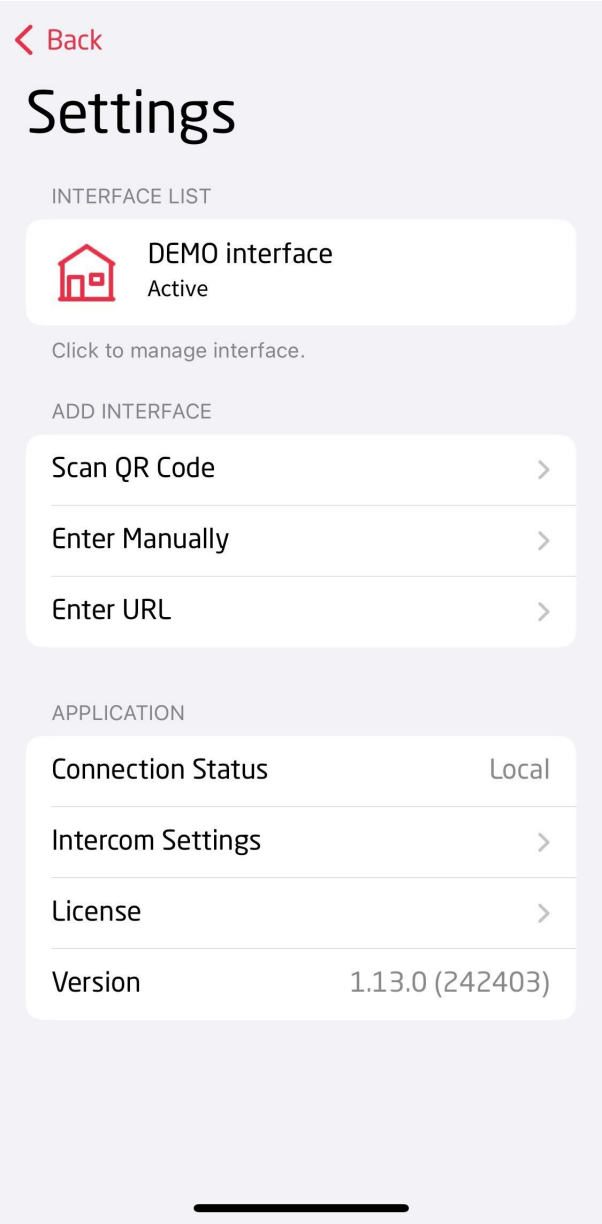
A. Android

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

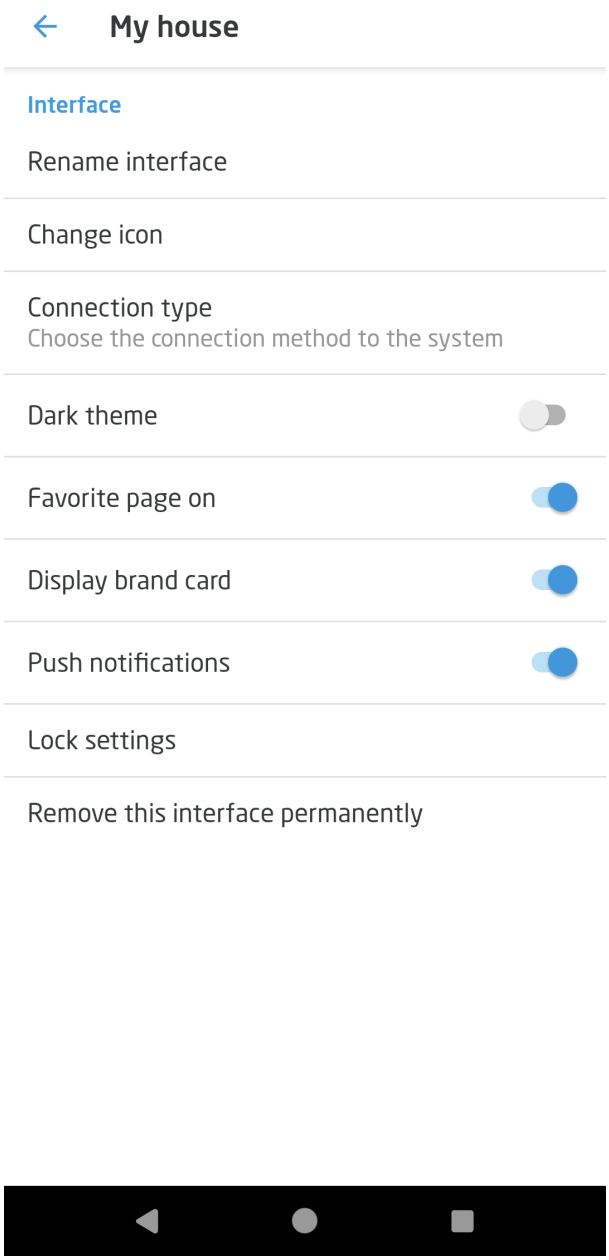
In the settings, we can add an interface by scanning the QR code, manually entering data or entering the URL address of the interface shared via the cloud. Here we can also see the current status of the connection to the system and the version of the current myGrenton application. The `License` button opens the page where the licenses are shown.



6.2. Interface Settings

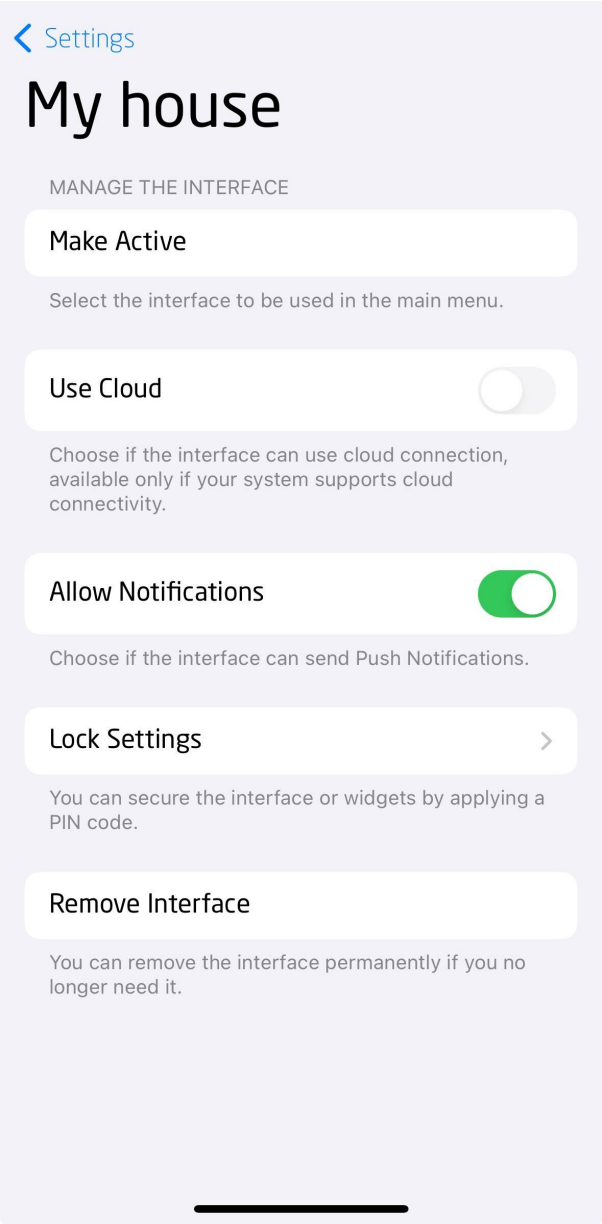
A. Android

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, select page visibility, display a card with a logo on the home page, set permissions to display push notifications, change lock settings and also remove the interface.



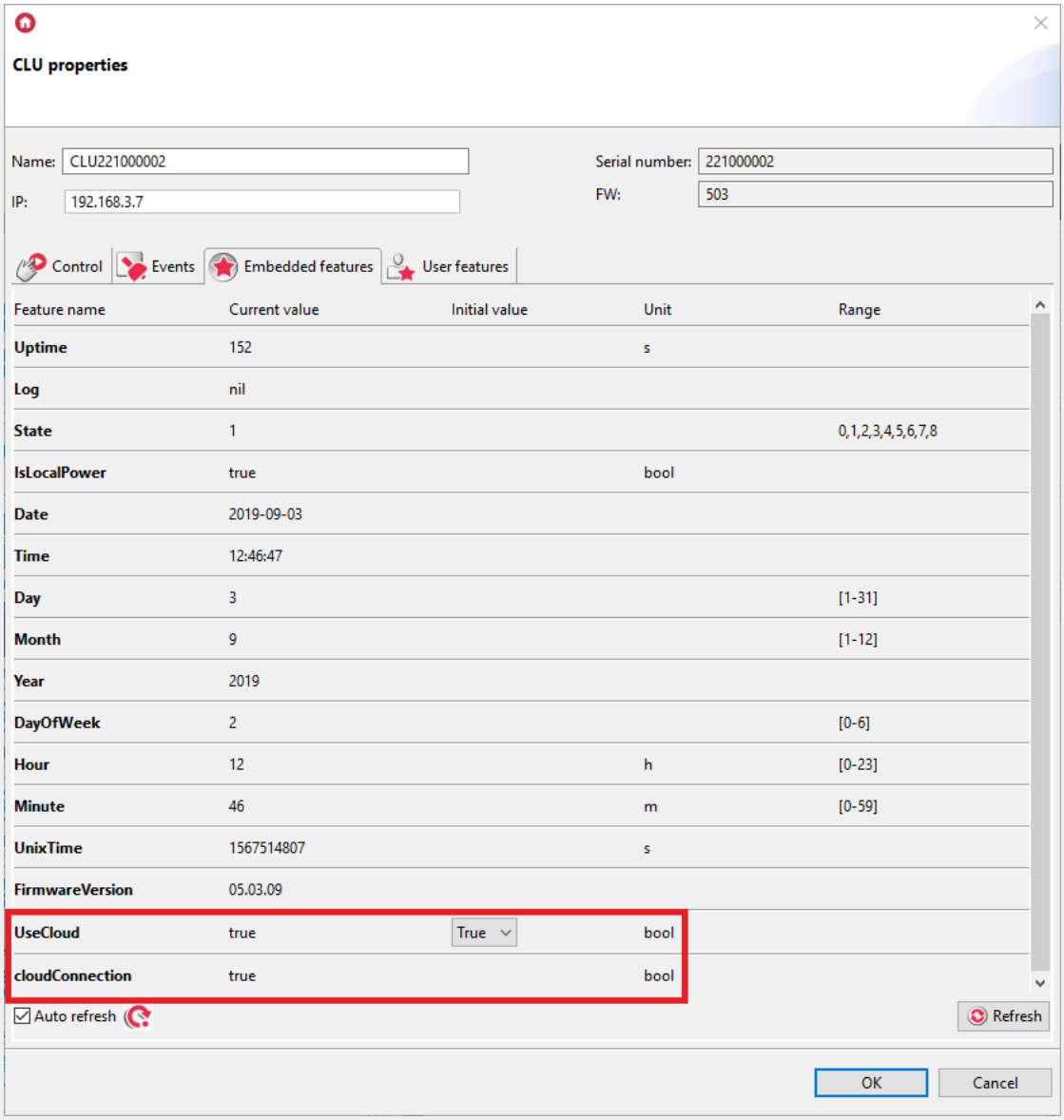
B. iOS

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, change lock settings or remove the interface.



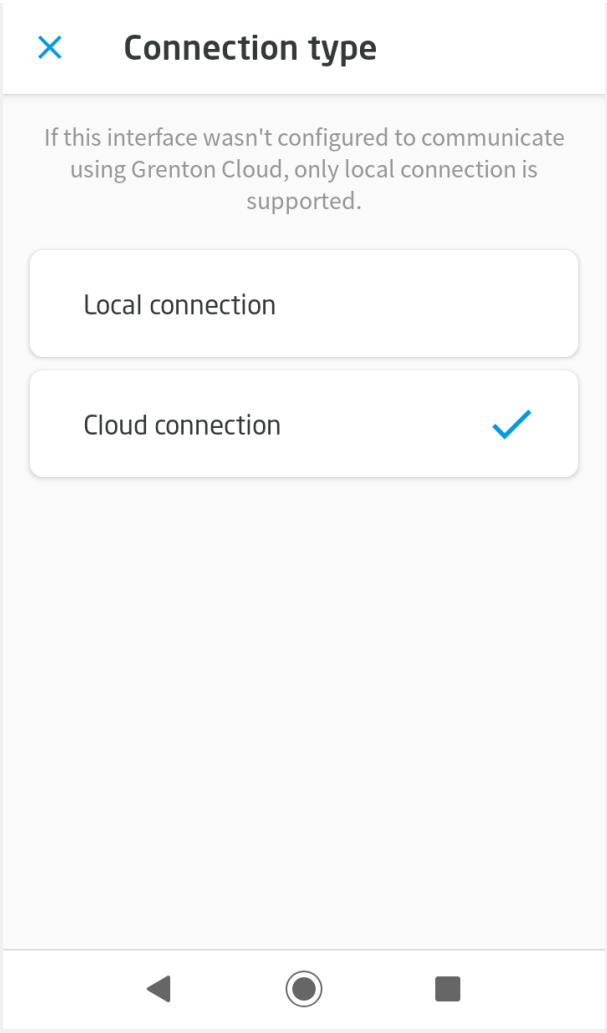
Note!
Required minimum CLU version for cloud support: **05.03.06**

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.



Note!

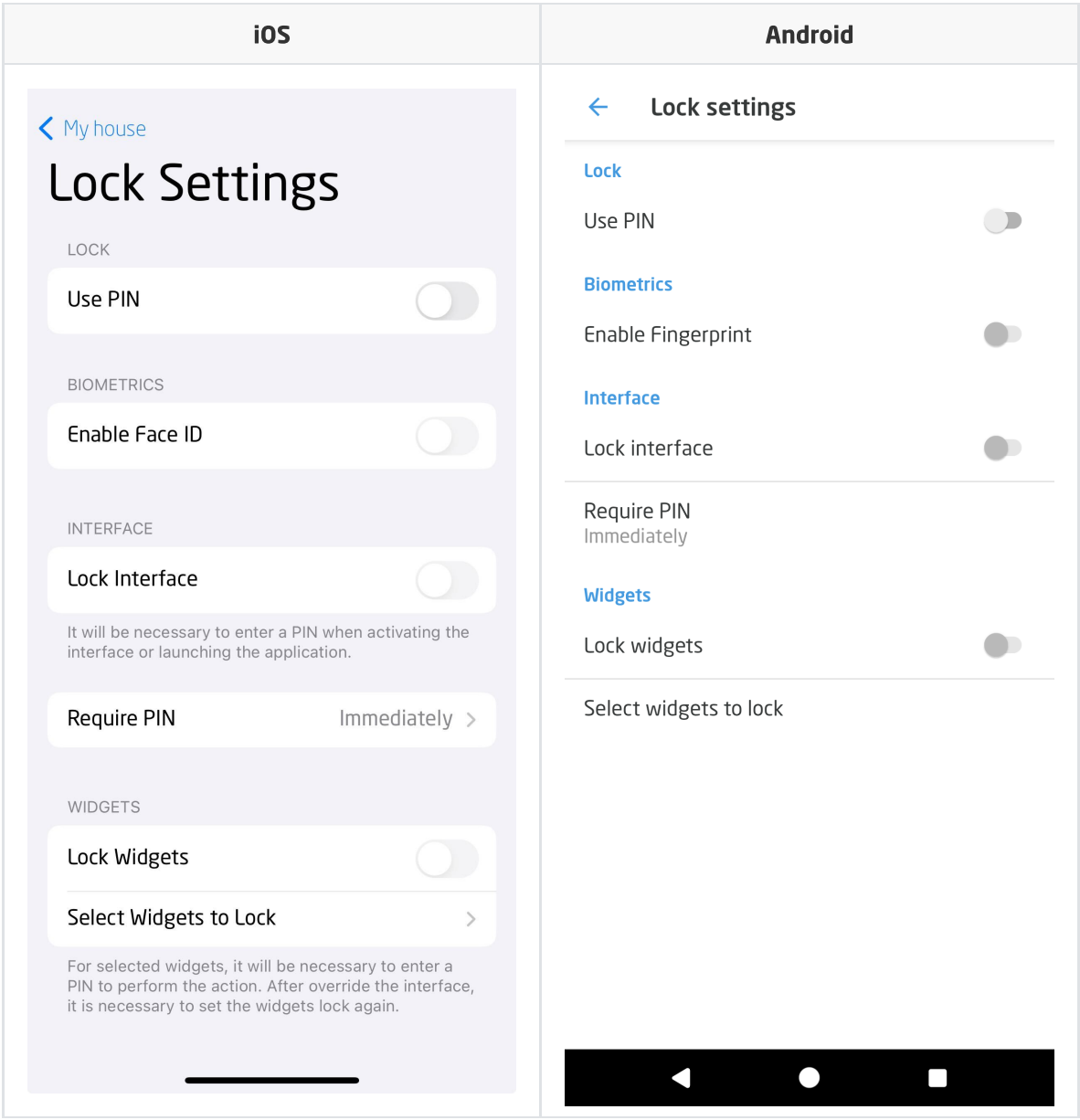
If the CLU has connected to the cloud, the interface sent to the mobile application on the Android device will use the remote connection by default. To switch to local communication, select the local system connection type. For devices running on iOS, the UseCloud parameter is turned off by default.



6.3. Interface and widget lock settings

Note!
This functionality is available for the myGrenton application version 1.9.18 (Android) / 1.12.1 (iOS) or higher.

Option available in the interface settings. Allows PIN configuration and locking of the interface or individual widgets. The request to unlock the interface will appear every time the myGrenton app is opened (immediately or after a specified time). By blocking only a single widget, the unblock request will appear when you try to perform an action or go to the widget details. Unlocking the interface or widget is possible using a PIN code or biometrics available on a given device.



Set or change PIN code

1. Go to Lock Settings, tap *Use PIN*.
2. Enter new PIN code (can be 4 to 8 digits).
3. Verify your new PIN.

Once the PIN has been set up, it is possible to unlock the interface and widgets using biometrics (Face ID or Touch ID for iOS devices, Fingerprint for Android) by clicking on the options `Enable Face ID` / `Enable Touch ID` (iOS) or `Use fingerprint` (Android). In addition, the option to change the PIN code appears.

> **Note!**
> If the interface is reloaded, the PIN code remains unchanged.

Lock interface

1. Go to Lock Settings, set PIN code.
2. Tap *Lock Interface*.

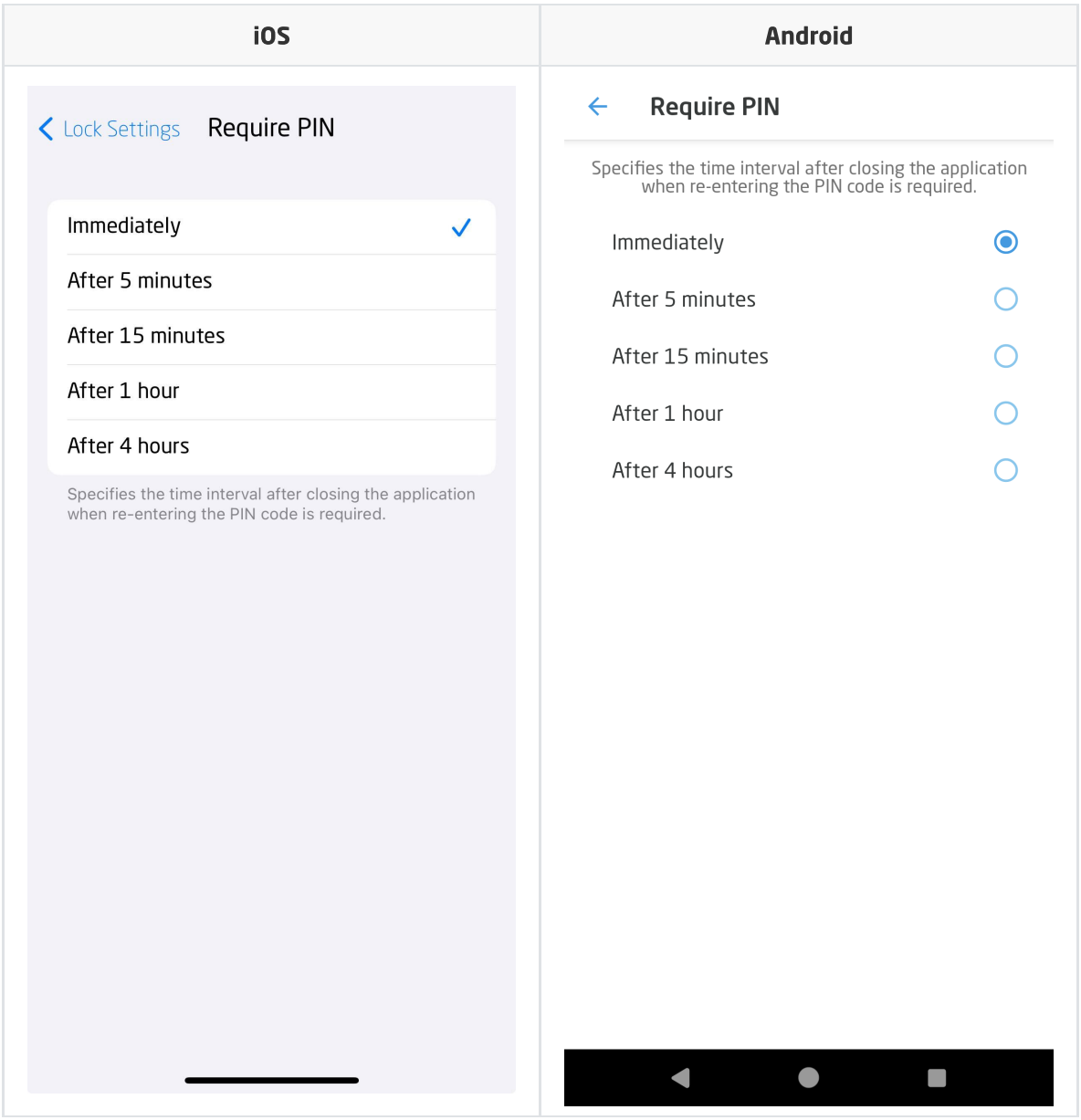
When the interface is locked, entering the PIN code is required in the following situations:

- Activating the interface,
- Activating or restarting the myGrenton app,

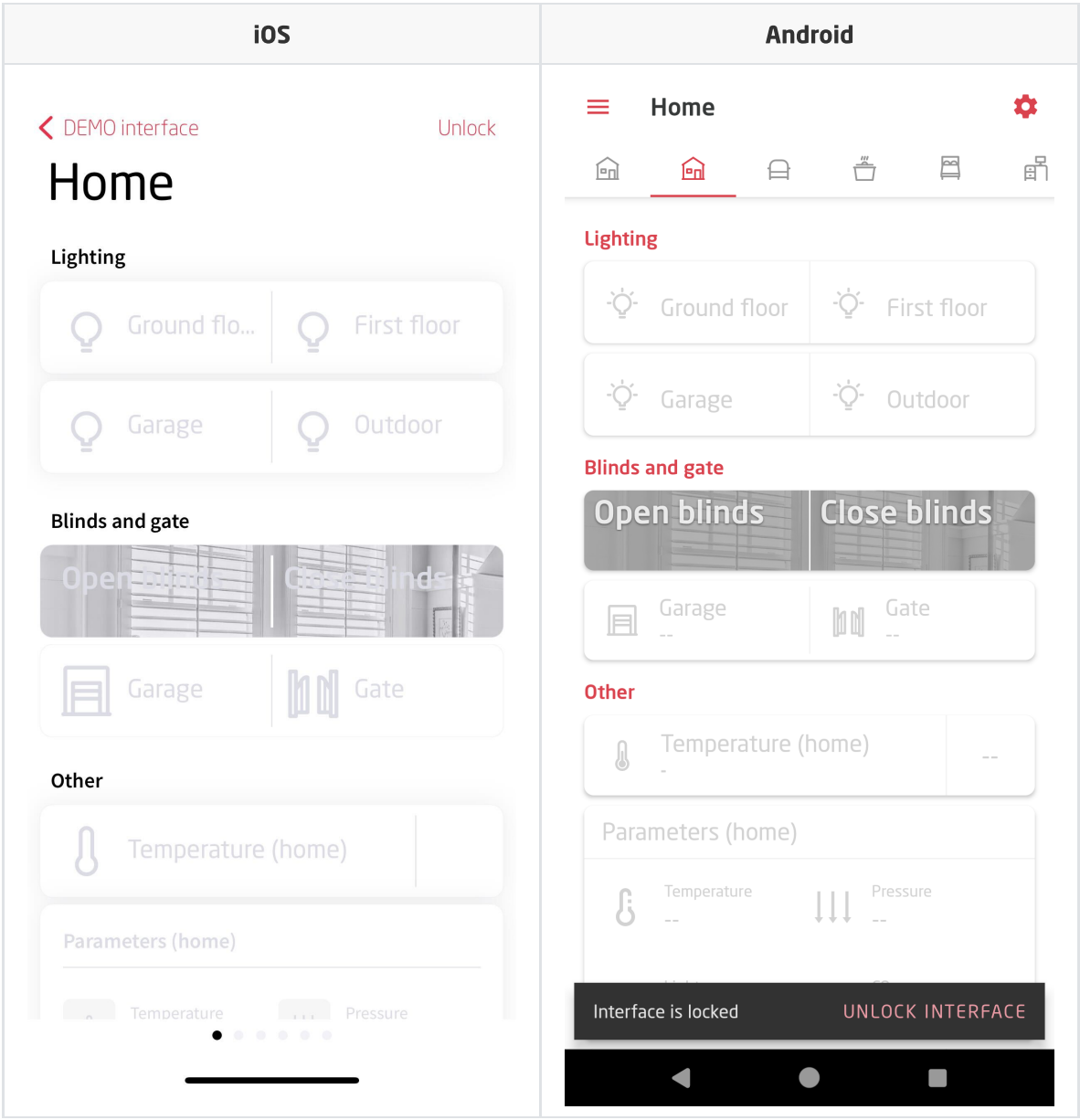
- Opening the interface lock settings.

It is possible to determine how long after closing or minimizing the application the interface will need to be unlocked again:

1. Tap *Require PIN* and then set the time.



If the PIN entry action is cancelled, the interface view will be displayed with inactive widgets (unable to read values, perform actions on the widget). To unlock, click on the `Unlock` button located in the top right corner of the screen (iOS) or `UNLOCK INTERFACE` located in the bar displayed at the bottom of the screen (Android).



Note!
Widgets will not be able to be used as shortcuts when the interface is blocked.

Lock Widgets

1. Go to Lock Settings, set PIN code.
2. Tap *Lock Widgets*.
3. Tap *Select Widgets to Lock* and then select the widgets you want to block.

Note!
Only widgets with a click action can be blocked.

Note!
If the interface is reloaded, it will be necessary to set the lock for individual widgets again.

Note!
Blocked widgets will not be able to be used as shortcuts.

Turn PIN off

1. Go to Lock Settings.
2. Tap *Use PIN*.

> **Note!**
> Once the PIN is disabled, all interface locking settings and widgets will be reset.

6.4. Intercom settings

> **Note!**
> Functionality available **only for the Grenton Intercom** and for the myGrenton application in version 1.10.20 (Android) / 1.13.0 (iOS) or higher.

Option available in the application settings. Allows SIP configuration for taking calls from Grenton entrance panel.

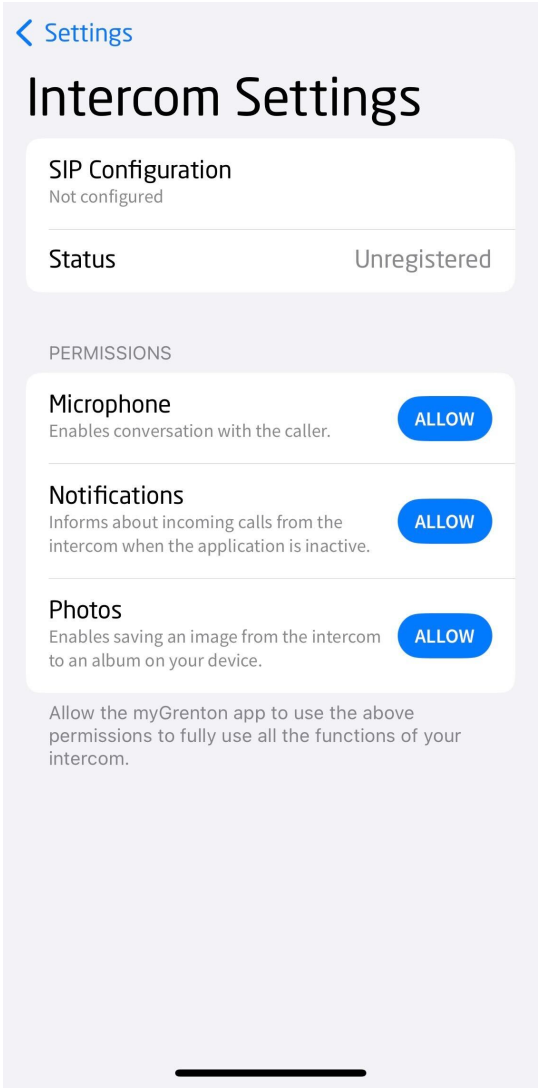
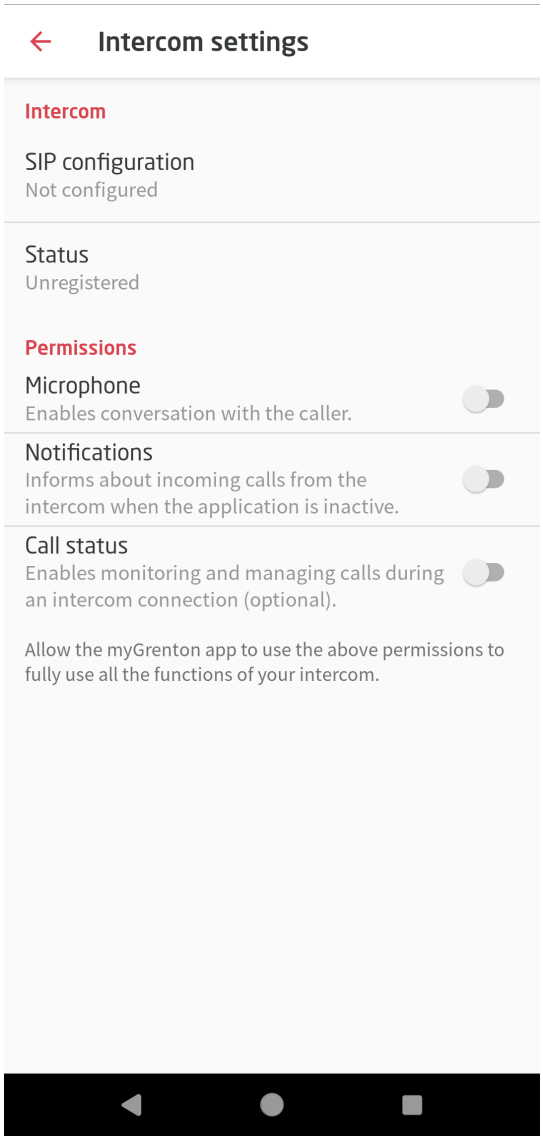
SIP Configuration

For the connection between the intercom and the myGrenton application, you must add your SIP number. The number must differ from the number indicated in the intercom settings. So, you need to have at least two SIP numbers to call from the intercom to the application. All information about registering a SIP account can be found here:

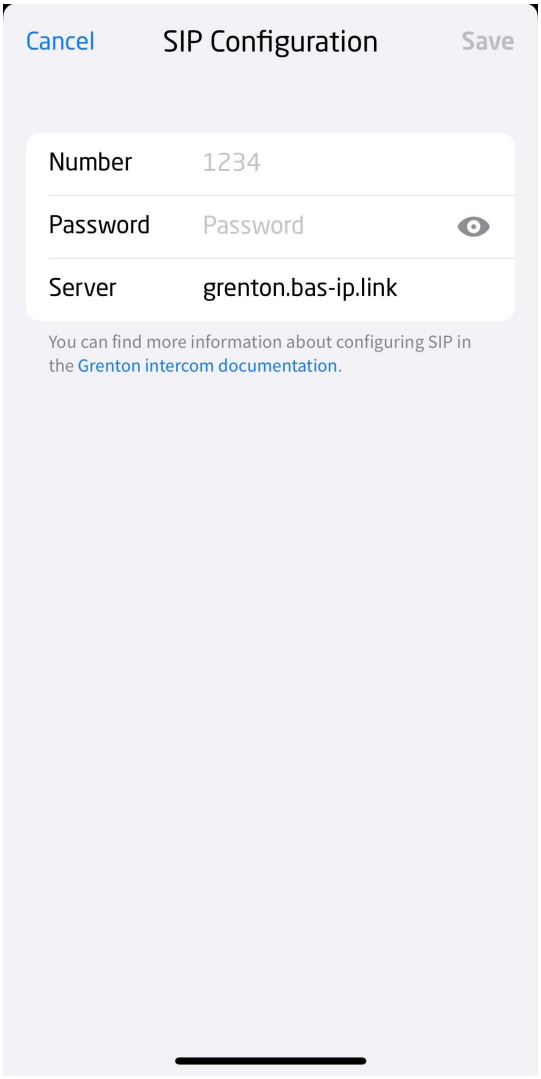
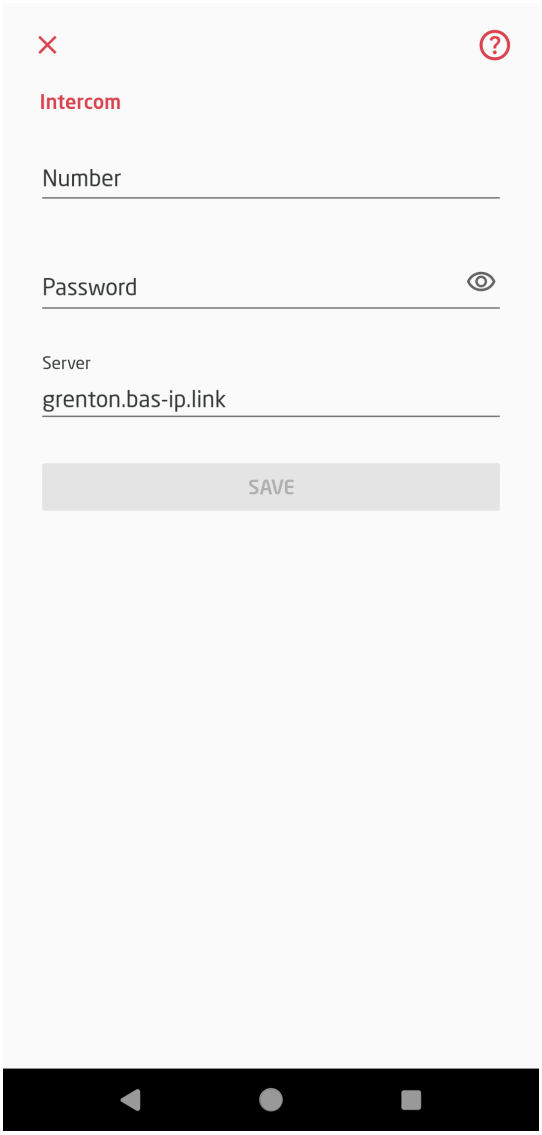
- https://grenton.com/sip-configuration

Once the SIP account is properly set up, add the number to the application:

1. Go to Intercom Settings.
2. Allow the myGrenton app to use the necessary permissions (tap "allow" for all permissions).

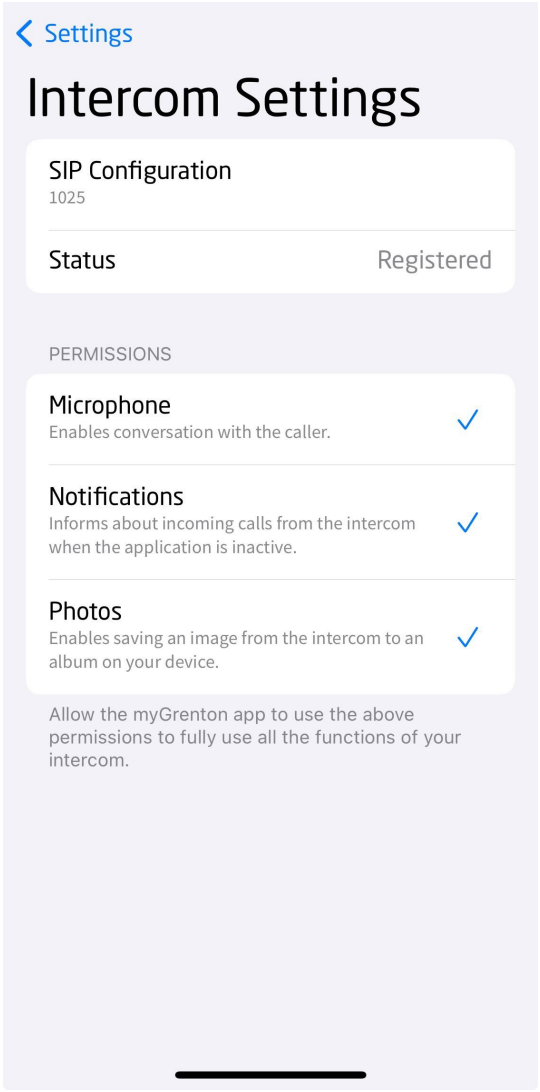
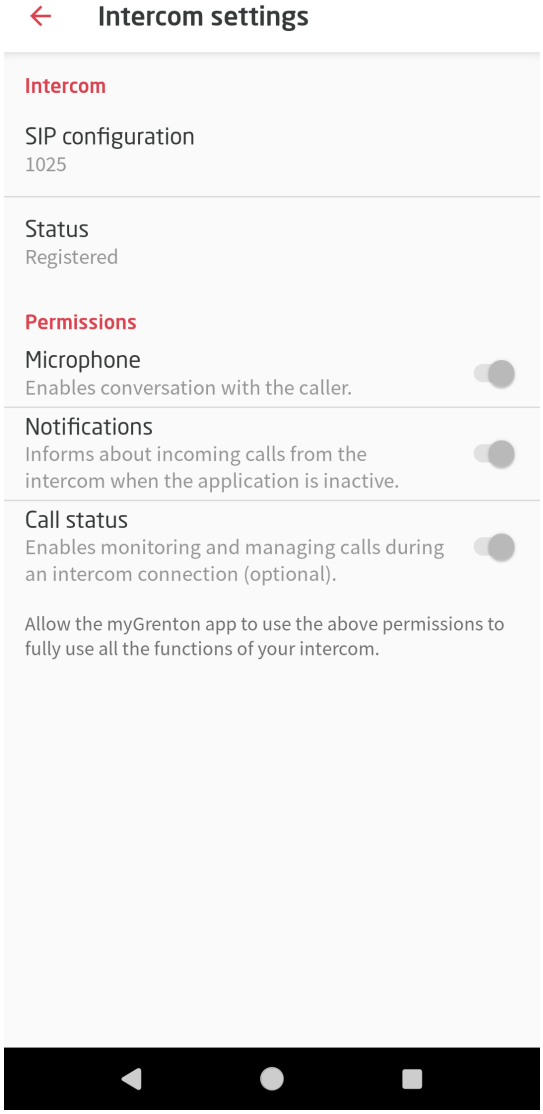
iOS	Android
	

3. Tap *SIP Configuration*.

iOS	Android
	

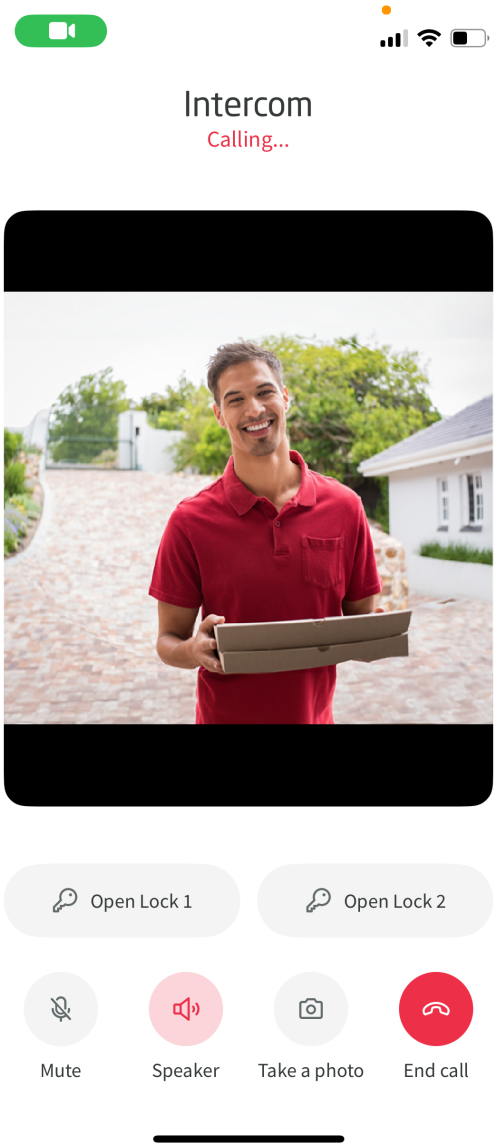
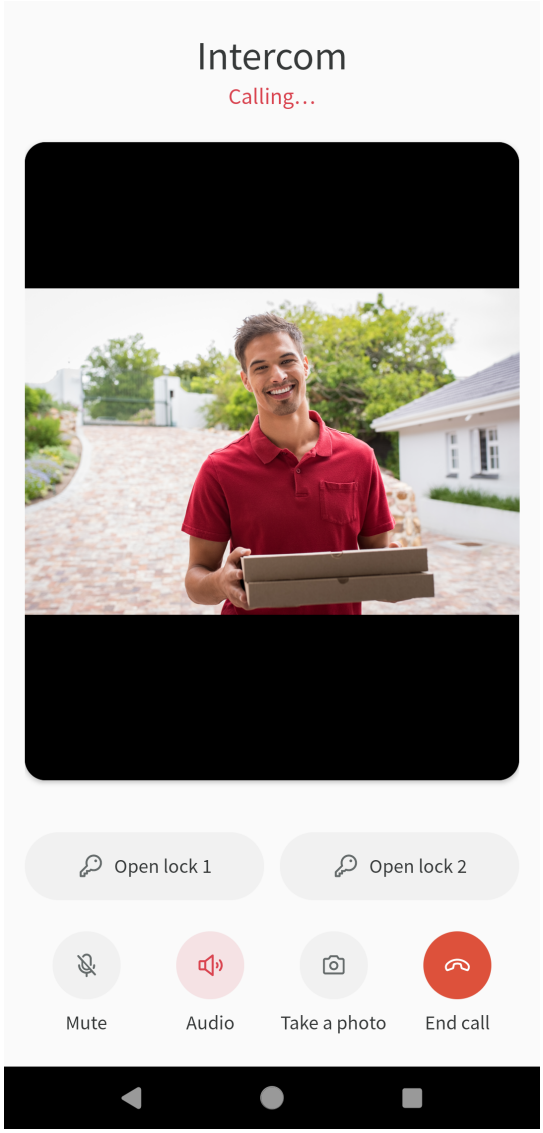
4. Enter information about SIP number and its password.
5. Tap *Save*.

When all steps are completed, the *Registered* status will be displayed.

iOS	Android
	

Call view

During a conversation, the following interface is displayed. Using the buttons in this interface, you can turn on/off microphone, speaker (by default, the speaker is on when the call is initiated), take a photo of the visitor, open the door and end the call:

iOS	Android
	

Intercom
Calling...
Open Lock 1
Open Lock 2
Mute
Speaker
Take a photo
End call

Intercom
Calling...
Open lock 1
Open lock 2
Mute
Audio
Take a photo
End call

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:

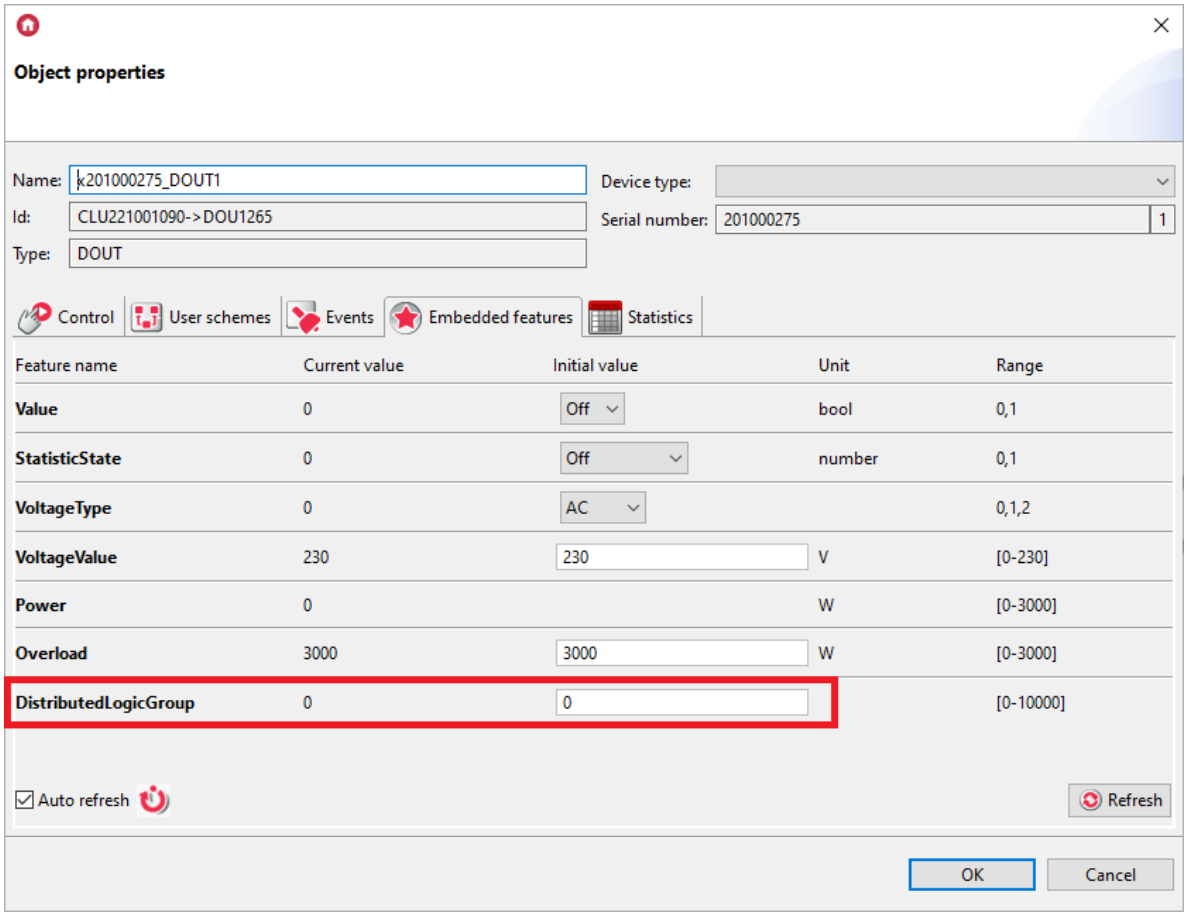
- GRENTON DIGITAL IN 6+3, DIN, TF-Bus (INP-209-D-01)
- GRENTON DIGITAL IN, Flush, TF-Bus (INP-210-T-01)
- GRENTON RELAY 4HP, DIN, TF-Bus (REL-204-D-01)
- GRENTON RELAY 2HP, DIN, TF-Bus (REL-202-D-01)
- GRENTON DIMMER MOSFET, DIN, TF-Bus (DIM-211-D-01)
- GRENTON I/O MODULE 8/8, DIN, TF-Bus (INO-288-D-01)
- 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.

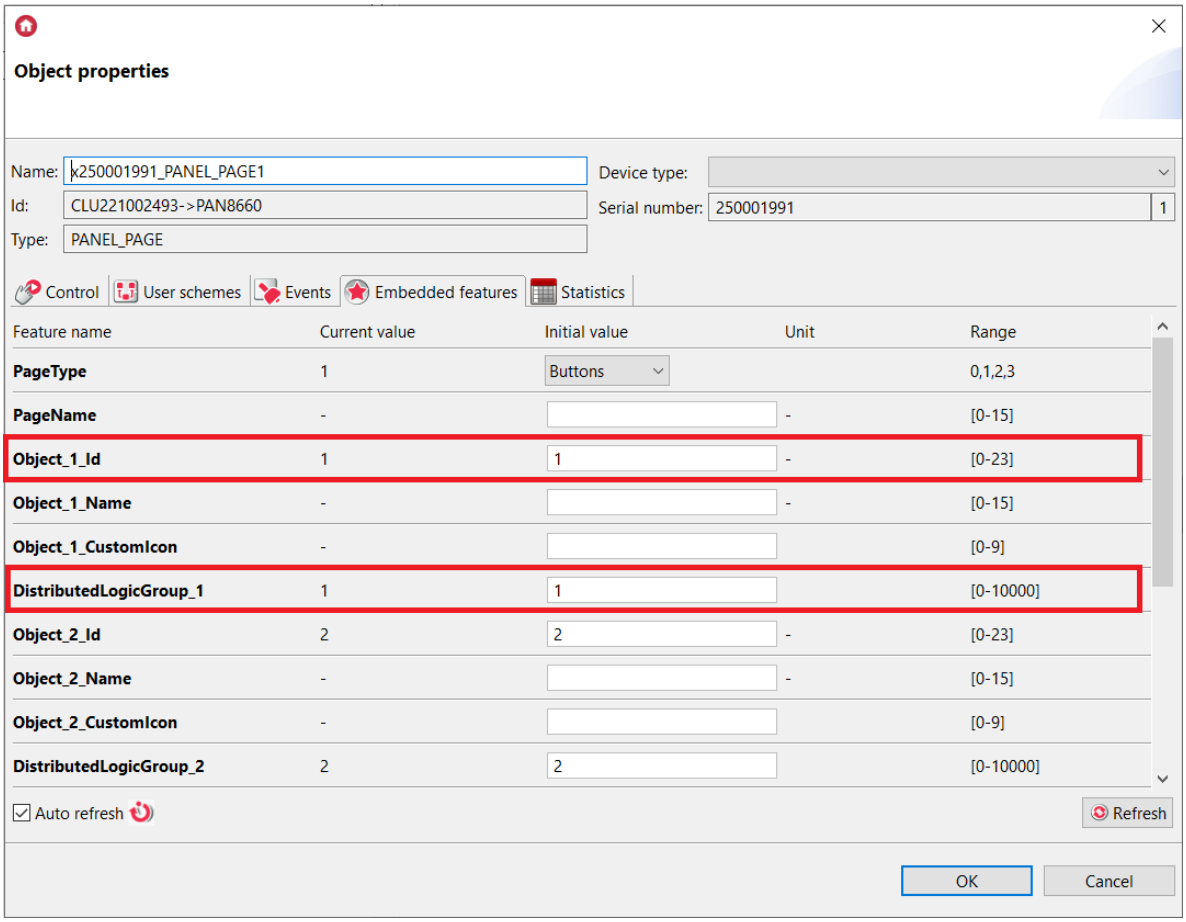
A. Mode configuration for IN / OUT modules



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



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.

XX. RS232 Controller

1. General information

The RS232 Controller module is a controller that allows integration with devices equipped with the RS232 interface.

2. Example of usage in scripts

2.1. Sending a command to the device without waiting for a response

Sending the ASCII command "PLAY":

```
CLU->SerialController->SetRepresentationType(1)  --  setting the data in ASCII format, can be configured from the built-in feature position.
CLU->SerialController->AddToTxBuffer("PLAY")  -- adding to the transmit buffer (Tx).
CLU->SerialController->SendTxBuffer(0)  -- sending the command without an end-of-line character (0), after sending, the transmit buffer (Tx) is cleared.
```

Sending the HEX command for three hexadecimal values: 0x12, 0xAB, and 0x34:

```
CLU->SerialController->SetRepresentationType(0)  --  setting the data in HEX format, can be configured from the built-in feature position.
CLU->SerialController->AddToTxBuffer(0x12) -- adding to the transmit buffer (Tx).
CLU->SerialController->AddToTxBuffer(0xAB) -- adding to the transmit buffer (Tx).
CLU->SerialController->AddToTxBuffer(0x34) -- adding to the transmit buffer (Tx).
CLU->SerialController->SendTxBuffer(0)  -- sending the command without an end-of-line character (0), after sending, the transmit buffer (Tx) is cleared.
```

2.2. Detecting the received command

Script checking if the received response contains the expression "Status: Play":

> The script analyzing should be set under the `OnReceive` event. To trigger the event for EVERY received message on Rx, you should set `ResponseSize = 1`.

```
x = CLU->SerialController->RxBuffer  -- Storing the buffer contents in variable x.
if (string.match(x, "Status: Play")) then  -- The string.match function will return "Status: Play" or nil. If it's nil, the condition will be treated as false.
 print("Status: Play recognized")
end
CLU->SerialController->ClearRxBuffer(0)  -- Clearing the entire (0-All) Rx buffer after analysis.
```

2.3. Detecting the received command with value analysis

Script checking if the received response contains the expression "Track: 25" and extracting the value 25:

> The script analyzing should be set under the `OnReceive` event. To trigger the event for EVERY received message on Rx, you should set `ResponseSize = 1`.

```
x = CLU->SerialController->RxBuffer  -- Storing the buffer contents in variable
x.
y = string.match(x, "Track: (%d+)")  -- The string.match function will return a
numeric value (%d+) or nil.
if y then  -- If y = nil, the condition will be treated as false.
 print("Track: " .. y)
end
CLU->SerialController->ClearRxBuffer(0)  -- Clearing the entire (0-All) Rx
buffer after analysis.
```

The variable y contains an expression in the string. To convert the value to a numeric representation, you should use the `tonumber()` function.

Script checking if the received response contains the expression "Temperature: 25.5°C" and extracting the value 25.5:

```
x = CLU->SerialController->RxBuffer  -- Storing the buffer contents in variable
x.
y = string.match(x, "Temperature: (%d+.%d+)°C")  -- The string.match function
will return a numeric value (%d+).(%d+) or nil.
if y then  -- If y = nil, the condition will be treated as false.
 print("Temperature: " .. y .. "°C")
end
CLU->SerialController->ClearRxBuffer(0)  -- Clearing the entire (0-All) Rx
buffer after analysis.
```

3. Object description

FEATURES

Name	Description
`RepresentationType`	Data representation type
`BaudRate`	Transmission speed
`WordLength`	Word length
`StopBits`	Number of stop bits
`Parity`	Bit parity control
`TxBuffer`	Transceiver buffer. Cleared automatically after calling SendTxBuffer
`RxBuffer`	Receive buffer
`ResponseSize`	Expected response size. This is the minimum number of bytes for which the OnReceive event will be triggered. If the message size is smaller than the specified number of bytes, the event will only trigger after accumulating the specified number of bytes in the buffer. If the value is set to 0, the OnReceive event will never be executed
`ResponseTimeout`	Response timeout

METHODS

Name	Description
`SetRepresentationType`	Sets the data representation type
`SetBaudRate`	Sets the transmission speed
`SetWordLength`	Sets the word length
`SetStopBits`	Sets the number of stop bits
`SetParity`	Sets the parity
`AddToTxBuffer`	Adds data to the transmit buffer
`SetResponseSize`	Sets the length of the expected response
`SetResponseTimeout`	Sets the response timeout
`ClearRxBuffer`	Clears the receive buffer. Operates on a first-in, first-out basis - the first element added to the queue will also be the first one removed
`ClearTxBuffer`	Clears the transmit buffer. Operates on a last-in, first-out basis - the last element added to the queue will be the first one removed
`SendTxBuffer`	Sends the transmit buffer

EVENTS

Name	Description
`OnReceive`	Event occurs when the controller has received data
`OnTransmit`	Event occurs when the controller sends data
`OnTimeout`	Event occurs when the response time has been exceeded
`OnOverflow`	Event occurs when the receive buffer is overflowed
`OnTransmitError`	Event occurs when data is sent incorrectly

---

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. ↵